De l'importance de valoriser l'expertise humaine dans l'annotation : application à la modélisation de textes en intentions à l'aide d'un clustering interactif

Erwan Schild

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

De l'Importance de Valoriser l'Expertise Humaine dans l'Annotation : Application à la Modélisation de Textes en Intentions à l'aide d'un Clustering Interactif

THÈSE

présentée et soutenue publiquement le 27 mars 2024

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Erwan SCHILD

Composition du jury

Président :	Pr. Kamel SMAILI
Rapporteurs :	Pr. Pascale KUNTZ-COSPEREC Pr. Mohamed NADIF
Examinateurs :	Dr. Adrien COULET Pr. Kamel SMAILI
Encadrants :	Dr. Jean-Charles LAMIREL Dr. Florian MICONI
Invités :	Dr. Gautier DURANTIN Pr. Pierre GANÇARSKI Dr. Thomas LAMPERT Dr. Mathieu POWALKA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503
En partenariat avec l'entreprise Euro Information, groupe Crédit Mutuel Alliance Fédérale

M i s e n p a g e a v e c l a c l a s s e t h e s u l .

Résumé

La tâche d'annotation, nécessaire à l'entraînement d'assistants conversationnels, fait habituellement appel à des experts du domaine à modéliser. Toutefois, l'annotation de données est connue pour être une tâche difficile en raison de sa complexité et sa subjectivité : elle nécessite par conséquent de solides compétences analytiques dans le but de modéliser les textes en intention de dialogue. De ce fait, la plupart des projets d'annotation choisissent de former les experts aux tâches d'analyse pour en faire des "super-experts".

Dans cette thèse, nous avons plutôt décidé mettre l'accent sur les connaissances réelles des experts en proposant une nouvelle méthode d'annotation basée sur un `Clustering Interactif`. Celle-ci se base sur une coopération Homme/Machine, où la machine réalise un *clustering* pour proposer une base initiale d'apprentissage, et où l'expert annote des contraintes `MUST-LINK` ou `CANNOT-LINK` entre les données pour affiner itérativement la base d'apprentissage proposée. Une telle annotation présente l'avantage d'être plus instinctive, car les experts peuvent associer ou différencier les données en fonction de la similarité de leur cas d'usage, permettant ainsi de traiter les données comme ils le feraient professionnellement au quotidien.

Au cours de nos études, nous avons pu montrer que cette méthode diminuait sensiblement la complexité de conception d'une base d'apprentissage, réduisant notamment la nécessité de formation des experts intervenant dans un projet d'annotation. Nous proposons une implémentation technique de cette méthode (algorithmes et interface graphique associée), ainsi qu'une étude des paramètres optimaux pour obtenir une base d'apprentissage cohérente en un minimum d'annotation. Nous réalisons également une étude de coûts (techniques et humains) permettant de confirmer que l'utilisation d'une telle méthode est réaliste dans un cadre industriel. De plus, afin que la méthode atteigne son plein potentiel, nous fournissons un ensemble de conseils, notamment : (1) des recommandations visant à cadrer la stratégie d'annotation, (2) une aide à l'identification et à la résolution des divergences d'opinion entre annotateurs, (3) des indicateurs de rentabilité pour chaque intervention de l'expert, et (4) des méthodes d'analyse de la pertinence de la base d'apprentissage en cours de construction.

En conclusion, cette thèse offre une approche innovante pour concevoir une base d'apprentissage d'un assistant conversationnel, permettant d'impliquer les experts du domaine métier pour leurs vraies connaissances, tout en leur demandant un minimum de compétences analytiques et techniques. Ces travaux ouvrent ainsi la voie à des méthodes plus accessibles pour la construction de ces assistants.

Mots-clés: `Apprentissage automatique` ; `Traitement automatique du langage naturel` ; `Annotation de contraintes` ; `Clustering sous contraintes` ; `Clustering interactif` ; `Assistant conversationnel`.

Abstract

Usually, the task of annotation, used to train conversational assistants, relies on domain experts who understand the subject matter to model. However, data annotation is known to be a challenging task due to its complexity and subjectivity. Therefore, it requires strong analytical skills to model the text in dialogue intention. As a result, most annotation projects choose to train experts in analytical tasks to turn them into "super-experts".

In this thesis, we decided instead to focus on the real knowledge of experts by proposing a new annotation method based on `Interactive Clustering`. This method involves a Human-Machine cooperation, where the machine performs clustering to provide an initial learning base, and the expert annotates `MUST-LINK` or `CANNOT-LINK` constraints between the data to iteratively refine the proposed learning base. Such annotation has the advantage of being more instinctive, as experts can associate or differentiate data according to the similarity of their use cases, allowing them to handle the data as they would professionally do on a daily basis.

During our studies, we have been able to show that this method significantly reduces the complexity of designing a learning base, notably by reducing the need for training the experts involved in an annotation project. We provide a technical implementation of this method (algorithms and associated graphical interface), as well as a study of optimal parameters to achieve a coherent learning base with minimal annotation. We have also conducted a cost study (both technical and human) to confirm that the use of such a method is realistic in an industrial context. Finally, we provide a set of recommendations to help this method reach its full potential, including : (1) advice aimed at framing the annotation strategy, (2) assistance in identifying and resolving differences of opinion between annotators, (3) rentability indicators for each expert intervention, and (4) methods for analyzing the relevance of the learning base under construction.

In conclusion, this thesis provides an innovative approach to design a learning base for a conversational assistant, involving domain experts for their actual knowledge, while requiring a minimum of analytical and technical skills. This work opens the way for more accessible methods for building such assistants.

In conclusion, this thesis provides an innovative approach to design a learning base for a conversational assistant, involving domain experts for their actual knowledge, while requiring a minimum of analytical and technical skills. This work opens the way for more accessible methods for building such assistants.

Keywords: `Machine Learning` ; `Natural Language Processing` ; `Constraints annotation` ; `Constrained clustering` ; `Interactive clustering` ; `Chatbot`.

Financements

Ce doctorat a été financé par ANRT (Association Nationale de la Recherche et de la Technologie) au sein d'un contrat CIFRE (n°2019.0289) entre l'entreprise Euro Information (groupe Crédit Mutuel Alliance Fédérale) et le laboratoire du LORIA (Université de Lorraine).

Remerciements

J'ai pu travailler pendant près de quatre ans sur un sujet passionnant, et, grâce au soutien et à la contribution de nombreuses personnes, je peux désormais vous présenter le fruit de mes intenses réflexions.

Par conséquent, je souhaite exprimer toute ma gratitude envers :

- ☑ Toi, lecteur, qui me fait l'immense honneur de lire ce manuscrit ! 👍
- ☑ Euro Information (partenaire industriel), le LORIA (partenaire académique) et l'ANRT : pour avoir contribué au financement et à l'encadrement de cette thèse CIFRE, me permettant ainsi de travailler dans de bonnes conditions ;
- ☑ Jean-Charles, mon encadrant académique : pour tout ce que tu as pu m'apprendre au cours de ces quatre années, pour ta disponibilité, et pour la confiance que tu m'as accordée tout au long de ce doctorat ;
- ☑ Florian, mon encadrant industriel : pour avoir subi à ma place une bonne partie de la paperasse administrative (😜) et pour m'avoir fait l'honneur d'être le premier doctorant chez Euro Information ;
- ☑ Gautier : pour ton aide inestimable tout au long de mon doctorat, pour m'avoir aidé à structurer mon sujet, pour m'avoir appris comment rédiger un article scientifique, et pour tes remarques toujours pertinentes sur mon travail ;
- ☑ Amélie : pour toutes les discussions que nous avons eues, pour l'intérêt que tu as porté à mes travaux, et pour tes encouragements ;
- ☑ Mathieu et toute l'équipe H330 : pour la bonne ambiance au bureau ainsi que le sérieux de nos réunions d'équipe (😜) ;
- ☑ Ma *machine à café*, ma *boîte de bretzels* et le *moteur de recherche Google*, sans qui cette thèse aurait été beaucoup plus compliquée que prévu ! 😅
- ☑ Marie, Jeremy et William : pour m'avoir aidé dans la révision de mes jeux de données ;
- ☑ Adrien, Amélie, Arthur, Baptiste, Bourhan, Iris, Julien, Lise, Marceau, Marie, Mathieu, Quentin, Robin et Thibaud : pour avoir participé en tant que ~~cobayes~~ opérateurs lors de mes expériences d'annotation de contraintes ;
- ☑ Clémentine, Thomas et Thomas, puis David, Esther et Marc (les élèves de l'École d'Ingénieurs Télécom Physique Strasbourg) : pour avoir contribué aux développements logiciels de mon `Clustering Interactif` lors de vos projets étudiants ;
- ☑ M'a su paire be relek triss : sang qui m'ont manu skry ceux raie il ysible ![1] ! [1] !
- ☑ Tous mes amis et ma famille : vous qui m'avez ~~supporté~~ soutenu jusqu'au bout (*Spéciale dédicace à ma très chère consoeur et à mes parents que j'aime tout beaucoup* 😄) ;
- ☑ et *bien entendu*, merci à ma merveilleuse épouse : pour avoir été une oreille attentive, pour tes petites attentions, et pour ta capacité naturelle à me faire sourire au quotidien : merci d'exister ! ❤

1. Ma superbe relectrice, sans qui mon manuscrit serait illisible !

Je dédie cette thèse à tous ceux que j'aime bien.
Si vous lisez ce message, alors vous en faîtes probablement partie.

Table des matières

Chapitre 1

Introduction

Idéalisation de l'IA aux yeux du grand public

L'Intelligence Artificielle (IA) a connu une démocratisation massive ces dernières années. Elle est considérée comme une révolution majeure de notre société, à tel point qu'il devient presque impossible de s'en passer :

— Vous avez besoin de trouver votre chemin ? utilisez votre GPS.

— Vous avez un problème avec une commande ou besoin d'un service après-vente ? un bot informatique est disponible jour et nuit pour traiter votre demande.

— Vous ne savez plus quelle série regarder ? Netflix peut faire des suggestions personnalisées.

— Vous avez les mains pleines de farine et vous voulez lancer un minuteur ou écouter de la musique ? Demandez-le à OK Google ou Alexa.

— Il vous manque une belle image pour votre présentation ? DALL-E peut la générer.

— Vous devez rédiger une dissertation en histoire-géographie ? ChatGPT s'en occupera.

— La classe LaTeX proposée par votre école doctorale ne compile pas[2] ? ChatGPT peut aussi identifier l'erreur et même la corriger...

Les modèles d'IA s'immiscent ainsi dans la plupart des activités de notre quotidien. Cependant, cette omniprésence est aussi source de confusion et d'incompréhension pour le grand public. En effet :

— L'IA peut être perçue comme une menace, soit parce qu'elle vole des emplois, soit parce qu'elle risquerait de devenir incontrôlable. Ces craintes sont notamment véhiculées par la culture populaire, à l'image de Terminator ou d'Ultron qui se sont retournés contre leurs créateurs.

— Les attentes des utilisateurs sont parfois trop élevées par rapport aux capacités réelles du modèle. Il en résulte alors un sentiment de frustration, en particulier lorsque l'utilisateur exprime un besoin urgent, mais que le modèle se contente de répondre qu'il n'a pas compris la question et que vous devriez reformuler votre demande.

— Le crédit accordé aux modèles d'IA est parfois excessif, au point que l'esprit critique des

2. Toute ressemblance avec une situation réelle ou ayant existé est purement fortuite !

utilisateurs s'efface petit à petit. Les capacités spectaculaires des derniers modèles génératifs accentuent davantage cette confiance aveugle, et il devient même difficile d'identifier les fausses informations générées tant elles semblent crédibles (*par exemple,* `ChatGPT` *peut vous mentir avec conviction en inventant certains détails dans ses réponses*).

L'idée reçue selon laquelle il est facile de concevoir un modèle d'`IA` explique en partie ces confusions. Encore une fois, la culture populaire véhicule cette image d'une conception accessible à tous et à moindres coûts. En reprenant l'exemple d'*Ultron*, il suffit au `Dr. Hank Pym` de « calquer ses schémas mentaux » pour créer le robot, comme si cela était un acte banal ; plus récemment, dans la série `Black Mirror` (S2 Ep1), le personnage de *Martha* se procure un avatar de son conjoint décédé sans trop de difficultés en communiquant simplement les messages et les photos de ce dernier, mais aucune mention n'est faite sur le processus de conception, excepté qu'il est « expérimental ».

Toutes ces illusions masquent ainsi un point pourtant fondamental à la conception des modèles d'`IA` : la nécessité d'avoir des données de qualité pour l'entraînement.

Désillusion quant à la simplicité de l'annotation de données

Bien qu'il ne soit pas limité à celà, nous pouvons résumer l'apprentissage automatique comme étant majoritairement un ensemble de méthodes permettant de reproduire un phénomène par l'exemple. En d'autres termes, il faut des données en qualité et en quantité suffisante pour concevoir la base d'apprentissage d'un modèle d'`IA`. C'est là qu'intervient la **tâche d'annotation** : celle-ci consiste à demander à un expert du métier, c'est-à-dire à un spécialiste du phénomène, d'enrichir les données pour leur attribuer une signification, de leur fournir de la valeur ajoutée, et ainsi permettre à la machine de comprendre et reproduire le phénomène.

Pour illustrer nos propos, prenons l'exemple des assistants conversationnels (*chatbot*). Ces assistants ont pour objectif de traiter automatiquement des requêtes exprimées en langage naturel. Pour ce faire, il est possible de réaliser une modélisation de textes en intentions de dialogue. À ce titre, des requêtes comme « *joue moi du jazz s'il te plaît !* » ou « *peux-tu lancer une playlist de Noël sur l'enceinte du salon !* » peuvent être modélisées par l'intention `jouer_musique`. La base d'apprentissage d'un assistant conversationnel est alors constituée d'un ensemble de textes qui ont été annotés en intention.

L'aspect élémentaire de notre exemple cache cependant toute la difficulté de cette tâche :

— Il est possible d'avoir une vaste diversité d'intentions (*jouer de la musique, allumer la lumière, consulter la météo, démarrer un minuteur, appeler sa maman, ...*) : la complexité grandit d'autant qu'il y a de cas d'usage modélisables ;

— L'interprétation du langage est complexe par essence : le vocabulaire employé peut être spécifique, une requête peut être ambiguë ou à double sens, une erreur grammaticale peut gêner la compréhension de la phrase, ...

— La modélisation en intentions est un exercice subjectif durant laquelle deux annotateurs peuvent avoir des avis différents : dans notre exemple, et avec les mots « *peux-tu lancer ...* », est-ce que l'assistant devrait effectuer une action, ou devrait-il simplement exprimer s'il en est capable ?

Ainsi, l'annotation de textes en intentions met en évidence la complexité de cette tâche. Ben

Hamner, cofondateur et directeur technique de l'entreprise `Kaggle`, résume ainsi cette problématique : **l'`IA` c'est « 1% d'écriture de code informatique, 9% d'analyse de ce qui ne va pas dans le code informatique, et 90% d'analyse de ce qui ne va pas dans les données d'entraînement »**.

Intervention difficile des experts métiers dans les projets de conception de base d'apprentissage

Pour absorber la complexité liée à la tâche d'annotation, un projet de conception d'une base d'apprentissage s'organise généralement de manière cyclique.

— En premier lieu, les experts métiers sont consultés lors d'ateliers d'idéation pour définir une première version de la modélisation (*dans le cadre de notre exemple sur les assistants conversationnels, ils définiraient la liste des intentions possibles*).

— Dans un second temps, les experts métiers parcourent l'ensemble des données pour les annoter en fonction de la modélisation définie ;

— Si certaines frictions apparaissent (*modélisation inadaptée, différences d'avis en annotateurs, ...*), le projet retourne à l'étape de modélisation pour proposer une nouvelle version révisée, et le cycle recommence...

Une telle approche a l'avantage de régler la complexité de la tâche par de petits ajustements, s'apparentant ainsi aux méthodes agiles. Toutefois, elle possède les inconvénients d'être chronophage et onéreuse. En effet, à chaque remise en cause de la modélisation, toutes les données annotées sont potentiellement à revoir pour s'assurer de leur compatibilité avec la nouvelle modélisation proposée. D'autre part, la définition d'une modélisation stable et cohérente, ainsi que l'encadrement de ses remises en cause potentielles, relève du domaine analytique, qui n'est pas le domaine principal d'expertise des annotateurs intervenant dans le projet. Il est donc nécessaire de former les experts métiers à la tâche d'analyse de données pour que leurs remarques soient le plus pertinentes possibles lors des revues de modélisations.

Nous touchons alors du doigt un paradoxe manifeste : **comment sommes-nous arrivés à la conclusion que la meilleure manière de faire intervenir un expert métier sur une tâche d'annotation, c'est en lui demandant une tâche pour laquelle il n'est pas expert ?** (*En effet, si nous voulions par exemple faire un assistant conversationnel sur un domaine gastronomique, nous engagerions a priori un chef cuisinier ou un restaurateur ; toutefois, il semblerait incongru d'engager ces profils dans le but de réaliser des analyses statistiques ou d'être compétents sur des questions linguistiques.*)

En conclusion, l'organisation traditionnelle des projets d'annotation ne semble pas résoudre la complexité de cette tâche : au contraire, elle semble plutôt l'ignorer en espérant qu'un opérateur humain suffisamment formé la résolve tout seul.

Recherche d'une méthode alternative pour modéliser le texte en intentions tout en valorisant l'intervention de l'expert

Au cours de ce doctorat, nous nous sommes demandé s'il était possible de changer la philosophie traditionnelle régissant les projets d'annotation, avec pour objectif de remettre l'expert métier au centre du processus. Pour cela, nous nous sommes restreints au cas d'usage de la modélisation du texte en intentions, en prenant comme contrainte le fait de toujours impliquer un expert métier pour ses vraies connaissances, et en lui demandant un minimum de connaissances analytiques ou techniques.

Nous avons donc proposé une méthodologie d'annotation assistée par la machine et basée sur la caractérisation de contraintes de similarité. Dans ce manuscrit, nous organisons la présentation et l'étude de cette méthodologie de la manière suivante :

— Au cours du Chapitre 2, nous présentons une revue de littérature sur la tâche d'annotation, son organisation traditionnelle ainsi que les nombreux défis qu'elle comporte. Pour mieux illustrer nos propos, nous utilisons des exemples inspirés de l'univers de la bande dessinée.

— Nous complétons la revue de littérature en expliquant le contexte de ce doctorat en Section 2.4 : ce complément nous permet de mettre en évidence la difficulté d'intervention des experts métiers dans un projet traditionnel d'annotation pour la conception d'assistants conversationnels.

— Le Chapitre 3 est dédié à la présentation de notre méthodologie d'annotation alternative basée sur un `Clustering Interactif`. La description de l'implémentation technique est consultable dans l'Annexe C.

— Dans le Chapitre 4, nous décrivons les six hypothèses que nous voulions vérifier sur notre méthodologie d'annotation : efficacité, efficience, coûts, pertinence, rentabilité et robustesse.

— Le Chapitre 5 fait le point sur l'ensemble des discussions et découvertes contenues des précédents chapitres, et comporte différents avis et conseils pratiques. Le chapitre entier est prévu pour être un guide d'utilisation synthétique de notre méthodologie d'annotation.

Le Chapitre 6 dresse la conclusion et clôt la discussion en abordant des thématiques et perspectives plus générales.

Chapitre 2

Revue de littérature sur la tâche d'annotation en intelligence artificielle

L'annotation (ou labellisation) de données est une tâche essentielle de l'apprentissage automatique, permettant de décrire des jeux de données nécessaires à l'entraînement et à l'évaluation de modèles d'intelligence artificielle. Dans ce chapitre, nous allons traiter les points suivants :

- ☐ Définir à quoi correspond une tâche de labellisation, et détailler la vaste organisation technique et méthodologique autour d'un projet d'annotation de données ;
- ☐ Présenter un aperçu des difficultés principales que peut rencontrer un projet d'annotation, ainsi que certaines techniques conçues pour s'en prévenir ;
- ☐ Expliquer le contexte industriel dans lequel ce doctorat s'inscrit et montrer l'intérêt d'assister la conception des jeux de données d'entraînement.

Sommaire

2.1 Présentation théorique de l'annotation

Tout d'abord, introduisons quelques définitions pour appréhender le concept d'« `annotation` » et donnons quelques exemples pour comprendre les enjeux qui y sont associés.

2.1.1 Définition et objectifs de l'annotation de données

2.1.1.a Qu'est que l'« `apprentissage automatique` » ?

Nous proposons la définition suivante inspirée de l'`ACM` (*Association for Computing Machinery*) : l'« `apprentissage automatique` » (ou « `Machine Learning` ») est une branche de l'intelligence artificielle dédiée au développement de méthodes permettant à l'ordinateur de **reproduire une tâche par l'exemple** : il n'est donc pas explicitement programmé pour réaliser cette tâche, mais il l'« apprend » à l'aide d'un modèle mathématique. Cet apprentissage peut être *supervisé* (l'interprétation des exemples est fournie par un humain), *non supervisé* (la machine déduit l'interprétation des données sans intervention humaine) ou *semi-supervisé* (mélange des deux précédentes approches).

Le *Machine Learning* permet ainsi d'automatiser l'analyse et la manipulation de certains phénomènes complexes tels que le langage, l'observation visuelle, la détection d'anomalies, le traitement acoustique, ...

> **Pour information :** Si vous voulez revoir les bases de l'apprentissage automatique, des livres comme Zhou, 2021 ou Raschka et Mirjalili, 2019 traitent des notions principales et de leur mise en application.

2.1.1.b Qu'est qu'un « `corpus d'entraînement` » ?

Pour concevoir un modèle en apprentissage automatique, il nous faut un ensemble d'exemples (textes, images, sons, vidéos, ou tout autre relevé d'informations) permettant de capturer le phénomène à appréhender : cela nous aide à la fois à le décrire et à mieux le comprendre. Nous utilisons alors les termes « `corpus d'entraînement` », « `jeu d'entraînement` » ou « `base d'apprentissage` » pour désigner cet ensemble de données.

Il est important de noter qu'un corpus n'est qu'un échantillon de taille finie d'un phénomène pouvant être infini ou indénombrable. Il est donc d'usage de valoriser cet échantillon s'il est « `représentatif` » du phénomène qu'il décrit, c'est-à-dire s'il capture bien le large éventail de variations que peuvent prendre les données (Biber, 1993).

> **Pour information :** Nous discuterons davantage de cette notion de représentativité dans la Section 2.3.1.a. D'autre part, si vous voulez mieux comprendre cette notion de corpus, vous pouvez vous référer à Sinclair, 2004 issu du livre *Developing Linguistic Corpora* (Wynne, 2004).

2.1.1.c Qu'est que l'« `annotation` » ?

Les données d'un corpus manquent parfois d'informations pour bien cerner un phénomène, il est alors nécessaire de faire intervenir un humain pour introduire des connaissances supplémentaires qui ne sont pas explicitement présentes dans ces données. Nous appelons « `annotation` »

(ou « étiquetage », « labellisation ») cette tâche consistant à décrire les données d'un corpus, et nous distinguons ainsi les données dites « brutes » (utilisées par les approches non supervisées) des données dites « annotées » (utilisées par les approches supervisées) en fonction de l'absence ou de la présence d'un complément d'informations.

Les informations renseignées peuvent porter sur la donnée entière ou sur une partie seulement, peuvent concerner des variables catégorielles (ensemble fini) ou numériques (ensemble infini), et peuvent aussi être cumulatives ou mutuellement exclusives. Dans la littérature, Garside et al., 1997 présentent l'annotation comme la tâche permettant de donner une « valeur ajoutée » aux données ; de son côté, Leech, 2004 précise que l'annotation est une action d'« interprétation » qui aide à la compréhension et à la reproduction d'un phénomène, mais aussi au contrôle du comportement des modèles d'apprentissage automatique.

2.1.2 Exemples de tâches d'annotation

Les définitions données dans la section précédente peuvent paraître abstraites car il est difficile de dépeindre la vaste diversité d'applications nécessitant des données labellisées. En effet, une tâche d'annotation répond toujours à un besoin précis, mais il y a une telle multiplicité de types de données (*données tabulaires, textuelles, visuelles, auditives, ...*) et de cas d'usages (*prédiction d'une valeur numérique (tâche de régression), prédiction d'une catégorie (tâche de classification), détection d'objets (tâche d'extraction), création de nouvelles données (tâche de génération), ...*) qu'une unique définition ne peut être que purement théorique.

Ainsi, nous estimons qu'il est préférable de compléter ces définitions par quelques exemples concrets. Nous pourrons ainsi mieux dresser le portait d'une tâche d'annotation, avec ses intérêts et ses complications. Pour cela, nous allons prendre le thème de la bande dessinée (BD) et explorer ensemble différents cas d'usage qui pourraient intéresser un auteur, un libraire ou un lecteur.

2.1.2.a Estimation du prix d'une bande dessinée.

Les acheteurs et les vendeurs de bandes dessinées s'interrogent forcément sur le juste prix de l'oeuvre qu'ils veulent acquérir ou céder. Répondre à cette question avec précision nécessite diverses informations, à la fois sur l'oeuvre (comme son identification ou l'avis de ses lecteurs), sur le document en tant que tel (comme son état de conservation), mais aussi sur le prestige de son édition (éditions originales ou de collection). Sans un regard d'expert, il est possible de trouver certaines oeuvres rares vendues pour presque rien sur le marché d'occasion, ou à l'inverse voir certaines BD être achetées à prix d'or alors que le document est en piteux état.

Afin d'aiguiller les acquéreurs, il est possible d'utiliser un modèle de régression[3] permettant de prédire le prix d'une BD à partir des différentes métadonnées à disposition. Mais pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples de transactions avec leur prix de vente. Nous pouvons structurer l'ensemble des informations nécessaires dans un tableau, et la tâche d'annotation consiste alors à renseigner pour chaque transaction :

— l'identification complète de la BD (titre, auteur, édition, ...),

— l'état du document grâce à un regard d'expert (l'état peut par exemple être défini par une variable catégorielle dont les valeurs seraient "Mauvais état", "Bon état", "Très bon état", "Neuf") ;

— le prix de la BD, estimé ou réel (défini par une variable numérique).

3. Pour plus de détails sur la régression : voir la revue de Maalouf, 2011 ; voir un exemple basé sur la méthode des moindres carrés dans Zdaniuk, 2014.

Un exemple de résultat d'annotation de ces données est disponible dans la TABLE 2.1.

Exemples :

Collection	N° : Titre	Édition	Note	État	Prix (€)
Lucky Luke	01 : La mine d'or de Dick Digger	1949	3.2/5	Très bon	5 000,00
Lucky Luke	12 : Les cousins Dalton	1958	4.3/5	Bon	40,00
Lucky Luke	12 : Les cousins Dalton	1962	4.3/5	Très bon	65,00
Lucky Luke	12 : Les cousins Dalton	1985	4.3/5	Très bon	6,00
Lucky Luke	15 : L'évasion des Dalton	1960	4.1/5	Mauvais	3,00
...					

TABLE 2.1 – *Exemple d'annotation du prix de vente de bandes dessinées en fonction de leur édition, de la note de leur lecteurs et de leur état (source : https://www.bedetheque.com/serie-213-BD-Lucky-Luke.html).*

Ainsi, si quelqu'un s'intéresse au prix d'une nouvelle bande dessinée pour lequel il n'y a pas de référence tarifaire, il peut interroger le modèle de régression qui proposera un prix en accord avec les exemples dont il dispose dans sa base d'apprentissage.

Pour information : De manière équivalente, il est possible de faire de la régression dans d'autres domaines, notamment pour prédire un volume, une surface, une quantité, ... La tâche d'annotation consistera à chaque fois à renseigner la valeur numérique à prédire en fonction des différentes données à disposition.

2.1.2.b Classification de l'état d'une bande dessinée à partir d'une photo.

Il est d'usage d'adapter le prix de vente d'un produit en fonction de son état, et nous avons intégré ce facteur dans l'estimation du prix d'une bande dessinée (voir exemple précédent). Cependant, l'état de conservation n'est pas une notion objective et chacun peut avoir des références différentes. Au final, c'est souvent un libraire qui détermine si l'oeuvre est en bon ou en mauvais état, et, sans un regard d'expert, nous pouvons omettre un détail ou nous tromper lors de notre appréciation.

Afin de nous aider à estimer l'état d'une bande dessinée, il est possible d'utiliser un modèle de `classification`[4] permettant, à partir d'une image, d'affecter à chaque `BD` une catégorie prédéfinie (par exemple : `"Mauvais état"`, `"Bon état"`, `"Très bon état"`, `"Neuf"`). Pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples d'images de `BD` associées avec leur catégorie d'état. La tâche d'annotation peut alors consister à renseigner pour chaque couverture de bande dessinée la catégorie d'état qui lui correspond le plus.

Un exemple d'annotation de la classification de l'image est disponible dans la FIGURE 2.1.

4. Pour plus de détails sur la classification : voir les revues de AIZED AMIN SOOFI et ARSHAD AWAN, 2017 ou de KOTSIANTIS et al., 2006 ; voir un exemple basé sur les machine à vecteurs de support (`SVM`) dans CORTES et VAPNIK, 1995.

Exemples :

FIGURE 2.1 – *Exemple d'annotation de l'état d'une BD (ici : MORRIS et GOSCINNY, 1950 et MORRIS et GOSCINNY, 1952). La première est en très bon état (couverture comme neuve, tranches légèrement usées, pages intactes) tandis que la seconde est en mauvais état (couverture usée, dos abîmé, traces sur les pages, ...).*

Ainsi, si quelqu'un s'interroge sur l'état d'une bande dessinée en sa possession, ce modèle peut identifier l'état le plus probable d'après les exemples disponibles dans sa base d'apprentissage.

Pour information : De manière équivalente, il est possible de faire de la classification sur d'autres données, comme par exemple la classification de textes pour identifier la langue de l'ouvrage. Dans l'exemple ci-dessous, les catégories proposées sont `"Français"`, `"Anglais"` et `"Allemand"`, et la tâche d'annotation consiste ici à associer à chaque texte une catégorie de langue.

« *Les cousins Dalton ont dévalisé la diligence.* » ⟹ `Français`
« *The Dalton cousins robbed the stagecoach.* » ⟹ `Anglais`
« *Die Dalton-Cousins haben die Postkutsche ausgeraubt.* » ⟹ `Allemand`

2.1.2.c Identification d'une bande dessinée à partir de sa couverture.

Identifier une bande dessinée n'est pas toujours facile, et recopier l'ensemble des informations l'identifiant peut prendre du temps. Les libraires ou les collectionneurs désirant faire l'inventaire des ouvrages en leur possession peuvent ainsi y passer de nombreuses heures, avec le risque de faire des erreurs lors de l'inscription des bandes dessinées dans leur registre.

Afin d'aider les collectionneurs, il est possible d'utiliser un modèle de `reconnaissance optique des caractères` (`OCR`)[5] pour extraire automatiquement les informations importantes présentes sur les couvertures d'une `BD` à identifier. Pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples de pages de couverture avec la position et la valeur des informations pertinentes à extraire. La tâche d'annotation peut alors consister à renseigner pour chaque couverture de bande dessinée :

— la position des informations en l'encadrant sur l'image (avec un rectangle par exemple) ;

— la valeur écrite dans l'encadré sur l'image.

Un exemple d'annotation de textes dans une image est disponible dans la Figure 2.2.

Exemples :

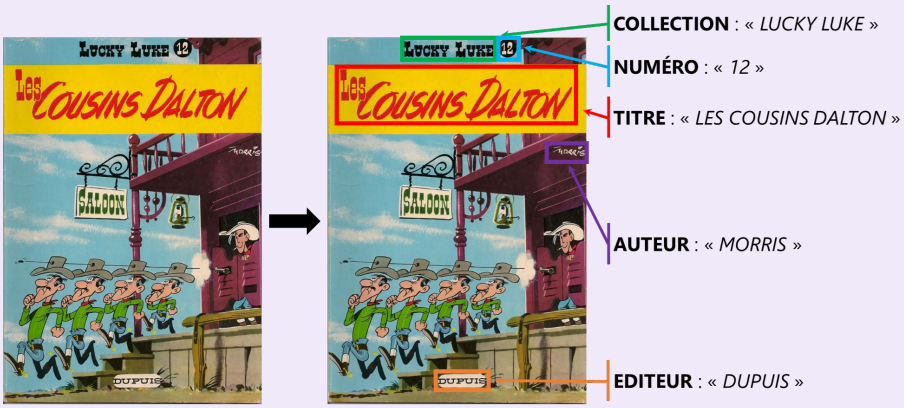

Figure 2.2 – *Exemple d'annotation de textes présents sur la couverture d'une bande dessinée (ici : Morris et Goscinny, 1958). Les informations essentielles telles que la collection, le numéro, le titre, l'auteur et l'éditeur y sont présentes.*

Ainsi, si quelqu'un veut identifier une nouvelle bande dessinée, il peut interroger le modèle d'extraction de caractères pour récupérer les informations textuelles présentes dans la couverture, à l'image des exemples disponibles dans sa base d'apprentissage.

Pour information : De manière équivalente, il est possible de réaliser une `reconnaissance d'entités nommées (NER)`[6] pour extraire les informations citées dans un texte. Dans l'exemple ci-dessous, les types d'entités présentes sont `"personnage"`, `"métier"`, `"argent"`, `"lieu"` et `"date"`. La tâche d'annotation consiste ici à identifier la position et le type de chaque entité présente.

> « ***Lucky Luke***$_{(personnage)}$, le ***cow-boy***$_{(métier)}$ solitaire, a attrapé les ***Dalton***$_{(personnage)}$ à ***Coyote Gulch***$_{(lieu)}$ et a touché ***50.000$***$_{(argent)}$ en les livrant au ***pénitencier***$_{(lieu)}$. Ils se sont évadés le ***jeudi suivant***$_{(date)}$. »

5. Pour plus de détails sur l'`OCR` : voir la revue de Berchmans et Kumar, 2014 ou de Awel et Abidi, 2019.

2.1.2.d Interprétation audio d'une bande dessinée.

Il est de plus en plus commun de trouver des livres disponibles avec une lecture audio. Ces livres audio, réalisés par une personne ou synthétisés par l'ordinateur, peuvent être à visée éducative ou simplement disponibles pour le loisir. Dans le cadre de notre exemple sur le thème des bandes dessinées, peu d'entre elles disposent d'une lecture audio. Une idée serait donc d'interpréter une lecture audio de ces bandes dessinées en synthétisant la voix des doubleurs de leurs adaptations télévisées (ou simplement d'un narrateur si l'oeuvre n'a pas été portée à l'écran).

Dans le but de créer ces `BD` audio, nous pourrions envisager d'utiliser des modèles de `synthèse vocale` (`TTS`) [7] pour générer automatiquement la lecture des bulles d'une bande dessinée. Pour entraîner de tels modèles, il est nécessaire d'avoir une base d'apprentissage contenant des exemples audio pour chacun des personnages (ici : les textes prononcés par les doubleurs de l'adaptation télévisée) avec la transcription de leurs paroles pour chaque audio. La tâche d'annotation peut alors consister à renseigner le personnage et les paroles qu'il a prononcées.

Un exemple d'annotation phonétique est illustré dans la Figure 2.3.

Exemples :

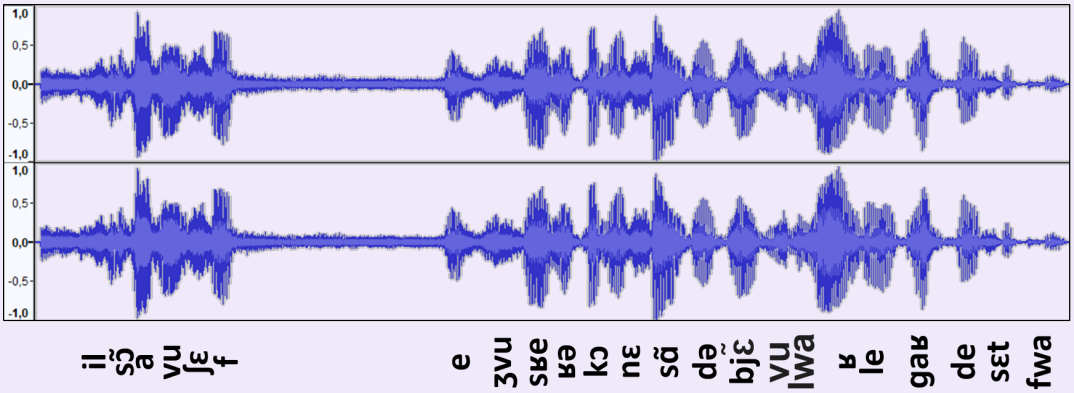

Figure 2.3 – *Exemple de paroles prononcées dans un enregistrement audio. Ici, la voix de Lucky Luke est interprétée par Jacques THEBAULT. Le texte annoté, c'est-à-dire celui prononcé dans l'audio, est « `Ils sont à vous chef, et j'vous s'rai reconnaissant de bien les garder cette fois.` ». Les phonèmes associés à chaque séquence de l'enregistrement sont disponibles en alphabet phonétique international.*

Ainsi, nous pourrions consulter le modèle de synthèse vocale d'un personnage (ici : celui de `Lucky Luke`) avec un nouveau texte à prononcer pour en obtenir une lecture audio dont la voix se rapproche des enregistrements de la base d'apprentissage (ici : celle de Jacques THEBAULT).

6. Pour plus de détails sur la reconnaissance d'entités nommées (`NER`) : voir les revues de Goyal et al., 2018 ou de Li et al., 2022.

7. Pour plus de détails sur la synthèse vocale : voir la revue de Kothadiya et al., 2020 ; voir un exemple d'architecture neuronale `end-to-end` dans Mu et al., 2021.

Pour information : Nous pourrions compléter le cas d'usage (1) en extrayant automatiquement le texte d'une planche de `BD` par `OCR`, (2) en détectant automatiquement le personnage prononçant la bulle de `BD` par classification, puis (3) en générant du texte à prononcer par le personnage par synthèse vocale. Bien entendu, la conception et l'enchaînement de ces différents modèles sont plutôt complexes, et chaque tâche de *Machine Learning* demande ses propres données annotées pour construire une base d'apprentissage.

2.1.3 Bilan concernant la présentation de l'annotation

Points à retenir :

- ☑ « `Annoter` » une donnée consiste à **ajouter un complément d'information** pour pouvoir mieux interpréter puis reproduire un phénomène.
- ☑ Le type d'annotation à réaliser **dépend du problème à traiter** : régression, classification, extraction d'information, génération ou synthèse de données, ...
- ☑ L'ensemble des données annotées peut être utilisé pour concevoir un modèle d'« `apprentissage automatique` » : il est alors appelé « `corpus d'entraînement` ».

2.2 Organisation usuelle d'un projet d'annotation

Dans la section précédente, nous avons présenté l'importance d'avoir des données annotées pour entraîner un modèle de *Machine Learning*. Nous allons maintenant détailler l'organisation de cette tâche d'annotation, identifier les compétences nécessaires aux intervenants du projet ainsi que les fonctionnalités essentielles des outils de labellisation.

2.2.1 Étapes clés du cycle d'annotation

L'organisation d'un projet d'annotation, établie aujourd'hui comme une référence, est proposée par Pustejovsky et Stubbs, 2012 et complétée dans Stubbs, 2013. Les auteurs y formalisent la conception et l'amélioration **cyclique** d'un modèle de *Machine Learning*. Ce cycle est appelé cycle `MATTER` en référence aux six étapes de conception qui le composent : ***M**odelize*, ***A**nnotate*, ***T**rain*, ***T**est*, ***E**valuate* et ***R**evise*. Ces étapes sont schématisées en Figure 2.4 et nous détaillons chacune d'entre elles ci-dessous.

Notes de l'auteur : Nous conseillons la lecture de Finlayson et Erjavec, 2016 pour son excellente revue de littérature qui détaille pas à pas le cycle `MATTER` tout en dressant la liste des points importants de chacune des étapes.

2.2.1.a Concevoir la base d'apprentissage (*Modelize*, *Annotate*).

Pour obtenir un modèle de *Machine Learning*, il faut avoir une base d'apprentissage de qualité. Comme nous l'avons dit précédemment, il faut disposer au départ d'un ensemble de données d'exemples qui représente fidèlement les différentes facettes du problème à modéliser

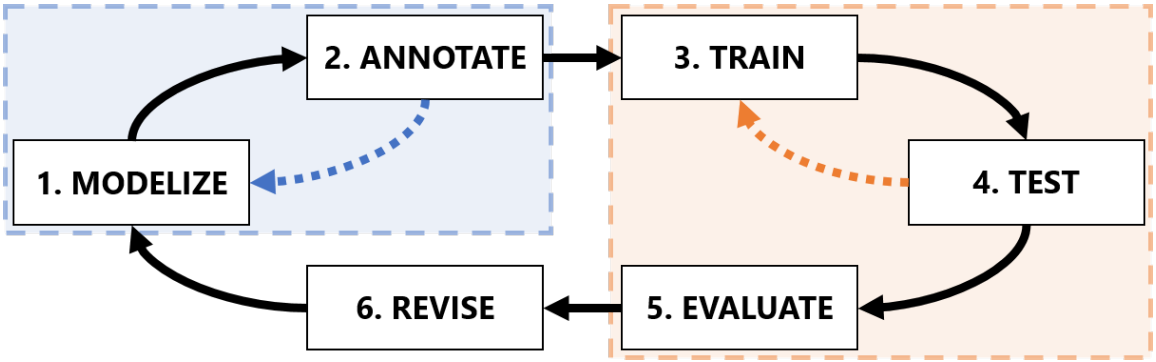

FIGURE 2.4 – *Cycle MATTER structurant un projet d'annotation en six étapes principales : **M**odelize, **A**nnotate, **T**rain, **T**est, **E**valuate et **R**evise.*
*Le carré bleu identifie le mini-cycle MAMA durant lequel la modélisation est adaptée en cours d'annotation, et le carré orange identifie le mini-cycle **T**rain-**T**est lors de la conception du modèle.*

(voir SECTION 2.3.1.A). Une phase de `collecte` de données est alors organisée : cette collecte peut se baser sur des extractions de bases de données ou de sites internet à disposition, sur des enquêtes réalisées auprès d'utilisateurs finaux, ou encore sur les avis éclairés d'experts du problème. Certaines données peuvent aussi être artificiellement créées afin de compléter la collecte pour les aspects du problème étant difficiles à observer. Une fois la collecte terminée, ces données brutes ont besoin d'être annotées pour pouvoir être exploitées.

Afin de garantir la qualité de cette labellisation, **il est important de ne pas précipiter la tâche d'annotation**. En effet, l'objectif de cette tâche peut considérablement changer en fonction du phénomène à décrire, des données à disposition et de la finalité du modèle de *Machine Learning* à entraîner. Il est donc fortement conseillé de bien **modéliser le problème**, c'est-à-dire de définir l'objectif de l'annotation, de clarifier en amont les modalités et les attendus de cette tâche, et de préciser les règles que devront suivre les opérateurs.

PUSTEJOVSKY et STUBBS, 2012 précisent notamment deux concepts importants de cette phase :

— la **modélisation** du problème, représentant de manière abstraite l'objectif à atteindre et décrivant ainsi la logique générale de l'annotation dans un *schéma d'annotation* ;

— les **spécifications**, compilant dans un *guide d'annotation* l'ensemble des règles concrètes à respecter pour mettre en application la modélisation.

Pour résumer cette distinction, la modélisation représente *quoi* annoter (*objectif, définition, valeurs possibles, ...*) alors que les spécifications décrivent *comment* annoter (*règles d'attribution, exemples et contre-exemples, règles de format, ...*).

> **Exemples :** PERROTIN et al., 2018, s'intéressant à la classification des conversations d'assistance en ligne en actes de dialogues, décrivent leur guide d'annotation dans ASHER et al., 2017. Nous y retrouvons (1) la modélisation avec la présentation des étiquettes possibles à annoter, et (2) les spécifications avec les définitions concrètes, des exemples, des restrictions d'attribution, et la gestion des données non pertinentes.
> Dans nos exemples précédents (cf. SECTION 2.1.2.B), nous avions modélisé le problème

de classification de l'état d'une bande dessinée en quatre classes : `"Mauvais état"`, `"Bon état"`, `"Très bon état"`, `"Neuf"`. Il faudrait désormais rédiger les spécifications avec des définitions concrètes et quelques exemples pour guider un annotateur, notamment pour l'aider à distinguer `"Bon état"` de `"Très bon état"`.

Bien entendu, il n'est pas toujours facile de modéliser un problème ni de rédiger un guide d'annotation adéquat. Nous reviendrons plus tard sur les caractéristiques de cette tâche pouvant introduire de la complexité (voir Section 2.3), mais il est important de souligner d'emblée les points élémentaires suivants :

— le besoin d'*interopérabilité* et de *réutilisabilité* : un projet d'annotation est toujours un investissement coûteux, il serait donc regrettable de perdre ou de ne pas pourvoir réutiliser ces données après ce projet. Par conséquent, il faut réfléchir au format des données ainsi qu'aux types de détails à fournir pour être sûr de pouvoir toujours exploiter les données si la modélisation évolue légèrement ou si un futur projet désire en bénéficier ;

— la balance entre *généralité* et *spécificité* : le niveau de détail requis dépend sans conteste du problème à modéliser : annoter trop peu de détails ne permet pas d'exploiter les données, mais en annoter trop peut rapidement complexifier la tâche et introduire des erreurs. Il faut donc trouver le juste milieu pour réaliser un travail de qualité.

Exemples : Dans la classification de langues exposée en Section 2.1.2.b, nous avons annoté chaque texte grâce à trois classes : `"Français"`, `"Anglais"` et `"Allemand"`.

— par souci d'*interopérabilité*, nous pourrions plutôt utiliser la norme `ISO 639-3` (International Organization for Standardization, 2007), soit les codes `"fra"`, `"eng"` et `"deu"`, afin de standardiser l'annotation et ainsi pouvoir partager plus facilement les données labellisées avec d'autres projets ;

— afin de présenter un cas simple, nous avions proposé un modèle avec trois langues communes pour une bande dessinée d'origine belge. Toutefois, nous aurions pu *spécialiser* davantage notre modèle en fonction des variations régionales en prenant en compte le Corse (`"cos"`) ou le Wallon (`"wln"`). Cette distinction peut être essentielle pour certaines sagas publiées dans ces langues (comme `Astérix & Obélix`), mais peut simplement être une source de confusion pour les autres (comme `Lucky Luke` qui n'est pas traduit en Corse).

Pour information : Pour aider à concevoir le guide d'annotation et afin de se poser les bonnes questions, Dipper et al., 2004 dressent une liste de définitions et de recommandations à prendre en considération. Bien que ces conseils soient issus du traitement de données linguistiques, ils permettent d'identifier les sections importantes d'un guide d'annotation en fonction des attentes des différents acteurs de l'annotation (*l'auteur, l'annotateur, l'explorateur de données, ...*) et de les rédiger en suivant certaines règles simples (*introduire les objectifs, ordonner les règles par complexité, traiter en premier les cas par défaut, trier les valeurs des variables catégorielles par ordre alphabétique, ...*). Des exemples reconnus pour leur bonne conception y sont notamment cités.

Lorsque le guide d'annotation est rédigé, la **phase de labellisation** peut commencer. Cette tâche est traditionnellement réalisée par un groupe d'experts choisis en fonction de leurs connaissances sur le problème à caractériser (*dans nos exemples sur les bandes dessinées, ce serait plutôt des libraires ou des collectionneurs*). Après leur avoir expliqué l'objectif de leur travail et partagé les règles de labellisation contenues dans le guide, les annotateurs se partagent les données et réalisent chacun une partie du corpus d'apprentissage.

> **Pour information :** C'est généralement à ce stade que la théorie rencontre la pratique : certaines règles d'annotation peuvent difficilement être applicables, certaines données peuvent être ambiguës ou hors-sujet, deux annotateurs peuvent aussi avoir des avis différents sur l'annotation la plus adéquate, ... Il est aussi important de rappeler que l'annotation est un acte d'interprétation, et que les données sont donc labellisées par un humain dont l'avis n'est pas infaillible. Pustejovsky et Stubbs, 2012 introduisent donc le premier sous-cycle `MAMA` en référence à la boucle entre ***M**odelize* et ***A**nnotate* qui peut avoir lieu tant que le guide d'annotation n'est pas adapté aux données manipulées ou que différents points de vue opposent les annotateurs.
>
> Par exemple, lors de l'annotation de la transcription audio en Section 2.1.2.d, il peut y avoir une voix principale accompagnée de plusieurs voix en arrière plan : une première adaptation du guide serait de clarifier si ces voix secondaires doivent être transcrites ou ignorées, voire si l'audio entier doit être considéré comme inexploitable. La réponse à cette question dépend bien entendu du phénomène à décrire et de l'objectif du modèle de *Machine Learning* à entraîner : dans notre cas, nous pourrions probablement annoter uniquement la voix principale et ignorer l'audio si le bruit gène la compréhension.

À la fin de l'annotation (ou du cycle `MAMA`), le corpus d'entraînement est disponible pour concevoir un modèle de *Machine Learning*.

2.2.1.b Concevoir le modèle (***Train***, ***Test***, ***Evaluate***).

La phase d'entraînement du modèle est l'étape centrale de l'apprentissage automatique. Toutefois, comme l'apprentissage se base sur des méthodes statistiques, il est important d'introduire une phase de test et d'évaluation pour s'assurer des performances du modèle obtenu. Il est donc courant de considérer une boucle de raffinement du modèle tant que les performances n'ont pas atteint un seuil acceptable.

En pratique, il est d'usage de **créer trois jeux de données** à partir de la base d'apprentissage qui vient d'être annotée :

— le jeu d'`entraînement` : c'est sur cette partie des données que le modèle de *Machine Learning* est conçu ;

— le jeu de `développement` (ou de validation) : le modèle entraîné est évalué sur ce jeu de données pour étudier son comportement, identifier ses forces et ses faiblesses, et ainsi permettre de le comparer à d'autres modèles entraînés pour cette même tâche ;

— le jeu de `test` : le modèle retenu est évalué sur ce jeu de test pour déterminer ses performances réelles.

Ainsi, le modèle représente la connaissance présente dans le jeu `entraînement`, il est étudié puis affiné grâce au jeu de `développement`, et est finalement évalué en fonction de ses performances sur le jeu de `test`. Il est encore une fois difficile d'être exhaustif sur les analyses et les

métriques à considérer, car elles dépendent fortement du type de problème que le modèle tente de résoudre. Une métrique basique est l'`accuracy` (ou taux de bonnes prédictions), décrivant simplement le nombre de fois où le modèle a fait une bonne proposition sur l'ensemble du test. Suivant le problème et le type de données, d'autres métriques usuelles peuvent être utilisées comme le `MSE` (*Mean Squared Error*) pour la prédiction de variables numériques (voir Wallach et Goffinet, 1987), le `F1-score` pour les variables catégorielles (voir Sasaki, 2007) ou le `WER` (*Word Error Rate*) pour la transcription de textes (voir McCowan et al., 2005). Dans tous les cas, une règle d'or est de ne pas utiliser le jeu de test lors de la phase de développement pour éviter tout biais de surapprentissage [8].

À la fin de ce cycle, le modèle de *Machine Learning* est à disposition, et ses performances théoriques sont celles obtenues avec le jeu de test.

2.2.1.c Revoir la base d'apprentissage (*Revise*).

Pour terminer cette boucle, il est parfois nécessaire d'envisager de corriger son modèle en remettant en cause la modélisation du problème et l'annotation des données. Voormann et Gut, 2008 formalisaient en effet ce besoin de réviser la conception d'une base d'apprentissage en observant les lacunes du modèle obtenu, et Pustejovsky et Stubbs, 2012 évoquent certaines révisions nécessaires de la modélisation dès la phase d'annotation (voir sous-cycle `MAMA` dans la Figure 2.4).

Diverses pistes peuvent mener à une évolution de la base d'apprentissage :

— le modèle de *Machine Learning* peut avoir de mauvaises performances, malgré son affinage lors de la phase de développement, ou peut manquer d'adaptabilité sur des données réelles ;

— la modélisation ou l'annotation peuvent devenir obsolètes car le phénomène modélisé évolue dans le temps ;

— un cas d'usage non identifié jusqu'à présent nécessite de nouvelles données pour être pris en compte ;

— le modèle peut ne pas convenir aux utilisateurs finaux par manque d'ergonomie ou à cause d'une utilisation non prévue initialement ;

— un nouvel algorithme de *Machine Learning a priori* plus performant peut requérir une modélisation différente pour traiter le problème.

> **Exemples :** Pour illustrer nos propos, prenons la tâche d'estimation du prix d'une bande dessinée (cf. Section 2.1.2.a) : il se peut que les prix annotés sur les transactions ne soient plus d'actualité à cause de l'inflation, et que les données doivent être réannotées pour prendre en compte les nouvelles valeurs du marché.
>
> D'autre part, la modélisation en tant que telle peut aussi être impactée : par exemple, dans le cadre de la classification de l'état d'une bande dessinée à partir d'une photo (cf. Section 2.1.2.b), nous pourrions constater à l'usage qu'il manque une catégorie `"Très mauvais état"` nécessaire pour trier d'emblée toute `BD` indigne à la vente.
>
> Enfin, il est possible que le modèle se comporte mal sur certaines données. Par exemple lors de l'identification d'une bande dessinée à partir de sa couverture (cf. Section 2.1.2.c), certains textes du décor pourraient être extraits à tort (comme le texte de la pancarte

8. Pour plus de détails sur le surapprentissage : voir Collins, 2017

> « *Saloon* » dans la FIGURE 2.2). Il faudra peut-être adapter l'annotation pour identifier les textes à ne pas extraire (avec une classe de rebus par exemple).

Nous bouclons ainsi le cycle `MATTER` qui préfigure le besoin d'une amélioration continue d'un modèle de *Machine Learning* pour que celui-ci soit le plus adapté à son environnement d'utilisation.

2.2.2 Portraits des acteurs intervenant sur un projet d'annotation

Au cours du cycle `MATTER`, nous pouvons constater que divers acteurs interviennent pour concevoir la base d'apprentissage et entraîner un modèle de *Machine Learning*. Cette diversité de métiers qui gravitent autour du traitement automatique des données semble difficile à détailler, tant à cause de leur grand nombre que de leurs subtiles différences. Pour avoir un aperçu, vous pouvez consulter les offres d'emplois du marché actuel (voir TEAM DATASCIENTEST, 2022 ou DATABIRD, 2023) ou certaines formations professionnelles (voir ISOZ, 2017) pour pouvoir faire la distinction entre *data scientist*, *data analyst*, *data librarian*, *data journalist*, *data architect*, *data engineer*, *data steward*, *data archivist*, ou encore *machine learning engineer*...

Afin d'avoir une approche moins commerciale de ces métiers, nous proposons plutôt de dresser les compétences requises aux diverses phases du cycle, à l'image de RADOVILSKY et al., 2018 qui présente les acteurs de la science des données grâce à quatre groupes de compétences :

1. les compétences **métiers** : elles sont liées aux connaissances et à l'expertise sur le phénomène à modéliser ou le problème à résoudre. C'est grâce à ces compétences qu'un acteur peut être apte à annoter une donnée ou à qualifier la pertinence de la prédiction d'un modèle de *Machine Learning*. Les métiers associés comportement notamment l'`expert métier` (*business expert*) ;

2. les compétences **analytiques** : elles concernent entre autres la modélisation du problème, la gestion des données, et les analyses statistiques sur les biais et les performances. C'est grâce à ces compétences qu'un acteur peut concevoir le guide d'annotation, estimer le taux d'accords inter-annotateurs, ou encore réaliser l'évaluation statistique d'un modèle de *Machine Learning*. Les métiers associés comportement notamment l'`analyste des données` (*data analyst*) ou le `scientifique des données` (*data scientist*) ;

3. les compétences **techniques** : elles portent sur l'ingénierie autour du modèle de *Machine Learning*, comme le choix du meilleur algorithme d'entraînement et réglage fin des hyperparamètres, l'archivage des différentes versions du modèle ainsi que son déploiement dans un environnement de production. Les métiers associés comportement notamment le `scientifique des données` (*data scientist*), l'`ingénieur en apprentissage automatique` (*machine learning engineer*) ou l'`architecte des données` (*data architect*) ;

4. les compétences en **gestion** ou en **communication** : elles permettent d'aborder le cadrage du projet et la définition des objectifs, ainsi que diverses aptitudes transverses comme l'établissement de rapports, la gestion de projet, la vérification des normes, ... Les métiers associés comportement notamment le `chef de projet` (*project leader*) ou le `responsable de la protection des données` (*data protection officer*).

Pour information : Nous pouvons compléter cette vision par compétences avec la vision donnée par Fort, 2017, selon laquelle il y a trois types d'experts lors d'un projet d'annotation :

- les **experts du corpus** de données, ayant par exemple les connaissances sur les bandes dessinées, s'approchant donc des compétences `métiers` ;
- les **experts de l'annotation**, ayant par exemple les connaissances sur l'annotation de textes dans une image, s'approchant donc des compétences `analytiques` ; et
- les **experts de la tâche** de *Machine Learning*, ayant par exemple les connaissances sur les techniques d'`OCR`, s'approchant donc des compétences `techniques`.

Ainsi, durant le cycle `MATTER`, nous pouvons voir les compétences ci-dessus se compléter :

1. la conception de la **base d'apprentissage** (étapes ***M**odelize* et ***A**nnotate*) nécessite :
   - des compétences de `gestion` pour cadrer l'objectif du modèle à entraîner, et ainsi définir l'objectif auquel doit répondre l'annotation de données ;
   - des compétences `analytiques` pour proposer une modélisation stable du phénomène et un guide d'annotation précis pour limiter les biais de conception ;
   - des compétences `métiers` pour vérifier que la proposition de modélisation est pertinente vis-à-vis du cas d'usage, mais aussi pour réaliser l'annotation des données.
2. la conception du **modèle de *Machine Learning*** (étapes ***T**rain*, ***T**est*, ***E**valuate*) nécessite :
   - des compétences `analytiques` pour gérer les jeux de données (*entraînement, développement, test*) et évaluer les performances statistiques du modèle ;
   - des compétences `techniques` pour manipuler l'écosystème de développement du modèle, régler les hyperparamètres, versionner les changements, et planifier la distribution du modèle sur un environnement de production ;
   - des compétences de `gestion` pour s'assurer du respect des normes et du caractère privé ou confidentiel de certaines données.
3. la **révision** de la base d'apprentissage (étape ***R**evise*) nécessite :
   - des compétences `métiers` pour identifier le manque de pertinence du modèle vis-à-vis de certains cas d'usage ;
   - des compétences `analytiques` pour disserter des performances réelles du modèle face à des données de production et remettre en question les précédents choix de modélisation pour espérer améliorer le modèle.

Nous noterons que les compétences transverses de **gestion** ou **communication** ne sont pas spécifiques à une étape du cycle `MATTER` (*le cadrage, la gestion de projet et l'établissement de rapports étant réalisés tout au long du projet*), alors que les compétences `métier` et `techniques` n'interviennent généralement pas au même moment du cycle : autrement dit, des experts métiers

croisent rarement des experts techniques et ne partagent donc que très rarement leurs connaissances.

2.2.3 Choix du logiciel d'annotation

Pour terminer la description de l'organisation d'un projet d'annotation, attardons nous sur le choix du logiciel à utiliser pour labelliser les données. S'il est vrai qu'une diversité d'applications existe pour répondre aux besoins des annotateurs, il est important de noter que l'absence de certaines fonctionnalités essentielles peuvent gêner le projet d'annotation.

Nous faisons référence à Finlayson et Erjavec, 2016 pour dresser ci-dessous une liste des fonctionnalités principales (voire essentielles) d'un logiciel d'annotation. Pour simplifier la lecture, nous proposons de regrouper ces fonctionnalités dans les catégories suivantes :

— répondre au **besoin d'annotation** : cette fonctionnalité est bien entendu obligatoire, car un logiciel ne permettant pas d'annoter les données ne sera d'aucune utilité. Cette remarque semble être une évidence, mais nous nous permettons aussi d'étendre l'avertissement aux logiciels n'étant pas destinés à l'annotation mais qui peuvent être détournés pour y répondre indirectement : de tels contournements peuvent introduire des biais et offrent généralement une expérience utilisateur assez médiocre ;

— intégrer le **guide d'annotation** : ce livrable issu de la phase de modélisation du cycle `MATTER` doit être facilement accessible aux annotateurs car il contient la documentation et les instructions à appliquer lors de la labellisation. Les logiciels permettant d'intégrer directement ces définitions (*avec exemples et contre-exemples*) ainsi que les règles d'annotation (*comme les labels mutuellement exclusifs, les détails obligatoires, ...*) ont donc un net avantage ergonomique pour respecter la modélisation définie et ainsi garantir la qualité de la base d'apprentissage ;

— autoriser l'**annotation multiple** et l'**annotation multimodale** : il est fréquent de devoir annoter une même donnée suivant des modélisations ou des paradigmes différents pour répondre à plusieurs cas d'usage (*en prenant l'exemple de l'annotation d'images, nous pouvons détourer les objets présents, identifier les textes inscrits, proposer une ou plusieurs catégories générales, proposer une description textuelle, ...*) ou encore de devoir annoter des données de différents types (*en combinant texte, image et voix comme dans l'annotation des sous-titres d'une vidéo*). Ainsi, les logiciels offrant la possibilité de labelliser plusieurs informations et de manipuler plusieurs types de données sont pertinents pour centraliser les annotations et de pouvoir facilement les réutiliser ;

— évaluer la **qualité de l'annotation** : les erreurs d'annotation et les divergences d'opinions sur la modélisation sont inévitables. Il est donc appréciable de pouvoir les identifier, soit sur la base d'une comparaison directe entre deux annotateurs, soit en comparant avec l'annotation la plus probable issue d'une base de référence. Il peut aussi être intéressant de pouvoir calculer les scores d'accord inter-annotateurs sur un même échantillon de données pour estimer la qualité de la base d'apprentissage, de pouvoir corriger les erreurs lors de revues d'annotation ou encore de trancher les cas de conflits apparents lors d'avis discordants ;

— permettre l'**interopérabilité** technique : il peut être frustrant de ne pas pouvoir réutiliser des annotations d'un projet à l'autre car le format de stockage n'est pas compatible. Par conséquent, les logiciels prenant en considération plusieurs formats de données (*`PNG/JPG`, `MP3/WAV`, `XLSX/XML/JSON`, ...*) et respectant les standards de la tâche d'annotation lors des

imports et exports sont à privilégier. De plus, il est conseillé de ne pas écrire directement les annotations dans les données (« ***[Lucky Luke]***`/(personnage)`*, le* ***[cow-boy]***`/(métier)` *solitaire, a ...* »), mais de les stocker dans des fichiers séparés pour garder une meilleure gestion et permettre les annotations multiples ;

— gérer le **flux de travail** et le **suivi de projet** : certaines fonctionnalités simples sont nécessaires à l'organisation de l'équipe d'annotation. Cela peut comprendre la répartition de la charge de travail entre plusieurs utilisateurs, l'historisation des changements pour permettre les retours arrières, la possibilité d'émettre des appels d'aide ou d'écrire des commentaires sur les choix d'annotation, ou encore l'accompagnement des nouveaux annotateurs lors de leur montée en compétence ;

— favoriser le **confort de l'annotateur** : le logiciel choisi sera utilisé au quotidien par l'équipe d'annotation, il semble donc essentiel d'offrir une expérience utilisateur agréable pour réaliser cette tâche. Cela peut passer par une customisation de l'interface afin d'être adaptée à l'objectif d'annotation et par le paramétrage de raccourcis claviers. Simplifier l'accès et l'installation du logiciel peut aussi s'avérer utile pour favoriser son adoption, en favorisant par exemple les applications web permettant plus facilement le travail collaboratif ;

— permettre des **annotations** et des **analyses avancées**. : la littérature scientifique regorge de techniques permettant d'assister un annotateur dans son travail (*pré-annotation, apprentissage actif, visualisation, interaction, ...*). Nous détaillons plusieurs de ces techniques dans la Section 2.3.

Considérant la diversité de cas d'usage d'annotation, une liste exhaustive des outils de labellisation n'est bien entendu pas possible. Nous tenons toutefois à présenter quelques exemples illustrés dans la Figure 2.5.

> **Notes de l'auteur :** Par expérience, nous constatons malheureusement que peu de projets industriels utilisent un outil d'annotation dédié, souvent au profit d'outils rudimentaires comme des traitements de texte ou des tableurs tels que `Microsoft Excel` (Microsoft Corporation, 2018). Une étude serait à mener pour étudier cette tendance et expliquer le manque d'intérêt porté aux outils spécialement conçus pour les tâches d'annotations. Peut-être que ces outils s'adaptent mal aux particularités des divers projets industriels, expliquant ainsi l'utilisation d'outils simplistes mais flexibles. Ou alors est-ce par méconnaissance des difficultés et des bais potentiels de l'annotation que ces outils aux fonctionnalités avancées ne sont pas employés ?

2.2.4 Bilan concernant l'organisation d'un projet d'annotation

> **Points à retenir :**
>
> ☑ En général, un projet d'annotation est **organisé en cycle** (`MATTER`) au cours duquel nous réalisons une modélisation abstraite des données que nous formalisons dans un guide (***M****odelize*) ; nous appliquons ce guide pour labelliser notre base d'apprentissage (***A****nnotate*) ; puis nous entraînons et testons un modèle de *Machine Learning* (***T****rain*,

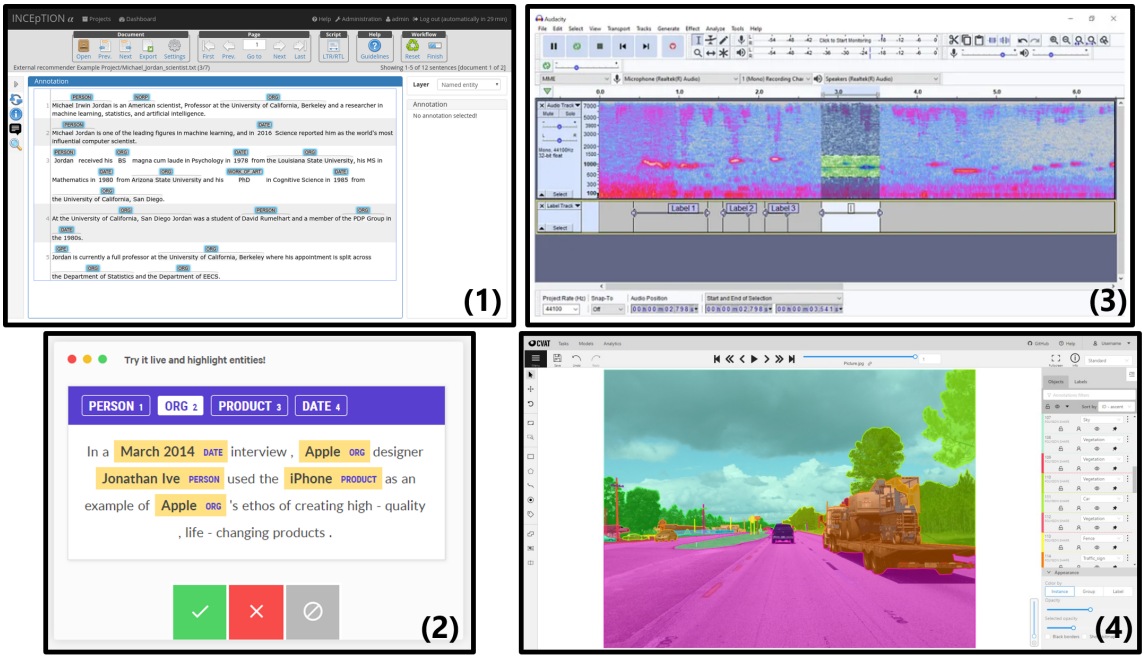

FIGURE 2.5 – *Quatre exemples d'outils d'annotation : (1) `INCEpTION` pour le texte (KLIE et al., 2018), (2) `Prodigy` pour le texte ou l'image (MONTANI et HONNIBAL, 2017), (3) `Audacity` pour l'audio (AUDACITY TEAM, 2000 et (4) `CVAT` pour l'image (CVAT.AI CORPORATION, 2019).*

Test et ***Evaluate***). Ensuite, l'évaluation du modèle peut mener à une révision de la modélisation des données en fonction des performances obtenues (***Revise***) ;

☑ Un tel projet d'annotation nécessite une **diversité de connaissances et de compétences** qui peuvent être réparties en quatre catégories : `métier`, `analytique`, `technique` et `gestion/communication` ;

☑ Un tel projet nécessite aussi un **outil d'annotation dédié** possédant certaines fonctionnalités essentielles comme la possibilité d'`intégrer le guide` d'annotation, de `contrôler la qualité` des annotations, de réaliser des `annotations multiples ou multimodales`, ou encore d'assimiler d'éléments de `gestion de projet`.

2.3 Aperçu des nombreux défis de l'annotation

Comme nous avons pu l'apercevoir dans les sections précédentes, le cycle d'annotation recèle de nombreuses zones d'ombres pouvant introduire des complications dans la conception d'une base d'apprentissage (BALEDENT, 2023). Pour aborder cette partie, nous verrons :

— qu'il y a une forte pression sur la **qualité des données** devant constituer le corpus d'entraînement (cf. SECTION 2.3.1) ;

— que ce standard de qualité entretient une **complexité inhérente** aux étapes de modélisation et d'annotation (cf. SECTION 2.3.2) ; et

— que cette complexité provoque des **différences de comportements** chez les annotateurs

(cf. Section 2.3.3).

En détaillant chacun de ces trois points, nous discuterons d'un ensemble de techniques et bonnes pratiques mises en avant dans la littérature pour limiter les désagréments d'un projet d'annotation. Nous identifierons aussi les freins récurrents pouvant intervenir dans les mises en application industrielles.

2.3.1 Défis concernant le besoin de qualité des données

Comme nous l'avons défini en Section 2.1.1.a, l'« `apprentissage automatique` » regroupe un ensemble de techniques dont l'objectif est de reproduire une tâche **par l'exemple** : il est donc normal de porter une attention particulière aux données utilisées, car la qualité du modèle de *Machine Learning* va fortement dépendre de la qualité de sa base d'apprentissage. Nous allons ici détailler trois défis actuels concernant cette création d'un jeu de données.

2.3.1.a Problèmes de représentativité

La phase de collecte de données est une étape importante du projet d'annotation. Malheureusement, la littérature scientifique associée à cette tâche est assez légère alors que c'est précisément à ce moment que ce joue une caractéristique cruciale de la future base d'apprentissage : la **représentativité** du phénomène à modéliser.

Cette notion est assez ambiguë, notamment car le terme technique « représentatif » fait écho à un mot de la vie courante qui peut avoir plusieurs sens. Dans Kruskal et Mosteller, 1979a et Clemmensen et Kjaersgaard, 2022, plusieurs usages communs de ce terme sont recensés :

— « *assertive claim* » : l'opérateur déclare que ses données sont représentatives du problème sans apporter d'arguments. Bien entendu, cette option est à bannir car elle n'apporte aucune information et peut cacher des vices de conception du modèle ;

— « *absence or presence of selective force* » : la représentativité du phénomène est supposée en sélectionnant des données de **manière aléatoire** et en limitant le nombre de critères de sélection à ceux nécessaires pour l'étude réalisée ;

— « *miniature* » : aussi appelée **sélection stratifiée**, cette approche consiste à dire qu'un échantillon est représentatif d'un phénomène si la proportion de chacune de ses parties y est respectée (*par exemple, un sondage peut respecter la répartition des tranches d'âge d'une population*) ;

— « *typical/ideal case* » : cette définition consiste à représenter chaque partie du phénomène par un **exemple emblématique** ou un **exemple moyen** (*par exemple, nous pouvons illustrer l'univers de la bande dessinée française par un numéro de la saga* ***`Astérix & Obélix`***) ;

— « *population coverage* » : ici, la représentativité est associée à la présence **exhaustive** de l'ensemble des caractéristiques importantes du phénomène avec au moins un exemple par caractéristique (*par exemple, dans l'arche de Noé, il devait y avoir au moins un couple d'animaux de chaque espèce*).

Pour compléter ces définitions, Kruskal et Mosteller, 1979b insistent sur le besoin de **définir avec précision la méthode d'*échantillonnage*** plutôt que l'*échantillon* lui-même : en effet, il est important de caractériser le phénomène à modéliser, de définir l'objectif de la collecte de données et de détailler comment cette collecte va être réalisée. Clemmensen et

Kjaersgaard, 2022 introduisent à leur tour trois mesures pour aider à caractériser une collecte : la *réflexion* (est-ce l'échantillon respecte la distribution de la population ?), la *couverture* (est-ce que l'échantillon illustre la diversité de la population ?) et la *présence de représentants* (est-ce que l'échantillon contient les exemples emblématiques de la population ?). De telles informations sont essentielles pour pouvoir juger la valeur d'un échantillon par rapport à un cas d'usage et déterminer s'il est réutilisable pour une autre application.

> **Exemples :** Illustrons nos propos avec la classification de l'état d'une bande dessinée à partir d'une photo (voir Section 2.1.2.b). Afin de représenter correctement le cas d'usage, nous pourrions collecter des exemples de `BD` couvrant l'ensemble des dégradations fréquemment identifiées par les libraires (couvertures froissées, pages déchirées, couleurs délavées, ...) et les intégrer de manière proportionnelle dans la base d'apprentissage. Nous pourrions aussi nous assurer de la présence de cas emblématiques permettant de catégoriser les `BD` en `"Mauvais état"`, `"Bon état"`, `"Très bon état"`, `"Neuf"`.
>
> Toutefois, cette base d'apprentissage ne serait peut-être plus représentative si nous voulions détecter la langue de la bande dessinée : il conviendrait alors de vérifier la répartition par langue en revoyant les trois mesures précédemment mentionnées.

La description d'un phénomène reste cependant une **tâche difficile**, notamment lorsque que celui-ci possède un ensemble vaste et abstrait de caractéristiques à décrire. Nous comptons généralement sur la loi des grands nombres pour espérer dresser un portrait fidèle du phénomène, mais cela impose parfois de traiter d'**immenses quantités de données**.

> **Exemples :** Considérons le traitement du langage : le vocabulaire employé peut concerner des dizaines de milliers de mots ; il existe des variantes régionales et divers jargons techniques ; certains termes peuvent avoir plusieurs sens et des expressions peuvent dépendre de leur contexte (comme l'humour ou les critiques). Pour représenter ces spécificités (listées de manière non exhaustive), une base d'apprentissage devra contenir de nombreux exemples afin de capturer les différents aspects du langage à traiter. Nous pouvons citer par exemple `MLSUM: The Multilingual Summarization Corpus` (Scialom et al., 2020), une base de 1.5 millions d'articles de journaux en 5 langues pour entraîner un modèle de résumé automatique, ou encore `The Multilingual Amazon Reviews Corpus` (Keung et al., 2020), une base de 1.26 millions de commentaires de produits en 6 langues pour entraîner un modèle de classification de la note d'un commentaire sur 5 étoiles.

Toutefois, la masse de données ne résout pas toujours tous les problèmes de représentativité. Une des difficultés récurrentes concerne les aspects peu fréquents d'un phénomène qui se retrouvent ainsi **sous-représentés** : si l'enjeu du modèle à concevoir consiste justement à détecter ou reproduire ces aspects, il peut être intéressant de volontairement biaiser les proportions du corpus d'entraînement pour mieux les illustrer. À l'inverse, des cas communs ou fréquents peuvent être **sur-représentés** : il est parfois nécessaire de limiter leur occurrence dans le corpus d'apprentissage pour ne pas concevoir un modèle véhiculant des généralités ou des stéréotypes. Dans les deux cas, **toute intervention va introduire un biais** : l'opération doit donc être réfléchie et judicieusement réalisée pour contribuer à la finalité du modèle, d'où l'intérêt de bien la documenter pour faire entendre ce que vous voulez signifier par « échantillon représentatif ».

Exemples : D'une part, considérons le besoin de détecter la langue d'une bande dessinée (voir Section 2.1.2.b). Il est fort probable que la base d'apprentissage contienne peu de données sur les parutions en langues régionales (en Corse, en Wallon, en Alsacien, ...). Nous pouvons donc être amenés à ajouter des exemples supplémentaires pour espérer mieux les détecter et ainsi augmenter la *couverture* de notre jeu de données.

D'autre part, regardons l'analyse du modèle de `Stable Diffusion`, réalisée par Nicoletti et Bass, 2023 sur la génération de portraits de personnes fictives à partir d'une description textuelle de leur métier. L'étude montre que le modèle tend générer des portraits d'hommes à la peau blanche pour le métier d'architecte ou d'ingénieur, des femmes pour le rôle de concierge ou encore des personnes à la peau noire pour illustrer la classe ouvrière (voir la Figure 2.6).

Figure 2.6 – *Répartition des couleurs de peau (1) et des genres (2) par métier lors de génération de portraits avec `Stable Diffusion` (étude menée par Nicoletti et Bass, 2023).*

Ici, ce modèle dépeint les inégalités de notre société en perpétuant des stéréotypes, et cela ouvre la question suivante : veut-on vraiment reproduire à l'identique cette représentation ? (*Pour grossir le trait : si nous voulions générer le scénario d'une nouvelle `BD Lucky Luke` à partir des `BD` déjà existantes, accepte-t-on que tous les personnages d'origine asiatique soient à la blanchisserie, et que par conséquent le prochain le soit aussi ?*)

Idées : Une piste pour équilibrer efficacement les corpus d'entraînement et permettre de corriger leurs biais consiste à utiliser des **données synthétiques** (Jaipuria et al., 2020). Ces données peuvent être créées manuellement ou être générées automatiquement (voir Shorten et al., 2021 pour une revue de génération de textes et Shorten et Khoshgoftaar, 2019 pour la génération d'image). Bien entendu, une telle approche doit rester réfléchie pour ne pas introduire davantage de biais et pour répondre à un objectif précis d'équilibrage du jeu de données.

Une dernière difficulté concerne l'**obsolescence** des données au cours du temps. En effet, peu de phénomènes sont immuables, et il est courant de devoir questionner la représentativité d'un problème pour prendre en compte de nouveaux aspects ou corriger des caractéristiques devenues inexactes.

Exemples : Ce problème est particulièrement impactant si nous voulons estimer le prix d'une bande dessinée (voir Section 2.1.2.a) car les oeuvres vont gagner ou perdre de la valeur avec le temps. Par exemple, en 1949, le premier album de `Lucky Luke` devait s'acheter pour quelques francs, alors qu'aujourd'hui, cette édition est estimée à plusieurs milliers d'euros.

De manière similaire, le traitement du langage constate aussi des évolutions au cours du temps (*nouveaux mots de vocabulaire, nouvelles expressions, influence de langues étrangères, ...*), imposant ainsi la mise à jour des jeux de données.

2.3.1.b Problèmes de bruits

La qualité d'une base d'apprentissage dépend fortement du bruit qu'elle contient. Ce bruit est inévitablement inséré lors de la collecte : d'une part, la méthode de collecte elle-même peut en introduire (*instrument de mesure faillible, erreur humaine, ...*) ; d'autre part, par souci de représentativité, le bruit intrinsèque du phénomène va être capturé (*forte variabilité, présence d'irrégularités, ...*). Un échantillon de données va donc forcément devoir se confronter à des étapes de prétraitements et de curation de données.

Nous nous inspirons de Maharana et al., 2022 et de Alasadi et Bhaya, 2017 pour dresser une liste de problèmes récurrents sur les données suite à une collecte :

— présence de **données non pertinentes** par rapport au cas d'usage : une collecte automatique ou aléatoire peut sélectionner des données n'ayant pas ou peu de rapport avec le phénomène à modéliser. Garder de telles données peut introduire de la confusion dans le modèle à entraîner ;

— dégradation des données par des **variations parasites** : comme décrit en introduction de cette partie, les bruits peuvent être intrinsèques au phénomène ou être introduits par la méthode de collecte. Il convient de lisser ou limiter ces bruits pour ne pas perturber le modèle ;

— **absence de valeurs** descriptives essentielles : cette absence peut venir d'une erreur de mesure, d'une méconnaissance du phénomène à caractériser, ou simplement d'un oubli. Cependant, un trou de description peut rendre inutile une donnée si cela concerne une caractéristique importante du phénomène ;

— présence d'**incohérences** ou d'**ambiguïté** entre les données : les données sont rarement catégoriques et plusieurs interprétations sont parfois possibles (voir la discussion sur la subjectivité en Section 2.3.3.a). Toutefois, des contradictions entre les données peuvent pénaliser le modèle à entraîner.

Exemples : Pour illustrer ces problèmes, considérons la tâche d'estimation du prix d'une bande dessinée (voir Section 2.1.2.a) :

— une donnée concernant le prix d'un roman ou d'une encyclopédie peut avoir été insérée par mégarde et pourrait être considérée comme donnée non pertinente pour ce cas d'usage ;

— un changement de typographie (*majuscules, minuscules, accents, ponctuation*) dans l'écriture d'un titre pourrait mal identifier une bande dessinée ;

— une information peut ne pas avoir été renseignée lors d'une transaction (l'année d'édition par exemple), alors que c'est une caractéristique importante de la prise de décision ;

— une même bande dessinée (avec les mêmes caractéristiques) peut avoir été vendue à deux prix radicalement différents, introduisant ainsi une légère ambiguïté dans les données.

Des problèmes similaires peuvent impacter le traitement du texte (*fautes grammaticales, fautes syntaxiques, erreurs sémantiques, ambiguïtés, omissions, ...*), des images (*flous, mauvais cadrages, colorimétries gênantes, ...*) et de l'audio (*bruits en arrière plan, saturations, coupures inopportunes, mots mâchés, ...*). Quelques exemples sont présentés ci-dessous dans la Figure 2.7.

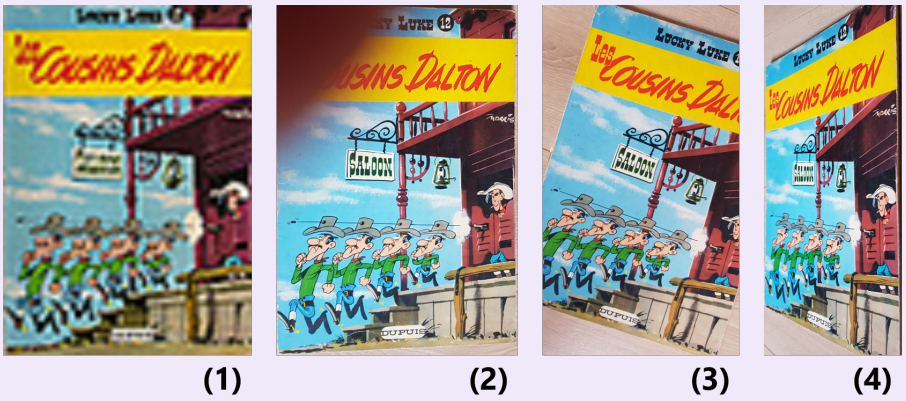

Figure 2.7 – *Exemples de bruits courants perturbant l'analyse d'une image : (1) le flou, (2) un doigt sur le capteur, (3) un problème de cadrage et (4) un problème d'angle de vue.*

Pour limiter l'impact du bruit dans les données, Alasadi et Bhaya, 2017 structurent les étapes de prétraitements de données entre quatre catégories :

— le **nettoyage des données** : cette étape consiste à compléter les données manquantes (*en prenant la valeur moyenne par exemple*), à filtrer les données aberrantes ou inintéressantes,

et surtout à lisser le bruit dans les données en gommant les variations parasites ;

— l'**intégration des données** : dans certains cas, plusieurs sources de données sont disponibles, il peut donc être intéressant de croiser ces sources de données pour augmenter la consistance de la base d'apprentissage et identifier les incohérences ;

— le **formatage des données** : pour exploiter facilement les données, certaines transformations sont parfois nécessaires pour limiter les ambiguïtés dues au leur format (*par exemple : normaliser une valeur entre* 0 *et* 1*, limiter les caractères spéciaux dans un texte*) ;

— la **réduction des données** : en réalisant une analyse approfondie des données, nous pouvons quelquefois constater que certaines caractéristiques présentes sur les données sont peu utiles et peuvent être supprimées pour réduire la complexité de la base d'apprentissage.

Baledent, 2023 rappelle néanmoins que le document source (ou ici : la donnée brute) doit rester accessible pour la phase d'annotation afin de ne pas manquer une information potentiellement intéressante.

Exemples : Reprenons les problèmes évoqués précédemment sur la tâche d'estimation du prix d'une bande dessinée :

— une donnée inintéressante peut simplement être supprimée ;

— normaliser les champs décrivant une bande dessinée en passant tout en minuscules limiterait les chances de mal l'identifier ;

— une année d'édition manquante pourrait être identifiée par l'étiquette `inconnue` et un prix manquant pourrait être complété par la moyenne du prix des `BD` ayant les mêmes caractéristiques ;

— un prix faussé pourrait être identifié comme incohérent en analysant les prix des `BD` ayant les mêmes caractéristiques ;

2.3.1.c Problèmes d'exploitation et de diffusion

En plus des difficultés techniques sur la réalisation d'une collecte de données, il y a aussi des contraintes législatives et stratégiques à prendre en compte.

D'une part, il faut considérer le fait que certaines données sont protégées et ne peuvent pas être collectées ou exploitées librement. C'est le cas de données soumises aux droits des **propriétés intellectuelles** qui empêchent cet usage : nous pouvons citer par exemple Loignon, 2023 qui évoque une levée de boucliers des médias français contre l'utilisation de leurs articles pour entraîner des modèles de langues, mais aussi le journal Les Echos, 2023 qui questionne la violation des **droits d'auteur** lorsque qu'un modèle est entraîné sur l'oeuvre d'un artiste et qu'il est capable de la reproduire.

Ces limites concernent aussi la Réglementation Générale européenne sur la Protection des Données (General Data Protection Regulation, 2016) restreignant les **collectes et usages non consentis** de données personnelles. Ainsi, il n'est pas possible d'entraîner n'importe quel modèle sur n'importe quelles données, et une telle contrainte impose de manipuler les données en garantissant l'anonymat et la confidentialité des personnes consentantes.

Exemples : Considérons la conception d'un modèle de synthèse vocale pour réaliser une `BD` audio (voir Section 2.1.2.d). D'une part, un tel projet nécessiterait déjà de demander les droits d'adaptation pour entraîner un modèle de synthèse vocale avec les doublage de l'adaptation télévisée de `Lucky Luke`. D'autre part, l'utilisation de la voix d'une personne se confrontera probablement à plusieurs restrictions pour éviter que ce modèle ne soit détourné pour des usages non consentis par le doubleur.

Pour aller plus loin, cette notion de confidentialité touche les données personnelles, mais aussi le **caractère stratégique** d'une organisation. En effet, dans le monde académique, les données manipulées sont le plus souvent publiques et peuvent être employées pour contribuer à la recherche scientifique. Mais dans le secteur industriel, les jeux de données sont liés au domaine d'activité de l'entreprise : ils ont généralement requis un investissement conséquent en temps et en moyens, et ils représentent donc son avantage concurrentiel (*par leur spécificité, leur caractère secret ou novateur, leur qualité compétitive, ...*). Il est donc rare de voir une entreprise partager ses jeux de données car elle pourrait perdre un de ses atouts stratégiques.

Une solution est de trouver des jeux de données **accessibles en *Open Source***. Plusieurs plateformes mettent en effet à disposition des données ou des modèles, comme *Hugging Face* (Hugging Face, 2016) ou `Zenodo` (Re3data.Org, 2013). Toutefois, deux limites subsistent à l'utilisation de ces données :

— les données mises à disposition publiquement sont souvent assez générales et ne reflètent pas la spécificité des cas d'usage de l'entreprise, limitant ainsi leur intérêt ;

— les données publiques ne sont pas forcément ouvertes à un usage commercial (*elles peuvent par exemple employer la licence `CC BY-NC 4.0`, Creative commons, 2013*), restreignant ainsi les seules applications aux domaines de la recherche et de la veille scientifique.

Pour ne pas faire de faux pas juridique, Rajbahadur et al., 2022 proposent une approche pour vérifier si une licence permet d'exploiter un jeu de données.

Pour information : Pour terminer, nous mentionnons aussi une proposition de législation européenne concernant la future réglementation des modèles d'intelligence artificielle (The Artificial Intelligence Act, 2021). Cette loi concerne les quatre objectifs suivants :

— « *veiller à ce que les systèmes d'`IA` mis sur le marché de l'Union et utilisés soient sûrs et respectent la législation en vigueur en matière de droits fondamentaux et les valeurs de l'Union* » ;

— « *garantir la sécurité juridique pour faciliter les investissements et l'innovation dans le domaine de l'`IA`* » ;

— « *renforcer la gouvernance et l'application effective de la législation existante en matière de droits fondamentaux et des exigences de sécurité applicables aux systèmes d'`IA`* » ;

— « *faciliter le développement d'un marché unique pour des applications d'`IA` légales, sûres et dignes de confiance, et empêcher la fragmentation du marché* ».

> Un besoin de traçabilité des données et des modèles se fait donc sentir, renforçant les recommandations à documenter et détailler les traitements et choix pour garantir la représentativité et la qualité des données des bases d'apprentissage.

2.3.2 Défis concernant la complexité inhérente à la tâche d'annotation

Selon Baledent, 2023, deux types de complexités sont à différencier :

— la **complexité propre au phénomène** que l'on veut modéliser : nous avons pu apercevoir celle-ci dans la Section 2.3.1.a, notamment en considérant la nécessité de collecter une grande quantité de données pour représenter la diversité du phénomène et le besoin de contrôler les bruits pour en assurer la qualité ; et

— la **complexité propre à la procédure d'annotation** mise en place : cet aspect est abordé brièvement dans la Section 2.2 en exposant le besoin d'établir une modélisation du problème avec son guide d'annotation, de recourir à un processus itératif pour affiner les problèmes de conception et différences d'interprétation (cycles `MATTER` et `MAMA`), ou encore d'employer des opérateurs ayant différentes compétences.

Dans cette section, nous allons voir comment analyser et mesurer cette complexité, puis nous nous intéresserons aux coûts qu'elle peut engendrer lors de la conception du jeu de données.

2.3.2.a Modélisation difficile du phénomène

Comme nous l'avons énoncé lors de la présentation au cours du cycle `MATTER` en Section 2.2.1.a, la modélisation est une étape importante du processus de conception car elle permet de clarifier ce qu'il faut annoter et avec quel(s) objectif(s). Toutefois, plusieurs aspects rendent cette tâche difficile à réaliser.

Tout d'abord, le projet d'annotation concerne généralement un cas d'usage précis dont la spécificité requiert l'intervention d'experts ayant des **connaissances métiers**. Cependant, de tels experts n'ont pas forcément les compétences analytiques requises pour établir une représentation abstraite de leurs connaissances. En effet, diverses questions sont à discuter concernant la modélisation à choisir :

— peut-elle réutiliser un standard existant ?

— sera-t-elle stable et explicite à l'annotation ?

— sera-t-elle traduite en un seul ou en plusieurs modèle(s) de *Machine Learning* ?

— sera-t-elle maintenable avec de grands volumes de données ?

Pour y répondre, des ateliers de conception sont généralement organisés avec les experts, permettant ainsi d'identifier les notions pertinentes à intégrer dans la modélisation finale du problème. Si le phénomène est intrinsèquement complexe, il est possible néanmoins que ces ateliers se réalisent en mode essai-erreur (à l'image du mini-cycle `MAMA`) jusqu'à trouver une modélisation qui puisse convenir, rendant cette tâche particulièrement pénible.

> **Exemples :** Pour se rendre compte de cette difficulté, il suffit d'essayer de détailler l'ensemble des actions que peut entreprendre un robot conversationnel en domotique

(comme `Alexa`, Alexa Internet, 2018), puis de modéliser les diverses instructions verbales pouvant déclencher ces actions. Nous pouvons vite tomber sur des débats d'opinion, notamment autour de la gestion des formulations ambigües ou d'actions différentes provenant d'instructions similaires (*voir l'exemple précédemment utilisé : est-ce que l'énoncé « peux-tu allumer... » demande à l'assistant d'effectuer une action, ou demande-t-il simplement exprimer s'il en est capable ?*)

💡 **Idées :** Quelques approches peuvent être mises en place pour assister la modélisation d'un problème :

- les approches **non supervisées** : elles consistent à laisser la machine extraire et structurer automatiquement la connaissance contenue dans les données collectées. Par exemple pour le traitement du texte, nous pouvons citer les méthodes d'exploration utilisant le regroupement automatique (*clustering*) comme `KMeans` (MacQueen, 1967), ou la modélisation thématique (*topic modeling*) comme `LDA` (Blei et al., 2003) ;
- les approches **semi-supervisées** : elles consistent à travailler main dans la main avec la machine pour explorer les données collectées. Nous pouvons citer par exemple les méthodes d'apprentissage actif (*active learning*) dont une revue est proposée par Settles, 2010 ;
- les approches **itératives** : elles consistent à concevoir d'abord un petit modèle dont le périmètre est limité, puis d'étendre pas à pas son périmètre suite aux retours de son utilisation. Une telle organisation fait bien entendu écho à la philosophie du cycle `MATTER`.

Toutefois, plusieurs handicaps pénalisent ces approches, notamment lors de déséquilibrage ou de bruits dans les données, ou lorsque l'interprétation nécessite une forte connaissance métier.

💬 **Notes de l'auteur :** Par expérience, nous constatons que peu d'approches non supervisées ou semi-supervisées sont utilisées, notamment parce que leurs résultats sont jugés peu pertinents sur des phénomènes complexes. Les experts métiers réalisent alors cette étape de modélisation manuellement, absorbant la complexité de cette tâche en organisant de nombreux ateliers de conception en mode essai-erreur.

C'est particulièrement le cas lors de la conception de la base d'apprentissage d'un assistant conversationnel (*task-oriented*) : la tâche de modélisation consiste généralement à identifier l'intention formulée dans une demande d'un utilisateur en fonction du verbe d'action (« *je veux **réserver** un billet* », « ***joue** moi du jazz !* », « *comment **éditer** une attestation ?* »). Cependant, la vaste diversité du langage permet rarement d'identifier les thématiques présentes dans une collecte de données. L'étape de modélisation est alors réalisée manuellement par les experts du domaine couvert par l'assistant conversationnel.

Lorsqu'une modélisation acceptable du phénomène a été définie, il est nécessaire de la structurer dans le but de **rédiger le guide d'annotation** associé (voir Section 2.2.1.a). Nédellec et al., 2006 ont pu montrer que la clarté de ce guide impacte directement la qualité des annotations : il est donc important de rédiger celui-ci avec soin pour donner à l'annotateur une vision

de son objectif (Fort et al., 2009). Cela requiert toutefois de solides compétences analytiques et pédagogiques, notamment pour définir, organiser et transmettre ce vaste ensemble de règles sans introduire de biais ni de confusion.

> **Rappel :** Dipper et al., 2004 ont dressé une liste de recommandations pour rédiger un guide d'annotation fiable.

Malgré tout, Baledent, 2023 rappelle que l'établissement d'une modélisation d'un phénomène **relève d'un choix**, et que celui-ci est par conséquent subjectif. En effet, plusieurs représentations peuvent convenir à un même cas d'usage, et celle retenue devra être la représentation qui correspondra le mieux à la finalité du modèle. Ce choix, aussi avisé soit-il, entrera inévitablement en friction avec certaines données collectées qui s'insèreront difficilement au sein de la modélisation choisie. Lors de la rédaction du guide d'annotation, il faut donc trouver l'équilibre entre entre la rigueur (*pour fiabiliser la qualité de la labellisation*) et la flexibilité (*pour pouvoir s'adapter aux données contrariantes*).

> **Exemples :** Dans la tâche de transcription d'un audio pour réaliser un modèle de synthèse vocale (Section 2.1.2.d), nous constatons que `Lucky Luke` mâche certains de ses mots (« [...] `j'vous s'rai reconnaissant` [...] ») : préférez-vous annoter strictement ce qu'il dit (*au risque de complexifier le modèle*) ou corriger la transcription (*au risque de ne pas reproduire la prononciation exacte du personnage*) ?

2.3.2.b Estimation de la complexité

Dans Fort et al., 2012, une approche analytique est proposée pour mesurer la complexité des tâches de modélisation et d'annotation. Six dimensions sont détaillées, réparties en trois axes :

— la complexité de **localisation** de l'annotation (*discrimination*, *délimitation*) : y a-t-il plusieurs parties à annoter dans une donnée ? quel est le ratio entre les parties à annoter et les parties potentiellement annotables ? est-ce que ces parties sont clairement délimitées ou ont-elles des bornes floues ?

— la complexité de **caractérisation** de l'annotation (*expressivité*, *dimensionnalité*, *ambiguïté*) : doit-on annoter une variable catégorielle ou numérique ? s'il y a plusieurs variables, y a-t-il des relations entre elles ? leur cardinalité est-elle finie ou infinie ? quelle est la proportion de confusion ou de désaccord d'interprétation de cette modélisation ?

— la complexité de **situation** de l'annotation (*poids du contexte*) : a-t-on besoin d'informations complémentaires pour être capable d'interpréter la donnée ?

Une autre manière de mesurer la complexité d'une tâche d'annotation consiste à **évaluer les différences entre annotateurs**. En effet, Gut et Bayerl, 2004 ont notamment montré qu'un nombre élevé de désaccords de labellisation peut mettre en avant une ambiguïté de modélisation, une différence d'interprétation entre annotateurs, ou encore une difficulté à saisir pleinement la subtilité des données manipulées : dans tous les cas, ces désaccords témoignent de la complexité de la tâche. Pour mesurer cet accord, différents scores ont été développés comme peut en témoigner la revue de Artstein et Poesio, 2008. Nous reviendrons plus en détails sur ces différences de comportement dans la Section 2.3.3.

Pour information : L'un des plus scores les plus utilisés est α de *Krippendorff* (Krippendorff, 2004). Le score est normalisé entre 0 (*aucun accord*) et 1 (*accord parfait*). Il est d'usage de considérer un accord fort lorsque le score est supérieur à 0.8, et l'accord reste acceptable si la valeur est supérieure à 0.667.

Exemples : Considérons l'identification d'une bande dessinée à partir de sa couverture, pour laquelle l'annotation de textes présents sur une image sont nécessaires pour entraîner un modèle de reconnaissance optique de caractères (voir Section 2.1.2.c). Nous pouvons voir que :

- la localisation des annotations est assez évidente car l'information est visuelle : (1) En général, il y a peu de texte sur une couverture de `BD`, l'information est donc facilement identifiable (*bonne discrimination*) ; (2) Cependant, il peut y avoir de légères différences sur la position exacte des encadrés identifiant les textes (*délimitation délicate*) ;
- la caractérisation concerne du texte mais son annotation est explicite : (1) L'annotation concerne du texte et des nombres, il y a donc une grande variabilité parmi les valeurs possibles (*forte expressivité*). (2) Plusieurs informations sont attendues, comme le nom de la collection, l'auteur, le titre, ou la date de parution (*dimensionnalité modérée*) ; (3) Toutefois, il y a assez peu de doute sur les valeurs à indiquer car ces dernières sont explicitement liées aux inscriptions sur l'image (*peu d'ambiguïté*) ;
- la situation de l'annotation n'a pas d'impact : en effet, peu d'informations complémentaires sont requises pour interpréter les textes de l'image (*poids du contexte faible*).

Au final, l'analyse révèle que cette tâche est relativement peu complexe. Le calcul d'un score d'accord entre annotateurs permettrait de confirmer cette hypothèse et d'en déduire la confiance que nous pouvons accorder à la qualité des données labellisées.

2.3.2.c Problèmes de coûts

Toute cette complexité a un impact non négligeable sur les coûts à engager dans un projet de conception de jeu de données. Nous allons ici détailler certains de ces coûts et donner quelques exemples notoires.

Tout d'abord, il y a des **charges de main d'oeuvre**. En effet, de nombreuses personnes aux compétences diverses vont s'investir dans un tel projet (voir Section 2.2.2), à la fois pour modéliser le phénomène mais aussi pour labelliser les données le représentant. Cependant, les experts qualifiés pour ces tâches sont souvent très demandés : ainsi, ces profils sont rares à l'embauche, souvent difficiles à mobiliser sur de longues durées, et ont généralement des tarifs horaires élevés. Une alternative consiste à former le personnel nécessaire, mais cette montée en compétence prend du temps et représente aussi un investissement important (*formations professionnelles, veille technologique, appréhension du phénomène à modéliser, ...*).

Ensuite, il faut considérer les **facteurs temporels** qui impactent le délai de mise en production du modèle de *Machine Learning*. En plus des délais de montée en compétences des experts, nous pouvons ajouter :

— le temps nécessaire à la *collecte de données*, pouvant prendre plusieurs semaines (*diffusion d'un sondage, temps de mesure d'un phénomène s'étalant sur la durée, ...*) ;

— le temps imposé par les *ateliers de conception* de la modélisation, pouvant prendre entre quelques heures et quelques semaines en fonction de sa complexité et des différences d'opinions entre experts ;

— le temps dédié à l'*annotation des données*, dépendant entre autres de la complexité de la modélisation et de la quantité à labelliser ; et

— le temps requis pour entraîner le modèle de *Machine Learning*.

À cela s'ajoute aussi la perspective de révision de la modélisation, et donc à la multiplication des coûts en appliquant plusieurs itérations du cycle `MATTER` pour obtenir un modèle convenable. Il est possible de réduire certains de ces coûts temporels en ajoutant davantage de personnes dans le projet, mais cela augmentera évidemment les charges de main d'oeuvre, et quelques aspects demeureront toutefois incompressibles (*les débats lors de la phase de modélisation par exemple*).

Enfin, il y a divers **coûts financiers à engager** en plus des recrutements et des conséquences des délais de réalisation. Nous pouvons notamment considérer l'achat ou le développement de l'infrastructure technique telle que de l'outil d'annotation et de stockage des données, les potentiels instruments de mesure du phénomène (*micros, caméras, sondes, ...*) ou encore les acquisitions de droits d'utilisation de données ou de modèles sous licence.

Exemples : Détaillons ici deux projets d'annotation dont le chiffrage est disponible :

— `Prague Dependency Treebank` (Böhmová et al., 2003) : l'annotation de 180 000 phrases tchèques (environ 1.8 million de *tokens*) pour l'analyse morpho-syntaxique a duré 5 ans (entre 1996 et 2003), impliquant 22 personnes. La somme totale engagée est estimée à 600 000 dollars ;

— `MS COCO` (Lin et al., 2014) : l'annotation d'objets présents dans 328 000 images (environ 2.5 millions d'objets répartis en 91 types différents) a nécessité environ 70 000 heures d'annotation.

Idées : Pour limiter les coûts, diverses solutions ont déjà été proposées :

— la **pré-annotation** : cette approche consiste à employer un petit modèle pour suggérer une annotation à l'annotateur, celui-ci pouvant alors l'approuver ou corriger. Fort et Sagot, 2010 démontre le gain de temps et de qualité d'une telle approche. toutefois, certains biais peuvent influencer négativement l'annotateur : Dandapat et al., 2009 souligne par exemple le risque de biais de confirmation (*influence de la machine et tendance de l'annotateur à être en accord avec celle-ci, notamment sur des données ambiguës*) ;

— l'**apprentissage actif** (*active learning*) : cette approche consiste à interagir avec la machine pour sélectionner les données les plus intéressantes à annoter (Settles, 2010). Nous pouvons par exemple entraîner un modèle avec les données déjà disponibles, le tester sur les données non annotées et sélectionner les données sur lesquelles ses prédictions sont le moins confiantes ;

— le **transfert d'apprentissage** (*transfert learning*) : cette approche consiste à réutiliser la connaissance contenue dans un modèle déjà existant dans le but d'exploiter certaines de ses fonctionnalités (Zhuang et al., 2021, Iman et al., 2023). Il a été montré qu'adapter un modèle pour un nouveau cas d'usage similaire peut se faire avec peu de données, réduisant ainsi drastiquement la charge d'annotation (Parnami et Lee, 2022) ;

— la **myriadisation** (*crowdsourcing*) : cette approche consiste à déléguer l'annotation à des plateformes collaboratives en ligne comme `Amazon Mechanical Turk` (Callison-Burch et Dredze, 2010) ou `Language ARC` (Fiumara et al., 2020). N'importe quel internaute peut alors contribuer à la tâche d'annotation, permettant ainsi de labelliser de grands volumes de données rapidement et à moindres coûts. Cette technique comporte toutefois un inconvénient majeur : les opérateurs ne sont généralement pas des experts et cer derniers sont rémunérés au volume labellisé. La qualité des annotations risque donc d'être délaissée au profit de la quantité (Sagot et al., 2011).

2.3.3 Défis concernant les différences de comportements d'annotation

Nous pouvons constater deux divergences de comportements :

— ceux entre deux annotateurs (voir Section 2.3.3.a) ;

— ceux d'un annotateur avec lui-même (voir Section 2.3.3.b).

2.3.3.a Différences inter-annotateurs

Comme nous avons pu le voir dans la Section 2.3.2.a, la modélisation d'un phénomène relève d'un choix subjectif. Il est donc normal de constater que cette subjectivité entraîne des différences de comportement d'annotation. Celles-ci se manifestent particulièrement sur des données ambiguës ou bruitées car les limites de la modélisation sont exposées. D'autres éléments comme le caractère abstrait d'une modélisation ou la présence de règles de labellisation trop floues peuvent aussi mener à des divergences d'interprétation entre opérateurs.

Bayerl et Paul, 2011 ont pu étudier un ensemble de **facteurs à l'origine de désaccords**. Nous en dressons une liste résumée ci-dessous :

— la *complexité* du problème : que celle-ci vienne intrinsèquement du phénomène à annoter ou du cas d'usage reflété dans la modélisation, nous observons que si l'annotation est difficile, alors les chances d'interpréter différemment les données sont plus élevées ;

— le *nombre* d'annotateurs : il est normal de constater que plus y a de personnes donnant leurs avis, plus les avis peuvent diverger ;

— le *niveau d'expertise* du phénomène : nous remarquons des divergences d'opinion entre les opérateurs n'étant pas spécialistes du phénomène et ceux détenant des connaissances sur celui-ci (un linguiste pour des données textuelles, un libraire pour des `BD`, ...) ;

— le *niveau de formation* à la tâche de labellisation : nous constatons des divergences entre les opérateurs habitués à l'outil et au guide d'annotation, et ceux découvrant la tâche de labellisation et sa mise en oeuvre pour la première fois.

Pour information : Bayerl et Paul, 2011 montrent aussi qu'il peut y avoir des différences entre deux annotateurs experts non formés à la tâche de labellisation, alors qu'il y a moins de divergences entre des annotateurs experts et non expert s'ils ont été formés tous les deux. Cela souligne bien l'**importance de la formation à la tâche d'annotation** et la nécessité de rédiger un guide d'annotation qui détaille l'objectif et les règles de labellisation dans le but d'en limiter la subjectivité.

Idées : Pour aller plus loin, il est recommandé d'organiser des sessions d'**adjudication** durant laquelle les opérateurs peuvent confronter leurs visions en annotant les mêmes données. Cela permet de mesurer un score d'accord entre annotateurs mais aussi de discuter des points de désaccords dans l'application du guide de labellisation. Une règle de bon sens consiste à confronter 2 annotateurs pour être capable d'identifier les points de désaccords, mais il faut être au moins 3 annotateurs pour en discuter et les résoudre. Bayerl et Paul, 2011 conseille même d'être au moins 5 annotateurs si le cas d'usage est critique.

2.3.3.b Différences intra-annotateur

Malgré la conception d'un guide et la rédaction de règles, malgré les sessions de revues dédiées à affiner ce guide et malgré les diverses pistes mises en oeuvre pour rendre l'annotation moins complexe, nous pouvons tout de même constater qu'un annotateur peut manifester des variations de comportement. Ces différences peuvent s'apparenter à des erreurs d'inattention, à des oublis de consignes, ou à d'autres fluctuations similaires.

Pour expliquer cela, nous pouvons nous baser sur la théorie de la **régulation de la charge de travail** (Sperandio, 1978, Sperandio, 1987). Celle-ci s'exprime de la manière suivante :

— D'une part, l'opérateur **estime la charge de travail** à accomplir (*la quantité, la difficulté, les délais, ...*). Concernant la tâche d'annotation, elle sera perçue comme **très élevée** (*complexité intrinsèque, volume conséquent à traiter, règles strictes de labellisation , ...*) ;

— D'autre part, l'opérateur **estime les ressources** à sa disposition (*ses capacités, ses connaissances, ses outils, ...*). Celles-ci seront plutôt perçues comme **minimes** face à cette montagne (*peu de compétences techniques, manque de formation, méconnaissance de la modélisation utilisée, ...*) ; ;

— Enfin, l'opérateur compare les deux estimations (*est-ce que la charge de travail et mes ressources sont à l'équilibre ?*) et **ajuste sa charge de travail** pour la proportionner aux ressources qu'il va engager. Dans notre cas, l'opérateur va probablement **essayer de réduire** (intentionnellement ou non) sa charge de travail pour qu'elle soit perçue comme acceptable vis-à-vis de ses ressources.

Ce principe explique alors certaines variations, notamment lorsque la modélisation est particulièrement complexe ou que l'annotateur est sous pression ou fatigué.

Exemples : Nous pouvons appliquer cette théorie lorsque l'opérateur n'a pas eu de congés depuis plusieurs semaines ou que la fin de semaine approche : ses capacités sont

perçues comme plus faibles à cause de la fatigue, nous pouvons alors constater des erreurs d'annotation même si la complexité de la tâche n'a pas changé.

Idées : Une première approche consiste à **essayer de simplifier la tâche** de labellisation (recommandation de Bayerl et Paul, 2011), permettant ainsi d'avoir une meilleure qualité avec une modélisation moins sophistiquée. Fort et al., 2012 rappellent que le contexte autour de la donnée semble aussi introduire un surplus d'information à gérer : diminuer ce contexte peut parfois alléger l'annotation mais risque en échange introduire plus d'ambiguïté. Baledent, 2023 expose le dilemme entre annoter un phénomène complexe en un seul jet et le découper en plusieurs tâches unitaires distinctes : si la première permet d'avoir une meilleure vue d'ensemble, la seconde peut alléger la charge de travail.

Il est aussi possible d'essayer de **rendre la tâche ludique** (*gamification*, Von Ahn, 2006) : plusieurs outils tentent en effet de faire oublier à l'opérateur qu'il est en train de travailler en déguisant sa mission sous la forme d'un jeu (par exemple : Guillaume et al., 2016 propose `ZombiLingo` pour l'annotation morpho-syntaxique de textes). Cette approche requiert toutefois une grande créativité pour réussir à produire cette illusion (Fort, 2017).

2.3.3.c Faible valorisation du rôle d'annotateur

Comme vu en Section 2.2.2, l'annotation repose principalement sur des opérateurs ayant une connaissance **métier** du phénomène à modéliser, ou des opérateurs formés spécifiquement à la tâche de labellisation. Ces personnes sont donc essentielles à la conception d'une base d'apprentissage.

Cependant, nous constatons que de plus en plus d'entreprises décident de sous-traiter ces tâches d'annotation à des plateformes de myriadisation (*crowdsourcing*, Howe, 2008)[9] comme `Amazon Mechanical Turk` (Callison-Burch et Dredze, 2010) ou `Language ARC` (Fiumara et al., 2020). Plusieurs études montrent en effet l'intérêt de faire participer une foule d'opérateurs non-experts (Snow et al., 2008, Fort, 2017), mais de **sérieuses questions éthiques** sont néanmoins soulevées par l'utilisation de ces plateformes.

Tout d'abord, nous pouvons remarquer que ces opérateurs sont souvent payés un salaire dérisoire proportionnel à la quantité de données qu'ils labellisent. Cela ouvre la porte à des dérives, comme la polémique autour de `ChatGPT` (OpenAI, 2023) où Perrigo et Zorthian, 2023 avaient dénoncé l'emploi de Kényans pour moins de 2$ de l'heure dans le but de corriger ce modèle. Nous pouvons aussi citer Dzieza, 2023 qui alerte sur l'apparition cette nouvelle classe de travail sous-payée, n'ayant pas de place claire dans le droit du travail, et généralement dévalorisée dans l'organisation des projets d'annotation.

D'autres part, Rowe, 2023 reporte aussi les impacts émotionnels et psychologiques causés par certaines tâches d'annotation. En effet, une mission récurrente consiste à modérer des contenus à caractères offensants (*insultes, violence, drogue, sexe, armes, ...*) : de nombreux annotateurs sombrent ainsi dans la dépression après avoir labellisé de telles données pendant des jours voire des semaines...

9. Myriadisation : Pour rappel, ces plateformes collaboratives permettent à n'importe quel internaute de contribuer à une tâche d'annotation, permettant ainsi de labelliser de grands volumes de données rapidement et à moindre coûts.

Pour information : Pour conclure concernant ces dérives éthiques, Valette, 2016 rédige une critique franche des stratégies d'annotation dans le domaine du traitement automatique du langage naturel. Celle-ci dénonce l'organisation actuelle où les experts linguistes sont devenus de simples sous-traitants n'ayant plus leur mot à dire dans la conception d'une modélisation : ils sont généralement traités avec mépris, ne récoltent pas les lauriers des projets à succès mais sont tenus pour responsables des projets en échec.

2.3.4 Bilan concernant les nombreux défis de l'annotation

Points à retenir : En dressant la liste des défis autour de la tâche d'annotation, nous avons pu voir que :

- ☑ L'enjeu d'un projet d'annotation consiste à **avoir des données de qualité** : celles-ci doivent être représentatives du problème à traiter, en quantité suffisante, avec un minimum de bruit, et leurs droits d'usage doivent être disponibles ;
- ☑ Cependant, la tâche de labellisation et son exigence de qualité **engendre de la complexité** : celle-ci peut venir de la difficulté à modéliser le phénomène ou de l'annotation elle-même, ce qui engendre un certains nombre de coûts ;
- ☑ Ainsi, cette complexité **créé des différences de comportements** : ces divergences s'expliquent notamment par la subjectivité de l'annotation et par la régulation de la charge de travail effectuée par l'opérateur lorsque cette charge est trop élevée.

2.4 Contexte du doctorat : comment assister la conception d'une base d'apprentissage pour un agent conversationnel bancaire en français ?

Durant ce doctorat, nous nous sommes intéressé à la **conception d'assistants conversationnels** (*chatbot*). En effet, leur utilisation en entreprise est de plus en plus courante (Goasduff, 2019, Costello et LoDolce, 2022), notamment pour l'automatisation de certaines tâches simples et l'accès aux informations de bases documentaires. La popularité de ces assistants vient entre autres de la possibilité de dialoguer directement avec la machine grâce à des requêtes exprimées en langage naturel, offrant ainsi un gain de confort, de disponibilité et de performance.

Par expérience, nous avons constaté que plusieurs critères sont nécessaires pour déployer un assistant conversationnel dans un contexte industriel :

— Il faut être capable de gérer le dialogue entre l'utilisateur et l'assistant (*comprendre la requête initiale, demander ou confirmer des informations complémentaires, ...*) ;

— Il faut être capable de contrôler le contenu des réponses de l'assistant et de s'assurer de ses performances (*répondre ou agir de manière adaptée, ne pas fournir de réponses contenant des informations confidentielles, ne pas répondre de manière indécente, ...*) ;

— Il est possible de donner à l'assistant l'accès à certaines ressources (*lire et écrire en base de données, exécuter des programmes tiers, ...*) mais il faut alors en garantir un certain niveau de sécurité (*se prémunir contre les requêtes malveillantes et les erreurs de manipulations*).

Pour toute ces raisons, **l'architecture traditionnelle des *chatbot* est plutôt orientée par tâches** (*task-oriented*), c'est-à-dire qu'elle manipule une abstraction du dialogue en intentions[10] et gère un paramétrage des réponses dépendant des intentions détectées (voir Chen et al., 2017 et Brabra et al., 2022).

> **Pour information :** L'Annexe B détaille plus amplement les différences entre les assistants *task-oriented* (approches symboliques) et les assistants *chat-oriented* (approches numériques ou génératives). Cette annexe se base notamment sur une revue des architectures de conceptions publiée par Chen et al., 2017.

Néanmoins, l'élaboration de tels assistants reste un **défi difficile à relever dans le monde industriel** :

— Le traitement du langage en tant que tel est un problème complexe : il faut traiter une grande variété de bruits et d'ambiguïtés de dialogue en plus d'un vocabulaire souvent spécifique au domaine de l'assistant (voir les problèmes de bruits et de représentativité en Section 2.3.1) ;

— La base d'apprentissage ainsi que les réponses de l'assistant peuvent contenir des données privées ou confidentielles : ainsi, il y a peu de données réutilisables à partir de sources publiques, et de fortes pressions sont exercées sur le contrôle du comportement du *chatbot* (voir les problèmes de droits d'utilisation et de confidentialité en Section 2.3.1) ;

— L'assistant doit parfois pouvoir manipuler un grand nombre de cas d'usages : sa modélisation peut alors représenter des dizaines d'intentions pour lesquelles des centaines de branches de dialogues peuvent être paramétrées, introduisant ainsi une grande complexité aux tâches de modélisation et d'annotation de sa base d'apprentissage (voir Section 2.3.2) ;

Pour surmonter ces difficultés, les entreprises font alors intervenir des experts aux compétences diverses (voir Section 2.2.2), notamment des experts analytiques pour concevoir une modélisation stable des textes en intentions, puis des experts métiers pour valider la pertinence de la modélisation proposée et annoter les données suivant cette modélisation.

Or au vu de la complexité d'un tel projet, des différences de comportements entre opérateurs, telles que des erreurs d'annotation ou des divergences d'opinion, sont inévitables (voir Section 2.3.3). Il est alors nécessaire de former les experts métiers aux tâches d'annotation et à certaines tâches d'analyse afin d'encadrer les discussions autour de certaines différences de comportements pouvant mener à des remises en cause de la modélisation abstraite de textes en intentions (voir étape *Revise* du cyle `MATTER`, Section 2.2.1.a). Au final, **cette organisation devient très coûteuse** car elle demande des formations analytiques à des experts métiers, nécessite l'organisation d'ateliers de modélisation en mode essai-erreur pour trouver une base

10. Intention de dialogue : en traitement automatique du langage naturel, une intention représente la compréhension de la demande formulée par un utilisateur au cours de la conversation. Elle est généralement définie par le verbe d'action de la demande, et est représentée par une étiquette. Par exemple, les requêtes « *joue moi du jazz s'il te plaît !* » ou « *peux-tu lancer une playlist de Noël sur l'enceinte du salon !* » peuvent être modélisées par l'intention `jouer_musique`. Pour plus d'information, consulter l'Annexe B.

d'apprentissage stable et pertinente, et fait intervenir des experts métiers sur une abstraction de leurs connaissances du quotidien.

Exemples : Au cours de ce doctorat, nous avons entre autres travaillé sur des assistants conversationnels à destination de conseillers bancaires et de leur clients. Ces assistants doivent traiter une large variété de sujets (banque, assurance, finance, ...) et peuvent donc rapidement contenir une centaine d'intentions pour plus d'un millier de branches de dialogue. La conception de la base d'apprentissage de tels assistants représente ainsi un réel défi d'organisation, sur plusieurs semaines, notamment pour faire intervenir des experts de la banque-assurance dans un projet d'intelligence artificielle, domaine dans lequel ils n'ont pas ou peu de connaissances...

Cependant, il pourrait être intéressant de remettre en question cette organisation des projets d'annotation où les experts métiers sont interrogés sur des compétences qui ne sont pas les leurs. Ainsi, dans le but de trouver une solution à cette problématique, **nous nous sommes alors posé la question suivante** :

Comment assister la phase de modélisation de textes en intentions pour concevoir la base d'apprentissage d'un assistant conversationnel en impliquant des experts métiers pour leurs vraies compétences et en leur demandant un minimum de bagages analytiques ou techniques ?

Idées : Pour répondre à cette problématique, nous nous sommes intéressés particulièrement à trois concepts issus de la littérature :

- aux techniques de *clustering*, permettant de déléguer à la machine la tâche de modélisation grâce à une segmentation des données sur la base de leurs similarités (Xu et Tian, 2015) ;
- à l'annotation de contraintes binaires sur les données, permettant de corriger le fonctionnement d'un algorithme de *clustering* en y introduisant de la connaissance métier (Lampert et al., 2018) ;
- aux techniques d'apprentissage actif, favorisant les interactions entre l'Homme et la Machine pour atteindre un objectif (Settles, 2010).

Ces trois concepts seront détaillés au début du chapitre suivant, et seront assemblés dans le but de concevoir une nouvelle méthode d'annotation basée sur un `Clustering Interactif`.

Chapitre 3

Proposition d'un *Clustering Interactif* pour assister la tâche de modélisation d'un jeu de données textuelles

Dans le chapitre précédent, nous avons vu les points essentiels suivants :

- ☑ L'étape de modélisation est nécessaire pour définir les objectifs et les règles d'un projet d'annotation ; or cette étape rencontre de nombreux défis qui la rendant particulièrement laborieuse (*complexité intrinsèque du phénomène, subjectivité des opérateurs, différences de comportements, ...*). ainsi, la modélisation est régulièrement révisée (cycle `MATTER`).
- ☑ La modélisation de textes en intentions pour entraîner un assistant conversationnel orienté par tâches n'échappe pas à ce constat, notamment à cause de la complexité du langage naturel, de la diversité d'intentions de dialogue à représenter, et de la pression sur le contrôle du comportement de l'assistant.
- ☑ Dans un cadre industriel, des experts métiers sont responsables de la modélisation et de l'annotation des données spécifiques ou confidentielles de l'entreprise ; or ces interventions requièrent des compétences analytiques et techniques dont les experts métiers ne disposent pas forcément ; de ce fait, la manipulation d'une modélisation abstraite de leurs connaissances est alors vécue comme une tâche pénible, nécessitant un grand nombre de formations et organisée sous la forme d'ateliers en mode essai-erreur.

Dans cette partie, nous cherchons une alternative à cette organisation traditionnelle, et nous proposerons une méthodologie d'annotation basée sur un `Clustering Interactif` visant à remplir un double objectif :

- ☐ Permettre d'assister la modélisation et l'annotation des données pour créer plus efficacement une base d'apprentissage destinée à la classification d'intentions d'un assistant conversationnel ;

☐ Redéfinir les tâches et les objectifs des différents acteurs afin de rester au plus proche de leurs compétences réelles, particulièrement en ce qui concerne l'intervention des experts métiers du projet.

Pour information : Cette proposition de méthode a été l'objet d'une présentation à la conférence `EGC (Extraction et Gestion des Connaissances)` (Schild, Durantin, Lamirel et Miconi, 2021 et (Schild, Durantin et Lamirel, 2021), et d'une extension dans le journal `IJDWM (International Journal of Data Warehousing and Mining)` (Schild et al., 2022). Nous reprenons ici certains des éléments présentés avec quelques détails supplémentaires.

Sommaire

3.1 Intuitions à l'origine d'un *Clustering Interactif*

Tout d'abord, détaillons trois intuitions qui nous ont permis de concevoir notre méthodologie d'annotation.

3.1.1 Utiliser une approche non supervisée pour créer une modélisation

Dans le but d'assister la phase de modélisation des données, une piste intéressante revient à déléguer cette tâche à la machine. En effet, grâce à une **classification non supervisées (*clustering*)**, un algorithme peut regrouper les données en fonction de leur similarité intrinsèque et ainsi suggérer une modélisation en intentions. Plusieurs algorithmes et méthodes connus peuvent être utilisés :

— le ***clustering*** `KMeans` (MacQueen, 1967) : cette méthode se repose sur la minimisation de l'inertie intra-classes en attribuant chaque donnée au barycentre de *clusters* le plus proche. Cette approche est l'une des plus répandues en raison de sa simplicité et de sa rapidité de calcul ;

— le ***clustering* hiérarchique** (Murtagh et Contreras, 2012) : cette méthode revient à fusionner itérativement les données les plus similaires dans un nouveau *cluster*. Plusieurs liens de similarité peuvent être implémentés (*le lien `single` fusionnant les deux clusters*

ayant les frontières les plus proches, le lien `complete` *fusionnant les deux clusters ayant les frontières opposées les plus proches, le lien* `average` *fusionnant les deux clusters ayant les barycentres les plus proches et le lien* `ward` *fusionnant les deux clusters qui donneront le prochain cluster le plus compact*) ;

— le ***clustering* spectral** (Ng et al., 2002) : cette méthode consiste à modéliser la matrice de similarité entre les données par ses vecteurs propres puis de regrouper ces derniers à l'aide d'un `KMeans`. Cette approche permet d'obtenir des *clusters* aux formes complexes ;

— le ***clustering*** `DBScan` (Ester et al., 1996) : cette méthode utilise la densité de données dans l'espace pour identifier des regroupements. Cette approche permet de découvrir des *clusters* aux formes complexes, si leur densité est suffisante.

— ...

Cependant, ces algorithmes non supervisés font régulièrement face à un ensemble de difficultés qui pénalisent leur utilisation. En effet, ces méthodes ont souvent du mal à traiter des données de grandes dimensionnalités ou en grand nombre (Steinbach et al., 2004), la présence de bruits peut rapidement perturber le fonctionnement d'un algorithme (Yang et Wang, 2004), et certains *clusters* aux géométries complexes peuvent être difficiles à identifier (Kriegel et al., 2011). De plus, le choix de certains hyperparamètres de ces méthodes n'est pas toujours simple, surtout en ce qui concerne le nombre de *clusters*, la méthode d'initialisation ou la mesure de distance à utiliser (Agarwal et al., 2011). Enfin, toutes ces difficultés se retrouvent dans le traitement du langage naturel, notamment à cause de la taille de vocabulaire importante, de la présence de nombreux bruits et de la vaste diversité et complexité des thématiques pouvant y être abordées.

> **Pour information :** La revue de Xu et Tian, 2015 détaille plusieurs algorithmes de *clustering*, en fonction de leurs forces et faiblesses sur les points mentionnés ci-dessus.

Ainsi, toutes ces limites étayent le fait qu'**un résultat brut d'une classification non supervisée est généralement perçu comme peu pertinent par les experts métiers**. Il est donc nécessaire d'introduire une intervention humaine dans le processus pour guider le fonctionnement d'un algorithme de *clustering*.

3.1.2 Corriger l'approche non supervisée avec une annotation de contraintes

Une variante aux approches non supervisées consiste à demander à un humain certaines informations nécessaires à leur amélioration nous considérons donc les **approches semi-supervisées**. Une manière efficace d'introduire les connaissances d'un expert dans le processus est l'ajout de contraintes.

Lampert et al., 2018 rappellent qu'il peut y avoir deux types de contraintes :

— les **contraintes sur les données** : nous parlons alors principalement des contraintes binaires `MUST-LINK` et `CANNOT-LINK` décrivant si deux données doivent ou ne doivent pas être dans un même *cluster* (Wagstaff et Cardie, 2000)

— les **contraintes sur les *clusters*** : ces contraintes peuvent concerner le nombre de *clusters* à trouver, leur taille minimale ou maximale, la distance minimale de séparation de leurs frontières, ...

Bien que d'autres méthodes permettent d'insérer des contraintes (*heuristiques non supervisées, transferts de connaissances préalables*), nous nous concentrons ici sur l'ajout manuel par un expert. Comme cet expert n'a *a priori* pas de connaissances techniques ou analytiques, ce dernier aurait du mal à manipuler des contraintes sur les *clusters*. Toutefois, ses connaissances métiers lui permettent de caractériser facilement la similarité entre deux données, et donc de décrire des contraintes de type `MUST-LINK` et `CANNOT-LINK` en répondant à la question : « *est-ce que les deux données traitent du même cas d'usage ?* ».

> **Notes de l'auteur :** La pierre angulaire de la méthode que nous proposons en Section 3.2 repose notamment sur le fait qu'il est difficile pour un expert métier de classer une question suivant une modélisation abstraite prédéfinie : cela l'éloigne de ses compétences métiers initiales, nécessite en contre-partie de nombreuses formations, et introduit de nombreuses erreurs d'annotation. De fait, il semble plus adéquat de demander à l'expert métier de discriminer deux questions sur la base de leurs similarités de cas d'usage métier.

Parmi les algorithmes de *clustering* sous contraintes connus, nous disposons des adaptations suivantes :

— le ***clustering*** `KMeans` **sous contraintes** comme `COP-KMeans` (Wagstaff et al., 2001) : dans cette version, l'attribution d'une donnée se fait au *cluster* dont le barycentre est le plus proche et où aucune contrainte n'est violée. Cette adaptation est relativement simple à mettre en oeuvre, mais elle peut mener à des cas de blocage où plus aucun *cluster* n'est accessible pour cause de violation de contraintes (*il est possible d'adapter l'algorithme en créant un nouveau cluster*) ;

— le ***clustering*** **hiérarchique sous contraintes** (Davidson et Ravi, 2005) : dans cette version, les données liées par des contraintes `MUST-LINK` sont d'abord fusionnées, puis le processus agglomératif commence en prenant garde de ne pas fusionner des *clusters* ayant des contraintes `CANNOT-LINK` entre eux. Cette adaptation est très simple à mettre en oeuvre car il suffit d'adapter le calcul de distance entre *clusters* ;

— le ***clustering*** **spectral sous contraintes** (Kamvar et al., 2003) : dans cette version, les coefficients de la matrice de similarité sont forcés à 0 (respectivement 1) si deux données sont liées par une contrainte `CANNOT-LINK` (respectivement `MUST-LINK`). Cette adaptation demande peu d'effort pour être mise en oeuvre, mais des modifications aussi drastiques de la matrice de similarité peut entraîner des changements imprévisibles de comportements ;

— le ***clustering*** `DBScan` **sous contraintes** comme `C-DBScan` (Ruiz et al., 2010) : dans cette version, la densité des données ainsi que les contraintes `CANNOT-LINK` sont utilisées pour identifier des *clusters* locaux qui seront ensuite fusionnés à l'aide des contraintes `MUST-LINK` Cette adaptation est plus compliquée à réaliser car elle change légèrement le fonctionnement interne d'exploration de la densité de l'espace de données.

— ...

Ces algorithmes de *clustering* sous contraintes sont ainsi pleins de potentiels : ils sont en effet capables de tirer parti de la similarité intrinsèque des données et des contraintes judicieusement placées de l'expert pour segmenter les données de manière adéquate et ainsi proposer une modélisation pertinente.

Exemples : Nous pouvons citer Lampert et al., 2019 où l'annotation de contraintes par un expert permet d'identifier efficacement des objets dans une image satellite.

Cependant, pour un jeu de données de taille N, il y a N^2 possibilités de contraintes à ajouter. L'estimation du placement des contraintes les plus appropriées et les plus efficaces pour corriger le *clustering* semble donc être un problème NP-difficile. De plus, si ce choix peut être visuel pour des images (comme dans l'exemple cité ci-dessus), il ne l'est pas pour des données textuelles dont la diversité et la complexité sont difficiles à appréhender. Il est donc important d'assister l'expert métier pour qu'il dispose ses contraintes en maximisant son impact dans la correction du *clustering*.

3.1.3 Tirer parti des avantages de l'apprentissage actif pour optimiser les interactions Homme/Machine

Une dernière piste intéressante est celle de l'apprentissage actif (*active learning*, voir Settles, 2010) : celle-ci prône les interactions Homme/Machine comme moyen d'atteindre un objectif qu'aucun ne peut atteindre séparément. Bae et al., 2021 listent par exemple un ensemble d'interactions possibles avec un algorithme de *clustering* : celles-ci peuvent concerner la manipulation et l'adaptation des résultats, l'adaptation des hyperparamètres des algorithmes, mais aussi des initiatives de la machine pour préparer la réalisation d'une tâche.

Dans notre cas, nous nous intéressons en particulier aux initiatives de la machine pour identifier la liste des contraintes à annoter pour corriger ou confirmer efficacement un résultat de *clustering*. Cela peut se faire par exemple à l'aide d'une heuristique sélectionnant les parties les moins sûres de la segmentation.

Points à retenir : Dans cette partie, nous avons exposé les intuitions suivantes :

- ☑ La phase de modélisation des données peut être déléguée à la machine en employant des algorithmes de classification non supervisée (*clustering*) ;
- ☑ Pour corriger le fonctionnement d'un algorithme de *clustering* et ainsi améliorer la pertinence de ses résultats, l'expert peut ajouter des contraintes binaires sur les données (`MUST-LINK` et `CANNOT-LINK`, *clustering* sous contraintes) ;
- ☑ Dans le but d'ajouter des contraintes de manière efficace, il est possible d'interagir avec la machine afin de maximiser l'impact de l'intervention de l'expert (*active learning*).

3.2 Description de notre *Clustering Interactif*

Sur la base des intuitions que nous venons de détailler, nous proposons la méthode suivante dans le but d'assister la modélisation et l'annotation d'une collecte de données brutes en une base d'apprentissage nécessaire à l'entraînement d'un assistant conversationnel.

3.2.1 Description générale

Notre méthode d'annotation, que nous appelons « `Clustering Interactif` », **repose sur l'alternance successive entre deux phases clefs** (voir FIGURE 3.1) :

- une phase d'**annotation de contraintes** par un expert permettant de caractériser la similarité entre deux données suivant leur cas d'usage métier ;
- une phase de **segmentation automatique** des données par une machine sur la base de la proximité sémantique des données et des contraintes précédemment annotées.

L'objectif recherché en associant ces deux phases est de **créer un cercle vertueux pour améliorer itérativement la qualité de la base d'apprentissage** en cours de construction. En effet, à chaque itération, l'expert métier obtiendra une proposition de segmentation des données qu'il pourra affiner dans le but de corriger le fonctionnement de la machine et ainsi d'obtenir une segmentation plus pertinente à l'itération suivante.

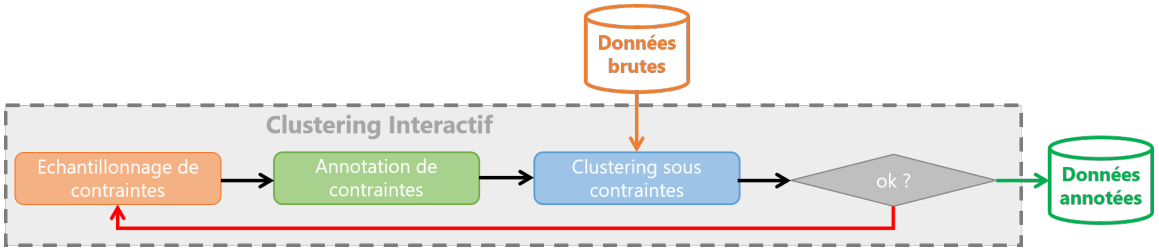

FIGURE 3.1 – *Schéma illustrant l'architecture du `Clustering Interactif`. La boucle principale enchaîne un échantillonnage de paires de données, une annotation de contraintes, et un clustering sous contraintes.*

3.2.2 Description détaillée

L'ALGORITHME 3.1 décrit formellement notre proposition de `Clustering Interactif` que nous détaillons ci-dessous.

Pour l'**initialisation** de la méthode (cf. ALGORITHME 3.1, *lignes 1 à 3*), nous définissons une liste vide de contraintes : tout au long du processus, nous y ajoutons les contraintes annotées par l'expert grâce à ses connaissances métiers (nous entrerons en détail en décrivant la phase d'annotation). Il faut aussi une première segmentation des données par la machine : celle-ci se réalise par l'exécution d'un algorithme de *clustering*. Nous estimons qu'il n'est pas du ressort de l'expert métier de choisir l'algorithme de *clustering* ni de régler ses hyperparamètres. Ces derniers pourront être déterminés par un *data scientist* en fonction du problème à traiter. Il est à noter que cette première segmentation des données est réalisée sans bénéficier de la connaissance de l'expert, il est donc peu probable que le résultat soit pertinent à ce stade.

Nous entrons dans le coeur de la boucle itérative par la phase d'**échantillonnage** (cf. ALGORITHME 3.1, *lignes 5 et 6*). Comme mentionné au préalable, savoir quelles contraintes ajouter

Données : données non segmentées
Entrées : budget à disposition
1 **initialisation** : créer une liste vide de contraintes ;
2 *optionnel* : évaluer les hyperparamètres de la segmentation automatique ;
3 **segmentation initial** : regrouper les données par similarité ;
4 **répéter**
5 | *optionnel* : évaluer les hyperparamètres de l'échantillonnage ;
6 | **échantillonnage** : sélectionner une partie de la segmentation à corriger ;
7 | **annotation** : corriger la segmentation en ajoutant des contraintes sur l'échantillon ;
8 | *optionnel* : réévaluer les hyperparamètres de la segmentation automatique ;
9 | **segmentation** : regrouper les données par similarité avec les contraintes ;
10 | **validation** : estimer la pertinence et la stabilité de la segmentation ;
11 | **coûts** : estimer le budget restant et les coûts restants à investir ;
12 **jusqu'à** *segmentation satisfaisante OU budget épuisé*;
13 **interprétation** : trier et nommer les *clusters* pour les exploiter ;
Résultat : données segmentées (i.e. base d'apprentissage)

ALGORITHME 3.1 – *Description en pseudo-code de la méthode d'annotation proposée employant le* `Clustering Interactif`.

pour corriger efficacement le *clustering* est un problème NP-difficile (le nombre de possibilités croît proportionnellement au carré du nombre de données). De plus, l'intervention d'experts est chiffrée et représente en général une partie des coûts à investir dans un projet (voir SECTION 2.3.2.C). Il est donc inconcevable de laisser un expert métier annoter des contraintes "*seul*" et "*au hasard*". Ainsi, pour optimiser ses interventions, il convient de déterminer là où l'expert aura le plus d'impact lors de sa transmission de connaissances. C'est pourquoi la phase d'échantillonnage est primordiale dans la méthode proposée : nous proposons d'y sélectionner des paires de données sur la base de leur similarité, de leur segmentation ou encore de leurs relations avec d'autres données déjà liées par des contraintes.

Sur la base de cet échantillon, l'expert peut entamer son étape d'**annotation de contraintes** (cf. ALGORITHME 3.1, *ligne 7*). Pour alléger la charge d'annotation, nous avons décidé de discriminer les données de l'échantillon par des contraintes binaires simples : `MUST-LINK` et `CANNOT-LINK`. Ces contraintes représentent respectivement la similitude ou la différence entre deux données, et seront utilisées pour regrouper ou séparer certaines données dans la prochaine segmentation. En fonction de l'orientation du projet et afin de rester au plus proche des compétences réelles de l'expert, la formulation de l'énoncé d'annotation doit être judicieusement définie : par exemple, les contraintes peuvent représenter une similitude sur la thématique concernée [11], sur l'action désirée [12], ou encore sur le besoin de l'utilisateur [13]. Nous noterons que des incohérences peuvent s'introduire, ayant pour conclusions de devoir à la fois considérer comme similaires et différentes deux données : ces incohérences peuvent être détectées grâce aux propriétés de transitivité des contraintes (voir la gestion des conflits en ANNEXE C.1.2).

Pour finir, la dernière phase de cette boucle est composée d'une nouvelle **segmentation** des données (cf. ALGORITHME 3.1, *lignes 8 et 9*). Cette segmentation devra respecter les contraintes préalablement définies par l'expert, nous nous tournons donc vers l'utilisation d'un *clustering*

11. Exemples de thématiques : *crédit* vs. *assurance* ; *sport* vs. *culture*, ...
12. Exemples d'actions : *souscrire* vs. *résilier* ; *activer* vs. *désactiver* ; *s'informer* vs. *réaliser*, ...
13. Exemple de besoins : *souscrire un crédit* vs. *souscrire une assurance* ; *s'informer en sport* vs. *s'informer en culture*, ...

sous contraintes. Au fur et à mesure des itérations, de plus en plus de contraintes seront ajoutées pour corriger le *clustering*. Ainsi, au bout d'un certain nombre d'itérations, la segmentation des données reflétera la vision que l'expert aura voulu transmettre. Comme précédemment, nous estimons qu'il n'est pas du ressort de l'expert métier de choisir l'algorithme de *clustering* ni de régler ses hyperparamètres. Ces derniers pourront être déterminés par un *data scientist* en fonction du problème à traiter, de l'itération en cours et des contraintes disponibles.

Comme la méthode est itérative, il faut pouvoir estimer des **cas d'arrêt** (cf. ALGORITHME 3.1, *lignes 10 à 12*). Le cas d'arrêt le plus évident n'est pas technique mais relatif aux coûts investis dans l'opération : si le projet n'a plus de budget dédié à l'annotation, il faudra créer la base d'apprentissage avec le résultat à disposition, quel que soit la pertinence de la segmentation obtenue sur les données. Ce cas d'arrêt par défaut peut malheureusement être synonyme d'échec pour le projet si les résultats sont inexploitables. D'autres cas d'arrêts peuvent être envisagés en fonction de la qualité ou de la pertinence de la segmentation. D'une part, nous pouvons comparer l'évolution de la segmentation des données : si les segmentations sont similaires sur plusieurs itérations, il est possible que la modélisation atteigne un optimum local. D'autre part, nous pouvons aussi comparer l'évolution de l'accord entre la segmentation obtenue et l'annotation de l'expert : en effet, si l'expert ne contredit plus la répartition proposée des données, il est probable que sa vision et la vision de la machine aient convergé. Dans les deux cas, l'analyse de l'expert métier reste nécessaire pour valider si la modélisation des données est pertinente ou si elle comporte encore des incohérences à corriger.

Lorsque la boucle itérative est finie, nous avons à disposition une segmentation des données qui a été corrigée par un expert et qui reflète ses connaissances métiers. La dernière étape consiste alors à **interpréter** ces *clusters* pour pouvoir les exploiter (cf. ALGORITHME 3.1, *ligne 13*). Cela commence par leur attribuer un nom au lieu de leur identifiant technique, de les définir en les rapprochant d'un cas d'usage métier, et éventuellement de les raffiner manuellement en supprimant certaines données aberrantes.

Exemples : La FIGURE 3.2 déroule l'initialisation et la première itération de la méthode sur un exemple fictif. Nous pouvons constater qu'entre les images **(2)** et **(5)**, la segmentation des données a évolué grâce à l'introduction de contraintes.

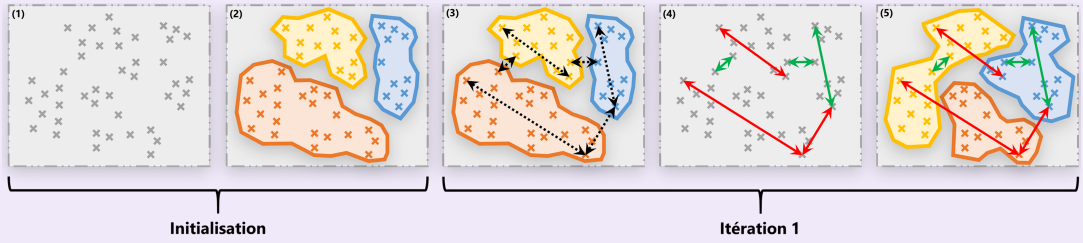

FIGURE 3.2 – *Exemple d'une itération de `Clustering Interactif`.*
Lors de l'initialisation, (1) correspond au jeu de données brut, et (2) correspond à une première segmentation des données en 3 clusters. Lors de l'itération 1 : (3) correspond à un exemple d'échantillonnage de 6 contraintes représentées par les flèches en pointillé, (4) correspond à la caractérisation de ces 6 contraintes par des liens `MUST-LINK` en vert et `CANNOT-LINK` en rouge, et (5) correspond à la nouvelle segmentation des données en 3 clusters respectant les 6 contraintes annotées. La prochaine itération se poursuivra par un nouvel échantillonnage de contraintes.

3.2.3 Descriptions techniques et implémentation

Au cours de ce doctorat, nous avons réalisé un ensemble d'implémentations en `Python` afin de mettre en oeuvre notre méthodologie de *Clustering Interactif*. Celle-ci est répartie en trois librairies :

1. `cognitivefactory-interactive-clustering`[14] (Schild, 2022a), regroupant les gestions de données et des contraintes, les algorithmes de *clustering* et d'échantillonnage ;
2. `cognitivefactory-interactive-clustering-gui`[15] (Schild, 2022b), intégrant la logique de la méthodologie dans une application web ;
3. `cognitivefactory-features-maximization-metric`[16] (Schild, 2023), disposant d'une méthode de sélection des patterns linguistiques caractéristiques d'un jeu de données labellisées, permettant ainsi d'analyser la pertinence d'un résultat de *clustering* en fonction du vocabulaire utilisé dans chaque *cluster*.

Exemples : La Figure 3.3 représente une capture d'écran de la page d'annotation de contraintes de l'application web intégrant notre méthodologie de `Clustering Interactif`.

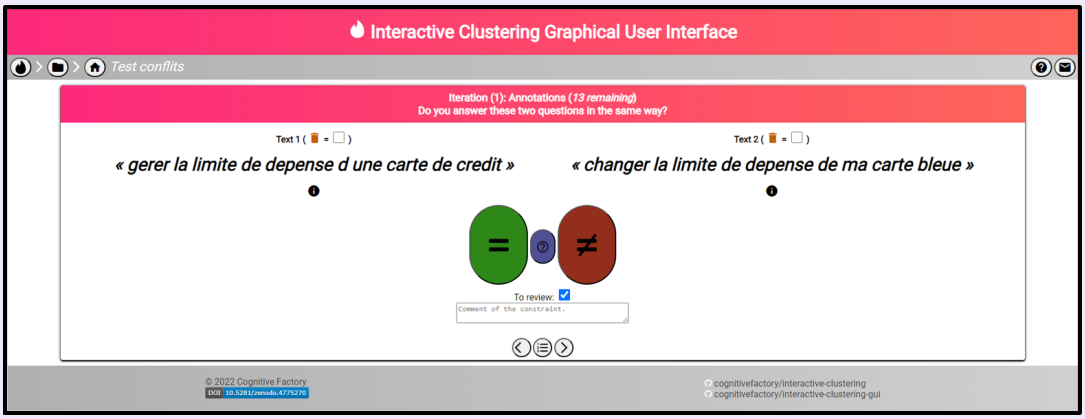

Figure 3.3 – *Capture d'écran de l'application web implémentant notre méthodologie de `Clustering Interactif` : **page d'annotation d'une contrainte**. Parmi les éléments importants, nous retrouvons les deux textes à annoter (disposés à gauche et droite de l'écran) et les boutons d'annotation (bouton « = » pour un `MUST-LINK`, bouton « ≠ » pour un `CANNOT-LINK`. Les autres fonctionnalités sont détaillées en Annexe C.2.*

Pour information : Ces implémentations sont présentées dans l'Annexe C. L'ensemble des détails techniques et des explications sur les choix d'implémentation y sont décrits.

14. https://pypi.org/project/cognitivefactory-interactive-clustering/
15. https://pypi.org/project/cognitivefactory-interactive-clustering-gui/
16. https://pypi.org/project/cognitivefactory-features-maximization-metric/

3.3 Perspectives portées par la méthode proposée.

Nous avons proposé une méthodologie d'annotation basée sur une interaction entre l'Homme et la Machine dans le but de soulager l'expert métier dans son intervention.

Grâce à une telle approche, nous pouvons espérer que :

- ☐ Les experts métiers n'auront désormais plus besoin de bagages analytiques ou techniques pour intervenir dans un projet d'annotation ;
- ☐ Les experts métiers pourront désormais participer à la modélisation d'une base d'apprentissage en ayant des discussions pragmatiques sur les cas d'usages des données manipulées ;
- ☐ Une telle méthodologie d'annotation permettra d'obtenir efficacement une base d'apprentissage stable et pertinente pour entraîner une modèle de classification d'intention ;
- ☐ Une telle méthodologie d'annotation sera réaliste en terme de délais et d'investissement financier.

Nous allons explorer diverses pistes pour confirmer ou infirmer ces perspectives dans le Chapitre 4, et nous détaillerons nos conclusions dans un guide d'utilisation qui sera présenté dans le Chapitre 5.

Chapitre 4

Étude de six hypothèses sur le *Clustering Interactif*

Dans le chapitre précédent, nous avons présenté une méthode de création d'un jeu de données d'entraînement pour un assistant conversationnel, que nous appelons « `Clustering Interactif` » :

- ☑ La méthode proposée repose sur la combinaison entre un regroupement automatique des données par la machine et l'annotation de contraintes binaires par un expert métier pour corriger le regroupement proposé ;
- ☑ Une telle approche devrait limiter les prérequis de compétences analytiques et techniques actuellement exigés d'un expert métier en les déléguant à la machine.
- ☑ En échange, l'expert se concentre d'avantage sur la transmission de ses connaissances avec une annotation caractérisant la similitude métier entre deux données.

Il existe des travaux similaires sur l'annotation de données visuelles (Lampert et al., 2019) et des revues sur les interactions possibles avec un algorithme de *clustering* (Bae et al., 2021), Cependant, peu d'études de la littérature scientifique ont exploré les possibilités d'interactions entre un expert métier et un algorithme de *clustering* sous contraintes dans le but de modéliser des données textuelles en intentions. Ainsi, dans cette partie, **nous étudions la faisabilité d'un tel `Clustering Interactif` pour des données textuelles** en explorant les six questions suivantes :

- ☐ Peut-on obtenir une base d'apprentissage à l'aide de notre proposition d'implémentation de la méthodologie du `Clustering Interactif` ? (cf. hypothèse d'**efficacité** en Section 4.1) ;
- ☐ Peut-on déterminer un paramétrage optimal de cette implémentation pour obtenir plus rapidement une base d'apprentissage ? (cf. hypothèse d'**efficience** en Section 4.2) ;
- ☐ D'après les données initiales, peut-on faire une approximation de l'investissement

nécessaire pour obtenir une base d'apprentissage exploitable ? (cf. hypothèse sur les **coûts** en SECTION 4.3) ;

- ☐ A un instant donné, peut-on estimer la pertinence métier d'une base d'apprentissage en cours de construction ? (cf. hypothèse de **pertinence** en SECTION 4.4) ;
- ☐ Au cours du processus de construction de la base d'apprentissage, peut-on aisément estimer les potentiels d'une étape de raffinage supplémentaire ? (cf. hypothèse de **rentabilité** en SECTION 4.5) ;
- ☐ Peut-on estimer l'influence d'une différence d'annotation dans la construction de la base d'apprentissage ? (cf. hypothèse de **robustesse** en SECTION 4.6).

Afin de vérifier ces différentes hypothèses, nous organisons un ensemble d'études basées soit sur des approches théoriques (*simulations des comportements et comparaisons à une vérité terrain grâce à un score de* `v-measure`[17] *(ROSENBERG et HIRSCHBERG, 2007)*), soit sur des approches empiriques (*expériences en situations réelles et analyses basées sur les compétences d'opérateurs métier*). L'imbrication de ces études est représentée dans la FIGURE 4.1 qui évoluera au cours des sections suivantes pour résumer les hypothèses vérifiées et annoncer l'hypothèse en cours d'étude.

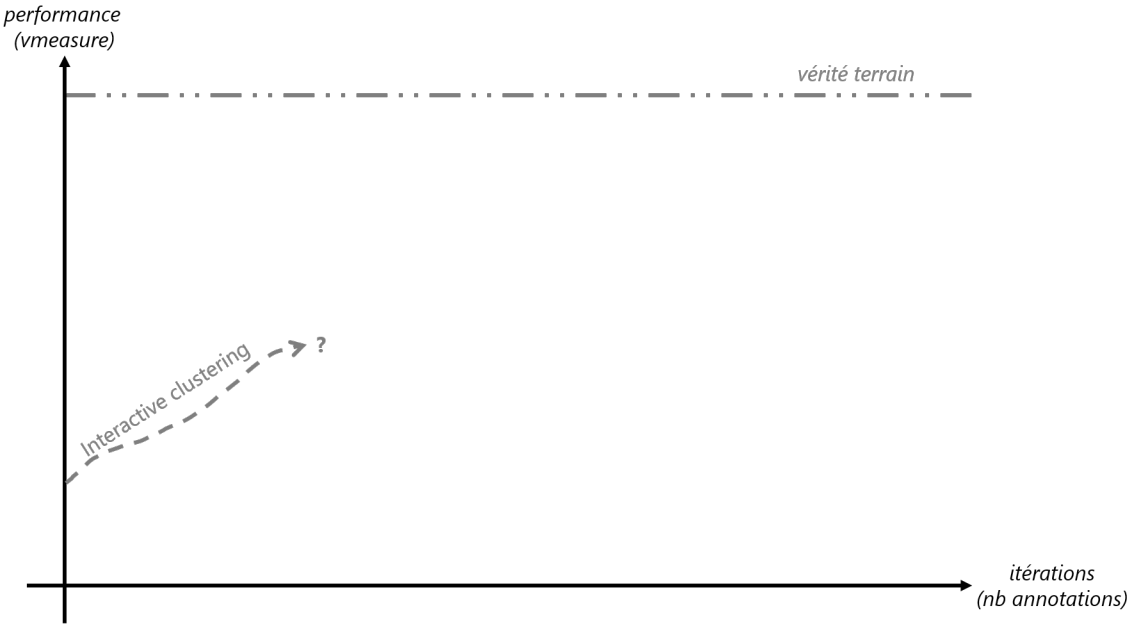

FIGURE 4.1 – *Illustration des études réalisées sur le* `Clustering Interactif` *(étape 0/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en v-measure) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations par un expert métier).*

17. `v-measure` : pour plus de détails sur cette mesure de qualité d'un *clustering*, consulter l'ANNEXE D.

Pour information :

— Les jeux de données utilisés pour ces études sont détaillés en Annexe A.

— Les implémentations du `Clustering Interactif` sont détaillées en Annexe C.

— Les scripts d'exécution et d'analyse de ces expériences, rédigés au sein de *notebooks* `Python` (Van Rossum et Drake, 2009) ou de scripts `R` (R Core Team, 2017), sont disponibles dans Schild, 2022c.

— Les exécutions des différentes expériences ont été réalisées sur des CPU `Intel(R) Xeon(R) CPU E5-2660 v4  2.00GHz` et parallélisées grâce à la librairie `multiprocessing`[18](utilisant un worker par CPU)

Sommaire

18. https://pypi.org/project/multiprocessing/

4.1 Évaluation de l'hypothèse d'efficacité

En premier lieu et afin de poser les bases de nos études, nous devons nous demander si notre implémentation du `Clustering Interactif` est fonctionnel et si elle permet d'atteindre son objectif. Nous aimerions donc vérifier l'hypothèse suivante :

Hypothèse d'efficacité

« **Une méthodologie d'annotation basée sur le `Clustering Interactif` permet d'obtenir une base d'apprentissage pour un assistant conversationnel qui respecte la vision donnée par l'expert métier au cours de l'annotation.** »

La FIGURE 4.2 illustre cette hypothèse et la perspective de convergence d'une base d'apprentissage en cours de construction vers sa vérité terrain.

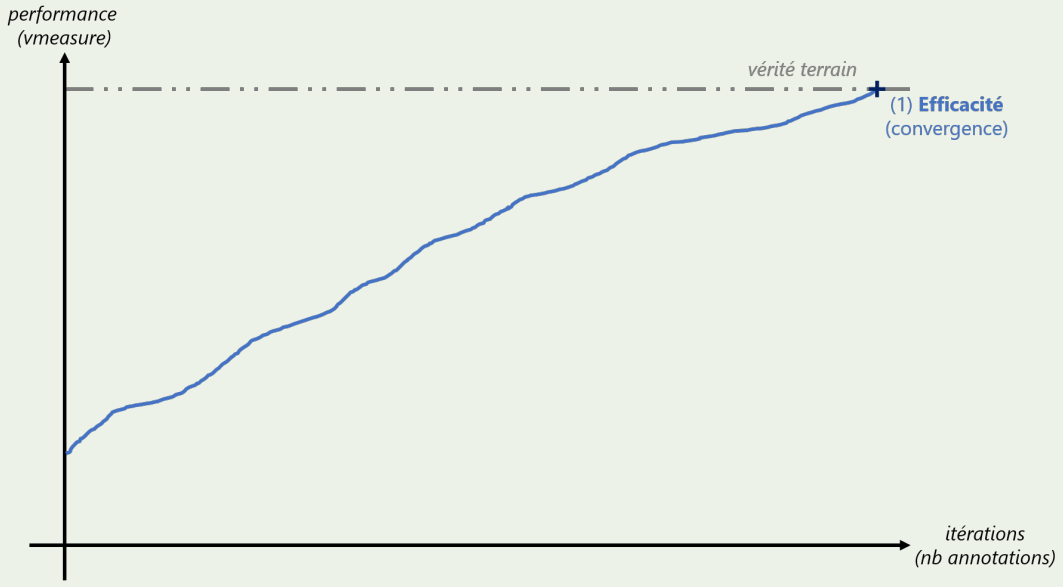

FIGURE 4.2 – *Illustration des études réalisées sur le `Clustering Interactif` (étape 1/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en v-measure) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations par un expert métier).*

Afin de vérifier cette hypothèse, nous mettons en place une **expérience de ré-annotation** d'une base d'apprentissage (qui servira ici de vérité terrain) à l'aide de notre méthode, en simulant l'annotation d'un expert, et nous critiquons l'évolution de la nouvelle base d'apprentissage obtenue ainsi que sa similitude avec la base d'apprentissage initiale (cf. SECTION 4.1.1).

4.1.1 Étude de convergence vers une vérité terrain préétablie en simulant l'annotation d'une base d'apprentissage et mesurant la vitesse de sa création

Nous voulons vérifier qu'une méthodologie d'annotation basée sur notre implémentation du `Clustering Interactif` permet de créer une base d'apprentissage pour un assistant conversa-

tionnel. Pour cela, nous prenons une base d'apprentissage employée pour entraîner un modèle de classification de textes, et nous utilisons ce jeu de données comme vérité terrain. L'objectif de cette expérience est de simuler la création de cette base d'apprentissage et de nous assurer que le résultat obtenu correspond à la vérité terrain.

> **Pour information :** Cette étude a été l'objet d'une présentation à la conférence `EGC (Extraction et Gestion des Connaissances)` (Schild, Durantin, Lamirel et Miconi, 2021), et d'une extension dans le journal `IJDWM (International Journal of Data Warehousing and Mining)` (Schild et al., 2022). Les résultats et la discussion de ces articles ont été mis à jour pour mieux s'intégrer au discours de ce manuscrit.

4.1.1.a Protocole expérimental

> **Attention :** Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Cependant, cette hypothèse forte n'est pas toujours vérifiée en pratique, surtout lorsque l'on manipule des données non structurées. L'impact de ce point sur les résultats obtenus est discuté en fin de partie, et nous nous y intéressons plus en détails dans la Section 4.6 (hypothèse de robustesse).

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.1.

Nous utilisons comme vérité terrain le jeu de données `Bank Cards (v1.0.0)` : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'Annexe A.1.

Lors de cette expérience, chaque tentative de la méthode commencera sur la version non labellisée de la vérité terrain à disposition, sans aucune contrainte connue à l'avance. À chaque itération de la méthode, nous simulons l'annotation de l'expert métier en comparant les labels de la vérité terrain : ainsi, deux données ont une contrainte `MUST-LINK` si elles ont le même label, et une contrainte `CANNOT-LINK` sinon. Cela traduit le prérequis d'avoir un annotateur qui soit capable, dans son domaine d'expertise, de différencier deux données selon leur cas d'usage. Une tentative de l'application de notre méthode s'arrête lorsque toutes les contraintes possibles entre les données ont été annotées par l'expert.

Pour cette étude, nous essayons une tentative pour chaque combinaison de paramètres de notre implémentation du `Clustering Interactif`. Cette implémentation est détaillée en Annexe C.1. Cela comprend les tâches et leurs paramètres respectifs suivants :

1. les **prétraitements** des données, avec les niveaux suivants : **aucun** (noté `prep.no`), **simple** (supprimant les minuscules, la ponctuation, les accents et les espaces blancs ; noté `prep.simple`), avec **lemmatisation** (noté `prep.lemma`) et avec **filtres** (supprimer les mots trop éloignés de la racine de l'arbre de dépendances syntaxiques ; noté `prep.filter`) ;

2. la **vectorisation** des données, avec les niveaux suivants : **TF-IDF** (noté `vect.tfidf`) et **SpaCy** (noté `vect.frcorenewsmd`) ;

Données : jeu de données annotées (vérité terrain)
Entrées : combinaisons d'algorithmes et de paramètres à tester
pour chaque *combinaison d'algorithmes et de paramètres à tester* **faire**
  initialisation (données) : récupérer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données ;
  vectorisation : transformer les données en vecteurs ;
  clustering initial : regrouper les données par similarité des vecteurs ;
  évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  répéter
    échantillonnage : sélectionner de nouvelles contraintes à annoter ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering : regrouper les données par similarité avec les contraintes ;
    évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  jusqu'à *annotation de toutes les contraintes possibles*;
  évaluation finale : espérer avoir un score d'équivalence de 100% avec la vérité terrain ;
Résultat : algorithmes et paramètres ayant un score d'équivalence de 100%

ALGORITHME 4.1 – *Description en pseudo-code du protocole expérimental de l'étude de convergence du `Clustering Interactif` vers une vérité terrain préétablie.*

3. le **clustering sous contraintes** des données, avec les niveaux suivants : **KMeans** (modèle *COP* noté `clust.kmeans.cop`), **Hiérarchique** (lien *single* noté `clust.hier.sing` ; lien *complete* noté `clust.hier.comp` ; lien *average* noté `clust.hier.avg` ; lien *ward* noté `clust.hier.ward`) et **Spectral** (modèle *SPEC* noté `clust.spec`). Le choix du nombre de *clusters* n'est pas étudié ici, et ce nombre est fixé au nombre de classes présentes dans la vérité terrain ;

4. l'**échantillonnage** des contraintes à annoter, avec les niveaux suivants : paires de données **purement aléatoires** (noté `samp.random.full`), paires de données **pseudo-aléatoires** (sélectionnant des données provenant d'un même *cluster* ; noté `samp.random.same`), paires de données **issues d'un même *cluster* et étant les plus éloignées** (noté `samp.farthest.same`) et paires de données **issues de *clusters* différents et étant les plus proches** (noté `samp.closest.diff`). Le choix de la taille d'échantillon n'est pas étudié ici, et cette taille est arbitrairement fixée à 50 [19].

> **Notes de l'auteur :** Il est difficile de discuter de l'influence du paramétrage du nombre de *clusters* : en effet, dans notre étude, nous désirons évaluer la capacité de notre méthode à atteindre une vérité terrain théorique ; or chercher un nombre de *clusters* différent du nombre de classes reviendrait à remettre en cause la vérité terrain que nous

19. Une taille d'échantillon de 50 contraintes à annoter semble *a priori* un bon compromis entre (1) ne pas donner trop de travail à un annotateur en une session et (2) donner suffisamment de nouvelles contraintes au *clustering* pour proposer un partitionnement plus pertinent des données. Ce choix sera discuté en fin de partie, et nous nous y intéresserons davantage dans la SECTION 4.3 (hypothèse sur les coûts).

essayons d'atteindre, nous empêchant donc d'évaluer la capacité de notre méthode... Devant ce cercle vicieux, nous laissons l'étude de ce problème ouvert à de futures études.

Il y a donc 192 combinaisons testées, et chaque tentative est répétée 5 fois pour contrer les aléas statistiques des algorithmes de *clustering* (*initialisation du* `clust.kmeans.cop`, ...) et d'échantillonnage (*choix des contraintes au hasard avec* `samp.random.full`, ...).

Pour évaluer l'équivalence entre la vérité terrain et notre segmentation des données obtenue au cours de la méthode, nous nous intéressons à l'évolution de la `v-measure` entre ces deux jeu de données. Si le score du calcul de la `v-measure` est de 100%, cela signifierait que le *clustering* final et la vérité terrain proposent une segmentation identique des données, donc que la vérité terrain a pu être retrouvée, et donc qu'il est possible d'obtenir une base d'apprentissage pour un assistant conversationnel à l'aide d'une méthodologie d'annotation basée sur le `Clustering Interactif`.

Pour information : Les scripts de l'expérience (*notebooks* `Python` (Van Rossum et Drake, 2009)) sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.1.1.b Résultats obtenus

La Figure 4.3 et la Table 4.1 représentent l'évolution moyenne de la `v-measure` du *clustering* en fonction du nombre d'itérations de la méthode. Les tentatives les plus rapides et les plus lentes sont représentées sur la figure.

Malgré une forte dispersion des résultats (écart-type de `v-measure` pouvant être supérieur à 20%, forte différence entre la tentative la plus rapide et la plus lente) et quelques sauts de performances (cf. à-coups de la tentative la plus lente sur la figure), une convergence générale vers la vérité terrain peut être constatée.

À l'itération 0, une tentative commence avec une moyenne de 19.05% de `v-measure` entre son *clustering* initial (sans contrainte) et la vérité terrain. Cette `v-measure` moyenne croît presque linéairement (pente de 0.97) jusqu'à l'itération 75 où elle atteint la performance de 92.08% (cf. Table 4.1).

Au delà de l'itération 75, la courbe de la `v-measure` moyenne tend vers une asymptote de 100% (cf. Figure 4.3). Cette asymptote est atteinte par toutes les 960 tentatives (192 combinaisons de paramètres, 5 tentatives pour chaque combinaison), la tentative l'ayant atteinte le plus tôt à l'itération 19 et celle le plus tard à l'itération 328.

La courbe se prolonge jusqu'à l'itération 393 pour que toutes les contraintes possibles sur le jeu de données puissent être annotées. Nous pouvons aussi noter que 756 tentatives (78.75%) convergent vers 100% de `v-measure` avant l'annotation exhaustive de toutes les contraintes : sur ces tentatives, la convergence peut ainsi s'observer en moyenne avec seulement 91.30% du nombre de contraintes possibles (min : 8.72%, max : 99.69%, écart-type : 18.60%).

4.1.1.c Discussion

Au regard des résultats décrits ci-dessus, les différentes simulations de la méthode ont bien convergé vers la vérité terrain (atteinte de l'asymptote à 100% de `v-measure`). Cette expérience

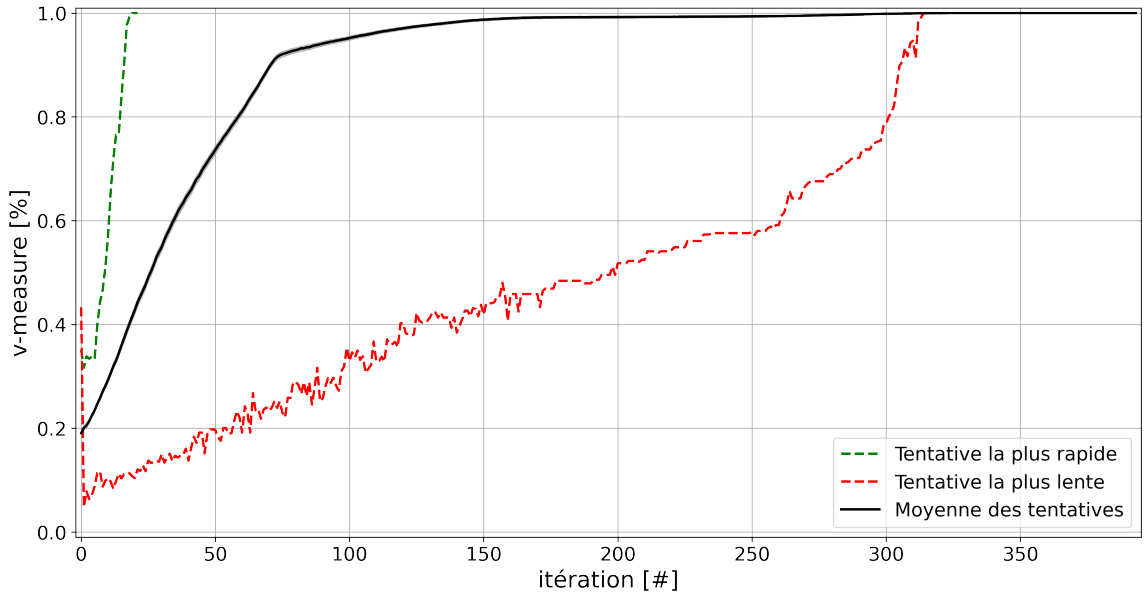

FIGURE 4.3 – *Évolution de la moyenne de la* `v-measure` *entre un résultat obtenu et la vérité terrain en fonction du nombre d'itérations de la méthode de* `Clustering Interactif`*, moyenne réalisée itération par itération sur l'ensemble des tentatives. Représentation des tentatives ayant été les plus rapides (des prétraitements* `prep.simple`*, une vectorisation* `vect.tfidf`*, un clustering* `clust.hier.comp` *ou* `clust.hier.ward`*, et un échantillonnage* `samp.closest.diff`*) et les plus lentes (un prétraitement* `prep.no`*, une vectorisation* `vect.tfidf`*, un clustering* `clust.spec`*, et un échantillonnage de contraintes* `samp.farthest.same`*) pour atteindre* 100 % *de* `v-measure`*.*

Annotations		Performances (`v-measure`)			
Itérations	Contraintes	Moyenne	Écart-type	Minimum	Maximum
0	0	19.05% (±0.43)	13.38%	03.42%	47.75%
25	1 250	49.09% (±0.82)	25.43%	09.09%	100.00%
50	2 500	73.66% (±0.77)	23.98%	16.78%	100.00%
75	3 750	92.08% (±0.54)	16.70%	21.74%	100.00%
100	5 000	95.19% (±0.41)	12.67%	26.93%	100.00%
125	6 250	97.43% (±0.29)	09.09%	34.99%	100.00%
150	7 500	98.73% (±0.23)	07.22%	38.14%	100.00%
328	16 400	100.00% (±0.00)	0.00%	100.00%	100.00%
394	19 700	100.00% (±0.00)	0.00%	100.00%	100.00%

TABLE 4.1 – *Détails de l'évolution de la moyenne de la* `v-measure` *entre un résultat obtenu et la vérité terrain en fonction du nombre d'itérations de la méthode de* `Clustering Interactif`*, moyenne réalisée itération par itération sur l'ensemble des tentatives.*

permet donc de confirmer plusieurs espoirs portés sur la méthode.

Tout d'abord, la vérité terrain a été retrouvée sans formaliser concrètement la structure de données. Là où une annotation par label aurait requis au préalable une définition des catégories possibles pour les données à étiqueter (création d'un "*type system*"), la méthodologie employant le `Clustering Interactif` a permis de faire émerger naturellement cette structure de données.

Cette émergence provient directement des contraintes annotées par l'expert métier, traduisant ainsi ses connaissances à l'aide d'instructions simples : *les données sont-elles ou non similaires ?* Cela représente un net avantage pour l'opérateur qui n'a ainsi pas à maintenir en mémoire une modélisation complexe de la structure de données, rendant la tâche d'annotation plus accessible.

D'autre part, ces contraintes ont fait l'objet d'une annotation guidée par les besoins de la machine afin de s'améliorer d'itération en itération (voir la croissance globale de la `v-measure` sur la FIGURE 4.3). Ainsi, l'expert métier corrige la base d'apprentissage à chaque itération : soit en affinant les *clusters* en cours de construction, améliorant ainsi la cohérence des *clusters* (cf. pentes croissantes) ; soit en remaniant les *clusters* mal formés pour repartir sur de bonnes bases, détériorant la cohérence des *clusters* le temps de la réorganisation (cf. oscillations ou pentes décroissantes). Une telle assistance limite ainsi le nombre de contraintes non utiles au *clustering*, même si certains paramétrages semblent plus efficaces que d'autres (voir la forte dispersion des résultats).

De plus, nous remarquons que 76.75% des tentatives convergent vers la vérité terrain sans bénéficier d'une annotation exhaustive de toutes les contraintes possibles. Cela montre l'intérêt des interactions Homme/Machine afin d'obtenir plus efficacement un résultat qu'un expert métier (aussi parfait soit-il) aurait obtenu seul. L'intérêt serait maintenant de déterminer la meilleure combinaison de paramétrages demandant d'annoter un nombre **suffisant** de contraintes afin d'obtenir ce même résultat de la manière la plus efficiente (cf. SECTION 4.2) et la plus robuste aux erreurs d'annotation (cf. SECTION 4.6).

Néanmoins, différentes pistes sont encore à explorer pour rendre le `Clustering Interactif` utilisable en situation réelle.

D'une part, nous échangeons le besoin de définir une structure de données contre la nécessité d'annoter un grand nombre de contraintes : pour 500 points de données, et en considérant que l'asymptote à 100% est atteinte en moyenne autour de l'itération 200, il faudrait 10 000 annotations de contraintes pour être exhaustif, ce qui correspond à près de 20 fois plus de contraintes que de données. Bien que l'annotation binaire demande *a priori* une charge mentale plus faible (HART et STAVELAND, 1988) et que l'opérateur n'a pas besoin de définir ou de maintenir en mémoire une structure de données complexe, un tel volume d'annotation représente tout de même une grande quantité de travail. Cela peut décourager les experts métiers en début de projet, surtout pour des projets ayant des jeux de données de plus grande taille. Toutefois, les résultats obtenus montrent une forte dispersion du nombre d'itérations nécessaire, et certaines tentatives ont été bien plus efficientes dans l'utilisation de leurs contraintes. La tentative la plus rapide a convergé à l'itération 19, soit 950 contraintes, ce qui est un volume d'annotation bien plus abordable ! Nous pouvons donc espérer trouver un paramétrage optimal de la méthode permettant de diminuer significativement le nombre moyen de contraintes nécessaires afin d'obtenir une base d'apprentissage exploitable avec un volume d'annotations acceptable. Cet aspect fait l'objet de l'étude décrite dans la SECTION 4.2 (hypothèse d'efficience).

D'autre part, le choix d'annoter toutes les contraintes possibles sur les données (**annotation exhaustive**) n'est pas forcément judicieux. En effet, si nous regardons la FIGURE 4.3, une moyenne de 90% de `v-measure` est déjà atteinte autour de l'itération 75, alors que l'asymptote à 100% n'est atteinte qu'au delà de l'itération 200. Afin d'être plus efficient, il faudrait envisager une **annotation partielle** permettant d'obtenir rapidement ces 90% de `v-measure` (cf. coude sur la FIGURE 4.3), quitte à affiner le résultat manuellement pour combler la "perte" moyenne de 10% de `v-measure`. Cet aspect sera ajouté à l'objectif de l'étude décrite dans la SECTION 4.2 (hypothèse d'efficience).

Pour finir, nous avons supposé dans cette étude que l'annotateur est un expert métier connais-

sant parfaitement le domaine traité. Cette hypothèse forte n'est *a priori* pas valable en situation réelle : En effet, des différences d'annotations peuvent intervenir (*ambiguïtés sur les données, méconnaissance du domaine, erreurs d'inattention, différence d'opinions entre annotateurs, ...*), ce qui peut entraîner des divergences ou des incohérences dans la construction de la base d'apprentissage. Il semble donc nécessaire d'étudier les impacts de ces incohérences, ainsi que de proposer une méthode pour les prévenir ou les corriger. Cet aspect sera traité à la fin de ce chapitre dans la SECTION 4.6 (hypothèse de robustesse).

4.2 Évaluation de l'hypothèse d'efficience

Suite à la validation de l'hypothèse d'efficacité (convergence de la méthode, cf. Section 4.1), nous voulons déterminer les paramètres optimaux de la méthode afin de converger le plus rapidement vers la vérité terrain. Nous aimerions donc vérifier l'hypothèse suivante :

Hypothèse d'efficience

« **La vitesse de convergence du `Clustering Interactif` peut être optimisée en ajustant différents paramètres afin de minimiser la charge de travail de l'opérateur. Nous étudions en particulier l'influence sur le nombre de contraintes requis des prétraitements des données, de la vectorisation des données, de l'échantillonnage des contraintes à annoter et du *clustering* sous contraintes.** »

La Figure 4.4 illustre cette hypothèse et la perspective d'une convergence "optimale" d'une base d'apprentissage en cours de construction vers sa vérité terrain.

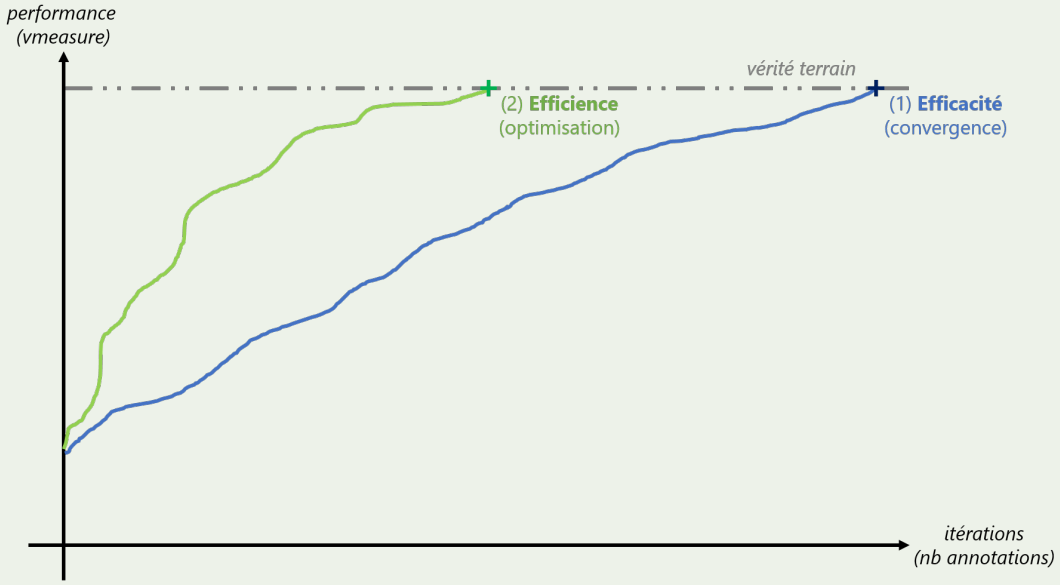

Figure 4.4 – *Illustration des études réalisées sur le `Clustering Interactif` (étape 2/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en v-measure) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations par un expert métier).*

Afin de vérifier cette hypothèse, nous mettons en place une expérience de ré-annotation d'une base d'apprentissage (qui servira ici de vérité terrain) à l'aide de notre méthode, en simulant l'annotation d'un expert, et nous réalisons l'analyse statistique de la taille d'effets de différents paramètres sur la **vitesse de convergence** du *clustering* itératif (cf. Section 4.2.1).

4.2.1 Étude d'optimisation des paramètres d'implémentation en analysant leurs tailles d'effets sur la vitesse de création d'une base d'apprentissage

Nous voulons étudier l'influence des paramètres de notre implémentation du `Clustering Interactif` sur la vitesse de création d'une base d'apprentissage pour un assistant conversa-

tionnel. Nous allons donc compléter le protocole expérimental de l'étude de convergence en Section 4.1.1 visant à simuler la création d'une base d'apprentissage.

> **Pour information :** Cette étude a été l'objet d'une présentation à la conférence `EGC (Extraction et Gestion des Connaissances)` (Schild, Durantin, Lamirel et Miconi, 2021), et d'une extension dans le journal `IJDWM (International Journal of Data Warehousing and Mining)` (Schild et al., 2022). Les résultats et la discussion de ces articles ont été mis à jour pour mieux s'intégrer au discours de ce manuscrit.

4.2.1.a Protocole expérimental

> **Attention :** Comme dans l'étude précédente, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Cependant, cette hypothèse forte n'est pas toujours vérifiée en pratique, surtout lorsque l'on manipule des données non structurées. L'impact de ce point sur les résultats obtenus est discuté en fin de partie, et nous nous y intéressons plus en détails dans la Section 4.6 (hypothèse de robustesse).

Pour résumer le protocole expérimental adapté, une description en pseudo-code est disponible dans l'Algorithme 4.2.

Données : jeu de données annotées (vérité terrain)
Entrées : combinaisons d'algorithmes et de paramètres à tester
pour chaque *combinaison d'algorithmes et de paramètres à tester* **faire**
  initialisation (données) : récupérer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données ;
  vectorisation : transformer les données en vecteurs ;
  clustering initial : regrouper les données par similarité des vecteurs ;
  évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  répéter
    échantillonnage : sélectionner de nouvelles contraintes à annoter ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering : regrouper les données par similarité avec les contraintes ;
    évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  jusqu'à *annotation de toutes les contraintes possibles*;
analyse : déterminer les tailles d'effets des algorithmes et paramètres ;
Résultat : meilleures combinaisons d'algorithmes et de paramètres

Algorithme 4.2 – *Description en pseudo-code du protocole expérimental de l'étude d'optimisation de la convergence du `Clustering Interactif` vers une vérité terrain pré-établie.*

En s'appuyant sur les résultats précédemment obtenus, nous allons analyser l'influence des différentes tâches employées (**prétraitements**, **vectorisation**, **clustering sous contraintes**,

échantillonnage) et de leurs paramètres sur la vitesse de convergence vers la vérité terrain. Nous utilisons à nouveau le jeu de données `Bank Cards (v1.0.0)` (cf. Annexe A.1) comme vérité terrain, sur lequel nous testons 192 combinaisons de paramétrages de notre implémentation du `Clustering Interactif` (voir Annexe C.1), et chaque tentative est répétée 5 fois pour contrer les aléas statistiques de certains algorithmes.

Comme lors de l'étude sur la convergence de la méthode, nous nous intéressons à l'évolution de la `v-measure` entre la vérité terrain et notre segmentation des données obtenue, et nous affinons notre évaluation en portant attention aux trois seuils d'annotations suivants :

1. le cas d'une **annotation partielle**, correspondant au nombre d'itérations nécessaire à la méthode pour avoir 90% de `v-measure`, c'est-à-dire un état de semi-parcours vers une convergence totale [20] ;
2. le cas d'une **annotation suffisante**, correspondant au nombre d'itérations nécessaire à la méthode pour avoir 100% de `v-measure`, c'est-à-dire avoir suffisamment de contraintes annotées par l'expert métier pour retrouver la vérité terrain ;
3. le cas d'une **annotation exhaustive**, correspondant au nombre d'itérations nécessaire à la méthode pour parcourir toutes les contraintes possibles sur les données, et ainsi retranscrire exhaustivement la vision de l'expert métier [21].

Enfin, nous utilisons une `ANOVA` à mesures répétées (Girden, 1992) afin de déterminer l'effet des paramètres de notre implémentation sur le nombre d'annotations requis pour converger vers la vérité terrain. Le test de `Tukey (HSD)` (Tukey, 1949) est utilisé pour les comparaisons post-hoc.

> **Pour information :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009) et des scripts `R` (R Core Team, 2017), sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.2.1.b Résultats obtenus

Pour obtenir une **annotation partielle** (*atteindre une `v-measure` de* 90%), la moyenne des itérations est de 59.04 (min : 11, max : 315, écart-type : 42.14), soit une moyenne de 2 951.81 annotations (min : 550, max : 15 750, écart-type : 2 106.72). La Figure 4.5 représente la répartition de ces itérations au cours des différentes tentatives. Nous pouvons noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec des prétraitements des données `prep.no` ou `prep.simple` ou `prep.lemma`, une vectorisation des données `vect.tfidf`, un *clustering* sous contraintes `clust.hier.sing`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 11 itérations, soit 550 annotations, dont 299 (respectivement 304 et 281) contraintes `MUST-LINK`.

20. Le seuil de 90% a été choisi au cours de l'étude de convergence (cf. hypothèse d'efficacité, Section 4.1, coude de la Figure 4.3).

21. Une annotation est *a priori* inutilisable en pratique (demande trop de contraintes, cf. hypothèse d'efficacité, Section 4.1), nous l'étudions toutefois pour avoir un point de comparaison.

- Les tentatives les plus lentes furent celles avec des prétraitements des données `prep.no`, une vectorisation des données `vect.tfidf`, un *clustering* sous contraintes `clust.spec`, et un échantillonnage de contraintes `samp.farthest.same`. Ces tentatives ont requis 315 itérations, soit 15 750 annotations, dont 1 032 contraintes `MUST-LINK`.

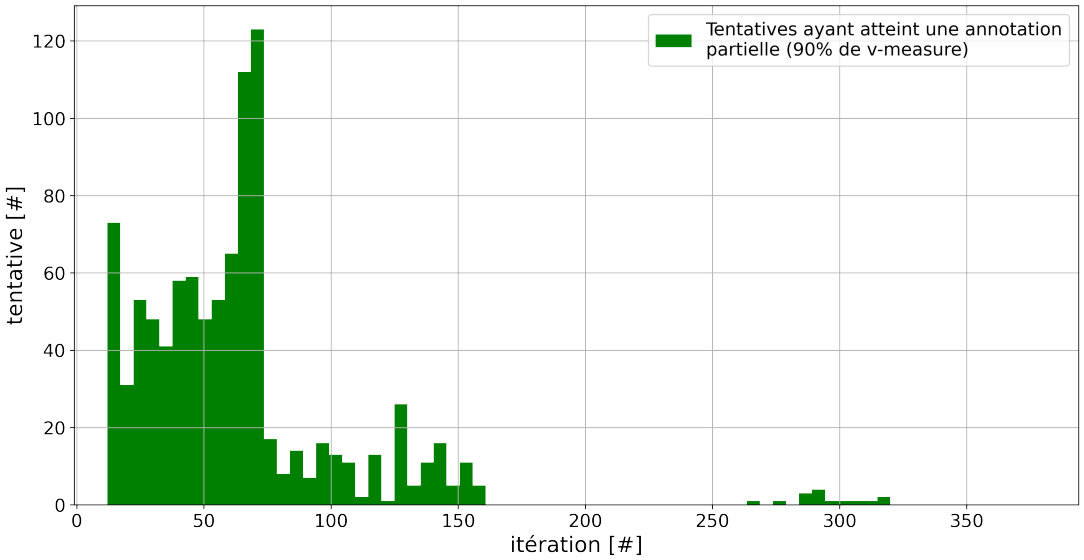

FIGURE 4.5 – *Répartition des tentatives en fonction de l'itération de la méthode pour laquelle elles atteignent le seuil d'une annotation partielle, c'est-à-dire l'itération pour laquelle elles parviennent à 90% de `v-measure` entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics pour simplifier l'affichage.*

La TABLE 4.2 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaire pour atteindre une **annotation partielle** (*atteindre une `v-measure` de* 90%). Les analyses de variance mettent en relief l'effet significatif sur cette convergence des prétraitements (`eta-carré` : 0.320, `p-valeur` : $< 10^{-3}$), de la vectorisation (`eta-carré` : 0.388, `p-valeur` : $< 10^{-3}$), du *clustering* (`eta-carré` : 0.866, `p-valeur` : $< 10^{-3}$) et de l'échantillonnage (`eta-carré` : 0.968, `p-valeur` : $< 10^{-3}$). L'analyse post-hoc de ces effets indique que le meilleur paramétrage moyen pour atteindre une **annotation partielle** repose sur les prétraitements `prep.simple`, la vectorisation `vect.tfidf`, le *clustering* `clust.hier.avg`, et l'échantillonnage `samp.closest.diff`. La moyenne du nombre d'itérations requises pour ce paramétrage est de 19.00 (écart-type : 0.79), soit 950 annotations (écart-type : 39.34).

Pour obtenir une **annotation suffisante** (*atteindre une `v-measure` de* 100%), la moyenne des itérations est de 76.29 (min : 19, max : 328, écart-type : 46.44), soit une moyenne de 3 801.19 annotations (min : 950, max : 16 400, écart-type : 2 314.91). La FIGURE 4.6 représente la répartition de ces itérations au cours des différentes tentatives. Nous pouvons noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec des prétraitements des données `prep.simple`, une vectorisation des données `vect.tfidf`, un *clustering* sous contraintes `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 19 itérations, soit 950 annotations, dont 638 (respectivement 641) contraintes `MUST-LINK`.

Description des facteurs analysés		Description statistique des itérations			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	p-valeur
prétraitements	prep.simple	61.90	(1)	0.32	0.320	$< 10^{-3}$ $(***)$
	prep.lemma	63.08	(2)			
	prep.no	63.70	(2)			
	prep.filter	71.90	(4)			
vectorisation	vect.tfidf	60.61	(1)	0.29	0.388	$< 10^{-3}$ $(***)$
	vect.frcorenewsmd	63.08	(2)			
clustering	clust.hier.avg	50.64	(1)	0.35	0.866	$< 10^{-3}$ $(***)$
	clust.kmeans.cop	52.43	(2)			
	clust.hier.sing	54.08	(3)			
	clust.hier.ward	72.41	(4)			
	clust.hier.comp	73.48	(5)			
	clust.spec	87.84	(6)			
échantillonnage	samp.closest.diff	33.66	(1)	0.32	0.968	$< 10^{-3}$ $(***)$
	samp.random.same	48.24	(2)			
	samp.random.full	65.83	(3)			
	samp.farthest.same	112.86	(4)			

Table 4.2 – *ANOVA du nombre d'itérations nécessaire pour l'obtention de 90% de v-mesure. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne* `Moyenne` *indiquent le classement des niveaux selon les analyses post-hoc.*

- Les tentatives les plus lentes furent celles avec des prétraitements des données `prep.no`, une vectorisation des données `vect.tfidf`, un *clustering* sous contraintes `clust.spec`, et un échantillonnage de contraintes `samp.farthest.same`. Ces tentatives ont requis 394 itérations, soit 16 400 annotations, dont 1 309 contraintes `MUST-LINK`.

La Table 4.3 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaire pour atteindre une **annotation suffisante**. Les analyses de variance mettent en relief l'effet significatif sur cette convergence des prétraitements (`eta-carré` : 0.987, `p-valeur` : $< 10^{-3}$), de la vectorisation (`eta-carré` : 0.991, `p-valeur` : $< 10^{-3}$), du *clustering* (`eta-carré` : 0.997, `p-valeur` : $< 10^{-3}$) et de l'échantillonnage (`eta-carré` : 0.998, `p-valeur` : $< 10^{-3}$). L'analyse post-hoc de ces effets indique que le meilleur paramétrage moyen pour atteindre une **annotation suffisante** repose sur les prétraitements `prep.lemma`, la vectorisation `vect.tfidf`, le *clustering* `clust.kmeans.cop`, et l'échantillonnage `samp.closest.diff`. La moyenne du nombre d'itérations requises pour ce paramétrage est de 34.60 (écart-type : 7.44), soit 1 730 annotations (écart-type : 372.00).

Enfin, pour avoir une **annotation exhaustive** (*annoter toutes les contraintes possibles*), la moyenne des itérations est de 88.98 (min : 20, max : 394, écart-type : 68.21), soit une moyenne de 4 431.34 annotations (min : 1 000, max : 19 656, écart-type : 3 405.16). La Figure 4.7 représente la répartition de ces itérations au cours des différentes tentatives. Nous pouvons noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec des prétraitements des données `prep.no` ou

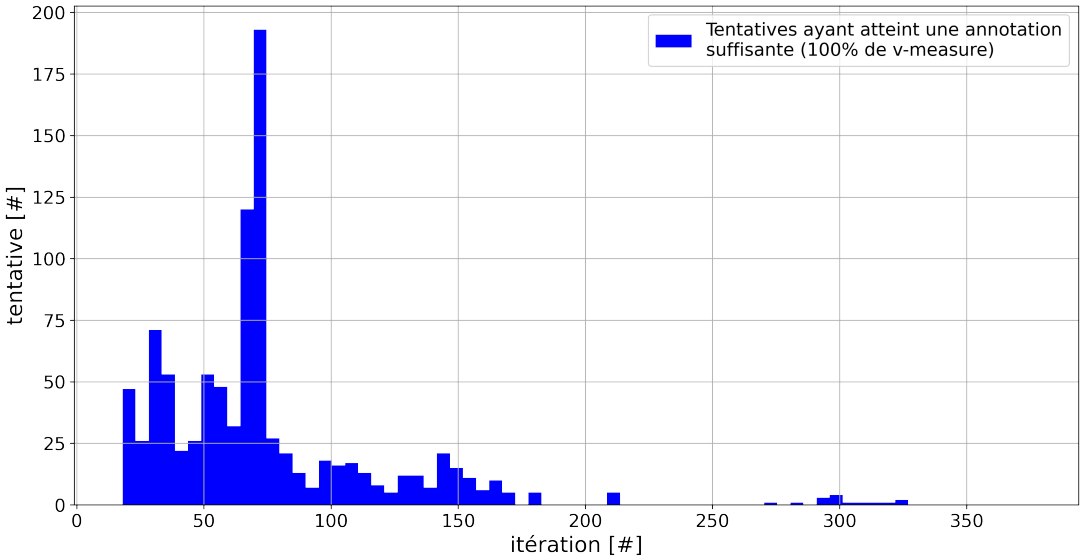

Figure 4.6 – *Répartition des tentatives en fonction de l'itération de la méthode pour laquelle elles atteignent le seuil d'une annotation suffisante, c'est-à-dire l'itération pour laquelle elles parviennent à 100% de* `v-measure` *entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics pour simplifier l'affichage.*

Description des facteurs analysés		Description statistique des itérations			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	`p-valeur`
prétraitements	`prep.lemma`	72.86	(1)	0.32	0.276	$< 10^{-3}$ $(***)$
	`prep.simple`	73.30	(2)			
	`prep.no`	75.24	(2)			
	`prep.filter`	83.77	(4)			
vectorisation	`vect.tfidf`	71.16	(1)	0.36	0.366	$< 10^{-3}$ $(***)$
	`vect.frcorenewsmd`	81.43	(2)			
clustering	`clust.kmeans.cop`	62.23	(1)	0.42	0.700	$< 10^{-3}$ $(***)$
	`clust.hier.avg`	65.13	(2)			
	`clust.hier.sing`	75.44	(3)			
	`clust.hier.ward`	80.44	(4)			
	`clust.hier.comp`	81.46	(5)			
	`clust.spec`	93.06	(6)			
échantillonnage	`samp.closest.diff`	50.29	(1)	0.39	0.950	$< 10^{-3}$ $(***)$
	`samp.random.same`	56.38	(2)			
	`samp.random.full`	71.95	(3)			
	`samp.farthest.same`	126.55	(4)			

Table 4.3 – *ANOVA du nombre d'itérations nécessaire pour l'obtention de 100% de v-mesure. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne* `Moyenne` *indiquent le classement des niveaux selon les analyses post-hoc.*

`prep.lemma`, une vectorisation des données `vect.tfidf`, un algorithme de *clustering* sous contraintes `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 20 itérations, soit 1 000 annotations, dont 653 (respectivement 668) contraintes `MUST-LINK`.

- Les tentatives les plus lentes furent celles avec des prétraitements des données `prep.simple`, une vectorisation des données `vect.frcorenewsmd`, un *clustering* `clust.hier.sing`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 394 itérations, soit 19 656 annotations, dont 682 contraintes `MUST-LINK`.

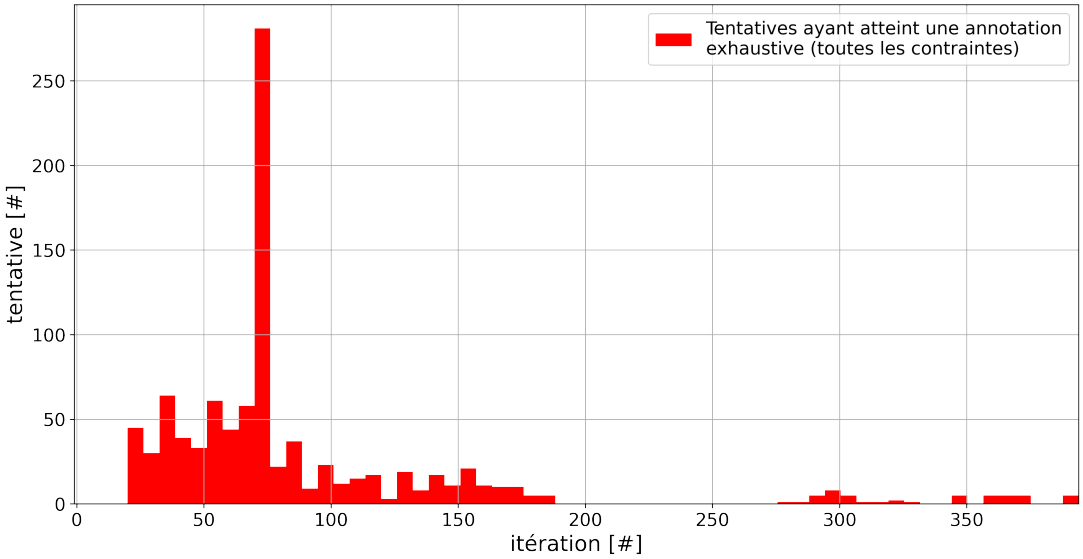

FIGURE 4.7 – *Répartition des tentatives en fonction de l'itération de la méthode pour laquelle elles atteignent le seuil d'une annotation exhaustive, c'est-à-dire l'itération pour laquelle toutes les contraintes possibles entre les données ont été annotées. L'histogramme est réduit à 60 pics pour simplifier l'affichage.*

La TABLE 4.4 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaire pour atteindre une **annotation exhaustive**. Les analyses de variance mettent en relief l'effet significatif sur cette convergence du prétraitement (`eta-carré` : 0.909, `p-valeur` : $< 10^{-3}$), de la vectorisation (`eta-carré` : 0.985, `p-valeur` : $< 10^{-3}$), du *clustering* (`eta-carré` : 0.999, `p-valeur` : $< 10^{-3}$) et de l'échantillonnage (`eta-carré` : 0.997, `p-valeur` : $< 10^{-3}$). L'analyse post-hoc de ces effets indique que le meilleur paramétrage moyen pour atteindre une **annotation exhaustive** repose sur les prétraitements `prep.lemma`, le vectorisation `vect.tfidf`, le *clustering* `clust.kmeans.cop`, et l'échantillonnage `samp.random.same`. La moyenne du nombre d'itérations requises pour ce paramétrage est de 32.60 (écart-type : 1.14), soit 1 630 annotations (écart-type : 57.00).

La FIGURE 4.8 représente les évolutions moyennes de la `v-measure` du *clustering* en fonction du nombre d'itérations de la méthode pour les différentes valeurs des facteurs analysés (prétraitements en haut à gauche, vectorisation en haut à droite, *clustering* en bas à gauche, échantillonnage en bas à droite). La FIGURE 4.9 représente cette même évolution pour les meilleurs paramétrages moyens destinés à atteindre les trois seuils d'annotation définis (partiel, suffisant, exhaustif), où nous constatons une baisse significative du nombre d'itérations nécessaire à la convergence par rapport à la moyenne des tentatives.

Description des facteurs analysés		Description statistique des itérations			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	`p-valeur`
prétraitements	`prep.lemma`	85.89	(1)	0.42	0.052	$< 10^{-3}$ $(***)$
	`prep.filter`	89.55	(2)			
	`prep.simple`	89.64	(2)			
	`prep.no`	90.81	(4)			
vectorisation	`vect.tfidf`	85.50	(1)	0.39	0.165	$< 10^{-3}$ $(***)$
	`vect.frcorenewsmd`	92.46	(2)			
clustering	`clust.kmeans.cop`	64.99	(1)	0.39	0.894	$< 10^{-3}$ $(***)$
	`clust.hier.avg`	78.54	(2)			
	`clust.hier.ward`	81.31	(3)			
	`clust.hier.comp`	82.49	(3)			
	`clust.spec`	93.78	(5)			
	`clust.hier.comp`	132.75	(6)			
échantillonnage	`samp.random.same`	57.23	(1)	0.42	0.930	$< 10^{-3}$ $(***)$
	`samp.random.full`	72.80	(2)			
	`samp.closest.diff`	98.38	(3)			
	`samp.farthest.same`	132.75	(4)			

Table 4.4 – *ANOVA du nombre d'itérations nécessaire pour annoter toutes les contraintes possibles. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne* `Moyenne` *indiquent le classement des niveaux selon les analyses post-hoc.*

4.2.1.c Discussion

L'objectif de l'étude est de trouver une implémentation "efficiente" du `Clustering Interactif` permettant d'obtenir une base d'apprentissage correctement annotée en un minimum d'annotations. Pour trouver si une telle implémentation existe et quels en sont les paramètres optimaux, nous avons analysé l'impact de différents paramétrages sur les tâches principales de la méthode (**prétraitements**, **vectorisation**, **clustering sous contraintes**, **échantillonnage**) en nous basant sur des simulations d'annotation d'un jeu de données.

Dans l'optique d'être efficient, nous excluons les tentatives d'annoter **exhaustivement** le jeu de données car la charge de travail estimée est trop importante. (cf. discussion de la Section 4.1 (hypothèse d'efficacité)) Nous préférons donc nous concentrer sur deux seuils d'annotation plus réalistes : celui d'une **annotation partielle** (atteindre 90% de `v-measure` avec la vérité terrain) et celui d'une **annotation suffisante** (atteindre 100% de `v-measure` avec la vérité terrain en un minimum de contraintes).

L'étude réalisée met en avant l'impact significatif des quatre tâches principales (**prétraitements**, **vectorisation**, ***clustering* sous contraintes**, **échantillonnage**) sur la vitesse de convergence de la méthode pour atteindre les seuils définis de 90% et 100% de `v-measure`. Il existe donc bien un paramétrage permettant d'optimiser l'implémentation proposée et de réduire le nombre de contraintes nécessaires à annoter :

1. pour une **annotation partielle** (90% de `v-measure`), le meilleur paramétrage moyen

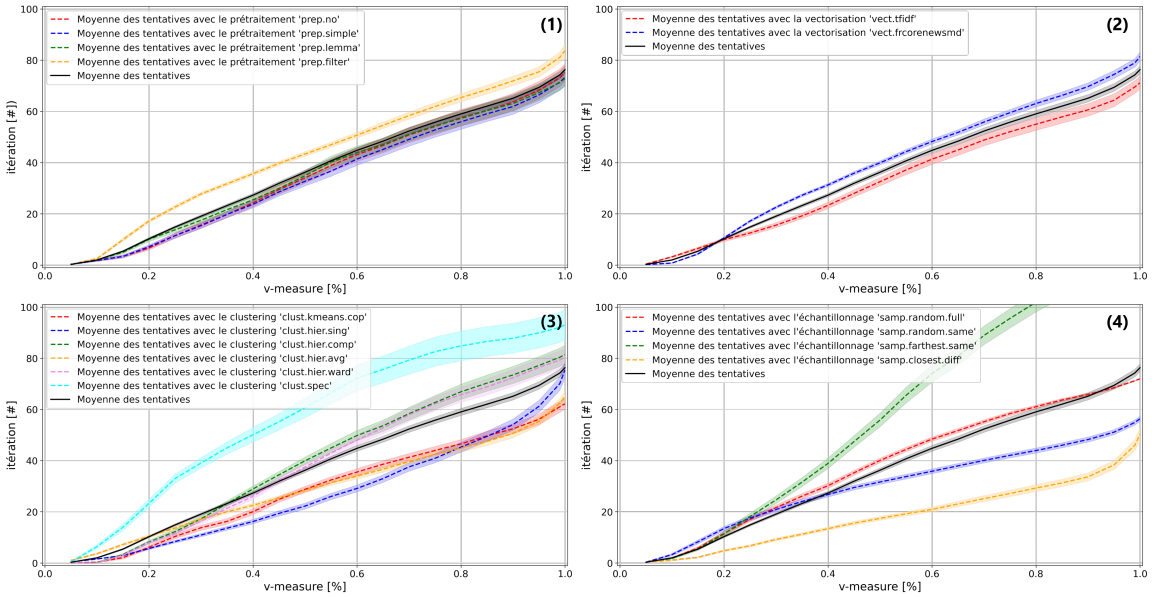

FIGURE 4.8 – *Évolution des moyennes du nombre d'itérations nécessaire de la méthode de* `Clustering Interactif` *pour obtenir un seuil défini de* `v-measure` *entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes valeurs que peuvent prendre les facteurs analysés, affichées par facteur : (1) prétraitements, (2) vectorisation, (3) clustering et (4) échantillonnage.*
Note : Le seuil d'annotation exhaustive (annoter toutes les contraintes possibles) n'étant pas exprimé en terme de `v-measure` *; ce seuil n'est pas affiché ici.*

est constitué des prétraitements simples (`prep.simple`), de la vectorisation `TF-IDF` (`vect.tfidf`), du *clustering* hiérarchique à lien moyen (`clust.hier.avg`) et de l'échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`). Avec ce paramétrage, il faut en moyenne 950 annotations de contraintes pour obtenir une `v-measure` de 90% ;

2. pour une **annotation suffisante** (100% de `v-measure`), le meilleur paramétrage moyen est constitué du prétraitement avec lemmatisation (`prep.lemma`), de la vectorisation `TF-IDF` (`vect.tfidf`), du *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et de l'échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`). Avec ce paramétrage, il faut en moyenne 1 750 annotations de contraintes pour obtenir une `v-measure` de 100% ;

3. le cas d'une **annotation exhaustive** (annoter toutes les contraintes possibles sur les données) n'est pas explicité ici mais peut se déduire des résultats décrits plus haut.

> **Notes de l'auteur :** Nous notons que les facteurs de prétraitements et de vectorisation sont significatifs mais ils possèdent toutefois de faibles valeurs de variance expliquée ($\eta^2 < 0.40$).
>
> Les facteurs de *clustering* et d'échantillonnage ont quant à eux des valeurs plus fortes ($\eta^2 > 0.70$), dénotant ainsi un plus grand pouvoir explicatif de la variance des résultats. Notre attention s'attarde particulièrement sur l'échantillonnage des données les plus

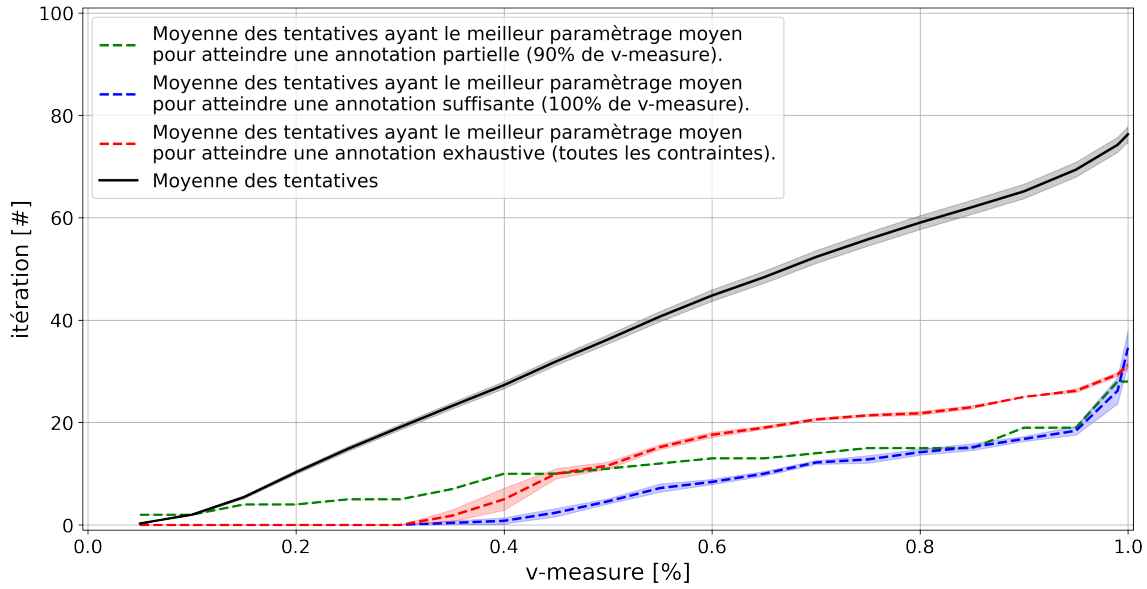

FIGURE 4.9 – *Évolution des moyennes du nombre d'itérations nécessaire de la méthode de* `Clustering Interactif` *pour obtenir un seuil défini de* `v-measure` *entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différents seuils d'annotations étudiés : l'annotation partielle (atteindre une* `v-measure` *de* 90%*), l'annotation suffisante (atteindre une* `v-measure` *de* 100%*) et l'annotation exhaustive (annoter toutes les contraintes possibles).*

> proches dans des *clusters* différents (`sampl.closest.diff`) qui s'avère très prometteur : cette sélection semble permettre de corriger efficacement la frontière des *clusters* en favorisant l'ajout de contraintes `MUST-LINK` sur des données séparées à tort. Or les corrections introduisant des contraintes `MUST-LINK` sont plus explicites que les contraintes `CANNOT-LINK` dont l'utilisation est plus ambiguë (*dans le premier cas, nous savons que les deux données sont désormais à mettre dans le même cluster ; dans le second, on sait qu'il faut séparer les deux données sans exprimer dans quels cluster ils doivent aller*). Ainsi, nous sommes d'avis que l'échantillonnage `sampl.closest.diff` est efficient car il permet d'introduire efficacement des contraintes dont l'utilité est immédiate pour corriger le *clustering*.

Ainsi, cette étude permet de répondre à la limite du nombre de contraintes requis (discutée dans l'hypothèse d'efficacité, SECTION 4.1). En effet, l'optimisation des paramètres de l'implémentation du `Clustering Interactif` permet de réduire considérablement le nombre de contraintes nécessaires pour obtenir une base d'apprentissage exploitable. En nous basant sur la TABLE 4.1 de l'étude de convergence, et dans le cadre de l'annotation d'un jeu de 500 données, nous sommes passé d'un paramétrage moyen nécessitant 3 750 (respectivement 10 000) contraintes à un paramétrage optimisé ne nécessitant que 950 (respectivement 1 750) contraintes pour atteindre un seuil de 90% (respectivement 100%) `v-measure`. L'ordre de grandeur de la charge de travail demandée aux annotateurs est donc située entre 2 et 4 fois la taille du jeu de données.

Cette estimation est plus raisonnable que celle réalisée en SECTION 4.1. De plus, en considérant que les annotations sont binaires et demandent *a priori* une charge mentale plus faible que les annotations par attribution de label ("*les données sont-elles similaires ?*" vs

"*quelle est l'étiquette de cette donnée ?*", cf. HART et STAVELAND, 1988), nous pouvons espérer que la charge totale nécessaire à l'annotation avec une méthodologie basée sur le `Clustering Interactif` est comparable à celle des méthodes traditionnelles. Nous confirmerons cette conclusion en étudiant le temps nécessaire à un opérateur pour annoter un lot de contraintes (cf. SECTION 4.3.1).

Afin de compléter cette analyse d'efficience, quelques pistes sont encore à explorer.

D'une part, une étude de coût est à réaliser pour trancher le choix de paramètres optimaux réalistes. En effet, il est intéressant d'étudier le coût machine (temps CPU utilisé) et le coût humain (temps d'annotation) afin d'affiner les choix techniques et de compléter les arguments sur l'utilisation en situation réelle d'une méthodologie d'annotation basée sur le `Clustering Interactif`. Cette étude sera l'occasion de rentrer en détail dans la comparaison de la charge demandée à l'annotateur, tant sur la durée que sur la complexité de la tâche d'annotation. Cet aspect sera traité dans la SECTION 4.3 (hypothèse des coûts).

D'autre part, l'étude réalisée se base sur des seuils de performance par rapport à une vérité terrain. Or en situation réelle, cette comparaison avec la vérité terrain n'est pas possible car elle est précisément en cours de conception (la base d'apprentissage finale devant être la vérité terrain). De plus, un tel score n'est pas le plus explicite pour un expert métier pour qui un score de `v-measure` n'est pas révélateur de la pertinence métier de la segmentation proposée des données. Dans un registre similaire, il est possible que l'évolution du partitionnement des données passe par plusieurs états stables et pertinents, mais que l'annotateur soit obligé d'affiner sa vision en annotant certaines contraintes ambiguës. Cela peut être la cas avec des *clusters* traitant en fin de compte de sujets très similaires (*ajouter des `MUST-LINK` pour les fusionner*) ou avec un *cluster* qui regroupe finalement plusieurs thématiques (*ajouter des `CANNOT-LINK` pour les segmenter*). Il manque donc une stratégie d'évaluation de pertinence de la base d'apprentissage en cours de construction, afin d'estimer la stabilité d'un partitionnement et la suffisance des annotations réalisées pour faire refléter la vision de l'annotateur dans le résultat obtenu. Cet aspect sera traité dans la SECTION 4.4 (hypothèse de pertinence).

Pour finir, comme pour l'étude de convergence réalisée en SECTION 4.1, nous avons supposé dans cette étude que l'annotateur est un expert métier connaissant parfaitement le domaine traité. Cette hypothèse forte n'est *a priori* pas valable en situation réelle : En effet, des différences d'annotations peuvent intervenir (*ambiguïtés sur les données, méconnaissance du domaine, erreurs d'inattention, différences d'opinions entre annotateurs, ...*), ce qui peut entraîner des divergences ou des incohérences dans la construction de la base d'apprentissage. Il semble donc nécessaire d'étudier les impacts de ces incohérences, ainsi que de proposer une méthode pour les prévenir ou les corriger. Cet aspect sera traité à la fin de ce chapitre dans la SECTION 4.6 (hypothèse de robustesse).

4.3 Évaluation de l'hypothèse sur les coûts

Dans les deux sections précédentes, nous avons estimé le paramétrage du `Clustering Interactif` le plus efficient pour atteindre 90% de `v-measure` avec la vérité terrain, correspondant à ce que nous appelons une annotation partielle. Toutefois, pour compléter l'étude de faisabilité technique de notre méthode, nous devons nous intéresser aux coûts (matériel et humain) à investir pour atteindre notre objectif. Nous aimerions donc vérifier l'hypothèse suivante :

Hypothèse sur les coûts

« **Il est possible d'estimer les coûts nécessaires d'une méthodologie d'annotation basée sur le `Clustering Interactif` pour obtenir une base d'apprentissage exploitable. Nous étudions en particulier les coûts relatifs au temps d'annotation, au temps de calculs des algorithmes, ainsi que la durée totale de la méthode en fonction de la taille du jeu de données.** »

La Figure 4.10 illustre cette hypothèse et la perspective de pouvoir caractériser la qualité de la base d'apprentissage en cours de construction en fonction d'un coût temporel au lieu d'un nombre abstrait d'itérations de la méthode.

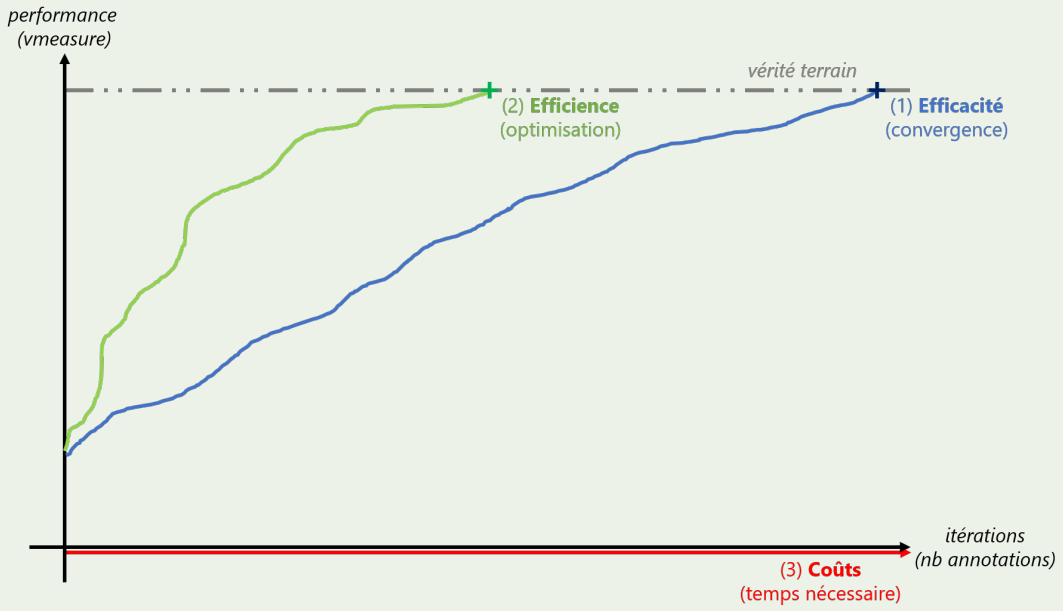

Figure 4.10 – *Illustration des études réalisées sur le `Clustering Interactif` (étape 3/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en v-measure) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine).*

Afin de vérifier cette hypothèse, nous organisons plusieurs expériences pour simuler et déterminer ces durées :

— une étude du **temps d'annotation** par un expert métier, temps mesuré lors d'une expérience d'annotation de contraintes faisant intervenir plusieurs opérateurs (cf.

Section 4.3.1) ;

— une étude du **temps de calcul** des algorithmes, temps modélisé en exécutant les différentes implémentations du `Clustering Interactif` avec diverses valeurs d'arguments (cf. Section 4.3.2) ; et

— une étude du **nombre de contraintes** nécessaires en fonction du nombre de données à traiter, nombre estimé en simulant la création d'une base d'annotation avec notre méthodologie sur des jeux de données de différentes tailles (cf. Section 4.3.3).

Nous exposons nos conclusions sur l'estimation du **temps total** à investir et réalisons une comparaison avec une organisation plus traditionnelle d'un projet d'annotation en Section 4.3.4.

4.3.1 Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle

Nous voulons estimer le temps nécessaire à un opérateur pour annoter un lot de contraintes. Pour cela, nous allons chronométrer plusieurs experts métiers en train d'annoter un même échantillon et modéliser le nombre de contraintes par minute, ainsi que son évolution au cours de plusieurs sessions d'annotation. De plus, nous aimerions aussi confirmer que l'ajout de contraintes dans notre contexte s'apparente à une tâche "intuitive", c'est-à-dire que l'annotation se fait dans la réaction et non dans la réflexion (voir Kahneman, 2011 qui distingue un *système 1* intuitif, rapide, de l'ordre de l'émotion, et un *système 2* plus lent, logique et réfléchi). Pour ce faire, nous estimons grossièrement le temps nécessaire à l'annotation réactive d'une contrainte, nous le comparons au temps moyen estimé lors de notre expérience, et nous nous demandons si la différence observée peut cacher un mécanisme cognitif plus complexe.

4.3.1.a Protocole expérimental

> ⚠ **Attention :** Dans cette étude, nous supposons que les annotateurs de l'expérience connaissent parfaitement le domaine traité dans le jeu de données, et qu'ils sont capables de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Afin de pourvoir faire cette hypothèse forte, et ainsi limiter les bruits dans l'analyse des résultats, le jeu de données devra traiter d'un sujet de culture générale (ne nécessitant donc pas de connaissance particulière) et des réviseurs supprimeront en amont et d'un commun accord les données trop spécifiques ou trop ambiguës.

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.3.

Nous allons procéder en plusieurs étapes. D'abord, il faut choisir un jeu de données approprié : pour valider notre hypothèse forte sur les compétences de nos annotateurs, nous cherchons un jeu de données traitant d'un sujet de culture générale. Pour cette expérience, nous avons donc choisi `MLSUM` : une collecte d'articles de journaux, classés par catégorie de publication et décrits par leur titre et leur résumé. Nous nous intéressons ici à la tâche de classification d'un titre d'article en fonction de sa catégorie de publication. Comme certains titres peuvent porter à confusion (un titre d'article n'étant pas toujours explicite sur son contenu), deux réviseurs (*une Data Scientist et moi-même*) sont chargés de choisir les données les plus explicites sur un échantillon d'un millier de données représentatives des catégories les plus communes. L'échantillon résultant, noté `MLSUM`

Données : jeu de données annotées (vérité terrain)
Entrées : plusieurs réviseurs, plusieurs annotateurs
initialisation : définir et revoir le jeu de données entre réviseurs ;
échantillonnage : sélectionner une base de contraintes avec `samp.rand.full` ;
temps théorique : estimation du temps nécessaire à l'annotation d'une contrainte ;
pour chaque *annotateur* **faire**
    tant que *la base de contraintes n'a pas été entièrement annotée* **faire**
        chronomètre : **START** ;
        annotation : annoter une partie des contraintes ;
        revue : revue des contraintes en conflits d'annotation ;
        chronomètre : **STOP** ;
        mesure : estimer la différence de chronomètre pour cette session ;
analyse : modéliser le temps d'annotation d'un lot de contraintes ;
Résultat : modélisation du temps d'annotation d'un lot de contraintes

ALGORITHME 4.3 – *Description en pseudo-code du protocole expérimental de l'étude du temps d'annotation d'un lot de contraintes par plusieurs experts métiers en situation réelle.*

`FR Train Subset (v1.0.0-schild)`, est composé de 744 titres d'articles rédigés en français et répartis en 14 classes (*économie*, *sport*, ...). Pour plus de détails, consulter l'ANNEXE A.2.

À partir de ces données, nous sélectionnons un lot de 1 000 contraintes à annoter. Comme nous nous intéressons exclusivement au temps d'annotation pour cette expérience (et que nous ne regardons pas le nombre d'itérations de la méthode), nous utilisons l'échantillonnage purement aléatoire (`samp.rand.full`). L'analyse de l'accord inter-annotateurs sera réalisé en SECTION 4.6.1.

Sur la base de cet échantillon, nous pouvons approximer le temps théorique nécessaire à l'annotation d'une contrainte à 6.8 secondes. En effet, il faut d'abord considérer la taille moyenne des titres d'article à lire et en déduire le temps dédié à la lecture des deux textes de la contrainte. En utilisant l'approximation d'une lecture silencieuse par un adulte à 238 mots par minute (BRYSBAERT, 2019) et en mesurant la taille moyenne des titres d'article à 10.1 mots, nous en déduisons que le temps de lecture d'un texte est environ de 2.55 secondes. Ensuite, il convient d'intégrer la durée de traitement cognitif requis pour estimer si les deux phrases sont similaires ou discordantes. À cet effet, nous retenons 1 seconde [22] (PURVES et BRANNON, 2013) en admettant que cette tâche est rapide. Enfin, nous ajoutons 1 seconde supplémentaire pour représenter la réaction motrice (clic de bouton) et le délai applicatif (rechargement de la page). Au total, nous obtenons ainsi un temps d'annotation moyen de 7 secondes. Bien entendu, cette durée reste approximative, mais elle nous permet de discuter de l'ordre de grandeur à manipuler durant l'annotation.

Ensuite, un groupe de 14 annotateurs vont annoter la sélection de 1 000 contraintes en plusieurs sessions. Les directives données aux opérateurs sont les suivantes :

— **Contexte de l'opérateur** : « *Vous êtes des **experts de la presse et de l'actualité**. Vous voulez classer des articles dans des catégories en fonction de leur titre. Vous ne savez pas précisément quelles catégories vous allez utiliser pour classer vos articles. Mais vous*

22. Nous pourrions faire le parallèle avec la composante `P600` communément admise en neuroscience pour caractériser la réaction provoquée par la dissonance grammaticale ou syntaxique d'une phrase. Nous arrondissons à 1 seconde pour garder une marge d'erreur.

savez ***caractériser la similitude*** *entre deux articles* ».

— **Contexte du jeu de données** : « *Le thème concerne les catégories d'articles de presse. La vérité terrain contient entre* 10 *et* 20 *catégories parmi les plus communes de la presse. La vérité terrain contient entre* 30 *et* 100 *articles par catégorie. Vous* ***pouvez regarder le jeu de données non annoté*** *autant que vous le voulez (disponible dans l'onglet* `TEXTS` *de l'application)* ».

— **Objectif de l'expérience** : « *Je veux savoir le temps nécessaire pour annoter un certain nombre de contraintes. Autrement dit :* ***pour annoter 1000 contraintes, combien de temps me faut-il ?*** ».

— **Consignes d'annotations** : « *Faites des séries de* ***15 minutes minimum*** *pour avoir de la régularité. Si possible,* ***isolez-vous*** *pour ne pas être dérangé et ne pas fausser les résultats. Pour chaque série,* ***notez le temps et le nombre de contraintes annotés****. Si vous ne savez pas quoi annoter (trop ambigu, vocabulaire inconnu, ...),* ***passez au suivant sans annoter*** *(vous êtes censés être des experts de la presse !)* ».

Pour réaliser l'annotation, les opérateurs auront accès à l'application web développée au cours de ce doctorat. Des captures d'écran sont disponibles en Figure 4.11 et Figure 4.12. Une description plus détaillée de l'application et de ses fonctionnalités est disponible en Annexe C.2.

Une fois les sessions d'annotations terminées, nous entraînons un modèle linéaire généralisé (*GLM*) pour estimer le temps d'annotation moyen pour un lot de contraintes (dont la taille est notée `batch_size`). Ce modèle sera caractérisé par le coefficient de détermination généralisé $\texttt{R}^2$ de *Cox et Snel* (Diamond et al., 1990), la log-vraisemblance `llf` (Edwards, 1992) et la log-vraisemblance `llf_null` du modèle *null*. Nous discutons aussi de l'évolution de la vitesse d'un opérateur au cours des différentes sessions d'annotation.

> **Pour information :** L'outil d'annotation utilisé est accessible dans Schild, 2022b. Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), ainsi que le projet à importer dans l'outil d'annotation, sont disponibles dans un dossier dédié de Schild, 2022c. Nous y utilisons entre autres les librairies `datetime`[23] et `statsmodels`[24] (Seabold et Perktold, 2010). De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.3.1.b Résultats obtenus

Durant cette expérience, 14 annotateurs ont participé à l'annotation de 1 000 contraintes aléatoires sur un jeu de données. Ces opérateurs travaillent tous dans un service informatique dédié à l'entraînement et l'amélioration de solutions de *Machine Learning* et sont répartis de la manière suivante :

— 4 femmes, 10 hommes ;

— 7 personnes entre 20 et 30 ans, 7 personnes entre 30 et 40 ans ;

23. https://pypi.org/project/datetime/
24. https://pypi.org/project/statsmodels/

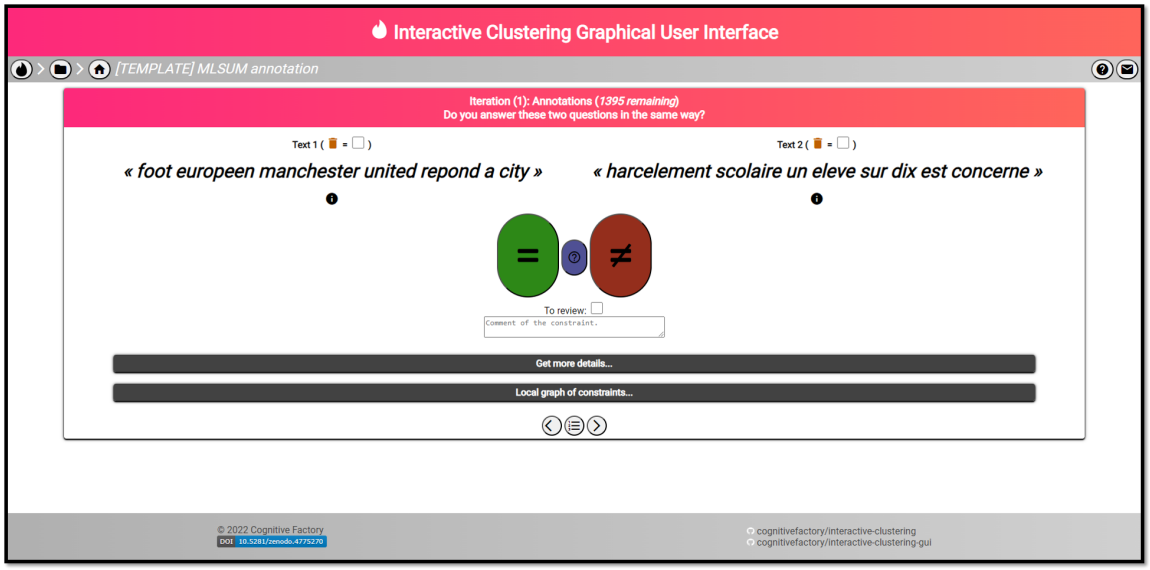

FIGURE 4.11 – *Capture d'écran de l'application web permettant d'utiliser notre méthodologie de `Clustering Interactif` : **page d'annotation de contraintes**. Les deux textes à annoter sont disposés à gauche et à droite de l'écran. Chacun dispose d'un case à cocher si le texte n'est pas pertinent à analyser (ambigu, hors périmètre, incompréhensible, ...).*
Les boutons à disposition permettent respectivement d'annoter un `MUST-LINK` si les données sont similaires (bouton « = »), un `CANNOT-LINK` si les données ne sont pas similaires (bouton « ≠ »), d'ignorer la contrainte pour laisser la main à l'algorithme de clustering (bouton en bleu), et d'ajouter un commentaire pour revoir la contrainte plus tard (case à cocher et champ de texte libre). Deux éléments déroulants permettent d'avoir des informations supplémentaires (metadata de sélection et de clustering, représentation graphique des liens entre contraintes annotées). Les boutons de navigation (boutons de flèches et de liste) sont disponibles en bas de page.

— 9 *Data Scientist*, 4 développeurs et 1 ergonome ;

— 1 personne ayant révisé le jeu de données, 3 le découvrant pour la première fois dont 1 ayant travaillé auparavant dans le secteur de la presse.

Par manque de disponibilités, 4 annotateurs n'ont que partiellement réalisé leur tâche : nous avons toutefois intégré leurs participations car elles contenaient au minimum 150 annotations.

D'après les observations, un annotateur réalisait en moyenne 170.7 contraintes par session d'annotation (min : 43, max : 547, médiane : 138, écart-type : 106.4) ce qui lui demandait en moyenne 23.1 minutes (min : 3.0, max : 92.0, écart-type : 14.4). De plus, la vitesse d'annotation moyenne était de 7.7 contraintes par minute (min : 3.5, max : 14.3, écart-type : 2.9).

Le modèle linéaire généralisé entraîné sur les mesures de temps d'annotations ($\texttt{R}^2$: 0.910, `llf` : −499.15, `llf_null` : −539.95) nous permet de déduire l'équation suivante :

$$\texttt{annotation\_time}\ [s] \ \propto\ 7.8 \cdot \texttt{batch\_size} \tag{4.1}$$

La FIGURE 4.13 représente cette modélisation du temps d'annotation en comparaison avec les mesures réalisées lors de l'expérience. Pour rappel, le temps théorique estimé précédemment est de 7 secondes par contrainte.

En ce qui concerne l'évolution de la vitesse d'annotation au cours des sessions, aucune tendance significative n'a été identifiée. La FIGURE 4.14 représente l'évolution de vitesse d'anno-

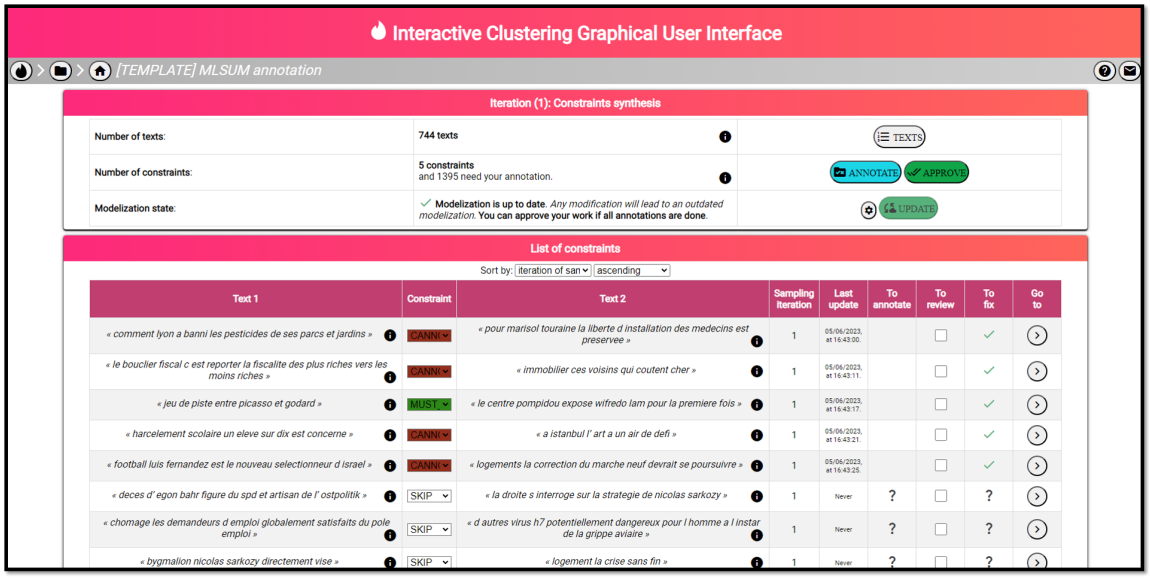

Figure 4.12 – *Capture d'écran de l'application web permettant d'utiliser notre méthodologie de* `Clustering Interactif` *:* ***page d'inventaire des contraintes à annoter****.*
La partie supérieure permet d'identifier le nombre de textes et de contraintes sur le projet, ainsi que les boutons destinés à calculer les transitivités entre les contraintes et à approuver le travail réalisé si aucune transitivité n'entre en conflit avec une contrainte annotée. La partie inférieure liste l'ensemble des contraintes du projet, avec les annotations réalisées, l'itération à laquelle la contraintes a été sélectionnée et annotée, si elle est à revoir ou si une incohérence la concernant est détectée.

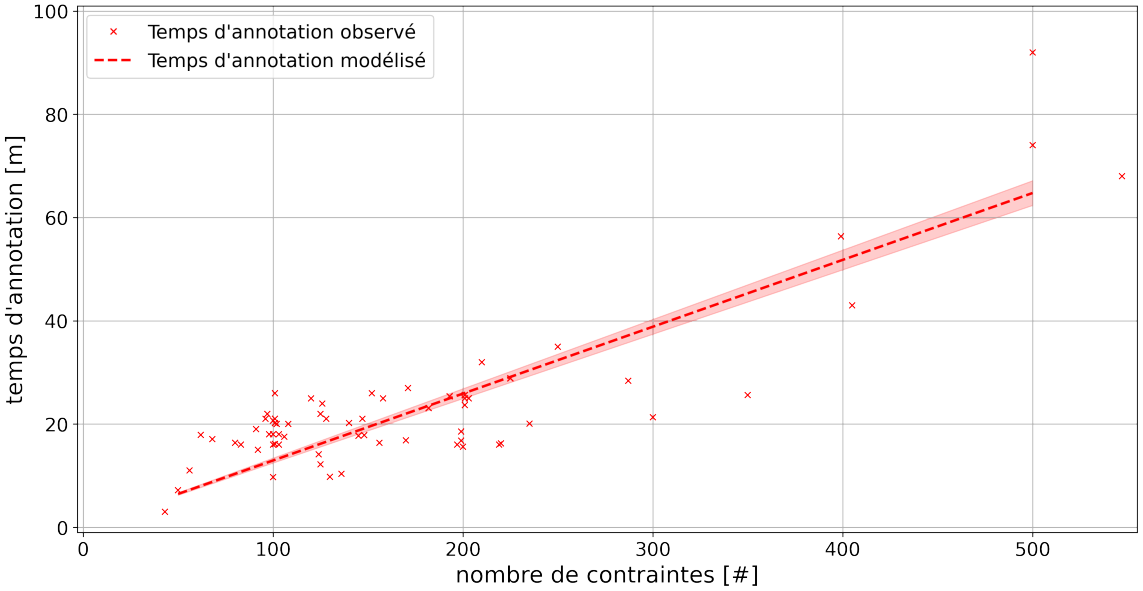

Figure 4.13 – *Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.*

tation pour quatre opérateurs (les deux plus rapides et les deux plus lents). Ces données sont l'objet d'une étude de cas dans la discussion ci-dessous.

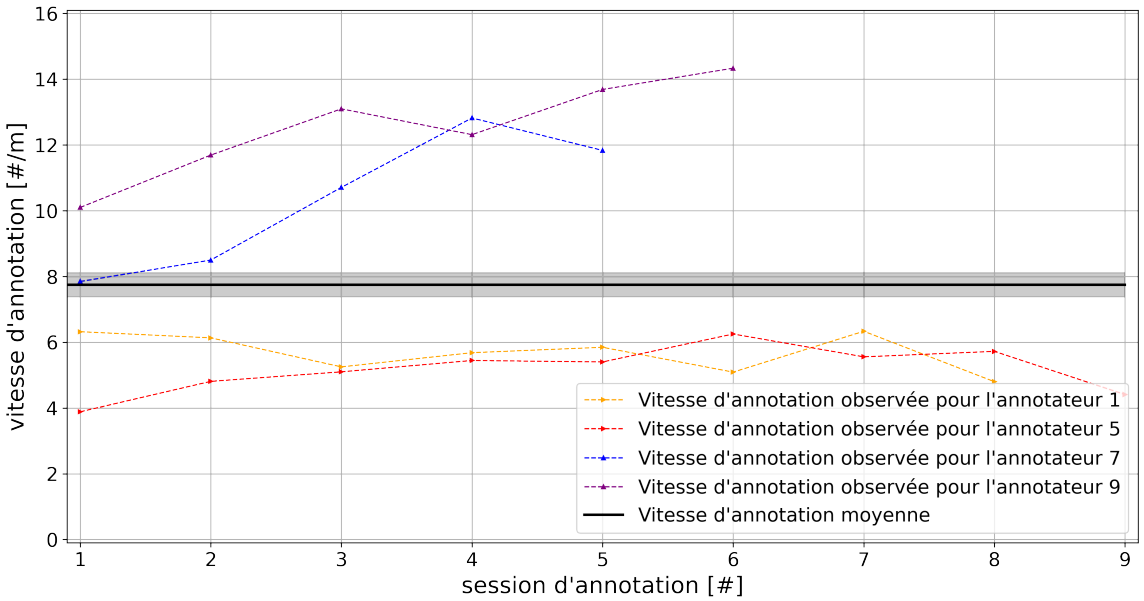

FIGURE 4.14 – *Étude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations.*

4.3.1.c Discussion

L'étude réalisée avec 14 annotateurs sur des lots de 1 000 contraintes a permis d'estimer à $7.8 \cdot$ `batch_size` le temps nécessaire (en secondes) pour annoter un lot de contraintes (cf. FIGURE 4.13).

Notes de l'auteur : Avant poursuivre la discussion, il est nécessaire de préciser qu'il est difficile de comparer ces résultats. D'une part, il y a une forte disparité des mesures, et il est idyllique de penser qu'une étude sur 14 annotateurs peut représenter la diversité du comportement humain sur une tâche aussi complexe que l'annotation de données textuelles. D'autre part, il y a un manque de repères concrets dans la littérature scientifique, entre autres à cause des nombreux facteurs intervenant dans une tâche d'annotation (*objectifs à réaliser, données à manipuler, nombre de choix proposés à l'opérateur, complexité sémantique des données, des compétences de l'opérateur, fréquence d'exécution de la tâche, ...*), mais aussi en raison du manque d'intérêt à l'analyse du temps nécessaire au profit de l'analyse de la cohérence et de la qualité intra- ou inter-annotateur (BALEDENT, 2023). De plus, les résultats peuvent différer en fonction des contraintes à caractériser : nous pouvons supposer que des paires de données très similaires ou très différentes sont simples à annoter, mais que des données plus ambiguës peuvent nécessiter davantage de temps pour être intégrées et étiquetées.

Pour pallier ce problème, nous proposons de confronter nos estimations à des mesures réalisées sur des tâches n'ayant pas le même objectif mais dont la complexité est comparable. Cette approche, bien qu'un peu rudimentaire, nous permettra ainsi de discuter des ordres de grandeur à manipuler.

Analyse de l'annotation d'une contrainte. En premier lieu, nous voulions confirmer que l'annotation de contraintes est une tâche intuitive dont la durée est caractéristique d'un mécanisme de réaction et non d'une réflexion. Pour ce faire, nous avons estimé la durée théorique d'une annotation à 7 secondes par contrainte, comprenant les temps de lecture, d'analyse de la similitude et d'action de l'opérateur, et nous avons mesuré la durée d'annotation réelle à 7.8 secondes par contrainte. Bien que l'écart constaté (0.8 seconde) soit compris dans les bornes d'approximation, cette différence n'est pas suffisamment insignifiante pour nous permettre d'exclure avec certitude la présence d'un mécanisme cognitif supplémentaire.

Pour nous permettre de discuter du temps d'annotation mesuré et de son approximation théorique, nous utilisons utiliser plusieurs points de référence extraits de (Snow et al., 2008). Dans cette étude, les auteurs délèguent quelques tâches d'annotation à `Amazon Mechanical Turk`[25] :

1. Tâche de "*désambiguïsation du sens des mots*" (Pradhan et al., 2007). Elle consiste à catégoriser le contexte de phrases comportant le mot "*président*" suivant 3 options préformatées (*dirigeant d'une entreprise, dirigeant des États Unis, dirigeant d'un autre pays*). Il y a 177 phrases à labelliser par 10 annotateurs, et la tâche a été réalisée en un total de 8.59 heures, soit une moyenne de 17.5 secondes par annotation. Nous utilisons ce point de repère car la tâche s'apparente à une annotation d'une tâche de classification (3 catégories).

2. Tâche de "*caractérisation de similitude des mots*" (Miller et Charles, 1991). Elle consiste à ordonner des paires de mots du plus similaire au moins similaire en fonction de leur proximité sémantique afin de mettre en avant les meilleurs paires de synonymes. Il y a 30 paires de synonymes à ordonner par 10 annotateurs, et la tâche a été réalisée en un total de 0.17 heures, soit une moyenne de 2.0 secondes par annotation. Nous utilisons ce point de repère car la tâche s'apparente à l'annotation de contraintes (*laquelle de ces deux paires est la plus adéquate ?*) qui ne fait pas appel à un mécanisme de réflexion (*peu de vocabulaire, similarité intrinsèque triviale entre deux mots*).

3. Tâche de "*reconnaissance de l'implication textuelle*" (Dagan et al., 2005). Elle consiste à confirmer ou infirmer si une phrase est la conséquence d'une autre (*l'annotation est donc binaire*). Il y a 800 paires de phrases à valider par 10 annotateurs, et la tâche a été réalisée en un total de 89.3 heures, soit une moyenne de 40.2 secondes par annotation. Nous utilisons ce point de repère car la tâche s'apparente à l'annotation de contraintes (*est-ce que l'implication est vraie ?*) faisant intervenir un mécanisme de réflexion (*comprendre une implication logique*).

En utilisant la tâche de "*désambiguïsation du sens des mots*" (1.), nous avons déjà un point de comparaison entre notre annotation de contraintes et une annotation classique de classes. Nous constatons une nette différence entre les deux approches (7.8 secondes par contrainte vs 17.5 secondes par donnée), cela met en avant qu'une annotation de contraintes est plus rapide qu'une annotation par label.

Ensuite, en utilisant la tâche de "*caractérisation de similitude des mots*" (2.), nous pouvons confirmer que notre approximation théorique (faisant intervenir un traitement cognitif de 1 seconde) semble adaptée pour représenter un mécanisme de l'ordre de la réaction. En effet, le coût très faible de la caractérisation d'une similarité entre deux mots (2.0 secondes par paire de mots) s'avère être en adéquation avec cette approximation (2.5 secondes par paire de phrases de taille

25. `Amazon Mechanical Turk` est une plateforme de travail collaboratif en ligne (*crowdsourcing*) : nous avions évoqué leurs fonctionnements, leurs avantages et leurs inconvénients en Section 2.3.2.c, notamment le risque de l'abandon de la qualité des annotations au profit de la quantité.

1).

Enfin, en utilisant la tâche de "*reconnaissance de l'implication textuelle*" (3.), nous comparons deux annotations binaires dont une fait nettement appel à un mécanisme de réflexion (déduction logique dans une implication). Nous constatons une très nette différence entre les deux approches (7.8 secondes par contrainte vs 40.2 secondes par implication), ce qui nous permet d'exclure la présence de mécanisme cognitif trop complexe.

Points à retenir : Mettant bout à bout ces comparaisons, et en gardant à l'esprit que ces références restent approximatives, nous pouvons conclure que (1) **l'annotation de contraintes est une tâche plus rapide qu'une annotation par label** et que (2) **cette annotation binaire ne semble pas faire pas intervenir de traitement cognitif complexe** (*c'est une piste à explorer plus en détails pour expliquer le gain de temps observé*).

Analyse d'une session d'annotation de contraintes. Nous avons analysé l'évolution de la vitesse d'annotation au cours des sessions d'annotation, en espérant observer une accélération des annotations au fur et à mesure que l'annotateur s'habitue avec la tâche, ainsi qu'un effet de fatigue pour des sessions d'annotations trop longues. Cependant, aucune de nos analyses n'a montré de résultats significatifs (on peut constater la forte dispersion des résultats grâce à la FIGURE 4.14). Nous ne pouvons donc pas conclure sur de telles tendances.

Notes de l'auteur : Nos intuitions initiales concernaient deux points :

— la diminution du **temps d'adaptation** au cours de sessions d'annotations : au fur et à mesure qu'il annote, l'opérateur pourrait entrer plus facilement dans sa tâche, lui permettant d'atteindre plus rapidement sa vitesse de croisière et ainsi gagner en efficacité sur plusieurs sessions. D'après ANDERSON, 2013, ce temps d'adaptation pourrait se définir en trois étapes : une phase déclarative (*besoin d'instructions détaillées, exécution lente et avec erreurs*), une phase associative (*quelques rappels clés suffisent pour retrouver les instructions, donc gain de vitesse*) et une phase autonome (*les consignes sont acquises, donc exécution rapide et sans erreur*) ;

— l'intervention d'un **effet de fatigue** : si une session d'annotation dure trop longtemps, l'opérateur pourrait perdre en efficacité par manque de concentration et augmenter ses chances de faire des erreurs. D'après JONES et al., 2015, la fatigue est considérée comme un inconfort qui s'installe après une tâche excessive, et ELKOSANTINI et GIEN, 2009 décrit cet état de fatigue par des capacités de travail réduites.

Ces différentes intuitions ont aussi été remontées par les annotateurs de notre expérience, mais aucun effet significatif n'a pu être observé.

Par extension, nous ne pouvons pas non plus conclure sur la taille optimale d'échantillon de contraintes à sélectionner pour une session d'annotation. Dans nos précédentes études, nous avions arbitrairement fixé la taille de lot à 50 pour bénéficier d'itérations brèves, permettant à l'algorithme de *clustering* de s'améliorer régulièrement avec les dernières contraintes. Mais des petits lots d'annotation démultiplient le nombre d'itérations à réaliser, et donc le nombre d'algorithmes à exécuter. Il serait donc judicieux d'adapter le nombre d'annotations à réaliser

pour améliorer l'expérience utilisateur en situation réelle de l'opérateur. Malheureusement, aucun repère significatif ne peut être déduit de nos résultats pour prédire la fin d'un temps d'adaptation ou le début d'un effet de fatigue.

Afin de proposer tout de même un ordre de grandeur de taille de lot, nous pouvons nous intéresser au nombre moyen de contraintes annotées lors des sessions réalisées par les opérateurs de notre expérience. Bien que ces informations n'aient pas été collectées initialement à cette fin, nous pouvons supposer que les opérateurs ont interrompu leur session pour se reposer (*par fatigue, ennui, agacement, ...*) ou répondre à une autre sollicitation (*intervention d'un collègue, mail important, pause café, ...*) Après un entretien avec les opérateurs de notre expérience, il semble y avoir deux possibilités : soit l'opérateur ne se fixait pas d'objectif et s'arrêtait par fatigue ; soit il se fixait un objectif (*de nombre ou de durée*), mais adaptait son prochain objectif en fonction de la fatigue ressentie en fin de session. Dans les deux cas, nous pouvons faire l'hypothèse que le nombre moyen d'annotation par session tend à représenter une borne supérieure de la taille maximale d'un lot à considérer pour ne pas entamer l'effet de fatigue. Sur l'expérience réalisée, cette moyenne est de 170.70 contraintes annotées par session (écart-type : 106.37, erreur standard : 13.19). En prenant en compte une marge d'erreur pour minimiser ce résultat, nous retenons 150 contraintes comme seuil à ne pas dépasser.

> **Points à retenir :** Pour une session d'annotation, **nous conseillons une taille d'échantillon entre 50 et 150 contraintes**. Attention aux échantillons trop petits qui multiplient le nombre d'itérations à réaliser ; Attention aussi aux échantillons trop gros qui peuvent introduire un effet de fatigue chez l'opérateur et casser la dynamique d'interactions avec la machine. La discussion finale de ce chapitre affinera cette fourchette grâce à une vue d'ensemble sur les coûts de la méthode.

Analyse des fonctionnalités de l'application d'annotation. Pour finir cette discussion, nous nous intéressons à l'utilisation du logiciel par nos opérateurs au cours de cette étude. En effet, il est logique de penser que la conception de l'application et les fonctionnalités dont elle dispose peut grandement impacter l'expérience utilisateur de notre méthodologie de *clustering* itératif. Un entretien avec les opérateurs a permis de remonter plusieurs pistes d'amélioration de son ergonomie.

Un premier point concerne l'affichage des textes d'une contrainte. Comme montré en FIGURE 4.11, nous avions choisi d'afficher des données normalisées à l'écran pour masquer le bruit provoqué par les accents, les majuscules, la ponctuation. Bien que cette fonctionnalité peut servir pour des textes bruts (*issus de conversations clients ou de forums par exemple*), cela a plutôt nuit à la compréhension des données par les opérateurs. Nous avons donc envisagé de laisser la données brutes à disposition, dans une infobulle ou grâce à une option permettant d'interchanger le format des données.

Une seconde proposition concerne l'ordre des contraintes à annoter. Nous pouvons facilement admettre que la compréhension rapide d'un grand nombre de textes est une tâche pénible. Pour soulager les opérateurs et limiter le nombre de changement de contexte, nous avons trié les contraintes par ordre alphabétique. Ainsi, toutes les contraintes associées à une même donnée peuvent être traitées à la suite. Cette solution peut faciliter la caractérisation d'une similitude en analysant à la chaîne plusieurs données du même type. À cet effet, une option de tri à été ajoutée sur la liste de contraintes à traiter (cf. FIGURE 4.12).

Enfin, une dernière idée concerne l'affichage des données à annoter. Nous avions jusqu'à présent considéré l'annotation entre deux données (cf. FIGURE 4.11), mais il peut être judicieux

d'afficher plusieurs données à caractériser simultanément. Une telle fonctionnalité permettrait ainsi de regrouper rapidement un grand nombre de données similaires ou de distinguer avec moins d'ambiguïté certaines données en s'appuyant sur leur voisinage.

> **Pour information :** Ces différentes évolutions sont en cours d'analyse ou ont déjà été intégrées dans l'application que nous proposons (Schild, 2022b).

4.3.2 Étude du temps de calcul nécessaire aux algorithmes implémentés en chronométrant des exécutions dans différentes situations

Maintenant que nous avons pu modéliser le temps nécessaire à un expert pour annoter un lot de contraintes, nous nous intéressons au temps nécessaire à la machine pour interpréter ces annotations et proposer une nouvelle segmentation des données.

Comme les différents algorithmes employés manipulent des contraintes sur les données, l'estimation théorique du temps d'exécution par l'analyse de la complexité ne semble pas fiable : en effet, quelques contraintes bien placées peuvent suffire à simplifier le fonctionnement d'un algorithme de *clustering* alors qu'une grande quantité de contraintes mal placées vont au contraire le pénaliser. Nous préférons donc une approche empirique en chronométrant plusieurs exécutions isolées des algorithmes intervenant dans notre implémentation du `Clustering Interactif` et en évaluant l'importance de leurs différents arguments d'entrée (la taille du jeu de données, le nombre de *clusters* et le nombre de contraintes annotées, ...). Nous profitons aussi de ces modélisations du temps de calcul pour confirmer le choix de paramétrage réalisé lors de l'étude d'efficience en Section 4.2, et ainsi faire un compromis entre l'algorithme le plus efficient et l'algorithme le plus rapide.

4.3.2.a Protocole expérimental

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.4.

Nous utilisons le jeu de données `Bank Cards (v2.0.0)` comme référence pour cette expérience : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 1 000 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'Annexe A.1. Cependant, un seul jeu de données ne nous permet pas d'analyser l'impact du nombre de données sur le temps d'exécution des algorithmes. Pour utiliser facilement plusieurs jeux de données de tailles différentes tout en maîtrisant leur contenu, nous avons donc dupliqué aléatoirement des données issues du jeu de référence en y insérant des fautes de frappes.

> **Attention :** Dans le cadre de cette étude, nous faisons l'hypothèse que cette création artificielle de données n'a pas d'impact majeur sur le temps d'exécution des différents algorithmes.

À l'aide de ces données, nous lançons plusieurs exécutions de chaque algorithme de notre implémentation du `Clustering Interactif` avec différentes variations de contexte d'utilisation. Cette implémentation est détaillée en Annexe C.1. Cela comprend les tâches, algorithmes et contextes d'utilisation suivants :

Données : jeux de données annotées (vérités terrains) de tailles différentes
Entrées : combinaisons d'algorithmes et de paramètres à tester

pour chaque *combinaison d'algorithmes et de paramètres à tester* **faire**
  initialisation (données) : récupérer ou générer le jeu de données ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  si *estimation de la tâche de* ***prétraitements*** **alors**
    chronomètre : **START** ;
    prétraitements (à étudier) : supprimer le bruit dans les données ;
    chronomètre : **STOP** ;
  sinon si *estimation de la tâche de* ***vectorisation*** **alors**
    prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
    chronomètre : **START** ;
    vectorisation (à étudier) : transformer les données en vecteurs ;
    chronomètre : **STOP** ;
  sinon si *estimation de la tâche de* ***clustering*** **alors**
    prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
    vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
    échantillonnage initial : sélectionner des contraintes avec `samp.rand.full` ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    chronomètre : **START** ;
    clustering (à étudier) : regrouper les données par similarité ;
    chronomètre : **STOP** ;
  sinon si *estimation de la tâche d'****échantillonnage*** **alors**
    prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
    vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
    échantillonnage initial : sélectionner des contraintes avec `samp.rand.full` ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering initial : regrouper les données avec `clust.kmeans.cop` ;
    chronomètre : **START** ;
    échantillonnage (à étudier) : sélectionner de nouvelles contraintes à annoter ;
    chronomètre : **STOP** ;

pour chaque *algorithme à modéliser* **faire**
  cadrage : définir les facteurs et les interactions intervenant dans la modélisation ;
  simplification : restreindre la modélisation aux facteurs les plus corrélés ;
  analyse : modéliser le temps d'exécution avec les facteurs retenus ;

Résultat : modélisation du temps d'exécution des différents algorithmes

ALGORITHME 4.4 – *Description en pseudo-code du protocole expérimental de l'étude du temps d'exécution des algorithmes du `Clustering Interactif`.*

1. le **prétraitements** des données...
   - avec les algorithmes suivants : **simples** (noté `prep.simple`), **avec lemmatisation** (noté `prep.lemma`) et **avec filtres** (noté `prep.filter`) ;
   - avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`) ;
2. la **vectorisation** des données...
   - avec les algorithmes suivants : **TF-IDF** (noté `vect.tfidf`) et **SpaCy** (noté `vect.frcorenewsmd`) ;
   - avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`) ;
   - le tout précédé par des prétraitements **simples** ;
3. le ***clustering* sous contraintes** des données...
   - avec les algorithmes suivants : **KMeans** (modèle *COP* noté `clust.kmeans.cop`), **Hiérarchique** (lien *single* noté `clust.hier.sing` ; lien *complete* noté `clust.hier.comp` ; lien *average* noté `clust.hier.avg` ; lien *ward* noté `clust.hier.ward`) et **Spectral** (modèle *SPEC* noté `clust.spec`) ;
   - avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`), le **nombre de contraintes annotées** (variant de 0 à 5 000 par pas de 500, noté `previous_nb_constraints`) et le **nombre de *clusters* à trouver** (variant de 5 à 50 par pas de 5, noté `algorithm_nb_clusters`) ;
   - le tout précédé par des prétraitements **simples** et une vectorisation **TF-IDF** et un échantillonnage initial **purement aléatoire** ;
4. l'**échantillonnage** des contraintes à annoter...
   - avec les algorithmes suivants : **purement aléatoire** (noté `samp.random.full`), **pseudo-aléatoire** (noté `samp.random.same`), **même *cluster* mais étant les plus éloignés** (noté `samp.farthest.same`) et ***clusters* différents mais étant les plus proches** (noté `samp.closest.diff`) ;
   - avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`), le **nombre de contraintes annotées** (variant de 0 à 5 000 par pas de 500, noté `previous_nb_constraints`), le **nombre de *clusters* existants** (variant de 10 à 50 par pas de 10, noté `previous_nb_clusters`) et le **nombre de contraintes à sélectionner** (variant de 50 à 250 par pas de 50, noté `algorithm_nb_constraints`) ;
   - le tout précédé par des prétraitements **simples**, une vectorisation **TF-IDF**, un *clustering* initial **KMeans** (modèle *COP*) et un échantillonnage initial **purement aléatoire** ;

Il y a donc 8 825 combinaisons d'algorithmes (`15` pour les prétraitements, 10 pour la vectorisation, 3 330 pour le *clustering*, 5 550 pour l'échantillonnage), et chaque combinaison est répétée

5 fois pour contrer les aléas statistiques des exécutions. De plus, chaque jeu de données est généré 5 fois pour contrer les aléas statistiques de création, donc il y a 220 625 exécutions d'algorithmes (375 pour les prétraitements, 250 pour la vectorisation, 82 500 pour le *clustering*, 137 500 pour l'échantillonnage).

Sur la base de ces mesures, nous cherchons à modéliser le temps d'exécution de chaque algorithme en fonction de son contexte d'utilisation (dépendant de ses arguments d'entrée), et les interactions doubles entre paramètres sont envisagées. Afin de réduire la complexité des modélisations, nous ordonnons les interactions de facteurs possibles en fonction de leur corrélation avec le temps mesuré (la corrélation `r` de *Pearson* (Kirch, 2008) est utilisée) et nous nous limitons aux variables responsables d'un maximum de la variance des mesures (la méthode d'*Elbow* (Thorndike, 1953) est utilisée pour choisir les facteurs pertinents). Sur cette base, nous entraînons un modèle linéaire généralisé (*GLM*, cf. Nelder et Wedderburn, 1972) pour représenter le temps d'exécution moyen de l'algorithme : ce modèle sera caractérisé par le coefficient de détermination généralisé `R²` de *Cox et Snel* (Diamond et al., 1990), la log-vraisemblance `llf` (Edwards, 1992) et la log-vraisemblance `llf_null` du modèle *null*. Pour finir, nous discutons des valeurs des coefficients obtenus sur l'impact du temps d'exécution.

> **Pour information :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. Nous y utilisons entre autres les librairies `datetime`[26] et `statsmodels`[27] (Seabold et Perktold, 2010). De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.3.2.b Résultats obtenus

En ce qui concerne la tâche de **prétraitements**, une première analyse montre que les modélisations des trois implémentations sont similaires (`p-valeur` : > 0.980). Nous faisons donc une seule modélisation.

Pour les algorithmes de prétraitements (`prep.simple`, `prep.lemma` et `prep.filter`), l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size` (`r` : 0.997). Le modèle linéaire généralisé retenu (`R²` : > 0.999, `llf` : -432.43, `llf_null` : $-1\ 353.98$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(prep)}\ [s] \ \propto\ 6.55 \cdot 10^{-3} \cdot \texttt{dataset\_size} \tag{4.2}$$

La Figure 4.15 représente cette modélisation du temps de calcul des algorithmes de prétraitements en comparaison avec les mesures réalisées lors de l'expérience.

En ce qui concerne la tâche de **vectorisation**, une première analyse montre que les modélisations des deux implémentations sont différentiables (`p-valeur` : $< 10^{-3}$). Nous faisons donc une modélisation par algorithme.

Pour les algorithmes de vectorisation `vect.tfidf`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut

26. https://pypi.org/project/datetime/
27. https://pypi.org/project/statsmodels/

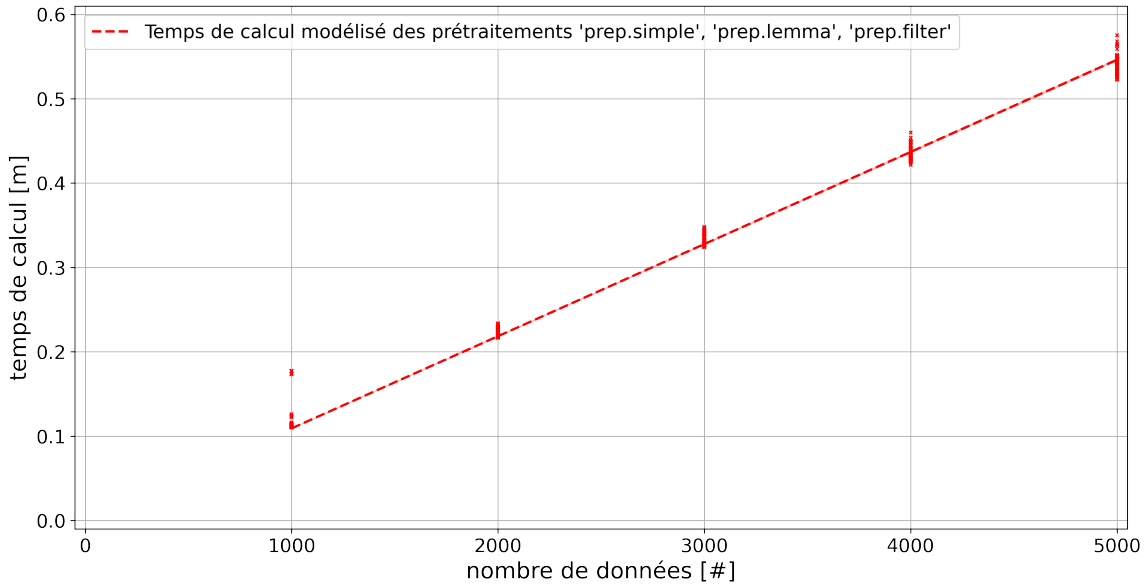

FIGURE 4.15 – *Estimation du temps nécessaire (en minutes) pour effectuer une tâche de* ***prétraitements*** *en fonction du nombre de données à traiter. Les paramétrages* `prep.simple`*,* `prep.lemma` *et* `prep.filter` *ayant des temps de calculs similaires, leurs modélisations n'ont pas été séparées.*

être réalisée à partir du facteur `dataset_size` (`r` : 0.977). Le modèle linéaire généralisé retenu (`R²` : > 0.999, `llf` : 259.89, `llf_null` : 70.04) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(vect.tfidf)}\ [s] \ \propto\ 9.16 \cdot 10^{-5} \cdot \texttt{dataset\_size} \tag{4.3}$$

Pour les algorithmes de vectorisation `vect.frcorenewsmd`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size` (`r` : 0.983). Le modèle linéaire généralisé retenu (`R²` : > 0.999, `llf` : -214.44, `llf_null` : -399.39) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(vect.frcorenewsmd)}\ [s] \ \propto\ 4.62 \cdot 10^{-3} \cdot \texttt{dataset\_size} \tag{4.4}$$

La FIGURE 4.16 représente ces modélisations de temps de calcul des algorithmes de vectorisation en comparaison avec les mesures réalisées lors de l'expérience.

En ce qui concerne la tâche de ***clustering* sous contraintes**, une première analyse montre que les modélisations des six implémentations sont différentiables (`p-valeur` : $< 10^{-3}$). Nous faisons donc une modélisation par algorithme.

> ⚠ **Attention :** Plusieurs exécutions des algorithmes de type *hiérarchique* ont été annulées pour les jeux de données de tailles supérieures à 4 000 car la durée excède plusieurs heures. Nous limitons donc l'analyse de `clust.hier.sing`, `clust.hier.comp`, `clust.hier.avg` et `clust.hier.ward` aux tailles de 1 000 à 3 000.

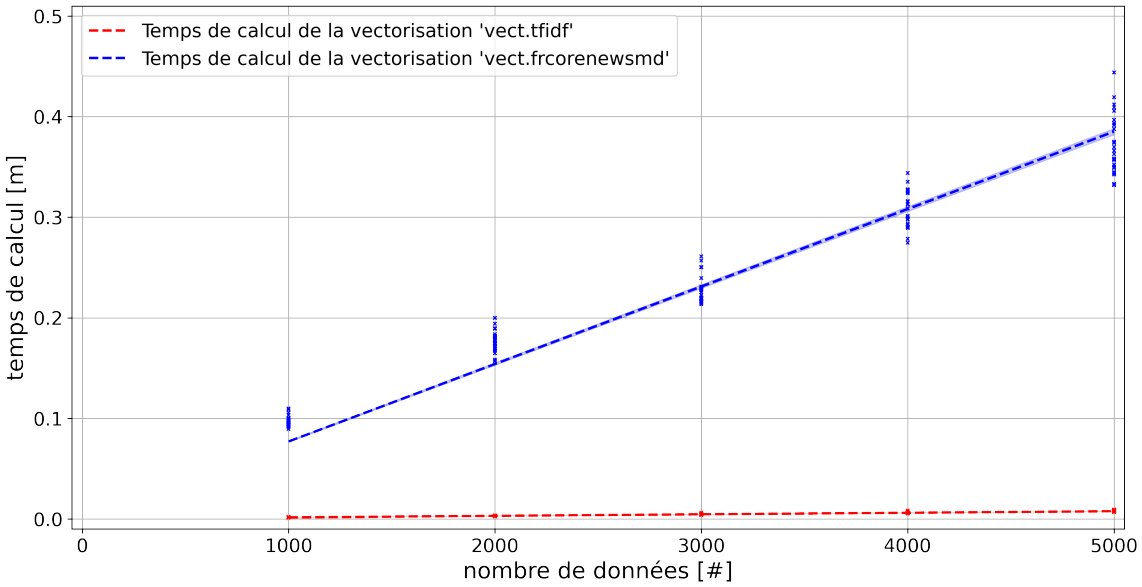

Figure 4.16 – *Estimation du temps nécessaire (en minutes) pour effectuer une tâche de **vectorisation** en fonction du nombre de données à traiter.*

Pour les algorithmes du *clustering* sous contraintes `clust.kmeans.cop`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size` (`r` : 0.837). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_clusters}$ (`r` : 0.545). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: 0.802, `llf` : $-9.37 \cdot 10^4$, `llf_null` : $-1.00 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(clust.kmeans.cop)}\ [s] \ \propto\ 1.45 \cdot 10^{-1} \cdot \texttt{dataset\_size} \tag{4.5}$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.sing`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ (`r` : 0.940). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_clusters}$ (`r` : 0.729). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: 0.987, `llf` : $-5.54 \cdot 10^4$, `llf_null` : $-6.10 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(clust.hier.sing)}\ [s] \ \propto\ 5.00 \cdot 10^{-4} \cdot \texttt{dataset\_size}^2 \tag{4.6}$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.comp`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ (`r` : 0.938). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_clusters}$ (`r` : 0.736). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: 0.984, `llf` : $-5.56 \cdot 10^4$, `llf_null` : $-6.11 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(clust.hier.comp)}\ [s] \ \propto\ 4.99 \cdot 10^{-4} \cdot \texttt{dataset\_size}^2 \tag{4.7}$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.avg`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale

et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.915). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_clusters}$ ($\texttt{r}$: 0.713). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: 0.981, $\texttt{llf}$: $-5.90 \cdot 10^4$, $\texttt{llf\_null}$: $-6.45 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(clust.hier.avg)}\ [s] \ \propto\ 8.51 \cdot 10^{-4} \cdot \texttt{dataset\_size}^2 \tag{4.8}$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.ward`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.945). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_clusters}$ ($\texttt{r}$: 0.734). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: 0.989, $\texttt{llf}$: $-5.57 \cdot 10^4$, $\texttt{llf\_null}$: $-6.14 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(clust.hier.ward)}\ [s] \ \propto\ 5.30 \cdot 10^{-4} \cdot \texttt{dataset\_size}^2 \tag{4.9}$$

Pour les algorithmes du *clustering* sous contraintes `clust.spec`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.658). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_clusters}$ ($\texttt{r}$: 0.595). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: 0.527, $\texttt{llf}$: $-7.89 \cdot 10^5$, $\texttt{llf\_null}$: $-8.27 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(clust.spec)}\ [s] \ \propto\ 8.18 \cdot 10^{-6} \cdot \texttt{dataset\_size}^2 \tag{4.10}$$

La FIGURE 4.17 représente ces modélisations de temps de calcul des algorithmes de *clustering* en comparaison avec les mesures réalisées lors de l'expérience.

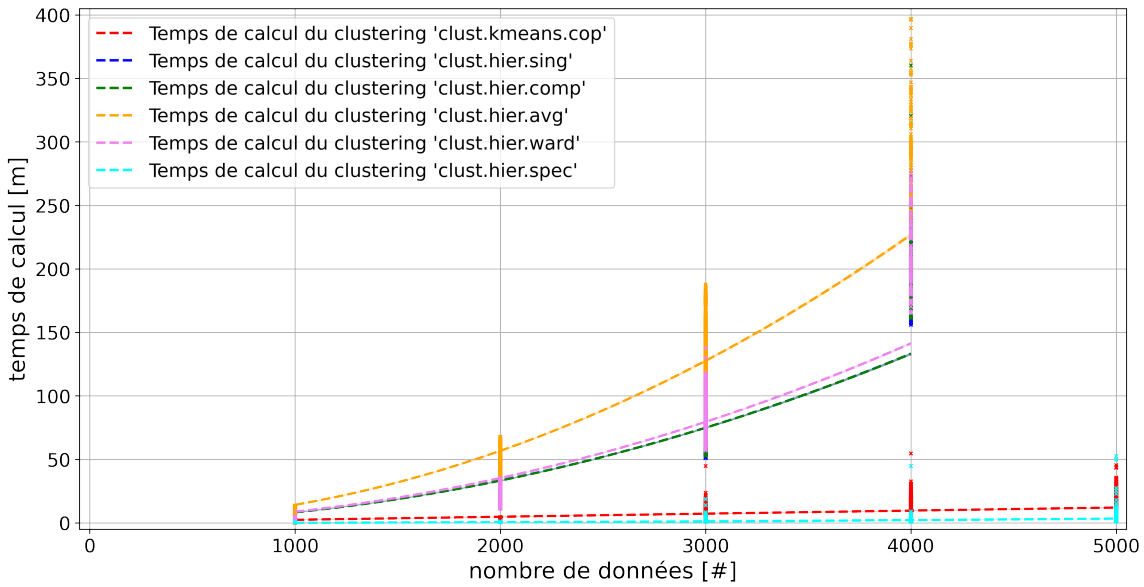

FIGURE 4.17 – *Estimation du temps nécessaire (en minutes) pour effectuer une tâche de* ***clustering*** *en fonction du nombre de données à traiter.*

En ce qui concerne la tâche d'**échantillonnage de contraintes**, une première analyse montre que les modélisations des quatre implémentations sont différentiables (`p-valeur` : $< 10^{-3}$). Nous faisons donc une modélisation par algorithme.

Pour les algorithmes de l'échantillonnage de contraintes `samp.rand.full`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.993). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{previous\_nb\_clusters}$ ($\texttt{r}$: 0.791). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: > 0.999, `llf` : $-4.52 \cdot 10^4$, `llf_null` : $-1.17 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(samp.rand.full)}\ [s] \ \propto\ 8.20 \cdot 10^{-7} \cdot \texttt{dataset\_size}^2 \tag{4.11}$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.rand.same`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.939). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{algorithm\_nb\_constraints}$ ($\texttt{r}$: 0.611). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: > 0.999, `llf` : $-3.20 \cdot 10^4$, `llf_null` : $-6.84 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(samp.rand.same)}\ [s] \ \propto\ 1.85 \cdot 10^{-7} \cdot \texttt{dataset\_size}^2 \tag{4.12}$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.farthest.same`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.981). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{previous\_nb\_clusters}$ ($\texttt{r}$: 0.700). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: > 0.999, `llf` : $-4.56 \cdot 10^4$, `llf_null` : $-1.02 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(samp.farthest.same)}\ [s] \ \propto\ 5.19 \cdot 10^{-7} \cdot \texttt{dataset\_size}^2 \tag{4.13}$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.closest.diff`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur $\texttt{dataset\_size}^2$ ($\texttt{r}$: 0.995). Le second facteur le plus corrélé (mais non retenu) est l'interaction $\texttt{dataset\_size}^2 \cdot \texttt{previous\_nb\_clusters}$ ($\texttt{r}$: 0.815). Le modèle linéaire généralisé retenu ($\texttt{R}^2$: > 0.999, `llf` : $-5.96 \cdot 10^4$, `llf_null` : $-1.36 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\texttt{computation\_time(samp.closest.diff)}\ [s] \ \propto\ 1.43 \cdot 10^{-6} \cdot \texttt{dataset\_size}^2 \tag{4.14}$$

La FIGURE 4.18 représente ces modélisations de temps de calcul des algorithmes d'échantillonnage en comparaison avec les mesures réalisées lors de l'expérience.

4.3.2.c Discussion

Dans cette étude, nous avons estimé le temps de calcul des différents algorithmes implémentés afin de confirmer le choix de paramétrage pour une convergence optimale (cf. hypothèse d'efficience en SECTION 4.2). Ces estimations ont été réalisées sur la base de plusieurs exécutions et fonction de divers contextes d'utilisation : nombre de données, nombre de contraintes annotées, nombre de contraintes à sélectionner, nombre de *clusters* existants, nombre de *clusters* à trouver.

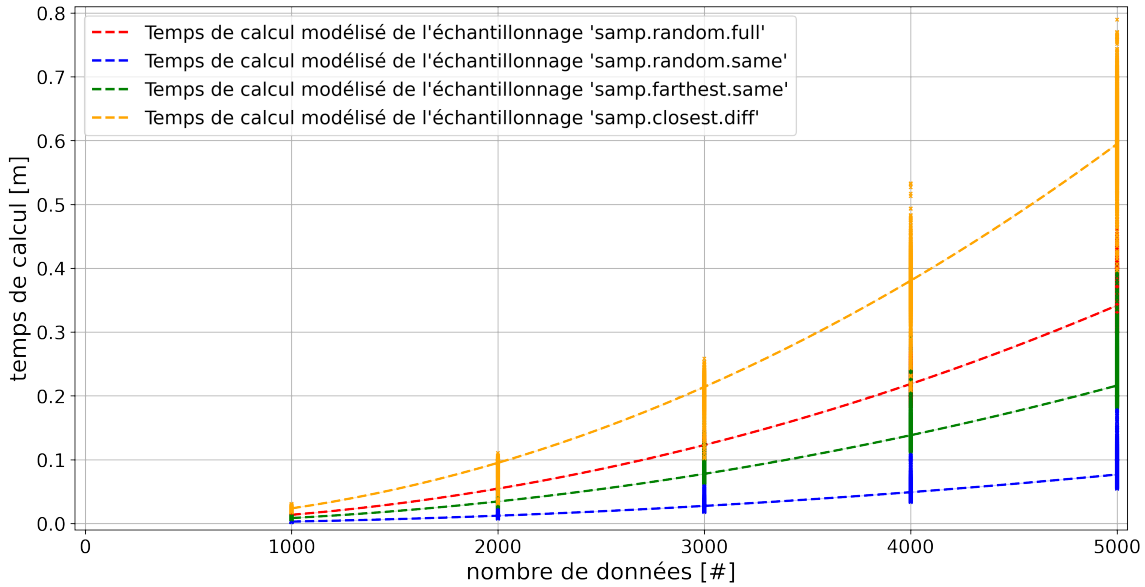

FIGURE 4.18 – *Estimation du temps nécessaire (en minutes) pour effectuer une tâche d'**échantillonnage de contraintes** en fonction du nombre de données à traiter.*

En premier lieu, nous pouvons constater que les différentes modélisations dépendent majoritairement de la taille du jeu de données manipulé (`dataset_size` ou $\texttt{dataset\_size}^2$) avec un score de corrélation `r` avec le temps mesuré généralement supérieur à 0.9 et des modèles *GLM* avec des coefficients de détermination généralisé $\texttt{R}^2$ généralement proches de 0.999. Bien que d'autres facteurs peuvent intervenir dans ces estimations (notamment les interactions doubles entre la taille du jeu de données et le nombre de *clusters* ou le nombre de contraintes), ces derniers semblent avoir un impact négligeable sur le temps d'exécution.

Notes de l'auteur : Certains paramétrages de la méthode du `Clustering Interactif` semblent cependant avoir un temps de calcul décroissant au cours des itérations, mais nous n'avons cependant pas pu montrer de tendances globales significatives. Il est probable que l'ajout de contraintes judicieusement placées permettent à certains algorithmes de *clustering* de s'exécuter plus rapidement, notamment lorsque ceux-ci exploitent les composants connexes du graphe de contraintes (cf. ANNEXE C.1.2). En effet, :

— les *clustering* hiérarchiques s'initialisent avec autant de *clusters* que de groupes de données liées entre elles par des contraintes `MUST-LINK` : or s'il y a plus de contraintes, alors les composants connexes sont davantage développés, donc il y a moins de *clusters* à initialiser et donc moins d'époques de l'algorithme ;

— le *clustering* `KMeans` (modèle `COP`) attire auprès d'un barycentre l'ensemble des données liées par un `MUST-LINK` : or s'il y a plus de contraintes, alors il y a des données attirées, donc les noyaux de *clusters* peuvent se stabiliser plus rapidement.

Toutefois, ces suppositions n'ont pas pu être démontrées, et certains contre-exemples tendent à conclure que ces comportements sont très dépendants du jeu de données manipulé et de l'ordre d'ajout des contraintes. Par exemple :

— l'ajout d'un trop grand nombre de contraintes `CANNOT-LINK` peut engendrer un surplus de vérifications pour estimer quelles formations de *clusters* sont autorisées sans violer de contraintes ;

— l'algorithme `KMeans` (modèle `COP`) peut osciller autour de plusieurs noyaux de *clusters* instables si les contraintes violent trop la similarité intrinsèque des données.

En ce qui concerne la tâche de *clustering*, nous notons des différences significatives dans les temps d'exécution des divers algorithmes implémentés. En effet, l'algorithme `KMeans` (modèle `COP`) est nettement plus rapide (complexité estimée en $\mathcal{O}(\texttt{dataset\_size})$, nécessitant quelques dizaines de minutes pour 5 000 données) que les implémentations du *clustering* hiérarchique (complexité estimée en $\mathcal{O}(\texttt{dataset\_size}^2)$, nécessitant plusieurs heures dès 3 000 données). Cette différence, visible en Figure 4.17, a un réel impact sur l'expérience utilisateur de l'opérateur. En effet, bien qu'il soit théoriquement plus efficient pour atteindre une annotation suffisante (cf. hypothèse d'efficience en Section 4.2), l'usage d'un *clustering* hiérarchique imposerait de longs temps d'attente à l'opérateur, interdisant des interactions rapides avec la machine. Or l'intérêt principal de notre méthodologie d'annotation à l'aide du `Clustering Interactif` repose sur ces interactions Homme/Machine via l'ajout régulier de contraintes pertinentes (cf. hypothèse d'efficacité en Section 4.1). Nous décidons donc d'exclure l'usage des algorithmes de *clustering* hiérarchique au profit du *clustering* `KMeans` (modèle `COP`).

Pour information : Dans le cadre d'un projet étudiants avec l'École d'Ingénieurs Télécom Physique Strasbourg (au cours de l'année 2022) visant à implémenter d'autres algorithmes de *clustering* sous contraintes, un raisonnement similaire a été utilisé pour filtrer les algorithmes. Ainsi, l'implémentation de `KMeans` (modèle `MPC`) a été exclue (complexité estimée en $\mathcal{O}(\texttt{dataset\_size}^3)$) et l'implémentation de la propagation par affinité écarte la gestion des contraintes `CANNOT-LINK` pour avoir un temps d'exécution comparable au *clustering* KMeans (modèle COP). L'algorithme `DBScan` (modèle `C-DBScan`) est quant à lui un rival possible avec une complexité estimée en $\mathcal{O}(\texttt{dataset\_size})$.

Les tâches de prétraitements (cf. Figure 4.15), de vectorisation (cf. Figure 4.16), et d'échantillonnage de contraintes (cf. Figure 4.18) ont des complexités presque négligeables en représentant moins de 10% des temps d'exécution du *clustering* (pour 5 000 données : moins de 1 minute pour les trois algorithmes, contre 12.1 minutes pour `clust.kmeans.cop` et 3.5 heures pour `clust.hier.sing`). Nous maintenons donc les paramétrages obtenus pour ces tâches en Section 4.2 sans analyses complémentaires, et nous utilisons l'estimation temporelle du *clustering* `clust.kmeans.cop` majorée de 10%.

Points à retenir : Dans l'optique d'atteindre de manière efficiente 90% de `v-measure`[28]avec un coût global minimal, nous retenons l'usage du **paramétrage favori** constitué des prétraitements simples (`prep.simple`), de la vectorisation `TF-IDF` (`vect.tfidf`), du *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et de l'échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`). Nous estimons le temps d'exécution de ce paramétrage avec l'équation suivante[29] :

$$\texttt{computation\_time(settings.favorite)}\ [s] \ \propto\ 0.17 \cdot \texttt{dataset\_size} \tag{4.15}$$

4.3.3 Étude du nombre de contraintes nécessaires à la convergence vers une vérité terrain préétablie en fonction de la taille du jeu de données

Avec les deux précédentes études, nous sommes capable d'estimer le temps nécessaire à un expert pour annoter des contraintes, et le temps nécessaire à la machine pour proposer un nouveau *clustering* adapté aux suggestions de l'expert. Pour poursuivre nos études et pouvoir estimer le coût total d'un projet d'annotation, il nous reste à estimer le nombre total de contraintes à devoir renseigner en fonction de la taille du jeu de données.

Pour cela, nous allons simuler la création de cette base d'apprentissage en adaptant le protocole utilisé lors de notre étude d'efficacité (cf. Section 4.1.1) : nous employons notre méthode de `Clustering Interactif` avec notre **paramétrage favori**[30] sur des jeux de données de différentes tailles et mesurons le nombre de contraintes nécessaires pour converger vers la vérité terrain.

4.3.3.a Protocole expérimental

> ⚠ **Attention :** Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble.

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.5.

Nous utilisons deux vérités terrains comme références pour cette expérience :

— le jeu de données `Bank Cards (v2.0.0)` : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 1 000 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'Annexe A.1 ;

— le jeu de données `MLSUM FR Train Subset (v1.0.0-schild)` : ce dernier concerne les titres d'articles de journaux issus des catégories de publication les plus communes. Il est composé de 744 titres d'articles rédigés et répartis en 14 classes (*économie*, *sport*, ...). Pour plus de détails, consulter l'Annexe A.2 ;

Cependant, deux jeux de données ne nous permettent pas d'analyser l'impact du nombre de données sur le nombre de contraintes nécessaires pour converger vers une vérité terrain. Pour utiliser facilement plusieurs jeux de données de tailles différentes tout en maîtrisant leur contenu,

28. 90% de `v-measure` : cas d'une annotation dite partielle, dont le paramétrage le plus efficient est constitué des prétraitements simples (`prep.simple`), de la vectorisation `TF-IDF` (`vect.tfidf`), du *clustering* hiérarchique à lien moyen (`clust.hier.avg`) et de l'échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

29. Temps du paramétrage favori : environ 2.8 minutes pour 1 000 données; environ 14.2 minutes pour 5 000 données.

30. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitements simples (`prep.simple`), vectorisation `TF-IDF` (`vect.tfidf`), *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

Données : jeux de données annotées (vérités terrains) de tailles différentes
pour chaque *jeux de données à tester* **faire**
  initialisation (données) : récupérer ou générer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
  vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
  clustering initial : regrouper les données par similarité avec `clust.kmeans.cop` ;
  évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  répéter
    échantillonnage : sélectionner des contraintes avec `samp.closest.diff` ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering : regrouper les données par similarité avec `clust.kmeans.cop` ;
    évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  jusqu'à *annotation de toutes les contraintes possibles*;
analyse : entraîner un modèle linéaire généralisé du nombre de contraintes nécessaires ;
Résultat : modélisation du nombre de contraintes nécessaires pour un jeu de données

ALGORITHME 4.5 – *Description en pseudo-code du protocole expérimental de l'étude du nombre de contraintes nécessaires pour converger vers une vérité terrain préétablie avec notre paramétrage favori du `Clustering Interactif`.*

nous avons donc dupliqué aléatoirement des données issues de ces jeux de référence en y insérant des fautes de frappes. La taille des jeux de données générés, notée `dataset_size`, varie entre 1 000 à 5 000 par pas de 250, et chaque taille de jeu est générée 3 fois pour contrer les aléas statistiques de création. Il y a donc 51 variations de chaque jeu de références, soit 102 jeux utilisés de tailles différentes.

> ⚠ **Attention :** Dans le cadre de cette étude, nous faisons l'hypothèse que cette création artificielle de données n'a pas d'impact majeur sur le nombre de contraintes nécessaires pour converger vers une vérité terrain.

Sur chacun de ces jeux générés, une tentative complète[31] de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori est exécuté, et chaque tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions. Il y a donc 510 tentatives de `Clustering Interactif` réalisées.

Pour chacune de ces tentatives, nous nous intéressons au nombre de contraintes nécessaires pour atteindre le seuil d'annotation partielle (caractérisé par 90% de `v-measure` entre la vérité terrain et la segmentation des données obtenue), et nous entraînons un modèle linéaire généralisé (*GLM*) pour modéliser le nombre de contraintes requis en fonction de la taille du jeu de données (noté `dataset_size`). Ce modèle sera caractérisé par le coefficient de détermination généralisé `R²` de *Cox et Snel* (DIAMOND et al., 1990), la log-vraisemblance `llf` (EDWARDS, 1992) et la log-vraisemblance `llf_null` du modèle *null*. Pour finir, nous discutons des valeurs des coefficients obtenus sur l'impact du nombre d'itérations de la méthode à prévoir.

31. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

Pour information : Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. Nous y utilisons entre autres la librairie `statsmodels` [32] (Seabold et Perktold, 2010). De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.3.3.b Résultats obtenus

Le modèle linéaire généralisé entraîné sur les mesures du nombre de contraintes requis pour atteindre 90% de `v-measure` ($\texttt{R}^2$: > 0.999, `llf` : $-4\ 327.6$, `llf_null` : $-4\ 942.9$) nous permet de déduire l'équation suivante :

$$\texttt{constraints\_needed(settings.favorite)}\ [\#] \propto 3.15 \cdot \texttt{dataset\_size} \tag{4.16}$$

La Figure 4.19 représente cette modélisation.

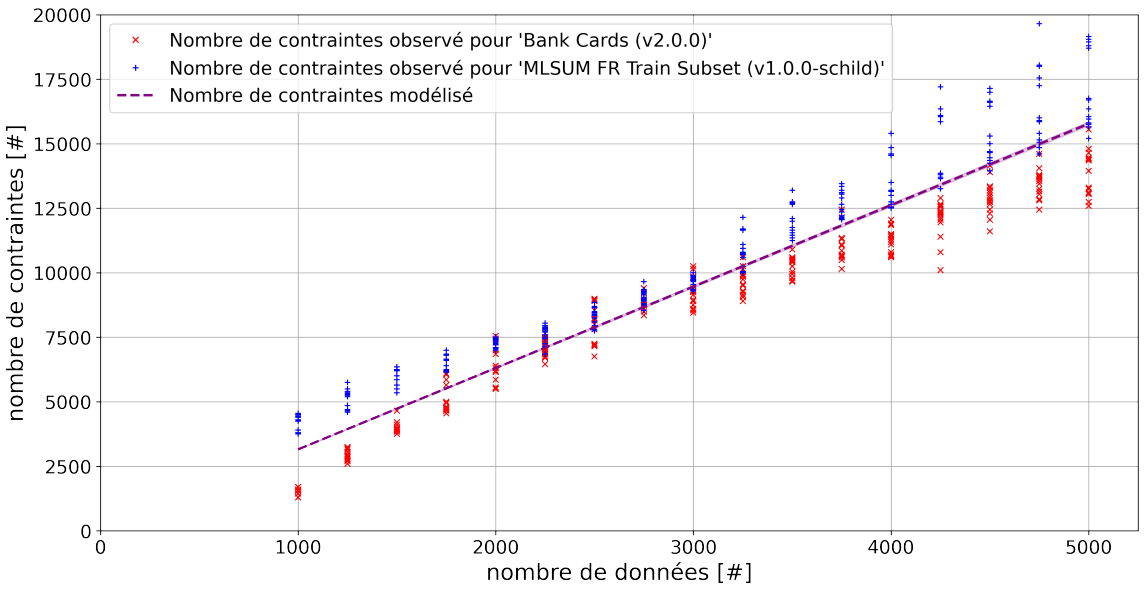

Figure 4.19 – *Estimation du nombre moyen de contraintes nécessaires à notre* ***paramétrage favori*** *du* `Clustering Interactif` *afin d'obtenir une annotation partielle (atteindre une* `v-measure` *de 90%) en fonction de la taille du jeu de données à modéliser.*

Notes de l'auteur : Nous pouvons considérer les points de références suivants :

— le nombre de contraintes possibles (avec doublons) est de $\texttt{dataset\_size}^2$ (*caractériser chaque couple de données présent dans la matrice d'adjacence*) ;

32. https://pypi.org/project/statsmodels/

— le nombre de contraintes possibles (sans doublons) est de $\frac{1}{2} \cdot (\texttt{dataset\_size}^2 - \texttt{dataset\_size})$ (*considérer la symétrie des contraintes, donc seul le triangle supérieur de la matrice d'adjacence a besoin d'être renseigné*) ;

— le nombre minimal de contraintes à annoter pour être exhaustif sur une partition en k *clusters* $\{K_1, K_2, ..., K_k\}$ est estimé à $\sum_{1 \leq i \leq k} (\|K_i\| - 1) + \sum_{1 \leq i \leq k} (k - i)$ (*il faut d'abord considérer les chemins minimaux pour parcourir les composants connexes avec des contraintes* `MUST-LINK`*, correspondant à* $\|K_i\| - 1$ *contraintes* `MUST-LINK` *pour chaque partition* $\|K_i\|$*, puis ajouter le nombre minimal des contraintes* `CANNOT-LINK` *pour distinguer chacun de ses composants connexes en cluster*, correspondant au nombre d'arrangements sans répétition de deux partitions).

La Figure 4.20 illustre ces propos sur un jeu d'exemples comportant 10 points de données réparties en 3 classes, et met en avant l'explosion du nombre de contraintes possible même sur un petit jeu de données (cf. 4.20 **(2)**).

Avec ces références, le nombre de contraintes est borné approximativement entre 1 035 et 499 500 pour un jeu de 1 000 données équilibré en 10 classes, et entre 6 175 et 12 497 500 pour un jeu de 5 000 données équilibré en 50 classes.

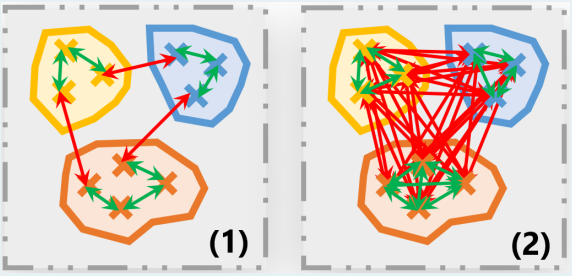

Figure 4.20 – *Exemple de caractérisation exhaustive d'un jeu de données (10 données, 3 classes) en ajoutant un nombre minimal de contraintes (cf.* ***(1)****) ou en ajoutant toutes les contraintes possibles (cf.* ***(2)****).*

4.3.3.c Discussion

L'objectif de cette étude était de déterminer le nombre moyen de contraintes à devoir annoter pour modéliser un jeu de données pour un accord 90% de `v-measure` avec la vérité terrain utilisée. Cette estimation, dépendant de la taille du jeu de données manipulé, est représentée par l'Equation 4.16.

Nous pouvons constater que la relation entre la taille du jeu de données et le nombre de contraintes à annoter est linéaire (pente de 3.15) : doubler la taille d'un jeu de données doublera donc la charge de travail incombant à l'expert métier. À première vue, une telle estimation représente une lourde charge d'annotation : **pour un jeu de 5 000 données, il faut caractériser 15 750 contraintes, ce qui correspond environ à 34 heures d'annotation** d'après l'Equation 4.1 ! Néanmoins, comme le nombre de contraintes possibles évolue en $\mathcal{O}(\texttt{dataset\_size})$, cette estimation aurait pu être bien pire et représenter 12 497 500 contraintes

(cf. FIGURE 4.20). Mieux encore, le nombre théorique minimal moyen de contraintes à annoter pour 5 000 données n'est que 2.55 fois plus faible que notre estimation (6 175 vs 15 750), alors que cette borne minimale nécessite un échantillonnage "*parfait*" permettant d'identifier le chemin minimal parcourant les *clusters*. Nous pouvons donc relativiser l'estimation faite avec l'EQUATION 4.16 et en conclure que notre méthode comporte un nombre de contraintes raisonnable à annoter.

Bien évidemment, une telle estimation est sensible au jeu de données utilisé comme référence (cf. FIGURE 4.19). Ici, la différence de pente mesurée est de 0.25 (`p-valeur` : > 0.999), soit un écart moyen d'environ 8% par rapport à la modélisation moyenne. Toutefois, comme l'impact semble limité, nous maintenons la modélisation moyenne représentée par l'EQUATION 4.16 pour la suite de nos estimations de coûts.

> **Notes de l'auteur :** Il n'y a pas davantage de matière à discussion pour cette étude, car le principal résultat (l'EQUATION 4.16) est un résultat temporaire nécessaire à l'estimation du coût global d'un projet utilisant une méthodologie de `Clustering Interactif`.

4.3.4 Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts

Résumons l'ensemble des modélisations réalisées lors des précédentes études (cf. sections 4.3.1, 4.3.2 et 4.3.3) afin d'estimer le coût total d'un projet d'annotation employant une méthodologie basée sur le `Clustering Interactif` et utilisant notre **paramétrage favori**[33]. Dans les notations, `dataset_size` représente la taille du jeu de données à modéliser, et `batch_size` représente le nombre de contraintes que l'expert annote à chaque itération.

4.3.4.a Synthèse des résultats

Tout d'abord, nous pouvons estimer le **temps moyen d'une itération de la méthode**, comprenant d'une part les temps d'exécution des algorithmes (*prétraitements*, *vectorisation*, *clustering*, *échantillonnage*) et d'autre part le temps d'annotation d'un lot de contraintes, grâce aux équations suivantes :

$$\begin{cases} \texttt{computation\_time}\ [s] & \propto 0.17 \cdot \texttt{dataset\_size} \\ \texttt{annotation\_time}\ [s] & \propto 7.8 \cdot \texttt{batch\_size} \\ \texttt{iteration\_time(sequential)}\ [s] & \propto \texttt{computation\_time} + \texttt{annotation\_time} \end{cases} \tag{4.17}$$

Ensuite, nous sommes en mesure d'anticiper le **nombre moyen de contraintes à annoter** pour modéliser le jeu de données avec un seuil de 90% de `v-measure`, et donc de déduire le nombre d'itérations nécessaire de la méthode, grâce aux équations suivantes :

$$\begin{cases} \texttt{constraints\_needed}\ [\#] & \propto 3.15 \cdot \texttt{dataset\_size} \\ \texttt{iterations\_needed}\ [\#] & \propto \frac{\texttt{nb\_constraints}}{\texttt{batch\_size}} \end{cases} \tag{4.18}$$

33. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitements simples (`prep.simple`), vectorisation `TF-IDF` (`vect.tfidf`), *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

Enfin, il suffit de combiner Equation 4.17 et Equation 4.18 pour estimer le temps total nécessaire à un projet d'annotation utilisant le `Clustering Interactif` (c'est-à-dire en enchaînant successivement des étapes d'échantillonnage, d'annotation et de *clustering*, cf. Figure 4.21 (1)) pour converger vers 90% de `v-measure` :

$$\left\{\texttt{total\_time(sequential)}\ [s] \quad \propto \texttt{iteration\_time} \cdot \texttt{iterations\_needed} \right. \tag{4.19}$$

Ces estimations globales sont représentées sur la Figure 4.22 en fonction de plusieurs tailles de jeu de données et plusieurs tailles de lots d'annotation.

4.3.4.b Discussion du coût total

L'objectif de cette section consistait à déterminer les coûts relatifs à la tâche d'annotation de contraintes par un expert métier et au temps d'exécution des algorithmes intervenant dans notre implémentation du `Clustering Interactif`. Pour cela, nous avons chronométré des annotateurs en situation réelle (cf. Section 4.3.1), estimé le temps de calcul de chaque algorithme implémenté (cf. Section 4.3.2) et trouvé le moyen de prédire le nombre de contraintes à annoter sur un jeu de données (cf. Section 4.3.3). Nous avons pu montrer qu'une annotation de contraintes est plus rapide qu'une annotation par label et nous conseillons, d'après notre analyse empirique, des sessions d'annotation de moins de 150 contraintes pour ne pas épuiser l'annotateur. Sur le paramétrage de la méthode, nous avons rejeté l'usage d'algorithmes de *clustering* de type hiérarchique à cause de leur lenteur, au profit du `KMeans` avec modèle `COP` (`clust.kmeans.cop`). Notre paramétrage favori, permettant d'atteindre 90% de `v-measure` avec un coût minimal, est ainsi constitué des prétraitements simples (`prep.simple`), d'une vectorisation `TF-IDF` (`vect.tfidf`), d'un *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et d'un échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`). Enfin, pour atteindre cet objectif de `v-measure`, le nombre moyen de contraintes à annoter avec notre méthodologie semble rester linéairement proportionnel à la taille du jeu de données à modéliser, ce qui est encourageant au regard de la combinatoire de contraintes possibles.

La mise en commun de ces résultats se retrouve dans Equation 4.17, Equation 4.18 et Equation 4.19. Comme le nombre de données à annoter est inversement proportionnel au nombre d'itérations à réaliser, nous avons le dilemme suivant : soit nous annotons de petits lots (50 contraintes) pour rapidement intégrer les contraintes et permettre à l'annotateur de se reposer régulièrement (*mais ce dernier va en contrepartie devoir attendre plus souvent la fin des exécutions du clustering*) soit nous annotons des lots plus conséquents (150 contraintes) pour diminuer le nombre d'itérations et exécuter moins de *clustering*, (*mais cela risque d'épuiser l'opérateur avec des grosses charges d'annotations*). Dans les deux cas, **le coût total semble élevé** : pour un jeu de 5 000 points de données, et avec des tailles d'échantillons comprises entre 50 et 150 contraintes, il faut entre 59 et 110 heures de travail (34 *heures d'annotations et entre* 25 *et* 76 *heures d'attente de la fin d'exécution d'algorithmes suivant la taille des lots*). En considérant une journée de travail de 7 heures, cela représente une charge contenue entre 8.4 et 15.7 jours pour avoir un jeu de donnée fiable à 90% de `v-measure`. Pour finir, nous ajoutons 1 jour pour combler l'écart théorique de 10% de `v-measure` en corrigeant manuellement le résultat obtenu, et 1 jour supplémentaire pour interpréter et nommer chaque *cluster* (voir Algorithme 3.1, *ligne 13*). Au final, pour un ensemble de 5 000 données, il faut donc entre 8.4 (+2) et 15.7 (+2) jours de travail à un expert métier pour obtenir une base d'apprentissage avec une méthodologie basée sur le `Clustering Interactif`.

Pour critiquer l'approximation que nous avons faite lors de cette section, nous essayons de comparer le temps nécessaire qu'il aurait fallu à un projet d'annotation traditionnel, comme décrit en SECTION 2.2.1 (cycle `MATTER`), et notre proposition de cycle d'annotation avec le `Clustering Interactif`, visible en FIGURE 4.21 (1). En nous basant sur un ensemble de 5 000 données à annoter, nous estimons qu'il faut : 1 jour de travail pour définir une représentation des données en modèle de classification ; 4 jours de travail pour annoter 5 000 données (à raison de 20 secondes par données, estimation haute inspirée de PRADHAN et al., 2007 où la "*désambiguïsation du sens des mots*", présentée comme une tâche de classification, demande 17.5 secondes par données) ; 2 jours de marge d'erreur pour changer de modèle de représentation s'il ne semble pas adapté aux données en cours d'annotation. Au total, nous estimons donc qu'il faut 5 (+2) jours de travail[34] à un expert métier pour obtenir une base d'apprentissage avec une méthodologie traditionnelle. **En l'état, nous pouvons donc conclure que le `Clustering Interactif` que nous proposons est en moyenne deux fois plus coûteux qu'une annotation manuelle classique (**8.4 (+2) **à** 15.7 (+2) **jours vs** 5 (+2) **jours), ce qui peut être un frein à son utilisation en situation réelle**.

Afin d'accélérer la phase d'annotation et de diminuer le nombre d'itérations, il est bien entendu possible d'augmenter la taille des échantillons de contraintes à annoter ou d'ajouter plusieurs opérateurs. Cependant, une telle solution ne permet toujours pas d'être compétitif avec l'annotation traditionnelle si cette dernière dispose aussi de plusieurs opérateurs (*avec* 2 *annotateurs : de* 6.5 (+2) *à* 13.3 (+2) *jours vs* 3 (+2) *jours ; avec* 4 *annotateurs : de* 4.8 (+2) *à* 12.1 (+2) *jours vs* 2 (+2) *jours*). Une autre piste, plus prometteuse, consiste plutôt à adapter notre méthode pour exploiter les temps d'attente lors de l'exécution d'un *clustering*.

4.3.5 Ouverture vers une annotation en parallèle du *clustering*

Notre méthodologie d'annotation basée sur le `Clustering Interactif` est pénalisée par la séquentialité des actions à réaliser. En effet, l'annotateur doit attendre la fin de l'exécution des algorithmes de *clustering* et d'échantillonnage avant de pouvoir travailler. D'après nos précédentes estimations sur un jeu de 5 000 données, l'opérateur doit attendre entre 25 et 76 heures en fonction de la taille de lots d'annotation choisie, et une idée à explorer consiste à optimiser ce temps d'attente.

L'amélioration envisagée consiste à **réaliser l'annotation de contraintes en parallèle de l'exécution du *clustering***, comme représenté dans la FIGURE 4.21 (2). Dans cette version, l'échantillonnage se base toujours sur le résultat du *clustering* de l'itération précédente, mais le *clustering* intègre les contraintes annotées avec un décalage d'une itération. Un tel changement permet de limiter le temps d'attente de l'opérateur et d'optimiser l'enchaînement des algorithmes.

Avec cette version, le temps nécessaire à une itération correspond à la durée la plus longue entre le temps d'annotation et le temps de calcul des algorithmes. Nous pouvons donc adapter l'EQUATION 4.17 par l'équation suivante :

$$\begin{cases} \texttt{computation\_time}\ [s] & \propto 0.17 \cdot \texttt{dataset\_size} \\ \texttt{annotation\_time}\ [s] & \propto 7.8 \cdot \texttt{batch\_size} \\ \texttt{iteration\_time(parallel)}\ [s] & \propto max(\texttt{computation\_time}, \texttt{annotation\_time}) \end{cases} \tag{4.20}$$

34. 5 (+2) jours de travail : Notons que cette estimation n'est bien entendu pas généralisable. En effet, le temps nécessaire aux phases de modélisation et de revues peut fortement augmenter si le cas d'usage est plus complexe. Par exemple, la classification de questions en une centaine d'intentions peut prendre plusieurs jours voire quelques semaines d'étude alors que la classification de sentiments sur trois niveaux (positif, négatif, neutre) est presque triviale.

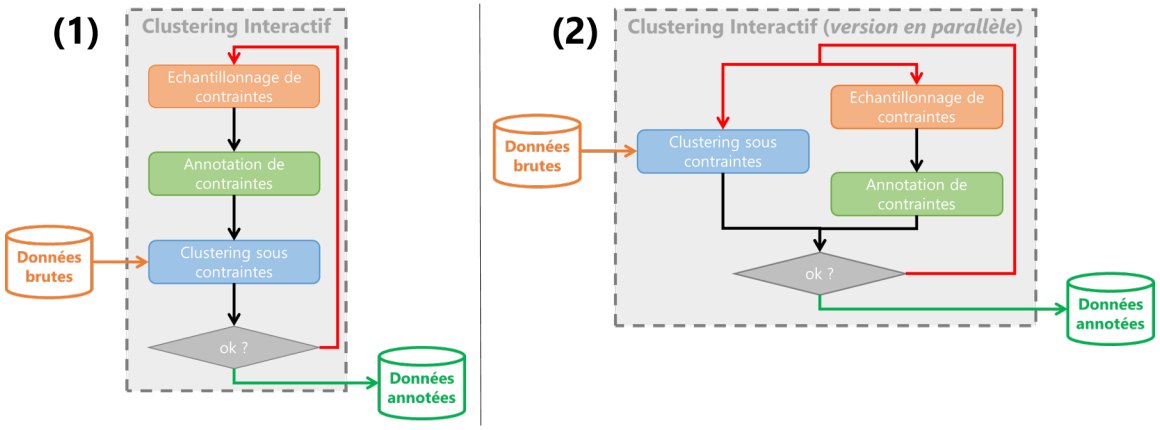

FIGURE 4.21 – *Schéma comparatif des architectures du* `Clustering Interactif` *: (1) représente la version séquentielle initialement présentée en* CHAPITRE 3 *où le clustering s'adapte avec les annotations de l'itération en cours ; (2) représente l'évolution en mode parallèle où le clustering s'adapte avec les annotations de l'itération précédente (décalage d'une itération).*

Ensuite, afin de limiter les pertes de temps (humain et machine), nous pouvons choisir une taille de lot d'annotations rendant ces deux durées équivalentes. Nous déduisons donc les changements suivants dans l'EQUATION 4.18 :

$$\begin{cases} \texttt{optimal\_batch\_size}\ [\#] \propto \dfrac{\texttt{computation\_time}}{7.8} \propto 0.0218 \cdot \texttt{dataset\_size} \\ \texttt{iterations\_needed}\ [\#] \propto \dfrac{\texttt{nb\_constraints}}{\texttt{optimal\_batch\_size}} \propto 144.5 \end{cases} \tag{4.21}$$

Enfin, il suffit de combiner EQUATION 4.20 et EQUATION 4.21 pour estimer le temps total nécessaire à un projet d'annotation pour converger vers 90% de `v-measure` utilisant la version parallèle du `Clustering Interactif` :

$$\begin{cases} \texttt{total\_time(parallel)}\ [s] \propto \texttt{iteration\_time} \cdot \texttt{iterations\_needed} \\ \texttt{total\_time(parallel)}\ [s] \propto 24.6 \cdot \texttt{dataset\_size} \end{cases} \tag{4.22}$$

Ces estimations mises à jour sont représentées sur la FIGURE 4.22 en fonction de plusieurs taille de jeu de données, et permettent de faire la comparaison avec la version séquentielle initialement présentée.

Nous pouvons déjà remarquer que le coût d'annotation du projet devient linéaire en nombre de données (pente de 24.6 secondes) et nécessite un nombre fixe de 145 itérations. En reprenant une base de 5 000 données et une marge de 2 jours pour corriger et nommer les *clusters*[35], cela représente 4.8 (+2) jours de travail, un estimation équivalente à une annotation traditionnelle qui nécessite 5 (+2) jours de travail d'après nos approximations. De plus, si nous ajoutons plusieurs opérateurs, cette version parallèle reste compétitive (*avec* 2 *annotateurs :* 2.5 (+2) *jours vs* 3 (+2) *jours ; avec* 4 *annotateurs :* 1.3 *jours vs* 2 (+2) *jours*). Cette découverte est très encourageante, car cela confirme qu'une méthodologie basée sur notre implémentation du `Clustering Interactif` **permet d'obtenir une base d'apprentissage avec un coût temporel équivalent à un projet traditionnel** utilisant une annotation par label. Cette méthode est d'autant plus intéressante qu'elle fait intervenir un mécanisme d'annotation rapide et intuitif pour un expert métier.

35. Nommer les *clusters* : voir ALGORITHME 3.1, *ligne 13*.

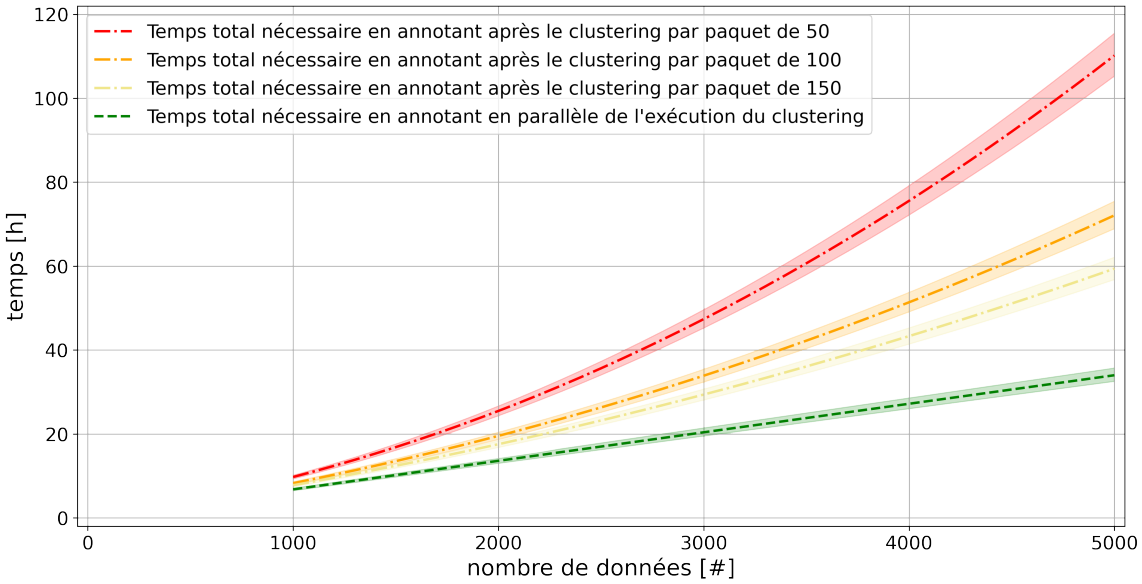

FIGURE 4.22 – *Estimation du temps total nécessaire (en heures) pour modéliser un jeu de données avec notre* ***paramétrage favori*** *du* `Clustering Interactif` *afin d'obtenir une annotation partielle (atteindre une* `v-measure` *de 90%), en fonction de plusieurs tailles de jeu de données, plusieurs tailles de lots d'annotation, et mettant en opposition l'approche séquentielle (annotation puis le clustering) et l'approche parallèle (annotation pendant le clustering).*

Points à retenir : Au cours de cette étude de coûts, nous avons pu déduire que :

- ☑ L'annotation d'une contrainte nécessite en moyenne 8 secondes : cette tâche est rapide et intuitive (cf. SECTION 4.3.1) ;
- ☑ Notre paramétrage favori, permettant d'atteindre 90% de `v-measure` avec un coût minimal, est constitué des prétraitements simples (`prep.simple`), de la vectorisation `TF-IDF` (`vect.tfidf`), du *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et de l'échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`). Ce paramétrage a un coût moyen de $0.17 \cdot$ `dataset_size` secondes (cf. SECTION 4.3.2) ;
- ☑ Une adaptation optimale de notre méthodologie consiste mettre en parallèle l'exécution du *clustering* avec l'annotation de contraintes afin de limiter les temps d'attente inutiles. Une telle méthode a un coût moyen de $24.6 \cdot$ `dataset_size` pour atteindre 90% de `v-measure`, auquel nous ajoutons 2 jours de travail pour raffiner les *clusters* et les nommer. Ce temps est compétitif avec une annotation traditionnelle (cf. SECTION 4.3.4) ;
- ☑ Cette étude met en avant l'intérêt des interactions homme-machine : (1) l'expert métier se recentre sur son domaine de compétence avec une caractérisation proche de ses connaissances ("*les données sont-elles similaires ?*") et (2) la machine optimise l'intervention de l'expert pour que ce dernier soit toujours pertinent dans ses contributions.

Dans les sections suivantes, nous allons nous intéresser à l'analyse des résultats de cette méthode. En effet, en situation réelle, nous n'avons pas accès à la vérité terrain car elle est justement en cours de construction. Il nous est donc impossible d'estimer notre seuil de `v-measure`, et donc incapable de s'arrêter à 90% de `v-measure`. Nous nous intéressons donc à l'estimation de la valeur métier d'un résultat de *clustering* (cf. hypothèse de pertinence en Section 4.4) et à la définition de cas d'arrêt indépendants d'une vérité terrain (cf. hypothèse de rentabilité en Section 4.5).

4.4 Évaluation de l'hypothèse de pertinence

Jusqu'à présent, nous avons analysé la performance et l'évolution des résultats de notre implémentation du `Clustering Interactif` en calculant la similarité (en `v-measure`) avec une vérité terrain. Cependant, une telle référence n'est pas accessible en situation réelle car l'objectif de notre méthode est précisément de construire cette vérité terrain. Nous devons donc nous intéresser à d'autres moyens pour estimer la pertinence et l'exploitabilité des bases d'apprentissage obtenues. Ainsi, nous aimerions vérifier l'hypothèse suivante :

Hypothèse de pertinence

« **Au cours d'une méthodologie d'annotation basée sur le `Clustering Interactif`, il est possible à un expert métier d'évaluer rapidement la pertinence de la base d'apprentissage en construction sans utiliser de vérité terrain.** »

La FIGURE 4.23 illustre cette hypothèse et la perspective de pouvoir caractériser la qualité de la base d'apprentissage en cours de construction en fonction d'une valeur métier exprimée par un expert.

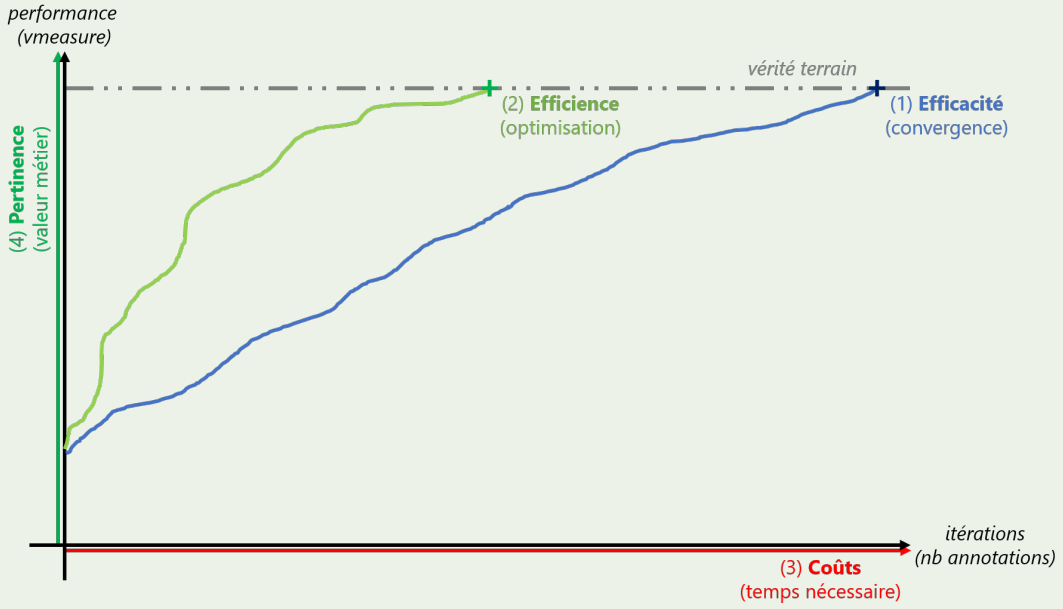

FIGURE 4.23 – *Illustration des études réalisées sur le `Clustering Interactif` (étape 4/6) en schématisant l'évolution de la **pertinence** (valeur métier évaluée par l'expert, exprimée en nombre de clusters) d'une base d'apprentissage en cours de construction, en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine).*

Afin de vérifier cette hypothèse, nous explorons trois approches :

— une **validation par un expert** du partitionnement des données obtenus, en parcourant manuellement le contenu des *clusters* et en donnant un avis sur l'exploitabilité de ces derniers (cf. SECTION 4.4.1) ;

— une analyse des **patterns linguistiques saillants** dans la base d'apprentissage à l'aide d'une stratégie de sélection des composantes principales d'un modèle (cf. Section 4.4.2) ; et

— une approche utilisant un **résumé automatique de thématique** par un large modèle de langage (`LLM`), permettant de décrire succinctement le contenu des *clusters* en une phrase (cf. Section 4.4.3).

4.4.1 Étude d'une validation manuelle et non assistée de la valeur métier d'une base d'apprentissage par un expert

Afin d'estimer la pertinence d'un résultat de *clustering*, notre première intuition consiste à demander simplement l'avis d'un expert sur la base d'apprentissage en cours de construction. En lui posant certaines questions, nous espérons obtenir une description qualitative de chaque *cluster* et ainsi déduire quand le résultat du `Clustering Interactif` devient exploitable pour définir et entraîner un modèle de classification.

4.4.1.a Protocole expérimental

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.6.

Données : jeux de données annotées (vérités terrains)
pour chaque *jeu de données à tester* **faire**
  initialisation (données) : récupérer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
  vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
  clustering initial : regrouper les données par similarité avec `clust.kmeans.cop` ;
  évaluation manuelle : juger de l'exploitabilité de chaque *cluster* ;
  labellisation manuelle : nommer chaque *cluster* exploitable ;
  répéter
    échantillonnage : sélectionner des contraintes avec `samp.closest.diff` ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering : regrouper les données par similarité avec `clust.kmeans.cop` ;
    évaluation manuelle : juger de l'exploitabilité de chaque *cluster* ;
    labellisation manuelle : nommer chaque *cluster* exploitable ;
  jusqu'à *annotation de toutes les contraintes possibles*;
analyse : afficher l'évolution de l'exploitabilité de chaque itération de *clustering* ;
Résultat : discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

Algorithme 4.6 – *Description en pseudo-code du protocole expérimental de l'étude de validation manuelle non assistée de la valeur métier d'une base d'apprentissage.*

Nous utilisons comme vérité terrain le jeu de données `Bank Cards (v1.0.0)` : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'Annexe A.1.

Sur ce jeu de données, nous exécutons une tentative complète [36] de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori [37] (voir Section 4.3), et cette tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions.

Au cours des itérations, un expert qualifie chaque *cluster* en donnant son avis sur sa valeur métier. Afin d'encadrer ses réponses, nous lui demandons d'analyser trois aspects :

— est-ce que le *cluster* a une thématique principale bien définie ? (*en effet, comment interpréter un cluster sans définition claire ?*)

— est-ce que le *cluster* est constitué par un nombre suffisant de données ? (*en effet, comment entraîner un modèle de classification sans données ?*)

— est-ce que le *cluster* n'est pas trop bruité ? (*en effet, comment avoir de bonnes performances si la base d'apprentissage n'est pas fiable ?*)

L'avis exprimé par l'expert métier est alors classé en trois niveaux :

— **exploitable** : le *cluster* possède (1) une thématique bien définie, (2) un nombre de données suffisant pour entraîner un modèle de classification et (3) peu de bruit ; ce *cluster* peut donc être exploité en l'état ou avec peu de modifications manuelles ;

— **partiellement exploitable** : soit le *cluster* est composé de plusieurs de thématiques (*deux ou trois*), soit il ne comporte pas assez de données (*moins d'une vingtaine*), soit il est bruité (*au moins un quart de bruit*) ; ce *cluster* donne une première base pour créer une classe, mais un travail manuel est nécessaire (*ajout de données, tri du bruit, ...*) ;

— **non exploitable** : soit le *cluster* ne contient pas ou contient trop thématique, soit c'est un *cluster* singleton ou un *cluster* de données trop hétérogènes, soit ce *cluster* est complètement bruité ; dans tous les cas, il n'est absolument pas exploitable sans un gros travail manuel.

Pour limiter la charge de travail de l'opérateur, nous ne demandons l'expertise que toutes les 5 itérations d'une tentative.

> ⚠ **Attention :** Par manque de personnes aptes à qualifier le jeu de données utilisé, les annotations de cette étude ont été réalisées par un seul opérateur (*moi-même, ayant participé à la création de ce jeu de données*). Nous supposons que cette contrainte n'est pas pénalisante pour l'analyse : en effet, dans une situation réelle, cet opérateur serait responsable des choix à entreprendre pour concevoir le jeu de données, il est donc le mieux placé pour juger de la pertinence d'un *clustering* par rapport à sa propre modélisation du problème. Les problèmes d'accords inter-annotateurs seront plutôt discutés en Section 4.6. Nous réalisons toutefois 5 tentatives différentes de la méthode pour limiter les biais intra-individuels.

> ℹ **Pour information :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild,

36. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

37. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitements simples (`prep.simple`), vectorisation `TF-IDF` (`vect.tfidf`), *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.4.1.b Résultats obtenus

La Figure 4.24 met en avant l'évolution de la pertinence moyenne estimée par l'opérateur sur la base des contenus des *clusters*. Nous allons nous intéresser à trois phases s'y distinguant.

À l'initialisation (itération 0), la majeure partie des *clusters* sont inexploitables (environ 60%) et seuls 35% d'entre eux semblent exploitables. Dans le top 3 des classes facilement identifiables à ce stade, nous retrouvons `gestion_sans_contact` (5/5), `consultation_solde` (3/5) et `gestion_carte_virtuelle` (3/5).

Nous constatons ensuite une première phase de remaniement des *clusters*, située entre les itérations 0 et 10, où le taux d'inexploitables chute au profit des *clusters* partiellement exploitables, dont la proportion augmente de 10% à près de 40%. À l'itération 10, le top 3 des classes identifiables mais bruitées ou en cohabitation dans un *cluster* sont `gestion_carte_virtuelle` (4/5), `alerte_perte_vol_carte` (4/5) et `commande_carte` (4/5).

Une seconde phase de consolidation se présente entre les itérations 10 et 25. Durant cette phase, les taux de *clusters* non exploitables et de partiellement exploitables diminuent alors que le taux d'exploitables monte en flèche (de 35% à 90% en 15 itérations). La majeure partie des `clusters` sont ainsi exploitables en l'état ou après la correction de quelques points aberrants. Après l'itération 25, le *cluster* le plus récalcitrant concerne un mélange des classes `alerte_perte_vol_carte` et `gestion_decouvert` (5/5).

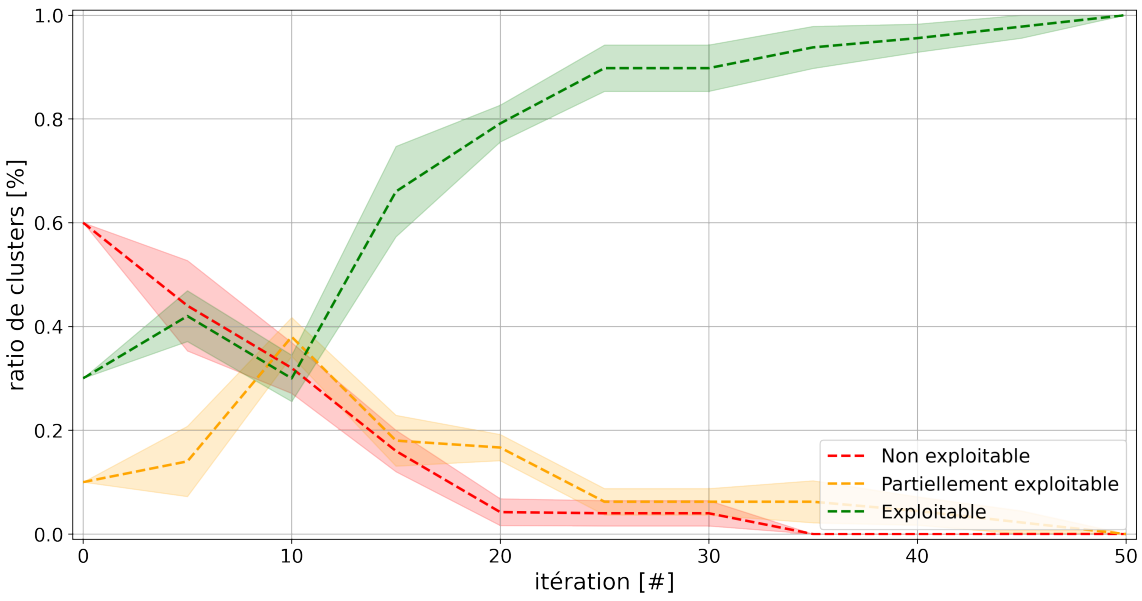

Figure 4.24 – *Évolution de la pertinence métier moyenne estimée manuellement au cours des itérations du résultat du `Clustering Interactif` avec notre paramétrage favori. Cette pertinence, exprimée en proportion du nombre de clusters, est retranscrite en trois niveaux : `exploitable` en vert, `partiellement exploitable` en orange, et `non exploitable` en rouge.*

4.4.1.c Discussion

Cette première étude visait à observer comment un expert métier peut interpréter un résultat de *clustering* proposé par notre méthode. Nous avons donc validé manuellement tous les *clusters* et essayé de les labelliser.

Comme l'analyse n'a pu être faite que par un seul opérateur, nous ne nous attardons pas sur la précision des taux d'exploitabilité des *clusters*. Nous pouvons déjà reconnaître que trois phases sont présentes :

— une première **phase exploratoire** (cf. itérations 0 à 10), où des premiers *cluster* partiellement exploitables apparaissent. Ces derniers contiennent souvent une thématique bruitée ou quelques thématiques mal séparées, ce qui permet toutefois à l'expert de se faire une idée des thématiques contenues dans la collecte de données. Cependant, s'arrêter à ce stade demanderait beaucoup de travail manuel pour obtenir une base d'apprentissage opérationnelle ;

— une seconde **phase de consolidation** (cf. itérations 10 à 25), où les *clusters* partiellement exploitables s'affinent. De plus en plus de *clusters* bien définis naissent, avec une seule thématique et peu de bruits. Ces derniers pourraient être extraits et exploités en l'état ;

— une **phase de parachèvement** (cf. itérations après 25), où la plupart des *clusters* sont exploitables en l'état mais quelques-uns nécessitent encore du travail. Cette phase est la moins rentable car les derniers *cluster* et points aberrants sont corrigés petit à petit, mais sans impact notable sur leur valeur métier.

Au cours de ces validations, il apparaît que la complexité réside dans l'analyse des *clusters* partiellement exploitables. En effet, les *clusters* totalement exploitables ou totalement inexploitables sont souvent simples à identifier. les premiers se repèrent facilement, surtout quand ils sont déjà présents lors d'itérations précédentes (exemple de la classe `gestion_sans_contact`, présente dès l'itération 0) ; les seconds, plutôt présents au début de la méthode, peuvent mélanger un grand nombre de thématiques et s'apparenter rapidement à des *clusters* poubelle. En revanche, les *clusters* partiellement exploitables peuvent avoir des limites subjectives, altérant parfois l'avis de l'expert.

Pour aller plus loin, la difficulté de cette tâche est aussi due aux contraintes suivantes :

— il faut être capable de maintenir en mémoire de grands ensembles de données ;

— il faut estimer la thématique principale à l'aide de données désordonnées ;

— il faut examiner la cohérence de cette thématique alors que le vocabulaire employé diffère ;

— il faut juger de l'importance du bruit contenu dans les *clusters* ;

— il faut prendre du recul pour repérer la dispersion d'une thématique dans plusieurs *clusters* ;

— ...

Tous ces facteurs peuvent donc nuire à la qualité, tant sur l'estimation de l'exploitabilité que sur le nommage des *clusters*.

Ainsi, nous déduisons aisément que l'expérience utilisateur proposée à l'expert métier induit une charge mentale élevée. Comme l'analyse de l'exploitabilité d'un *cluster* semble être une tâche complexe, il semble inconcevable de demander sa réalisation sur tous les *clusters* de toutes les

itérations ! De plus, sans aide supplémentaire et sans vis-à-vis pour se confronter à ses conclusions d'analyse, l'opérateur peut difficilement vérifier la cohérence et la reproductibilité de son travail. Il est donc nécessaire de considérer des pistes d'amélioration pour ne pas abandonner l'expert métier à lui-même dans cette tâche cruciale.

Quelques pistes peuvent être étudiées pour accompagner l'opérateur :

— employer plusieurs experts pour valider par consensus leurs appréciations sur un résultat de *clustering* : cette méthode est efficace pour rattraper les thématiques non identifiées et pour s'accorder sur la valeur d'un *cluster* ;

— préparer le travail d'analyse en réalisant une étude linguistique des *clusters*, et permettre ainsi d'identifier grossièrement les thématiques présentes en fonction du vocabulaire employé (cf. Section 4.4.2) ;

— automatiser une partie du travail d'analyse en utilisant les capacités des larges modèles de langage (*large language models*, `LLM`), afin d'alléger la charge de travail demandée aux experts métiers (cf. Section 4.4.3).

4.4.2 Étude des patterns linguistiques pertinents à l'aide de la Maximisation des Traits pour assister la validation d'une base d'apprentissage

Nous venons de conclure que la validation manuelle d'un résultat de `Clustering Interactif` est fonctionnelle, mais qu'elle souffre d'une très mauvaise expérience utilisateur. Afin d'améliorer cet aspect, une première idée consiste à mettre en valeur le vocabulaire caractéristique de chaque *cluster* et d'examiner si les informations recueillies sont utiles à l'appréciation de leur valeur métier.

4.4.2.a Protocole expérimental

Pour résumer le protocole expérimental adapté, une description en pseudo-code est disponible dans l'Algorithme 4.7.

Nous nous appuyons sur le même protocole que l'expérience précédente (cf. Section 4.4.1) : nous utilisons donc comme vérité terrain le jeu de données `Bank Cards (v1.0.0)`, nous réalisons 5 tentatives complètes de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori (voir Section 4.3), et nous demandons toutes les 5 itérations à un expert de qualifier les *clusters* obtenus entre trois catégories (**exploitable**, **partiellement exploitable** et **non exploitable**).

Cependant, avant de demander l'avis de l'expert, nous réalisons une analyse du vocabulaire employé. Pour cela, nous utilisons une sélection des patterns linguistiques pertinents basée sur la maximisation des traits (notée `FMC`, cf. Lamirel et al., 2017) : cette méthode permet de trouver les composantes vectorielles caractéristiques et discriminantes de chaque *cluster* en attribuant un score à chaque couple (`cluster`, `composante`). Dans notre cas, en utilisant une vectorisation basée sur la fréquence du vocabulaire dans un document comme `TF-IDF` (Sparck Jones, 1972), nous pouvons déterminer les mots les plus représentatifs de chaque *cluster*. Ainsi, nous pouvons à la fois décrire chaque groupe de questions par une liste de mots clés caractéristiques, mais aussi surligner ces mots dans les questions à parcourir pour attirer l'attention de l'expert sur ce qui semble être statistiquement discriminant.

Nous adaptons aussi la tâche de l'expert afin qu'il donne son avis sur chaque *cluster* en répondant aux questions suivantes :

Données : jeux de données annotées (vérités terrains)
pour chaque *jeu de données à tester* **faire**
  initialisation (données) : récupérer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
  vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
  clustering initial : regrouper les données par similarité avec `clust.kmeans.cop` ;
  analyse linguistique : caractériser les *clusters* grâce à la `FMC` ;
  évaluation assistée : juger de l'exploitabilité de chaque *cluster* ;
  labellisation assistée : nommer chaque *cluster* exploitable ;
  répéter
    échantillonnage : sélectionner des contraintes avec `samp.closest.diff` ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering : regrouper les données par similarité avec `clust.kmeans.cop` ;
    analyse linguistique : caractériser les *clusters* grâce à la `FMC` ;
    évaluation assistée : juger de l'exploitabilité de chaque *cluster* ;
    labellisation assistée : nommer chaque *cluster* exploitable ;
  jusqu'à *annotation de toutes les contraintes possibles*;
analyse : afficher l'évolution de l'exploitabilité de chaque itération de *clustering* ;
Résultat : discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

ALGORITHME 4.7 – *Description en pseudo-code du protocole expérimental de l'étude des patterns linguistiques pertinents pour vérifier la valeur métier d'une base d'apprentissage.*

— est-ce que la liste des patterns linguistiques caractéristiques du *cluster* est suffisamment complète pour permettre d'identifier une thématique principale **bien définie** ? (*en effet, comment interpréter un cluster sans définition claire ?*)

— est-ce que la liste des patterns linguistiques caractéristiques du *cluster* identifie plusieurs thématiques ou bruits dans le *cluster* ? (*en effet, comment avoir de bonnes performances si la base d'apprentissage n'est pas fiable ?*)

Idées : Nous pourrions aussi analyser l'impact ergonomique qu'apporte la mise en exergue des patterns linguistiques pertinents dans le texte de chaque *cluster*. Une telle étude pourrait ainsi mesurer le gain de temps et de qualité par rapport à une validation manuelle non assistée comme présentée en Section 4.4.1.

Pour information : Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C. L'implémentation de la maximisation des traits, aussi détaillée en Annexe C, est accessible ici dans Schild, 2023.

4.4.2.b Résultats obtenus

Attention : Par manque de moyens, la vérification manuelle des *clusters* n'a pas été réalisée. Nous présentons donc simplement quelques exemples d'analyses linguistiques réalisées grâce à la `FMC`.

Prenons quelques *clusters* et suivons l'évolution de leur analyse linguistique au cours des itérations. Nous nous référons aux tableaux ci-contre pour connaître le top 10 des termes caractéristiques des différents *clusters* et nous réalisons un extrait de questions issues de ces *clusters* avec une mise en évidence des termes caractéristiques dans le texte.

D'abord, prenons l'exemple suivi dans la Table 4.5. Nous suivons ici l'évolution d'un *cluster* bien formé dès l'itération 0. En effet, il est aisé de juger ce *cluster* exploitable et d'en déduire sa thématique : le *cluster* est de taille suffisante (plus de 45 questions), son vocabulaire caractéristique est fourni (36 à 38 patterns) et les patterns mis en avant sont cohérents (*sans contact*, *paiement sans contact*, `activer le`, *nfc*, ...). En parcourant le contenu de ce *cluster*, nous arrivons rapidement à associer sa thématique à la classe `gestion_sans_contact` de la vérité terrain.

Ensuite, prenons l'exemple décrit dans la Table 4.6. À l'itération 0, il est impossible d'exploiter ce *cluster* : il n'y a que deux patterns mis en avant, et les questions semblent toutes traiter de sujets différents. À l'itération 10, le résultat semble un peu plus exploitable. Nous pouvons d'ailleurs déduire deux thématiques principales : la première, mise en avant par des patterns linguistiques de type "*numero*", "*online*", et "*de carte virtuelle*", permet d'imaginer un sujet sur la gestion des numéros de cartes virtuelles ; la seconde, identifiée par les termes "*débloquer*" et "*réactiver*", oriente plutôt vers la création d'un thème pour débloquer une carte

Identification du *cluster*	Analyse linguistique (avec la FMC)	Aperçu du *cluster* (avec emphase)
Tentative : 1 Itération : **0** Cluster : 0 Avis initial : Exploitable	- sans contact - contact - sans - mode - le mode - sur ma carte - le sans contact - le sans - paiement sans contact (Total : 36)	- **activer le** moyen de **paiement nfc sur ma carte** gold - **enlever** le **mode sans contact** de ma carte - gerer le **mode** de **paiement nfc sur ma carte** - je souhaite gerer le **mode** nfc sur mes cartes bancaires - l option **sans contact ne fonctionne pas sur ma carte** - modifier le **mode sans contact** - modifier le **mode** nfc **sur ma carte** de paiement - peut on annuler le paiement **sans contact** - puis je **activer le sans contact** depuis l application (Total : 46)
Tentative : 1 Itération : **15** Cluster : 0 Avis initial : Exploitable	- sans contact - contact - sans - sur ma - mode - le mode - sur ma carte - nfc - le sans contact (Total : 38)	- **activer le** moyen de paiement **nfc sur ma** carte gold - **enlever** le **mode sans contact** de ma carte - gerer le **mode** de paiement **nfc sur ma** carte - je souhaite gerer le **mode nfc** sur mes cartes bancaires - l **option sans contact** ne fonctionne pas **sur ma** carte - modifier le **mode sans contact** - modifier le **mode nfc sur ma carte de paiement** - peut on annuler le paiement **sans contact** - puis je **activer le sans contact** depuis l application (Total : 50)

TABLE 4.5 – *Extrait de l'analyse linguistique de clusters exploitables dès la première itération. ces clusters représentent la thématique* `gestion_sans_contact` *entre l'itération* 0 *(initialisation) et l'itération* 15 *(atteinte de la vérité terrain). La troisième colonne expose un aperçu du contenu des clusters en mettant l'emphase sur les termes caractéristiques identifiés grâce à la* FMC*, et la deuxième colonne représente le top* 10 *de ces termes les plus caractéristiques pour chaque cluster.*

bancaire. Quelques bruits sont présents, mais ce *cluster* à l'itération 10 peut être associé aux classes `gestion_carte_virtuelle` et `deblocage_carte`. Dès l'itération 15, ces thématiques se séparent en deux *clusters* (1 et 4) : nous pouvons les identifier à l'aide de leurs listes de termes bien fournies.

4.4.2.c Discussion

Cette seconde étude avait pour objectif de proposer une assistance à la validation manuelle des *clusters* par un expert métier afin d'en qualifier la pertinence métier. Pour cela, nous avons choisi une analyse linguistique à l'aide de la maximisation de traits pour mettre en avant les mots représentatifs et discriminants de chaque groupe de questions. Nous discutons ici de l'intérêt d'une telle analyse.

Tout d'abord, concernant les *clusters* jugés exploitables, nous constatons que la FMC permet d'identifier aisément la thématique principale. C'est le cas dans les exemples présentés dans les TABLE 4.5 et TABLE 4.6), où les thématiques sont bien représentées par leurs patterns (`gestion_sans_contact` avec "*mode*", "*sans*", "*contact*", "*nfc*"; `gestion_carte_virtuelle` avec "*numero*", "*virtuel*", "*online*"; `deblocage_carte` avec "*debloquer*", "*deverrouiller*", "*carte*"). De plus, la présence des différentes variantes d'un même pattern (*avec ses pluriels, au sein d'un groupe de termes, ...*) permet rapidement d'intégrer le champ lexical présent, aidant ainsi à interpréter le *cluster* comme probablement exploitable.

Les thématiques présentes dans les *clusters* partiellement exploitables sont aussi identifiables

Identification du *cluster*	Analyse linguistique (avec la FMC)	Aperçu du *cluster* (avec emphase)
Tentative : 1 Itération : **0** Cluster : 1 Avis initial : Non exploitable	- carte avalee - nouvelle carte bancaire (Total : 2)	- ai je le droit d avoir un decouvert bancaire - bonjour pouvez vous debloquer ma carte merci - carte bancaire avalee - choisir une **nouvelle carte bancaire** - comment signaler un vol de carte bleue - diminuer le plafond d une carte gold - le rapatriement est il couvert par ma carte bancaire - que faire pour activer une carte bancaire virtuelle - quelle est ma situation financiere (Total : 157)
Tentative : 1 Itération : **10** Cluster : 2 Avis initial : Partiellement exploitable	- un numero - numero - de carte virtuelle - un numero de - numero de carte - numero de - numeros - debloquer ma - débloquer ma carte (Total : 25)	- activer les **achats** avec **un numero virtuel** - comment **debloquer ma** mastercard - comment **reactiver sa** carte - j aimerai **debloquer ma** carte svp - pouvez vous **debloquer ma** carte - obtenir une carte **online** - ou en est ma situation financiere - ou puis je gerer mes **numeros virtuels** - supprimer **une carte virtuelle** (Total : 80)
Tentative : 1 Itération : **15** Cluster : 2 Avis initial : Exploitable	- virtuelle - carte virtuelle - un numero - numero - de carte virtuelle - un numero de - numero de carte - numero de - numeros (Total : 34)	- **activer les achats** avec **un numero virtuel** - comment consulter **ses numeros** de carte **virtuelle** - comment obtenir **un numero** de carte **virtuelle** - comment supprimer **un numero** de carte **online** - **creer** une carte bancaire **virtuelle** - faire un achat avec **un numero** de carte **online** - j aimerai utiliser une carte **virtuelle** - ou peut on gerer **ses numeros virtuels** - supprimer **un numero** de carte **virtuel** (Total : 49)
.....	.....	.....
Tentative : 1 Itération : **15** Cluster : 4 Avis initial : Exploitable	- reactiver - debloquer - debloquer ma - debloquer ma carte - bloquee - reactiver sa - reactiver ma - deverrouiller - reactiver sa carte (Total : 24)	- bonjour pouvez vous **debloquer** ma carte merci - comment **deverrouiller** sa carte - comment reutiliser une carte bancaire **bloquee** - **debloquer** sa carte apres trois **mauvais** codes - j ai besoin de **deverrouiller** ma carte de paiement - j ai retrouve ma carte puis je la **reactiver** - je souhaite **debloquer** ma carte bleue - pouvez vous **debloquer** ma carte - **reactiver** une carte suspendue (Total : 48)

Table 4.6 – *Extrait de l'analyse linguistique de clusters évoluant de non exploitables à exploitables. ces clusters représentent la conception des thématiques* `gestion_carte_virtuelle` *et* `deblocage_carte`*, entre l'itération* 0 *(initialisation) et l'itération* 15 *(atteinte de la vérité terrain). La troisième colonne expose un aperçu du contenu des clusters en mettant l'emphase sur les termes caractéristiques identifiés grâce à la* FMC*, et la deuxième colonne représente le top* 10 *de ces termes les plus caractéristiques pour chaque cluster.*

grâce à la FMC, mais à moindre mesure. En effet, comme plusieurs thématiques se mélangent, l'ensemble des termes caractéristiques est aussi plus hétérogène, des variantes ne sont pas identifiées, et des patterns dénués de sens peuvent être mis en avant. Par exemple, dans le *cluster* 2 de la tentative 1 à l'itération 0, deux classes sont présentes avec leurs termes caractéristiques

(`consultation_solde` avec "*solde*" et "*compte*" ; `gestion_carte_virtuelle` avec "*carte virtuelle*" et "*numero*"), mais certains termes quelconques sont pourtant présentés comme représentatifs ("*de mes*", "*avec un*") et d'autres termes intéressants ne sont pas identifiés ("*compte*" et "*numeros*" ne sont présents qu'au singulier). Ces petits indices peuvent donc aiguiller l'expert pour identifier des ajustements nécessaires ou des *clusters* réalisés sur des bases fragiles.

Enfin, en ce qui concerne les *clusters* jugés non exploitables, la `FMC` permet de confirmer leur manque de cohérence (pour le *cluster* 1 de la tentative 2 à l'itération 0 : "*carte gold*", "*numeros virtuels*", "*découvert bancaire*", "*carte paiement differe*", ...) ou leur absence de valeur métier (seulement 2 termes caractéristiques pour le *cluster* 1 de la tentative 1 à l'itération 0). Nous pouvons aussi constater que les termes estimés comme caractéristiques pour ces *clusters* non exploitables sont souvent des groupes de mots trop spécifiques (dans notre dernier exemple : "*numeros virtuels*" uniquement au pluriel), ce qui est plutôt caractéristique de termes qui distinguent le *cluster* des autres mais qui ne l'identifient pas (voir Lamirel et al., 2017 pour comprendre la balance entre *Features Recall* (identification) et *Features Predominance* (discrimination)).

Cependant, une telle approche pour qualifier les *clusters* risque de ne pas répondre à nos attentes en l'état, car certains problèmes identifiés dans la section précédente subsistent (4.4.1). Entre autres, malgré une abstraction des *clusters* par leurs termes les plus caractéristiques (cf. deuxième colonne dans les Table 4.5 et Table 4.6) et une mise en exergue dans le texte (cf. troisième colonne de ces tableaux), il faut toujours parcourir un grand nombre de données pour juger de la pertinence métier, ce qui reste une tâche chronophage s'il faut la réaliser à chaque itération. Ainsi, même si le travail est simplifié par l'identification rapide des termes importants du corpus, la charge de travail reste élevée.

D'autre part, les résultats remontés par l'analyse linguistique sont à affiner davantage pour faciliter leur interprétation. Par exemple, les mots pourraient être réduits à leur forme racine (lemmatisation) afin d'éviter de considérer les nombreuses variations possibles (formes conjuguées, pluriels, ...) et les mots vides (*stopwords*) pourraient être supprimés pour accentuer l'absence de termes caractéristiques à valeur métier dans les *clusters* inexploitables. En ce qui concerne l'affichage des patterns identifiés dans les textes, un code couleur pourrait être introduit pour faciliter la lecture, avec l'usage de nuances de couleurs pour marquer la prépondérance d'un terme en fonction du *cluster*. L'exemple ci-dessous (issu du *cluster* 0 de la tentative 1 à l'itération 0) illustre l'intérêt de cette coloration : nous y distinguons facilement que le *cluster* traite des paiements sans contact (`NFC` en vert) mais que les cartes *Visa* (en rouge) ont dû être regroupées au sein d'un autre *cluster*.

> **Exemples :** Possibilité d'affichage en couleur d'un texte et de son analyse linguistique : les patterns caractéristiques du *cluster* auquel appartient le texte sont en vert, les patterns caractéristiques des autres *clusters* en rouge, les patterns non caractéristiques en noir.
>
> « *est ce que le mode de paiement nfc est disponible sur ma carte visa* »

Mais malgré ces améliorations de l'approche linguistique, il est probable qu'elle soit trop éloignée des compétences réelles des experts : en effet, ces derniers disposent de connaissances sur leur domaine métier (dans notre cas, des sujets concernant la banque, l'assurance et la finance), mais ils ne disposent pas forcément d'aptitudes permettant d'examiner les nuances linguistiques présentes dans un *cluster*, ni l'impact de la prépondérance des pluriels, des bigrams, des mots vides de sens, ...

Notes de l'auteur : Nous utilisons notre expérience personnelle pour avancer cet avis. En effet, lors de précédents travaux industriels (non publiés), nous avons réalisé une modélisation thématique (*topic modeling*) sur des corps de mails en utilisant la `LDA` (Blei et al., 2003). Comme de coutume, nous avons réalisé une abstraction en énumérant le vocabulaire employé par chaque *topic*. Puis, nous avons fait intervenir des experts métiers afin d'estimer la valeur de chaque *topic*, de les raffiner, et de les nommer grâce aux abstractions réalisées. Toutefois, nous avons constaté que les experts intervenant dans ce projet avaient du mal à manipuler de telles abstractions, notamment car ils n'arrivaient pas à se projeter sur les *topic* peu ou pas exploitables. Au final, les experts ont préféré modéliser manuellement le corpus de mails, sans utiliser les résultats de la `LDA`.

Une partie de cet échec est probablement lié à l'utilisation en tant que telle de la `LDA` (peu d'interactivité entre l'expert et l'algorithme de modélisation), mais nous avons aussi eu des retours directs des experts sur leur difficulté à utiliser des champs lexicaux pour juger de la qualité et de la cohérence d'une modélisation. Ainsi, même si une abstraction linguistique semble sémantiquement pertinente, il se peut que les intervenants du projet ne soient pas à l'aise avec son utilisation. Une étude spécifique à ce sujet pourrait vérifier ce ressenti.

Pour conclure, nous retenons que l'analyse linguistique est pleine de potentiel et qu'elle permet de mettre en exergue des mots importants lors de l'affichage du contenu des *clusters*. Toutefois, nous conservons quelques doutes concernant l'expérience utilisateur d'une telle approche pour des experts métiers qui, n'étant pas des experts en linguistique, pourraient se perdre dans des considérations trop techniques durant leur analyse. De fait, si une approche linguistique est trop abstraite pour qualifier un *cluster*, nous nous intéressons pour la suite à une approche plus pragmatique pour identifier sa thématique principale (cf. Section 4.4.3).

4.4.3 Étude d'un résumé automatique des *clusters* à l'aide d'un large modèle de langage

Comme nous l'avons vu dans la section précédente, une analyse linguistique peut paraître trop abstraite pour un expert métier. Nous nous intéressons donc à un moyen de simplifier l'identification de thématiques dans un *cluster*, et envisageons l'automatisation de cette tâche en utilisant les capacités d'un large modèle de langage (`LLM`). En effet, plusieurs de ces modèles ont montré leur efficacité sur les tâches de résumés de documents (J. Zhang et al., 2019, Lewis et al., 2019, Radford et al., 2019, Brown et al., 2020), une fonctionnalité que nous allons adapter [38] et étudier ici.

4.4.3.a Protocole expérimental

Pour résumer le protocole expérimental adapté, une description en pseudo-code est disponible dans l'Algorithme 4.8.

Nous nous appuyons sur le même protocole que l'expérience précédente (cf. Section 4.4.1) : nous utilisons donc comme vérité terrain le jeu de données `Bank Cards (v1.0.0)`, nous réalisons 5 tentatives complètes de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori (voir Section 4.3), et nous demandons toutes les 5 itérations à un expert de qualifier

38. Nous nous inspirons notamment de Alammar et Grefenstette, 2022 qui proposent une boîte à outils en `Python` permettant de nommer des *topics* en utilisant `BERT`.

Données : jeux de données annotées (vérités terrains)
pour chaque *jeu de données à tester* **faire**
  initialisation (données) : récupérer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
  vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
  clustering initial : regrouper les données par similarité avec `clust.kmeans.cop` ;
  synthèse automatique : résumer les thématiques des *clusters* par un `LLM` ;
  évaluation assistée : juger de l'exploitabilité de chaque *cluster* ;
  labellisation assistée : nommer chaque *cluster* exploitable ;
  répéter
    échantillonnage : sélectionner des contraintes avec `samp.closest.diff` ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
    clustering : regrouper les données par similarité avec `clust.kmeans.cop` ;
    synthèse automatique : résumer les thématiques des *clusters* par un `LLM` ;
    évaluation assistée : juger de l'exploitabilité de chaque *cluster* ;
    labellisation assistée : nommer chaque *cluster* exploitable ;
  jusqu'à *annotation de toutes les contraintes possibles*;
analyse : afficher l'évolution de l'exploitabilité de chaque itération de *clustering* ;
Résultat : discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

ALGORITHME 4.8 – *Description en pseudo-code du protocole expérimental de l'étude d'un résumé automatique des clusters à l'aide d'un large modèle de langage pour vérifier la valeur métier d'une base d'apprentissage.*

les *clusters* obtenus entre trois catégories (**exploitable**, **partiellement exploitable** et **non exploitable**).

Cependant, avant de demander l'avis de l'expert, nous utilisons un large modèle de langage pour résumer automatiquement le contenu du *cluster* à analyser. Pour cela, nous utilisons le modèle `gpt-3.5-turbo` mis à disposition en appel `API` via la librairie `openai` [39] de l'entreprise `OpenAI` [40]. Le *prompt* du modèle, adapté à l'usage de notre jeu de données, est composé de trois parties :

— un **contexte d'utilisation**, destiné à centrer les réponses sur le domaine général traité : « *Tu es un expert des secteurs banque, assurance et finance.* » ;

— une **description de la tâche** avec les consignes de restitution : « *Résume-moi en une phrase la thématique traitée dans les textes suivants :* » ;

— les **textes** du *cluster*, énumérés sous la forme d'une liste à puces. Si la taille maximale du *prompt* ne peut pas prendre en compte l'ensemble des données du *cluster*, il est possible de ne prendre qu'un échantillon.

Exemples : Exemple de *prompt* pour le *cluster* 0 de la tentative 1 à l'itération 0.

Tu es un expert des secteurs banque, assurance et finance.
Résume-moi en une phrase la thématique traitée dans les textes suivants :

— *activer le moyen de paiement nfc sur ma carte gold*

— *enlever le mode sans contact de ma carte*

— *gerer le mode de paiement nfc sur ma carte*

— *... (43 autres) ...*

À l'aide de ces résumés, nous pouvons ainsi préparer le travail de l'expert en mettant en avant une synthèse des informations contenues dans chaque *cluster*.

Nous adaptons aussi la tâche de l'expert afin qu'il donne son avis sur chaque *cluster* en répondant aux questions suivantes :

— est-ce que le résumé est suffisamment précis pour identifier une thématique principale bien définie dans le *cluster* ? (*en effet, comment interpréter un cluster sans définition claire ?*)

— est-ce que le résumé identifie plusieurs thématiques ou bruits dans le *cluster* ? (*en effet, comment avoir de bonnes performances si la base d'apprentissage n'est pas fiable ?*)

Idées : Nous pourrions aussi analyser l'impact ergonomique qu'apporte l'automatisation de la synthèse thématique de chaque *cluster*. Une telle étude pourrait ainsi mesurer le gain de temps et de qualité par rapport à une validation manuelle non assistée, comme présentée en Section 4.4.1. Ceci pourrait faire l'objet d'études ultérieures.

39. https://pypi.org/project/openai/
40. `OpenAI` est une entreprise fournissant des services d'intelligence artificielle sur le Cloud. Elle est entre autres connue pour les modèles d'IA `DALL-E` (Ramesh et al., 2021) et `GPT` (Brown et al., 2020, OpenAI, 2023).

Attention : Par manque de personnes aptes à qualifier le jeu de données utilisé, les annotations de cette étude ont été réalisées par un seul opérateur (*moi-même, ayant participé à la création de ce jeu de données*). Nous supposons que cette contrainte n'est pas pénalisante pour l'analyse : en effet, dans une situation réelle, cet opérateur serait responsable des choix à entreprendre pour concevoir le jeu de données, il est donc le mieux placé pour juger de la pertinence d'un *clustering* par rapport à sa propre modélisation du problème. Les problèmes d'accords inter-annotateurs seront plutôt discutés en Section 4.6. Nous réalisons toutefois 5 tentatives différentes de la méthode pour limiter les biais intra-individuels.

Pour information : Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.4.3.b Résultats obtenus

La Figure 4.25 met en avant l'évolution de le pertinence moyenne estimée par l'opérateur sur la base des résumés automatiques des *clusters*. Comme lors de l'analyse manuelle (cf. Section 4.4.1), il est normal de retrouver une tendance générale à la diminution du nombre de *clusters* inexploitables au profit de *clusters* exploitables. La différence principale réside dans l'absence du pic de croissance du nombre de *clusters* partiellement exploitables, dont l'apogée était précédemment situé à l'itération 10.

Pour aller plus loin, reprenons les *clusters* que nous avons utilisés comme cas d'étude lors de l'analyse linguistique à l'aide de la maximisation des traits (cf. Section 4.4.2).

D'abord, reprenons l'exemple de l'évolution d'un *cluster* bien formé dès l'itération 0, précédemment détaillé dans la Table 4.5 et dont les résumés automatiques sont présentés dans la Table 4.7. Nous constatons effectivement que les synthèses générées par le modèle identifie sans ambiguïté le thème du paiement sans contact, même si le résumé de l'itération 0 énumère longuement les actions réalisables sur ce sujet.

Ensuite, intéressons-nous à l'exemple de l'évolution des *clusters* en cours de formation détaillé dans la Table 4.5 et dont les résumés automatiques sont présentés dans la Table 4.7. À l'itération 0, le résumé proposé est une longue énumération de 9 thématiques différentes sur la gestion de carte bancaires : puisque le jeu de données entier traite des cartes bancaires, nous identifions clairement ce *cluster* comme non exploitable. À l'itération 10, nous ne distinguons plus que deux sujets principaux : le déblocage de carte, et l'utilisation de numéros de cartes virtuelles, ce qui est en accord avec notre précédente analyse. À partir de l'itération 15, ces deux thématiques se retrouvent bien séparées dans deux *clusters* différents, et chacune est identifiable via le résumé proposé : nous notons toutefois que si la thématique principale est identifiée, alors l'énumération de détails est plutôt portée sur les actions réalisables avec cette thématique ("*création*", "*activation*", "*suppression*" pour la classe `gestion_carte_virtuelle`).

D'une manière générales, la majorité des résumés automatiques permettent d'identifier sans ambiguïté les mêmes thématiques qu'une validation manuelle.

— En ce qui concerne les itérations où le *clustering* atteint la vérité terrain, tous les résumés permettent d'identifier les thématiques présentes (190 *clusters* concernés, toutes itérations

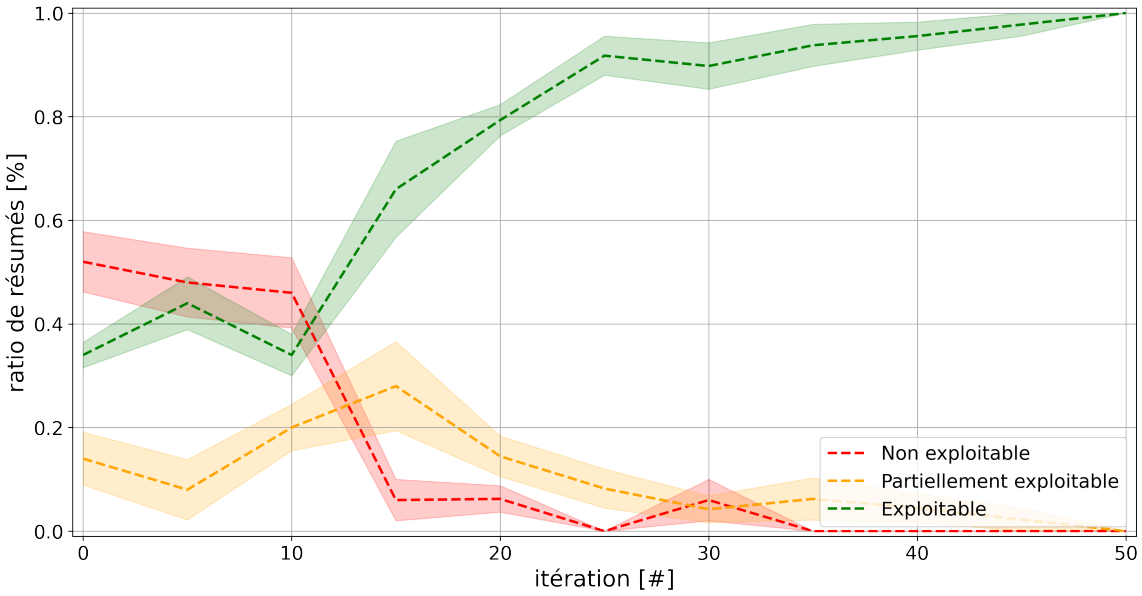

Figure 4.25 – *Évolution de la pertinence métier moyenne en fonction du nombre d'itérations de la méthode. Cette pertinence, exprimée en proportion du nombre de clusters, est estimée sur la base du résumé automatique des clusters par un large modèle de language et est retranscrite en trois niveaux :* `exploitable` *en vert,* `partiellement exploitable` *en orange, et* `non exploitable` *en rouge.*

Identification du *cluster*	Résumé automatique du *cluster* (LLM)
Tentative : 1 Itération : **0** Cluster : 0 Avis initial : `Exploitable`	La thématique traitée dans ces textes est la gestion de l'activation, la désactivation, la modification ou la nécessité d'utiliser le paiement sans contact ou le NFC (Near Field Communication) sur les cartes de paiement bancaires.
Tentative : 1 Itération : **15** Cluster : 0 Avis initial : `Exploitable`	Les textes traitent de la gestion et de l'utilisation du paiement sans contact (NFC) sur les cartes bancaires.

Table 4.7 – *Extrait de de l'analyse de résumés automatiques de clusters exploitables dès la première itération. ces clusters représentent la thématique* `gestion_sans_contact` *entre l'itération 0 (initialisation) et l'itération 15 (atteinte de la vérité terrain). La seconde colonne expose le résumé obtenu en appelant un large modèle de langage (*`gpt-3.5-turbo`*) sur une tâche de résumé.*

confondues) ;

— En ce qui concerne les autres itérations (dont le *clustering* ne correspond pas encore à la vérité terrain), les résumés automatiques permettent d'identifier les mêmes thématiques qu'une validation manuelle dans 86% des cas (331 cas sur 386 *clusters*, toutes itérations confondues).

Identification du *cluster*	Résumé automatique du *cluster* (avec un `LLM`)
Tentative : 1 Itération : **0** Cluster : 1 Avis initial : `Non exploitable`	La thématique des textes est la gestion des cartes bancaires, incluant la sécurité, les pertes, les vols, les blocages, les récupérations, le changement ou la commande de cartes, les garanties et assurances, les découverts, les paiements virtuels, les numéros virtuels et les plafonds de paiement autorisés.
Tentative : 1 Itération : **10** Cluster : 2 Avis initial : `Partiellement exploitable`	Les textes traitent principalement de la gestion des cartes de paiement, de leur déblocage ou leur réactivation, et de l'utilisation de numéros de carte virtuelle pour les achats en ligne.
Tentative : 1 Itération : **15** Cluster : 2 Avis initial : `Exploitable` ..... Tentative : 1 Itération : **15** Cluster : 4 Avis initial : `Exploitable`	Les textes concernent la gestion et l'utilisation des numéros de carte virtuelle pour les achats en ligne, notamment la création, l'activation, la suppression et la gestion de ces numéros virtuels. ..... La thématique traitée dans ces textes est le déblocage, le déverrouillage ou la réactivation de cartes bancaires bloquées.

TABLE 4.8 – *Extrait de de l'analyse de résumés automatiques de clusters évoluant de non exploitables à exploitables. ces clusters représentent la conception des thématiques* `gestion_carte_virtuelle` *et* `deblocage_carte`*, entre l'itération* 0 *(initialisation) et l'itération* 15 *(atteinte de la vérité terrain). La seconde colonne expose le résumé obtenu en appelant un large modèle de langage (*`gpt-3.5-turbo`*) sur une tâche de résumé.*

4.4.3.c Discussion

Cette troisième et dernière étude sur l'estimation de la valeur métier d'un résultat de `Clustering Interactif` avait pour but de simplifier la tâche d'analyse de l'expert en évaluant les capacités d'un large modèle de langage à synthétiser les thématiques présentes. Nous comparons les résultats obtenus avec ceux obtenus dans les deux sections précédentes.

En premier lieu, nous constatons que l'approche est efficace, car elle permet à l'expert, dans près de 86% des cas, de parvenir aux mêmes conclusions sur l'analyse de la valeur métier d'un *cluster*. Concernant les *clusters* exploitables, la synthèse générée est généralement très explicite, à l'image des résultats présentés dans la TABLE 4.7, ce qui rend la tâche de labellisation des *clusters* presque triviale. Nous retrouvons notamment ces résultats lorsque le *clustering* atteint la vérité terrain : toutes les résumés permettent de décrire sans ambiguïté les thématiques traitées, confirmant à la fois que le jeu de données est bien annoté et que le *clustering* est exploitable. De manière similaire, les *clusters* non exploitables peuvent aussi être rapidement identifiés, notamment par la présence de longues énumérations incohérentes et des tournures de phrases ambiguës, à l'image du premier exemple de la TABLE 4.8. Les résultats obtenus ci-dessus permettent donc de conclure sans hésitation que l'approche est adéquate pour assister un expert dans sa tâche d'évaluation.

En plus de la justesse des résumés obtenus, il y a aussi un gain réel de confort pour l'expert

métier. En effet, la tâche de celui-ci consiste désormais simplement à lire une description textuelle et à confirmer si elle correspond à un cas d'usage métier. Ainsi, plus besoin de parcourir de grands ensembles de données ou de réaliser des analyses linguistiques complexes : l'exercice est simple, peu chronophage, et est centré sur les compétences réelles de l'expert. Nous pourrions aller plus loin en déclinant cette approche sur d'autres analyses, par exemple en réduisant cette synthèse à quelques mots pour nommer le *cluster*, ou en demandant d'identifier les potentielles données aberrantes à supprimer pour augmenter la cohérence du regroupement de questions.

Bien entendu, l'automatisation de cette tâche peut aussi orienter l'expert vers d'autres conclusions, voire le mener vers une fausse piste. Nous estimons à 14% le taux de différences d'identification de thématiques avec une approche purement manuelle, et nous constatons que la majorité des écarts entre ces approches réside dans les cas de figure suivants :

— le modèle peut générer des hallucinations n'ayant rien à voir avec les données en entrée : c'est le cas avec le *cluster* 9 de la tentative 2 à l'itération 10, où le *cluster* est composé de 2 questions (« *Comment obtenir une Mastercard ?* » et « *Désactiver les numéros virtuels.* ») et où le résumé parle de "*sécurisation des transactions bancaires*" ;

— le résumé peut être trop concis et certaines thématiques peuvent être ignorées dans la synthèse : l'expert sera tenté de conclure à un *cluster* exploitable alors qu'il est potentiellement bruité ;

— à l'inverse, les données aberrantes ou isolées d'un *cluster* peuvent influencer le résumé : cela peut donner l'illusion que plusieurs thématiques sont présentes alors qu'une seule l'est réellement (voir aussi Falke et al., 2019 qui met en avant l'importance de l'ordre des informations transmises au modèle) ;

— le résumé peut aussi mettre en avant des thématiques auxquelles l'expert n'aurait pas pensé : c'est le cas dans les *clusters* 7 et 8 de la tentative 1 à l'itération 0, où les thématiques `gestion_carte_mastercard` et `gestion_carte_visa` sont proposées dans la synthèse, mais auraient probablement été considérées par un expert (*il n'y a pas de différences vraiment significatives entre les deux réseaux de cartes bancaires pour justifier une gestion séparée, même si les clusters sont bien formés*).

Ces différents exemples confirment qu'une étape de validation reste nécessaire. Celle-ci n'a pas besoin d'être systématique, elle peut être réalisée pour confirmer la fin des itérations de `Clustering Interactif`, limitant ainsi la charge de travail de l'expert.

Toutefois, le principal obstacle à l'utilisation des larges modèles de langage concerne des problématiques techniques pour disposer, installer et utiliser ces modèles. En effet, à l'heure de rédaction de ce manuscrit (juillet 2023), l'entraînement requiert des serveurs aux configurations pharaoniques, composés de plusieurs milliers de `GPU`, du stockage et de la bande passante pour manipuler des téraoctets de données, et de toute l'infrastructure auxiliaire nécessaire pour contrôler et refroidir les machines, soit un investissement de plusieurs millions voire milliards de dollars. L'inférence n'est pas simple non plus, car les machines doivent entre autres disposer de suffisamment de `RAM` pour charger les modèles et de `GPU` pour les exécuter. Ainsi, seules quelques entreprises peuvent investir et mettre à disposition ce genre d'infrastructures, souvent sous forme de services hébergés sur le *Cloud*, comme la plateforme *Meta Reasearch Super Cluster* (Lee et Sengupta, 2022) ou le service *Microsoft Azure AI* (Roach, 2023)) [41].

41. À titre d'exemple, prenons `Llama 2` de *Meta* (Touvron et al., 2023) : pour l'entraînement, il a fallu 2 000 `GPU` sur près de 3 millions d'heures (temps `GPU` cumulé) pour un modèle de 70 millions de paramètres.

Une autre limite en découlant concerne la confidentialité des données. En effet, comme l'utilisation des `LLM` se fait via des services externes, il est possible que certaines plateformes améliorent leurs modèles à l'aide des données communiquées. Ainsi, Huang et al., 2022 et O'Neill et Connor, 2023 mettent en avant les risques qu'un modèle soit capable d'intégrer puis de restituer des données privées ou confidentielles. Dans cette expérience, nous n'avions pas de craintes car les données sont publiques, mais cela peut pénaliser un projet manipulant des données plus sensibles.

En conclusion, l'utilisation d'un `LLM` semble très prometteuse pour aider un expert métier à estimer la valeur métier d'un partitionnement de données. Nous constatons notamment un réel impact sur l'expérience utilisateur car la synthèse automatique d'un *cluster* réduit drastiquement la charge et la complexité de travail de l'expert. Néanmoins, nous notons plusieurs zones d'ombre quant à la confiance que nous pouvons apporter à l'automatisation de cette tâche, tant sur l'utilisation des modèles (infrastructure technique lourde, confidentialité des données, ...) que sur l'exploitation de leurs résultats (hallucinations, ambiguïtés, ...).

4.4.4 Mise en commun des stratégies d'évaluation de la pertinence métier d'un résultat de *Clustering Interactif*

Points à retenir : Au cours de cette étude de pertinence, nous avons pu voir que :

- ☑ La tâche de validation et de labellisation de résultats de *clustering* est plutôt complexe et fastidieuse si elle n'est pas assistée : il est donc conseillé de faire intervenir plusieurs experts afin de consolider leurs analyses et de confronter les points de vue ;
- ☑ Nous pouvons utiliser des larges modèles de langage (`LLM`) pour réaliser une synthèse automatique d'un *cluster* : cela permet d'estimer rapidement la pertinence d'un regroupement de questions et d'identifier les thématiques qui y sont présentes, offrant ainsi un gain d'efficacité et de confort aux experts métiers (cf. Section 4.4.3) ;
- ☑ Cette approche automatisée n'étant pas infaillible, il est recommandé de croiser différentes approches d'analyses, et de toujours vérifier manuellement le contenu des *clusters* avant d'arrêter le projet d'annotation : pour faciliter cette revue, il est possible de réaliser une analyse linguistique grâce à la `FMC` pour identifier les termes caractéristiques de chaque *cluster* et les mettre en avant dans le texte (cf. Section 4.4.2).

Pour compléter l'étude que nous venons de réaliser, il peut être intéressant de définir un cas d'arrêt des itérations du `Clustering Interactif`, afin de pouvoir prédire quand stopper l'annotation et quand demander aux experts d'évaluer la pertinence de la base d'apprentissage obtenue. Nous étudions cet aspect dans la prochaine section (cf. hypothèse de rentabilité en Section 4.5).

4.5 Évaluation de l'hypothèse de rentabilité

Dans les études précédentes, le cas d'arrêt de notre méthodologie d'annotation basée sur le `Clustering Interactif` était conditionné à la vérité terrain. En effet, nous utilisions un seuil de 90% de `v-measure`, caractérisant une annotation dite "partielle" de la base d'apprentissage. Cependant, une telle référence n'est pas accessible en situation réelle car l'objectif de notre méthode est précisément de construire cette vérité terrain. Nous devons donc nous intéresser à d'autres moyens pour estimer la rentabilité d'une itération supplémentaire, et pouvoir ainsi définir de nouveaux cas d'arrêt pour le `Clustering Interactif`. Pour cela, nous aimerions vérifier l'hypothèse suivante :

Hypothèse de rentabilité

« **Au cours d'une méthodologie d'annotation basée sur le `Clustering Interactif`, il est possible d'estimer la rentabilité d'une itération supplémentaire de la méthode, et ainsi d'établir des cas d'arrêt indépendants d'une vérité terrain pour obtenir une base d'apprentissage satisfaisante.** »

La Figure 4.26 illustre cette hypothèse et la perspective de pouvoir estimer le rapport entre le gain de pertinence obtenu et le coût nécessaire pour l'obtenir.

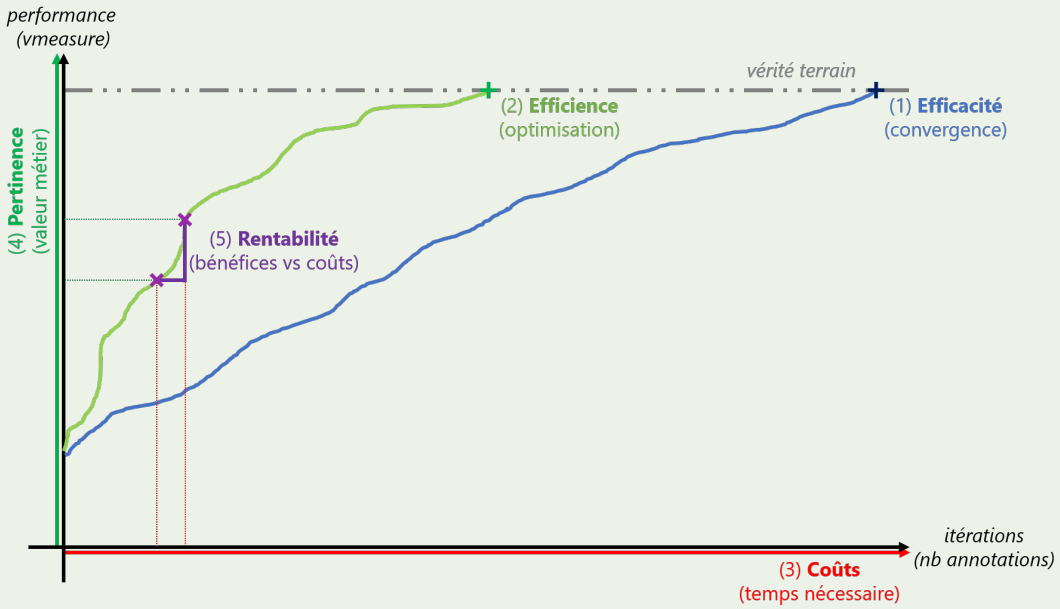

Figure 4.26 – *Illustration des études réalisées sur le `Clustering Interactif` (étape 5/6) en schématisant l'évolution de la **pertinence** (valeur métier évaluée par l'expert, exprimée en nombre de clusters) d'une base d'apprentissage en cours de construction, en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine), ainsi que la **rentabilité** de chaque itération de la méthode (rapport entre le gain potentiel de pertinence et le coût à investir).*

Afin de vérifier cette hypothèse, nous explorons deux approches :

— l'évolution de l'**accord entre l'annotation de l'expert et le *clustering*** sur lequel est

basé l'échantillon d'annotation, permettant d'estimer si la machine doit encore être corrigée par l'annotateur (cf. Section 4.5.1) ; et

— l'évolution de la **différence entre deux *clustering* successifs**, permettant de mesurer s'il y a eu des changements visibles dans le partitionnement des données après l'ajout des dernières contraintes (cf. Section 4.5.2).

4.5.1 Étude de l'évolution d'accord entre l'annotation et le *clustering*

Nous cherchons à trouver un cas d'arrêt du `Clustering Interactif` ne nécessitant pas de comparaison avec une vérité terrain, et notre première intuition concerne l'étude des annotations réalisées. En effet, à chaque itération, l'expert annote un échantillon de contraintes dans le but de confirmer ou de corriger le *clustering* de l'itération précédente. Or, après un nombre suffisant d'itérations, le *clustering* commence à se stabiliser : il devrait donc y avoir davantage d'annotations qui confirment le *clustering* que d'annotations qui le corrigent, puis n'avoir que des accords entre les annotations et le *clustering*. Ainsi, nous allons étudier l'évolution du nombre de contraintes annotées qui approuvent le partitionnement des données obtenu, et essayer d'adapter cette analyse en cas d'arrêt pour notre méthode d'annotation.

4.5.1.a Protocole expérimental

> ⚠ **Attention :** Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble.

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.9.

Nous utilisons comme vérité terrain le jeu de données `Bank Cards (v1.0.0)` : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'Annexe A.1.

Sur ce jeu de données, nous exécutons une tentative complète [42] de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori [43] (voir Section 4.3), et cette tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions. À chaque itération, un lot de 50 contraintes est sélectionné puis annotés en simulant l'action d'un expert métier, et nous évaluons l'accord entre ces nouvelles annotations et la proposition de partitionnement des données, partitionnement réalisé par le *clustering* à l'itération précédente :

— il y a **accord** lorsqu'une contrainte de deux données issues d'un même *cluster* est annotée `MUST-LINK`, ou lorsqu'une contrainte de deux données issues de deux *clusters* différents est annotée `CANNOT-LINK` (cf. Figure 4.27 (1)) ;

42. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

43. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitements simples (`prep.simple`), vectorisation `TF-IDF` (`vect.tfidf`), *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

```
   Données : jeux de données annotées (vérités terrains)
1  pour chaque jeu de données à tester faire
2      initialisation (données) : récupérer les données et la vérité terrain ;
3      initialisation (contraintes) : créer une liste vide de contraintes ;
4      prétraitements : supprimer le bruit dans les données avec prep.simple ;
5      vectorisation : transformer les données en vecteurs avec vect.tfidf ;
6      clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
7      répéter
8          échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
9          simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
10         intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
11         rentabilité : calculer l'accord entre l'annotation et le clustering précédent ;
12         clustering : regrouper les données par similarité avec clust.kmeans.cop ;
13     jusqu'à annotation de toutes les contraintes possibles;
14 analyse 1 : afficher l'évolution de l'accord entre annotation et clustering ;
15 analyse 2 : calculer la corrélation entre le score d'accord et le score de performance ;
   Résultat : discussion sur la rentabilité d'après l'accord entre annotation et clustering
```

ALGORITHME 4.9 – *Description en pseudo-code du protocole expérimental de l'étude de l'évolution d'accord entre l'annotation et le clustering.*

— il y a **désaccord** lorsqu'une contrainte de deux données issues d'un même *cluster* est annotée `CANNOT-LINK`, ou lorsqu'une contrainte de deux données issues de deux *clusters* différents est annotée `MUST-LINK` (cf. FIGURE 4.27 (2)).

Nous pouvons ainsi calculer un score d'accord défini par le ratio entre le nombre d'accords et le nombre de contraintes annotées. Pour nous permettre de discuter de l'utilité de ce score pour prédire la stabilisation du *clustering* et ainsi définir un cas d'arrêt de notre méthodologie d'annotation, nous calculons aussi le score de corrélation entre cet accord et la performance obtenue à l'aide d'une vérité terrain (la corrélation `r` de *Pearson* (KIRCH, 2008) est utilisée).

> **Idées :** Nous concentrons l'étude sur notre paramétrage favori (voir SECTION 4.4.3). Cependant, afin de compléter notre discussion avec d'autres points de comparaison, nous analysons aussi les autres paramétrages implémentés, notamment les meilleurs paramétrages moyens identifiés lors de l'hypothèse d'efficience (voir SECTION 4.2).

> **Pour information :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en ANNEXE A et en ANNEXE C.

4.5.1.b Résultats obtenus

La FIGURE 4.28 représente l'évolution moyenne du score d'accord entre annotation et *clustering* pour les quatre paramétrages mis en avant lors de nos études. Nous pouvons constater une tendance générale à la croissance de ce score d'accord : pour le paramétrage favori **(4)**, l'accord

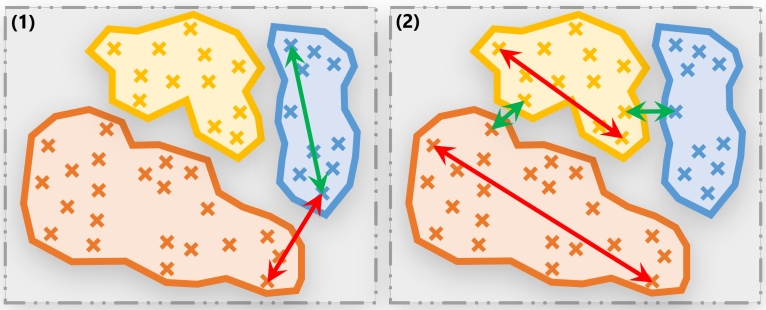

FIGURE 4.27 – *Exemples d'accords et de désaccords entre les annotations d'une itération et le résultat du clustering de l'itération précédente. Des contraintes* `MUST-LINK` *(flèches vertes) et* `CANNOT-LINK` *(flèches rouges) sont représentées dans deux situations : (1) montre des cas d'accords (*`MUST-LINK` *dans un même cluster,* `CANNOT-LINK` *entre deux clusters différents), et (2) montre des cas de désaccords (*`MUST-LINK` *entre deux clusters différents,* `CANNOT-LINK` *dans un même cluster).*

est plutôt faible au début de la méthode (inférieur à 45% avant l'itération 15), puis devient de plus en plus fort (dépassant les 60%) pour finalement atteindre les 100% vers l'itération 45.

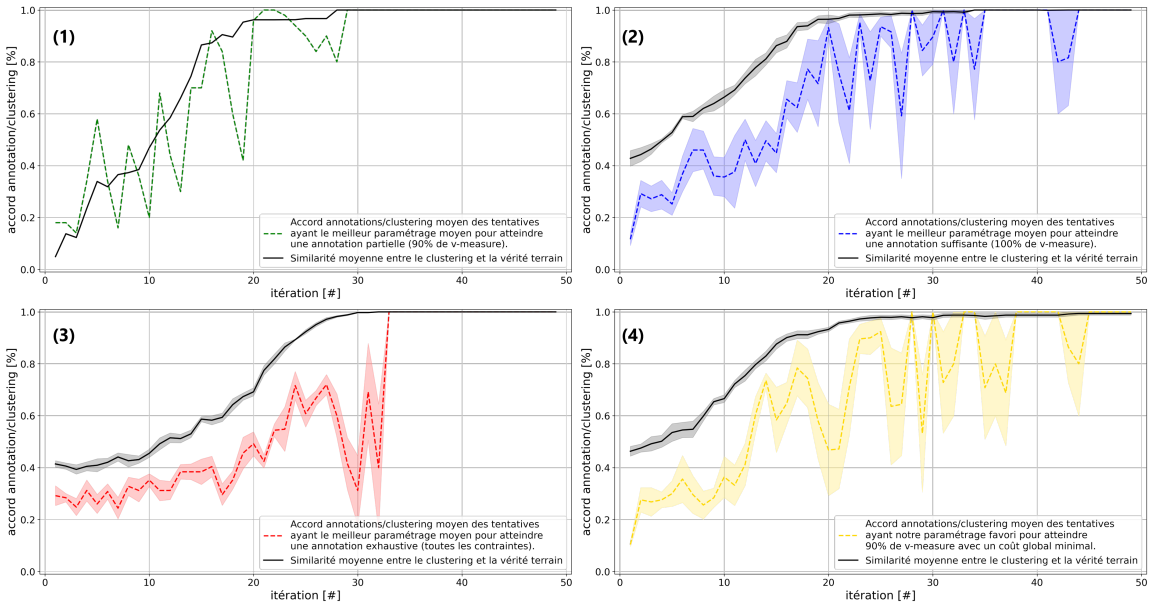

FIGURE 4.28 – *Évolution au cours des itérations de l'accord entre l'annotation de contraintes d'un expert et le résultat de clustering sur lequel est basé l'échantillonnage de contraintes. Ces accords sont exprimés grâce à des lots de 50 contraintes annotées. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori. À titre d'information, les courbes en noir représentent l'évolution de la* `v-measure` *entre le clustering et la vérité terrain.*

La Table 4.9 contient le score de corrélation entre cet accord et la performance théoriques obtenue grâce à la vérité terrain. Cette corrélation est modérée : 0.49 sur l'ensemble des tentatives, 0.69 sur les tentatives utilisant notre paramétrage favori.

Paramétrage	Corrélation `r`
Meilleur paramétrage moyen pour une annotation partielle **(1)**	0.92
Meilleur paramétrage moyen pour une annotation suffisante **(2)**	0.74
Meilleur paramétrage moyen pour une annotation exhaustive **(3)**	0.57
Paramétrage favori **(4)**	0.69
Moyenne des 960 tentatives	0.49

Table 4.9 – *Score de corrélation `r` de Pearson entre la performance du clustering obtenu à l'aide d'une vérité terrain (`v-measure`) et le score d'accord entre annotation et clustering.*

Cependant, la tendance constatée est aussi saccadée par de nombreux pics pouvant faire perdre ou gagner jusqu'à 40% d'accord entre deux itérations. Des chutes d'accord peuvent intervenir au niveau des itérations où la similarité du *clustering* avec la vérité terrain est pourtant forte, comme c'est le cas autour des itérations 29 et 36 où l'accord chute de plus de 25% alors que la `v-measure` avec la vérité terrain est constamment au dessus de de 95%.

Les autres paramétrages représentés dans **(1)**, **(2)** et **(3)** comportent des tendances similaires (corrélation forte mais variations soudaines d'accord, chute d'accords malgré des *clustering* aux performances élevées, ...).

4.5.1.c Discussion

Dans cette étude, nous avons analysé l'évolution de l'accord entre les annotations et le partitionnement de données proposé par un *clustering* dans l'espoir de définir un cas d'arrêt de notre méthodologie d'annotation qui soit indépendant d'une vérité terrain préétablie. Cependant, en considérant les résultats obtenus, ce score d'accord ne semble pas répondre à cet objectif.

Tout d'abord, malgré une corrélation acceptable avec la performance théorique du *clustering* (moyenne à 0.49, voir Table 4.9), l'évolution du score d'accord reste instable. En effet, les nombreuses variations et saccades rendent toute analyse de rentabilité difficile, voire impossible, ce qui ne permet pas de définir un cas d'arrêt pour notre méthode d'annotation.

> **Exemples :** Concernant l'évolution du paramétrage favori (Figure 4.28 **(4)**), nous ne pouvons pas précisément définir à partir de quelle itération les résultats semblent intéressants, car le score d'accord oscille longuement entre 50% et 100% avec des pics de plus de 25% entre deux itérations.

> **Notes de l'auteur :** Après réflexion, ce score d'accord est probablement infructueux à cause du fonctionnement même de notre méthode, dont l'objectif est de corriger le partitionnement des données en utilisant un minimum de contraintes. En effet, dans le cadre de l'optimisation des paramètres réalisée en Section 4.2, nous avons retenu dans notre paramétrage favori la sélection des contraintes les plus proches entre deux *clusters* différents

(`samp.closest.diff`) : cette sélection permet ainsi de décrire efficacement l'emplacement des frontières de *clusters*.

Or, cet échantillonnage reste une méthode non supervisée : lors des premières itérations, les contraintes sélectionnées ont de bonnes chances de mettre en avant une frontière mal positionnée, mais au fur et à mesure que des contraintes s'ajoutent, les nouvelles contraintes ont moins de chances de trouver des bordures de *clusters* qui ne soient pas encore caractérisées. De ce fait, il se peut que les dernières sélections n'identifient aucune nouvelle frontière, qu'elles se concentrent sur des frontières déjà bien positionnées ou déjà décrites par d'autres contraintes, ou qu'elles nécessitent plusieurs itérations pour caractériser des frontières complexes (le comportement des autres méthodes de sélections représentées en FIGURE 4.28 peut être illustré par des raisonnements similaires). L'ensemble de ces cas de figures peut ainsi expliquer les nombreuses saccades dans l'évolution du score d'accord : tantôt la sélection semble pertinente, tantôt la sélection semble inutile.

Pour aller plus loin, nous pouvons aussi critiquer le score de corrélation qui ne semble pas montrer de lien fort entre les performances théoriques et les accords calculés, tant sur l'ensemble des tentatives que pour le paramétrage favori. Il est même rare d'observer des chutes importantes d'accords qui soient accompagnées d'une variation significative de `v-measure` avec la vérité terrain. Au final, ce score d'accord n'est donc pas vraiment représentatif de la rentabilité d'une itération ou de l'évolution de la pertinence du *clustering*.

Notes de l'auteur : Pour expliquer cette absence de corrélation, il est possible que l'analyse des annotations ait été une idée infructueuse : les 50 contraintes annotées peuvent peut-être exprimer un désaccord avec le précédent *clustering*, mais ce n'est pas pour autant que l'ajout de nouvelles contraintes impacte significativement la pertinence globale du partitionnement des données.

En conclusion, **le score d'accord entre l'annotation courante et le *clustering* précédent n'est pas adéquat pour estimer un cas d'arrêt de notre méthode d'annotation**, principalement car il est trop instable et qu'il ne représente pas bien les bénéfices obtenus à chaque itération. Ainsi, comme l'analyse de l'annotation n'est pas fructueuse, nous nous tournons vers l'analyse basée sur les différences entre deux résultats de *clustering*

4.5.2 Étude de l'évolution de la différence entre deux *clustering* consécutifs

Nous venons de conclure que l'analyse de l'accord entre l'annotation et le partitionnement des données ne permet pas d'estimer la rentabilité d'une itération de notre méthode d'annotation. Parmi les explications possibles, nous avons mis en cause l'analyse du lot de contraintes annotées : en effet, ce n'est pas parce que l'annotation de contraintes est en désaccord avec le précédent partitionnement des données que les correctifs associés auront un impact significatif sur le prochain partitionnement. Ainsi, nous voulons analyser l'évolution de la différence entre deux *clustering* successifs : en effet, si une itération apporte des correctifs ayant un impact, alors il devrait y avoir des différences visibles entre les deux itérations de *clustering*.

4.5.2.a Protocole expérimental

> ⚠ **Attention :** Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble.

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.10.

```
   Données : jeux de données annotées (vérités terrains)
 1 pour chaque jeu de données à tester faire
 2     initialisation (données) : récupérer les données et la vérité terrain ;
 3     initialisation (contraintes) : créer une liste vide de contraintes ;
 4     prétraitements : supprimer le bruit dans les données avec prep.simple ;
 5     vectorisation : transformer les données en vecteurs avec vect.tfidf ;
 6     clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
 7     répéter
 8         échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
 9         simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
10         intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
11         clustering : regrouper les données par similarité avec clust.kmeans.cop ;
12         rentabilité : calculer la différence entre les deux précédents clustering ;
13     jusqu'à annotation de toutes les contraintes possibles;
14 analyse 1 : afficher l'évolution de la différence entre deux clustering consécutifs ;
15 analyse 2 : calculer la corrélation entre le score de différence et le score de performance ;
   Résultat : discussion sur la rentabilité d'après la différence entre clustering
```

Algorithme 4.10 – *Description en pseudo-code du protocole expérimental de l'étude de l'évolution de la différence entre deux clustering consécutifs.*

Nous nous appuyons sur le même protocole que l'expérience précédente (cf. Section 4.5.1) : nous utilisons comme vérité terrain le jeu de données `Bank Cards (v1.0.0)`, nous réalisons 5 tentatives complètes de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori (voir Section 4.3), et nous simulons l'annotation par un expert d'un lot de 50 contraintes à chaque itération.

Cependant, au lieu de calculer un score d'accord entre annotation et *clustering*, nous estimons la différence entre le *clustering* précédent et le *clustering* obtenu grâce aux dernières annotations. Cette différence entre deux *clustering* X et Y est obtenue par la formule $1 - \texttt{v-measure}(X, Y)$ où la `v-measure` caractérise la ressemblance entre deux partitionnements des données. Pour nous permettre de discuter de l'utilité de ce score pour prédire la stabilisation du *clustering* et ainsi définir un cas d'arrêt de notre méthodologie d'annotation, nous calculons aussi le score de corrélation entre cette différence et la performance obtenue à l'aide d'une vérité terrain (la corrélation `r` de *Pearson* (Kirch, 2008) est utilisée).

> 💡 **Idées :** Comme précédemment, nous concentrons l'étude sur notre paramétrage favori (voir Section 4.4.3). Cependant, afin de compléter notre discussion avec d'autres

points de comparaison, nous analysons aussi les autres paramétrages implémentés, notamment les meilleurs paramétrages moyens identifiés lors de l'hypothèse d'efficience (voir Section 4.2).

Pour information : Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.5.2.b Résultats obtenus

La Figure 4.29 représente l'évolution moyenne du score de différence entre deux *clustering* pour les quatre paramétrages mis en avant lors de nos études. Nous pouvons constater une tendance générale à la décroissance vers 0% de ce score de différence : pour le paramétrage favori, la différence moyenne entre deux *clustering* est initialement comprise entre 25% et 35% jusqu'à l'itération 10, elle chute ensuite pour être inférieure à 5% après l'itération 20, et elle termine enfin en oscillant très légèrement (±1%) autour de 0% jusqu'à la fin des annotations.

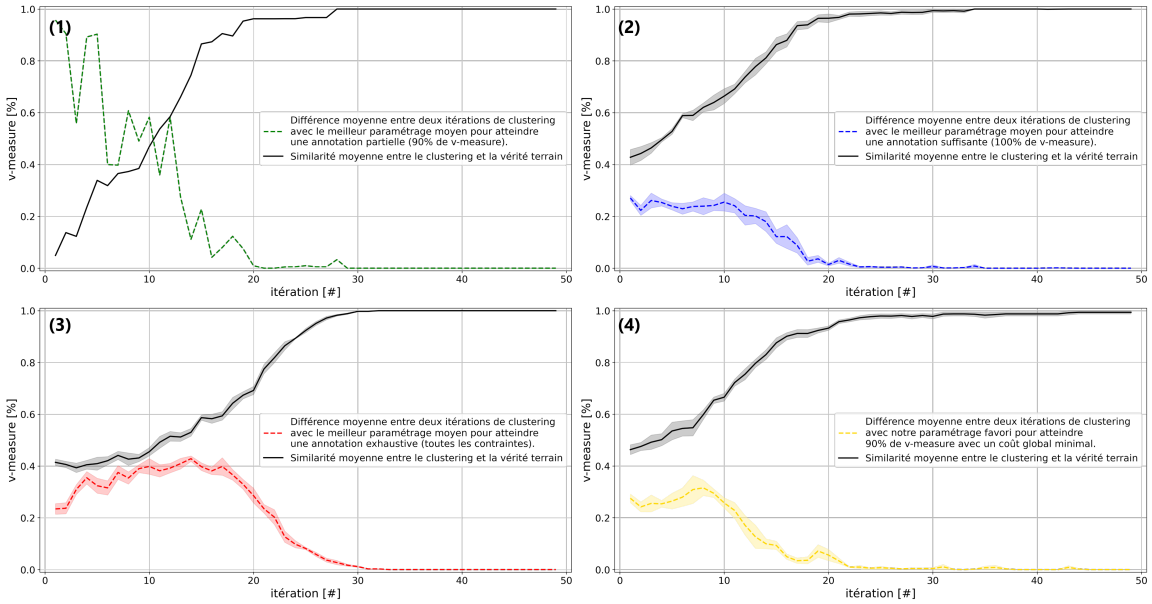

Figure 4.29 – *Évolution de la différence de résultats entre deux itérations de clustering. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori. À titre d'information, les courbes en noir représentent l'évolution de la* `v-measure` *entre le clustering et la vérité terrain.*

La Table 4.10 contient le score de corrélation entre cette différence et la performance théorique obtenue grâce à la vérité terrain. Cette corrélation est forte : 0.75 sur l'ensemble des

tentatives, 0.93 sur les tentatives utilisant notre paramétrage favori. La FIGURE 4.29 confirme cette corrélation :

— un score de `v-measure` avec la vérité terrain proche de 100% est accompagné d'un score de différence proche de 0% (après l'itération 20 pour **(1)**, après l'itération 20 pour **(2)**, après l'itération 30 pour **(3)** et après l'itération 22 pour **(4)**) ;

— une croissance de performance est généralement accompagnée d'un score non nul de différence (voir **(2)** et **(4)** entre les itérations 0 et 20), et plusieurs pics de performance sont accompagnés de scores forts de différence (particulièrement visible sur **(1)** vers l'itération 5 et entre les itérations 10 et 15) ;

— il est toutefois à noter que l'inverse n'est pas vrai : un score non nul de différence n'accompagne pas forcément une croissance de performance, mais peut simplement caractériser un changement de partitionnement, comme c'est le cas dans **(3)** entre les itérations 0 et 10 où des modifications ont lieu (score de différence non nul) mais où la performance par rapport à la vérité terrain stagne.

Paramétrage	Corrélation
Meilleur paramétrage moyen pour une annotation partielle **(1)**	0.96
Meilleur paramétrage moyen pour une annotation suffisante **(2)**	0.92
Meilleur paramétrage moyen pour une annotation exhaustive **(3)**	0.85
Paramétrage favori **(4)**	0.93
Moyenne des 960 tentatives	0.75

TABLE 4.10 – *Score de corrélation `r` de Pearson entre la performance du clustering obtenu à l'aide d'une vérité terrain (`v-measure`) et le score de différence entre deux clustering consécutifs.*

Les autres paramétrages représentés dans **(1)**, **(2)** et **(3)** comportent des tendances similaires (*décroissance générale, forte corrélation avec la performance théorique*) à quelques détails près (***(1)** commence avec des scores de différence très forts avant de décroître avec de nombreux pics ; **(3)** croît légèrement avant d'entamer sa décroissance, ...*).

4.5.2.c Discussion

Dans cette étude, nous avons analysé l'évolution du score de différence entre deux itérations de *clustering* dans l'espoir de définir un cas d'arrêt de notre méthodologie d'annotation qui soit indépendant d'une vérité terrain préétablie.

Tout d'abord, nous pouvons affirmer qu'il y a une forte corrélation entre l'évolution de ce score de différence et l'évolution du score de performance (voir TABLE 4.10 : `r` moyen de 0.75 ; `r` supérieur à 0.85 pour les paramétrages mis en avant). Cette corrélation est confirmée visuellement grâce à la FIGURE 4.29 : plus les différences entre *clustering* sont faibles, plus les performances des *clustering* sont fortes.

Un point d'attention est toutefois à retenir : une modification du partitionnement des données n'entraîne pas forcément un gain de performance (voir **(3)** entre les itérations 0 et 10 et **(4)** entre les itérations 0 et 8). Nous ne pouvons donc pas conclure que l'analyse de la différence entre deux itérations de *clustering* permet de caractériser totalement la rentabilité d'une itération.

Cependant, nous pouvons tout de même nous servir de ce score pour définir un cas d'arrêt pour notre méthodologie d'annotation lorsque la différence entre deux *clustering* est faible. Pour

cela, il nous suffit de fixer un seuil bas du score de différence en dessous duquel il n'est plus rentable de faire de nouvelles itérations de la méthode car les performances ne s'améliorent plus significativement. Une analyse manuelle ou semi-manuelle (voir hypothèse de pertinence en Section 4.4) reste nécessaire pour confirmer la valeur métier du résultat obtenu.

Idées : Si nous restons sur notre seuil théorique de 90% de `v-measure` (voir Section 4.2) et que nous nous basons sur la Figure 4.29 **(4)**, nous pouvons visuellement fixer ce seuil autour de 5% de différences. Le réglage fin de ce seuil pourra être le sujet de futures analyses complémentaires.

En conclusion, **le score de différences entre deux résultats de *clustering* semble être un bon indicateur pour estimer un cas d'arrêt de notre méthodologie d'annotation**. Nous proposons d'utiliser un seuil par défaut de 5% pour implémenter ce cas d'arrêt.

4.5.3 Mise en commun des stratégies d'évaluation de la rentabilité d'une itération de la méthode et définition d'un cas d'arrêt indépendant d'une vérité terrain.

Points à retenir : Au cours de cette étude de rentabilité, nous avons pu voir que :

- ☒ l'analyse du score d'accord entre l'annotation courante et le *clustering* précédent ne permet pas d'estimer la rentabilité d'une itération, ni de définir un cas d'arrêt de notre méthodologie d'annotation (cf. Section 4.5.1) ;
- ☑ l'analyse des différences entre deux itérations de *clustering* est une approche prometteuse pour estimer la rentabilité d'une itération (cf. Section 4.5.2), bien qu'une modification significative entre deux résultats de *clustering* n'implique pas forcément un gain de performance (*les deux clustering peuvent être différents mais avoir des* `v-measure` *avec la vérité terrain équivalentes*) ;
- ☑ l'usage de différences entre deux itérations de *clustering* permet de définir un cas d'arrêt de notre méthodologie d'annotation : si les différences sont faibles (*par exemple : inférieures à* 5%), alors les performances stagnent ou plafonnent ; il peut alors être intéressant d'interrompre le `Clustering Interactif` après avoir vérifié manuellement la pertinence des résultats obtenus (cf. Section 4.4.4).

Pour terminer nos différentes analyses, il convient maintenant d'anticiper la présence de différences d'annotation. En effet, nous avons fait jusqu'à présent l'hypothèse que l'annotateur ne se trompe jamais et que deux annotateurs n'ont jamais de désaccords, mais cette hypothèse forte n'est pas toujours vérifiée en pratique. Pour estimer l'impact de ces incohérences d'annotation, nous devons donc réaliser une analyse de robustesse de notre méthode d'annotation : celle-ci sera réalisée en Section 4.6.

4.6 Évaluation de l'hypothèse de robustesse

Dans les précédentes études, nous avons presque toujours analysé le `Clustering Interactif` en supposant que l'annotateur connaissait parfaitement le domaine traité par le jeu de données et qu'il était capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Bien entendu, cette hypothèse forte n'est pas toujours vérifiée en situation réelle : l'interprétation du langage peut contenir certaines ambiguïtés, l'opérateur peut faire des erreurs d'inattention, et deux annotateurs peuvent avoir des avis contraires sur un même sujet. Or, comme notre méthode d'annotation est itérative, elle est *a priori* sensible aux dérives de fonctionnement liées à ce type de contradictions. Dans cette section, nous nous intéressons donc à la robustesse du `Clustering Interactif` en présence d'incohérences dans les contraintes et aux moyens de les contrer. Pour cela, nous aimerions donc vérifier l'hypothèse suivante :

Hypothèse de robustesse

« **Au cours d'une méthodologie d'annotation basée sur le `Clustering Interactif`, il est possible d'estimer le taux d'incohérences dans les contraintes ainsi que leur impact sur les résultats de la méthode.** »

La FIGURE 4.30 illustre cette hypothèse et la perspective d'estimer l'impact de différences d'annotations sur le nombre d'itérations de la méthode.

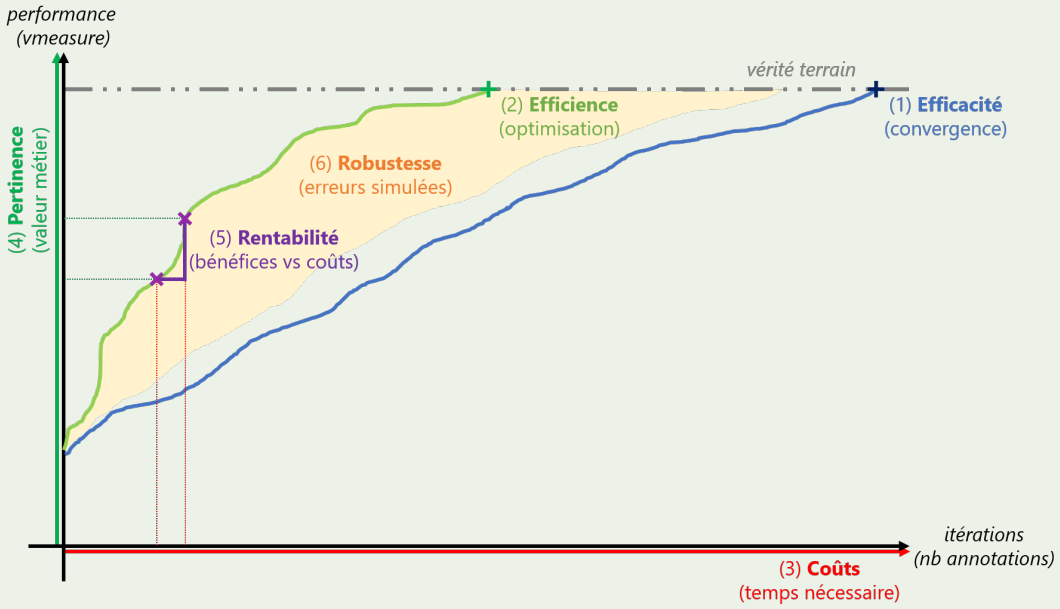

FIGURE 4.30 – *Illustration des études réalisées sur le `Clustering Interactif` (étape 6/6) en schématisant l'évolution de la* ***pertinence*** *(valeur métier évaluée par l'expert, exprimée en nombre de clusters) d'une base d'apprentissage en cours de construction, en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine), ainsi que les* ***estimations d'erreurs*** *représentant l'impact de différences d'annotation sur le nombre d'itérations nécessaire à la méthode.*

Afin de vérifier cette hypothèse, nous organisons trois expériences :

— une étude de cas d'un **score inter-annotateurs** obtenu lors d'une annotation de contraintes en situation réelle avec plusieurs opérateurs, permettant d'estimer une borne maximale du taux de désaccords d'annotation pour les autres études (cf. Section 4.6.1) ;

— une étude de l'**impact d'une erreur d'annotation** et de l'**intérêt de la corriger**, en simulant l'insertion d'erreurs d'annotation et en mesurant la similarité du *clustering* obtenu avec la vérité terrain (cf. Section 4.6.2) ;

— une étude de l'**impact de la subjectivité de l'annotation sur la divergence des résultats de *clustering***, en simulant l'insertion de différences d'annotation et en mesurant la similarité entre les *clustering* obtenus (cf. Section 4.6.3).

4.6.1 Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle

Afin d'affiner le champ de recherche de nos futures études, nous voulons analyser le score d'accords inter-annotateurs calculé lors d'une expérience d'annotation de contraintes par plusieurs experts métiers en situation réelle. Pour cela, nous reprenons l'expérience de la Section 4.3.1 visant à estimer le temps moyen d'annotation d'un lot de contraintes, et nous réutilisons les résultats de cette expérience pour en estimer l'accord inter-annotateurs. Comme l'objectif de cette précédente étude n'était pas d'étudier la qualité des annotations, aucun guide ni aucune règle d'annotation précise n'avaient été fournis : nous espérons donc pourvoir **estimer une borne maximale grossière du désaccord entre annotateurs** lors de l'utilisation de notre méthode, permettant ainsi d'affiner notre discussion.

4.6.1.a Protocole expérimental

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.11.

Données : jeu de données annotées (vérité terrain)
Entrées : plusieurs réviseurs, plusieurs annotateurs
1 **initialisation** : définir et revoir le jeu de données entre réviseurs ;
2 **échantillonnage** : sélectionner une base de contraintes équilibrée ;
3 **pour chaque** *annotateur* **faire**
4 **tant que** *la base de contraintes n'a pas été entièrement annotée* **faire**
5 **annotation** : annoter une partie des contraintes ;
6 **revue** : revue des contraintes en conflits d'annotation ;
Résultat : modélisation du score inter-annotateurs sur le lot de contraintes

Algorithme 4.11 – *Description en pseudo-code du protocole expérimental de l'étude du score inter-annotateurs d'annotation d'un lot de contraintes par plusieurs experts métiers en situation réelle.*

Concernant la précédente expérience d'annotation en situation réelle, nous avions procédé en plusieurs étapes. D'abord, il fallait choisir un jeu de données approprié : pour valider notre hypothèse forte sur les compétences de nos annotateurs, nous cherchions un jeu de données traitant d'un sujet de culture générale. Pour cette expérience, nous avions donc choisi `MLSUM` : une collecte d'articles de journaux, classés par catégorie de publication et décrits par leur titre et leur résumé. Nous nous intéressions ici à la tâche de classification d'un titre d'article en fonction

de sa catégorie de publication. Comme certains titres pouvaient porter à confusion (un titre d'article n'étant pas toujours explicite sur son contenu), deux réviseurs (*une Data Scientist et moi-même*) furent chargés de choisir les données les plus explicites sur un échantillon d'un millier de données représentatives des catégories les plus communes. L'échantillon résultant, noté `MLSUM FR Train Subset (v1.0.0-schild)`, est composé de 744 titres d'articles rédigés en français et répartis en 14 classes (*économie*, *sport*, ...). Pour plus de détails, consulter l'Annexe A.2.

À partir de ces données, nous avions sélectionné un lot de 400 contraintes à annoter. Pour faciliter l'analyse, l'échantillonnage fut un tirage aléatoire équilibré d'après la vérité terrain en 200 `MUST-LINK` et en 200 `CANNOT-LINK`.

Ensuite, 3 annotateurs ont annoté la sélection des 400 contraintes en plusieurs sessions. Les directives données aux opérateurs étaient les suivantes :

— **Contexte de l'opérateur** : « *Vous êtes des **experts de la presse et de l'actualité**. Vous voulez classer des articles dans des catégories en fonction de leur titre. Vous ne savez pas précisément quelles catégories vous allez utiliser pour classer vos articles. Mais vous savez **caractériser la similitude** entre deux articles* ».

— **Contexte du jeu de données** : « *Le thème concerne les catégories d'articles de presse. La vérité terrain contient entre* 10 *et* 20 *catégories parmi les plus communes de la presse. La vérité terrain contient entre* 30 *et* 100 *articles par catégorie. Vous **pouvez regarder le jeu de données non annoté** autant que vous le voulez (disponible dans l'onglet `TEXTS` de l'application)* ».

— **Objectif de l'expérience** : « *Je veux savoir le temps nécessaire pour annoter un certain nombre de contraintes. Autrement dit : **pour annoter 1000 contraintes, combien de temps me faut-il ?*** ».

— **Consignes d'annotations** : « *Faites des séries de **15 minutes minimum** pour avoir de la régularité. Si possible, **isolez-vous** pour ne pas être dérangé et ne pas fausser les résultats. Pour chaque série, **notez le temps et le nombre de contraintes annotés**. Si vous ne savez pas quoi annoter (trop ambigu, vocabulaire inconnu, ...), **passez au suivant sans annoter** (vous êtes censés être des experts de la presse !)* ».

Pour réaliser l'annotation, les opérateurs eurent accès à l'application web développée au cours de ce doctorat. Des captures d'écran sont disponibles en Figure 4.11 et Figure 4.12. Une description plus détaillée de l'application et de ses fonctionnalités est disponible en Annexe C.2. Il est à noter que l'autre réviseur a aussi participé à l'annotation de ces contraintes : nous avons retenu ses résultats, mais nous les analyserons séparément du groupe d'annotateurs.

Pour cette étude, nous allons calculer le score d'accord inter-annotateurs global et deux à deux. Pour ce faire, nous utilisons l'α de *Krippendorff*[44] (Krippendorff, 2004) implémenté dans la librairie `simpledorff`[45] (Perry, 2021). Nous rappelons qu'un accord est considéré comme *faible* si $\alpha < 0.667$, *acceptable* si $0.667 \leq \alpha < 0.8$, *fort* si $0.8 \leq \alpha < 1.0$ et *parfait* si $\alpha = 1.0$. De plus, un score α négatif représente une opposition d'accord.

> **Pour information :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild,

44. Choix de l'α de *Krippendorff* : Le κ de *Cohen* (Landis et Koch, 1977) est aussi fréquemment utilisé, mais il n'est pas adapté pour plus de deux opérateurs ni pour manipuler des absences d'annotations.
45. https://pypi.org/project/simpledorff/

2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.6.1.b Résultats obtenus

Durant cette expérience, 4 opérateurs (1 réviseur, 3 annotateurs) ont participé à l'annotation de 400 contraintes issues d'un tirage aléatoire équilibré d'après la vérité terrain en 200 `MUST-LINK` et en 200 `CANNOT-LINK`. Ces opérateurs travaillent tous dans un service informatique dédié à l'entraînement et l'amélioration de solutions de *Machine Learning* et sont répartis de la manière suivante :

— 2 femmes, 2 hommes ;

— 4 personnes entre 20 et 30 ans ;

— 4 *Data Scientist* ;

— 1 personne ayant révisé le jeu de données, 3 le découvrant pour la première fois.

La Table 4.11 expose les accords des opérateurs (1 réviseur, 3 annotateurs) par rapport à la vérité terrain. Nous pouvons constater les points suivants :

— le réviseur possède un accord *fort* avec la vérité terrain ($\alpha = 0.892$), confirmant sa connaissance de celle-ci ;

— les autres annotateurs, ne connaissant pas la vérité terrain, ont tous un accord *acceptable* avec celle-ci $\alpha \geq 0.685$; d'après le taux d'accord brut, le pourcentage de désaccords d'annotations concerne entre 13% et 16% de contraintes pour chaque annotateur ;

— malgré ces désaccords, l'accord inter-opérateurs global par rapport à la vérité terrain reste *acceptable* ($\alpha = 0.697$ et $\alpha = 0.735$).

Opérateurs	Accord avec la vérité terrain		
	Accord brut	α Krippendorff	Interprétation α
1 (réviseur)	94.75%	0.892	*fort*
7 (annotateur)	87.50%	0.750	*acceptable*
9 (annotateur)	84.25%	0.685	*acceptable*
12 (annotateur)	87.00%	0.737	*acceptable*
Annotateurs 7, 9 et 12	72.00%	0.697	*acceptable*
Tous (1, 7, 9 et 12)	71.75%	0.735	*acceptable*

Table 4.11 – *Score d'accord avec la vérité terrain des 4 opérateurs (1 réviseur, 3 annotateurs) sur un lot commun de 400 contraintes (200 `MUST-LINK`, 200 `CANNOT-LINK`). L'accord brut représente le pourcentage de contraintes ayant la même annotation et l'accord α représente la mesure de Krippendorff. Les numéros d'opérateurs correspondent à leurs identifiants lors de l'expérience.*

La Table 4.12 expose les accords inter-opérateurs (1 réviseur, 3 annotateurs). Nous pouvons constater les points suivants :

— les accords deux à deux concernent à chaque fois au moins 80.25% des contraintes ;

— une seule paire d'opérateurs est considéré comme ayant un accord *faible* ($\alpha = 0.597 < 0.667$) ;

— les autres annotateurs, ne connaissant pas la vérité terrain, ont tous un accord *acceptable* avec celle-ci $\alpha \geq 0.685$; d'après le taux d'accord brut, le pourcentage de désaccords d'annotations concerne entre 13% et 16% de contraintes pour chaque annotateur ;

— malgré ces désaccords, l'accord inter-opérateurs global reste *acceptable* ($\alpha = 0.669$ et $\alpha = 0.715$).

Opérateurs	Accord inter-annotateurs		
	Accord brut	α Krippendorff	Interprétation α
1 (réviseur) et 7 (annotateur)	91.50%	0.825	*fort*
1 (réviseur) et 9 (annotateur)	86.00%	0.711	*acceptable*
1 (réviseur) et 12 (annotateur)	89.50%	0.780	*acceptable*
7 (annotateur) et 9 (annotateur)	85.75%	0.714	*acceptable*
7 (annotateur) et 12 (annotateur)	85.25%	0.699	*acceptable*
9 (annotateur) et 12 (annotateur)	80.25%	0.597	*faible*
Annotateurs 7, 9 et 12	75.50%	0.669	*acceptable*
Tous (1, 7, 9 et 12)	74.00%	0.715	*acceptable*

Table 4.12 – *Score d'accord inter-opérateurs (1 réviseur, 3 annotateurs) sur un lot commun de 400 contraintes (200 `MUST-LINK`, 200 `CANNOT-LINK`). L'accord brut représente le pourcentage de contraintes ayant la même annotation et l'accord α représente la mesure de Krippendorff. Les numéros d'opérateurs correspondent à leurs identifiants lors de l'expérience.*

4.6.1.c Discussion

L'objectif de cette étude est l'estimation du taux de désaccords d'annotation qui peuvent apparaître en utilisant notre méthode. Pour cela, nous avons réutilisé les annotations réalisées dans une précédente expérience (voir l'étude du temps d'annotation en Section 4.3.1), en espérant ainsi déterminer une borne maximale de ce désaccord entre annotateurs.

Pour rappel, nous avions fait l'hypothèse que les opérateurs étaient des experts du domaine qu'ils annotent. Afin de valider cette hypothèse, nous avions choisi une vérité terrain traitant d'un sujet de culture générale (ici : la presse) et deux réviseurs avaient revu cette vérité terrain en amont dans le but de supprimer au mieux les données ambiguës. Cette hypothèse est *a priori* valable en considérant que les opérateurs ont eu de bons scores d'accord avec la vérité terrain : un accord *fort* pour le réviseur ($\alpha = 0.892$), et des accords *acceptables* pour les annotateurs ($\alpha \geq 0.685$).

Cependant, nous constatons aussi que nos consignes d'annotations n'étaient probablement pas assez explicites. En effet, aucun annotateur n'a d'accord *fort* avec la vérité terrain ($\alpha < 0.750$), et le score d'accord inter-opérateurs des trois annotateurs est simplement *acceptable*. Nous pouvons donc assimiler cette expérience à un projet d'annotation dont les règles sont légèrement ambiguës, empêchant ainsi d'avoir un accord *fort* entre les annotateurs.

Ainsi, nous utilisons les taux de désaccords bruts pour affiner notre champ de recherche pour nos prochaines simulations d'erreurs (voir Section 4.6.2) et de désaccords (voir Section 4.6.3) :

— Sur la base des résultats rapportés dans la TABLE 4.11, nous observons un accord brut avec la vérité terrain sur au moins 84.25% des contraintes annotées, c'est-à-dire que le désaccord maximal concerne au plus 15.75% des annotations ;

— Sur la base des résultats rapportés dans la TABLE 4.12, nous observons un accord brut inter-annotateurs sur au moins 80.25% des contraintes annotées, c'est-à-dire que le désaccord maximal concerne au plus 19.75% des annotations .

Par conséquent, et afin de prendre en compte ces résultats et de s'accorder une marge supplémentaire, **nous décidons de considérer 25% comme une borne maximale des taux d'erreurs de désaccords dans nos futures simulations**.

> **Notes de l'auteur :** Il est à noter qu'un taux d'erreurs ou de désaccords de 25% semble toutefois considérable pour une simple annotation binaire. En effet, nous avons pu voir dans la TABLE 4.12 qu'un taux de désaccords de 19.75% était déjà considéré comme représentatif d'un accord *faible*. De ce fait, un taux de désaccords plus important pourrait plutôt être imputé à une mauvaise organisation du projet d'annotation (*par exemple avec des opérateurs non formés à la labellisation ou des règles d'annotation minimalistes*).
>
> Dans les deux prochaines études, nous allons tout de même simuler des erreurs et des désaccords jusqu'à 25%, mais nous garderons à l'esprit que des désaccords de cet ordre de grandeur sont plutôt caractéristiques d'un problème de gestion de projet.

4.6.2 Étude de l'impact d'une erreur d'annotation et l'intérêt de la corriger

Dans cette seconde étude, nous cherchons à estimer la robustesse du `Clustering Interactif` en nous intéressant plus particulièrement aux erreurs d'annotation d'un seul opérateur. En effet, comme nous l'avons vu en SECTION 2.3.3.B, des **différences de comportements intra-annotateur** peuvent être observées au cours de la labellisation, ces différences s'apparentant à des erreurs d'inattention ou des contradictions. Nous allons ici simuler de telles erreurs dans l'annotation de contraintes dans le but :

— d'observer leur détection par le gestionnaire de contraintes ;

— d'évaluer la perte de performances par rapport à la vérité terrain si les contradictions détectées ne sont pas corrigées.

4.6.2.a Protocole expérimental

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'ALGORITHME 4.12.

Nous utilisons comme vérité terrain le jeu de données `Bank Cards (v1.0.0)` : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'ANNEXE A.1.

Sur ce jeu de données, nous exécutons une tentative complète [46] de la méthode du `Clustering`

46. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

Données : jeu de données annotées (vérité terrain)
Entrées : liste de stratégies de correction de conflits et de taux d'erreurs à insérer
pour chaque *stratégie de correction et taux d'erreurs à insérer* **faire**
  initialisation (données) : récupérer les données et la vérité terrain ;
  initialisation (contraintes) : créer une liste vide de contraintes ;
  prétraitements : supprimer le bruit dans les données avec `prep.simple` ;
  vectorisation : transformer les données en vecteurs avec `vect.tfidf` ;
  clustering initial : regrouper les données par similarité avec `clust.kmeans.cop` ;
  évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  répéter
    échantillonnage : sélectionner des contraintes avec `samp.closest.diff` ;
    choix des erreurs : définir les contraintes erronées ;
    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
    si *stratégie de correction naïve* **alors**
      intégration naïve : ajouter les nouvelles contraintes au gestionnaire de contraintes, et ignorer les conflits avec le gestionnaire de contraintes ;
    sinon si *stratégie avec correction* **alors**
      intégration corrective : ajouter les nouvelles contraintes au gestionnaire de contraintes, et ré-annoter les annotations en conflit ;
    clustering : regrouper les données par similarité avec `clust.kmeans.cop` ;
    évaluation : estimer l'équivalence entre le *clustering* et la vérité terrain ;
  jusqu'à *annotation de toutes les contraintes possibles*;
  analyse locale : afficher l'évolution de la similarité entre les *clustering* de la tentative courante et la vérité terrain ;
analyse générale : déterminer l'impact des stratégies de correction en fonction des taux d'erreurs insérées ;
Résultat : discussion sur l'impact des erreurs et l'intérêt de les corriger

ALGORITHME 4.12 – *Description en pseudo-code du protocole expérimental de l'étude d'impact d'une erreur d'annotation et l'intérêt de la corriger.*

`Interactif` en utilisant notre paramétrage favori [47] (voir Section 4.3). Toutefois, contrairement aux précédentes expériences, nous allons ajouter un pourcentage de contraintes erronées à chaque itération pour simuler les variations de comportement de l'annotateur :

— Le taux d'erreurs insérées, variant de 0% à 25% [48] par pas de 5%, reste fixe tout au long d'une même tentative de notre méthode : nous pouvons ainsi analyser l'impact d'un taux d'erreur fixe sur les résultats au cours des itérations ;

— Les contraintes erronées à insérer sont tirées aléatoirement parmi le lot de contraintes qui aurait été échantillonnées au cours d'une tentative sans erreurs : ainsi, nous pouvons comparer itération par itération toutes ces simulations car elles partagent la même base de contraintes (aux valeurs de `MUST-LINK` et `CANNOT-LINK` près) ;

Puisque nous introduisons des erreurs d'annotation, des conflits peuvent apparaître dans le gestionnaire de contraintes. Pour rappel, un conflit est détecté dans le cas où l'ajout d'une nouvelle contrainte annotée contredit ce qui a été précédemment déduit grâce aux propriétés de transitivité des contraintes de types `MUST-LINK` et `CANNOT-LINK` (voir Figure C.1 en Annexe C.1.2). Pour les traiter, nous allons tester deux approches :

— une approche *naïve* ignorant les conflits : si la prochaine contrainte à ajouter est incompatible avec la base de contraintes déjà intégrées au gestionnaire, alors nous ignorons simplement son existence sans remettre en question les précédentes annotations ;

— une approche *avec correction* des conflits : pour simuler la correction d'un expert, nous recréons à chaque itération le gestionnaire de contraintes en intégrant d'abord les contraintes correctes puis les contraintes erronées ; ainsi, les conflits ne peuvent arriver qu'à l'ajout d'une contrainte erronée, et il suffit de réévaluer la contrainte par rapport à la vérité terrain pour simuler la correction de l'expert.

Ainsi, il y a donc 6 taux d'erreurs d'annotation à simuler, chacun suivant 2 approches de gestion de conflits ; chacune de ces simulations d'erreurs sera répétée 10 fois sur chaque tentative complète de la méthode pour contrer les aléas statistiques des tirages de contraintes erronées, ce qui représente 120 simulations par tentative. Enfin, chaque tentative complète de `Clustering Interactif` est répétée 5 fois pour contrer les aléas statistiques des exécutions, ce qui représente un total de 600 tentatives complètes à simuler.

Enfin, nous affichons l'évolution de la performance moyenne du *clustering* obtenu en fonction des divers taux d'erreurs simulées, et nous discutons de la significativité de la perte de performances due à l'absence de corrections des conflits.

> **Pour information :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

47. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitements simples (`prep.simple`), vectorisation `TF-IDF` (`vect.tfidf`), *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

48. Choix de 0% à 25% : nous utilisons ici les estimations grossières des bornes maximales d'erreurs réalisées en Section 4.6.1.

4.6.2.b Résultats obtenus

La FIGURE 4.31 représente l'évolution moyenne de la `v-measure` du *clustering* en fonction du nombre de contraintes annotées au cours des itérations de la méthode, et cette évolution est déclinée pour les 6 taux d'erreurs simulées et les 2 approches de gestion des conflits. Les contraintes utilisées sont basées sur les échantillonnages réalisés au cours des tentatives n'introduisant pas d'erreurs d'annotation par rapport à la vérité terrain : comme les mêmes contraintes sont utilisées (aux valeurs d'annotations près), toutes les courbes sont comparables point par point.

> ⚠ **Attention :** Toutefois, il est important de noter que les tentatives sans erreurs ont besoin de maximum 3 000 contraintes pour annoter toutes les contraintes possibles et leurs transitivités (moyenne : 2 488, écart-type : 327). Au delà de cette limite, il faudrait échantillonner de nouvelles contraintes pour les tentatives introduisant des erreurs, mais les bases de contraintes utilisées ne seraient alors plus comparables. Nous décidons donc de tronquer les différentes courbes à 3 000 contraintes, que la convergence ait lieu ou non, et nous analysons les résultats partiels obtenus pour ce nombre de contraintes.

Tout d'abord, observons l'approche *naïve*, ignorant simplement les conflits (voir FIGURE 4.31 **(1)**). Nous pouvons constater que les performances (basées sur la vérité terrain) plafonnent très rapidement si des incohérences sont introduites : dès 5% d'erreurs, le score de `v-measure` stagne autour de 65% à partir de 1 000 contraintes, alors qu'une performance théorique d'environ 90% serait attendue pour une tentative idéale n'introduisant pas d'erreurs. D'autre part, si le taux de contraintes erronées est supérieur à 15%, nous pouvons remarquer que ces performances diminuent avant de stagner à des valeurs inférieures à 40% de `v-measure`. Pour finir sur cette première approche, tout porte à croire que ces faibles seuils de performances perdurent au delà des 3 000 contraintes (figure tronquée) : les tentatives s'éloignent donc significativement de la vérité terrain si des incohérences d'annotation sont insérées.

Observons désormais l'approche *avec correction*, réévaluant les contraintes erronées par rapport à la vérité terrain lorsqu'un conflit est détecté par le gestionnaire de contraintes (voir FIGURE 4.31 **(2)**). Nous pouvons constater que les tentatives introduisant des contraintes erronées subissent aussi un retard de performances, mais celui-ci est bien plus faible que celui encaissé par les approches naïves. Nous observons que toutes les courbes restent globalement croissantes, bien que le score moyen de 90% de `v-measure` par rapport à la vérité terrain n'est pas atteint en moins de 3 000 contraintes par les tentatives ayant un taux d'erreurs supérieur à 15%. Nous pouvons toutefois espérer que cette convergence se poursuit au delà des 3 000 contraintes (figure tronquée).

4.6.2.c Discussion

L'objectif de cette étude est l'analyse de l'impact des différences de comportements intra-annotateur sur les résultats de notre méthode, plus particulièrement l'intérêt de corriger les incohérences d'annotations lorsqu'elles sont détectées par le gestionnaire de contraintes. Pour cela, nous avons comparé deux approches : une approche naïve ignorant simplement les conflits, et une deuxième approche les corrigeant lorsque ces derniers sont détectées.

D'après l'analyse de la FIGURE 4.31, nous pouvons clairement déduire que l'absence de correction des incohérences pénalise significativement la méthode. En effet, sur le jeu de données utilisé comme vérité terrain, nous constatons une perte irréversible d'au moins 35% de `v-measure`

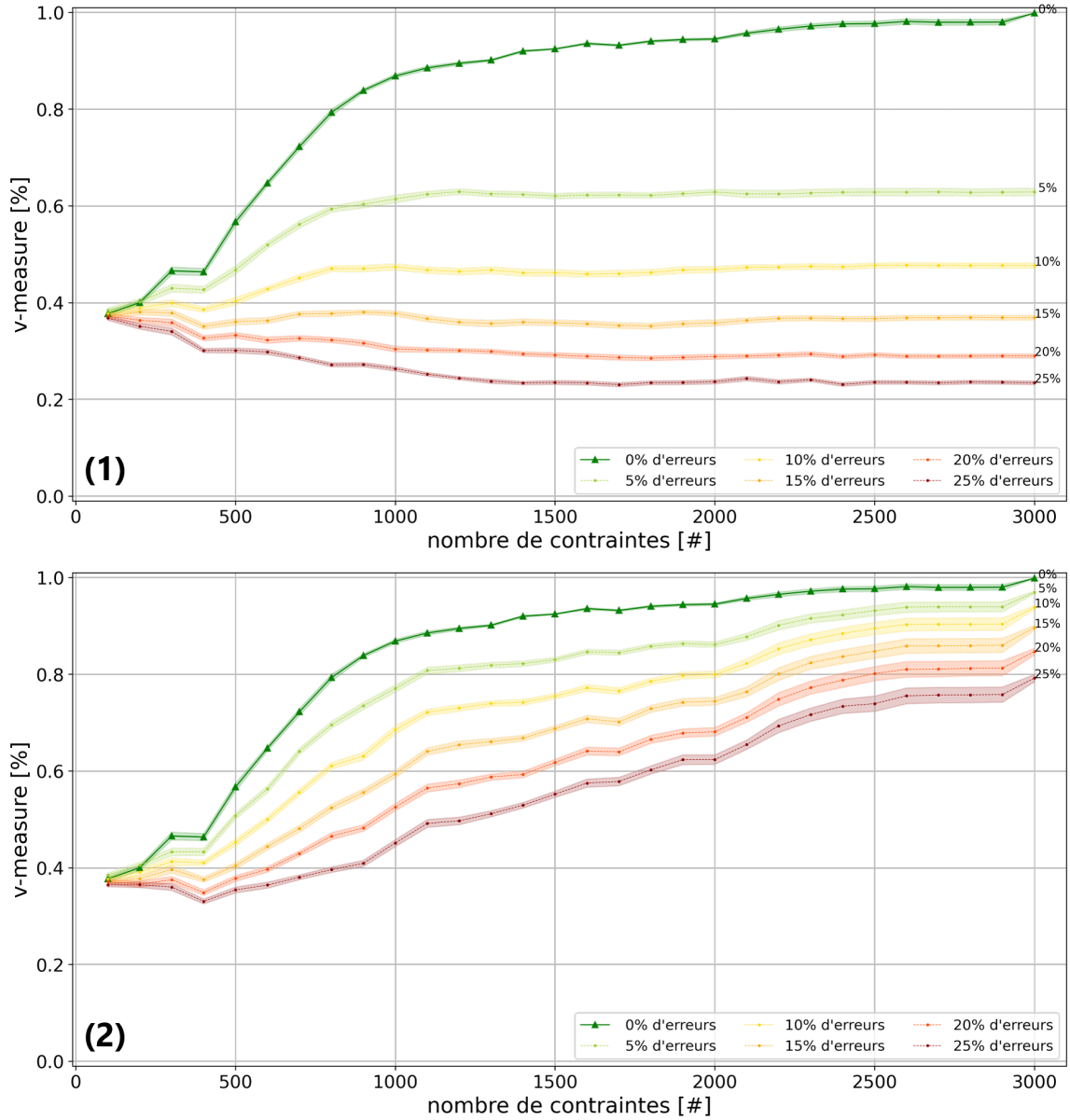

FIGURE 4.31 – *Évolution des similitudes moyennes (calculées en terme de* `v-measure`*) des résultats de clustering des tentatives introduisant des erreurs d'annotation par rapport à la vérité terrain au cours des itérations. Les dégradés de couleurs des courbes représentent les déclinaisons de ces évolutions en fonction des différents taux d'annotations erronées (allant de 0% et 25%). (1) représente l'approche naïve ignorant les conflits d'annotation et (2) représente l'approche corrigeant les conflits détectés par le gestionnaire de contraintes. Toutes les courbes sont tronquées à 3 000 contraintes (nombre maximum de contraintes nécessaires à une tentative n'introduisant pas d'erreurs pour converger vers la vérité terrain).*

dès l'introduction de 5% d'erreurs d'annotation, caractérisant ainsi une dérive importante des résultats si les conflits d'annotations ne sont pas corrigés. Toutefois, la mise en oeuvre d'un mécanisme simple de correction semble estomper en partie ces régressions de performances et permet à certaines tentatives introduisant des erreurs de rester compétitives (*toutes les tenta-*

tives peuvent espérer atteindre 80% *de* `v-measure` *en corrigeant leurs incohérences, alors que l'approche naïve les condamnait à une* `v-measure` *plafonnée*). Ces observations dépendent bien entendu fortement du jeu de données utilisé pour réaliser ces simulations. Nous pouvons néanmoins assurément conclure en faveur de la **nécessité de vérifier la base de contraintes et de corriger toute incohérence s'y trouvant**.

Une telle conclusion confirme la sensibilité aux erreurs de cette approche incrémentale : si une erreur apparaît, elle peut rapidement se propager et impacter les résultats de la méthode. Dans notre cas, nous avons optimisé l'implémentation de notre `Clustering Interactif` pour obtenir un corpus d'apprentissage pertinent en un minimum d'annotations de contraintes. Or obtenir un nombre minimal implique de limiter la redondance parmi les contraintes annotées. De ce fait, certaines incohérences mal placées peuvent fortement influencer la base de contraintes et ainsi faire diverger les résultats.

Pour contrer ce problème, nous avons deux pistes pouvant être explorées :

— **introduire intentionnellement de la redondance** dans la base de contraintes annotées à l'aide d'une nouvelle méthode d'échantillonnage : ce moyen tire parti des propriétés de transitivité du graphe de contraintes pour identifier d'éventuelles incohérences d'annotation isolées ou masquées. Cette piste a cependant le désavantage d'ajouter de nouvelles contraintes peu informatives dans le simple but de mieux détecter certaines incohérences, introduisant donc un léger coût supplémentaire pour l'annotateur ;

— **confronter plusieurs des annotateurs** sur les mêmes contraintes : en faisant annoter les mêmes contraintes par deux opérateurs différents, les erreurs d'annotation peuvent être révélées. Cette piste engendre aussi un surcoût car elle nécessite d'embaucher plusieurs annotateurs.

> 💬 **Notes de l'auteur :** Dans les deux cas, il y a un choix à faire : **veut-on privilégier la qualité** (*ajouter de la redondance et une double vérification, au prix d'un surcoût d'annotation*) **ou la rapidité de conception** (*optimiser l'annotation pour une convergence en un minimum de contraintes, au risque d'introduire des incohérences dans la modélisation*) **?**

4.6.3 Étude de l'impact de la subjectivité de l'annotation sur la divergence des résultats obtenus

Dans la section précédente, nous nous sommes intéressé aux différences de comportement intra-annotateur et aux pistes permettant de limiter les erreurs d'annotation d'un seul opérateur. Cependant, nous devons aussi nous intéresser aux **différences inter-annotateurs** : en effet, nous avons pu voir en Section 2.3.3.A que la labellisation est une tâche subjective, et que la complexité du phénomène à modéliser ainsi que la diversité de profils d'annotateurs peuvent introduire des différences d'annotation. Mais ces différences ne sont pas synonymes d'erreurs, elles peuvent aussi être les témoins de **différences d'opinion entre les experts**.

> 💬 **Notes de l'auteur :** Pour aller plus loin, il est même mal avisé de parler d'*erreurs* d'annotation car cela suppose qu'une comparaison avec une vérité terrain soit possible. Or, en situation réelle, cette vérité terrain est précisément en cours de construction à l'aide de

notre méthodologie de `Clustering Interactif` : l'expert est responsable de la vision qu'il veut appliquer aux données durant son annotation, il est donc difficile de parler d'erreurs sans porter de jugement sur sa vision. Il convient donc mieux de parler de *différences* d'annotation si deux experts ont une divergence de point de vue sur une donnée à annoter.

Dans cette dernière étude, nous allons donc estimer la robustesse du `Clustering Interactif` en nous intéressant plus particulièrement aux différences d'annotation entre deux opérateurs. Pour cela, nous allons simuler de telles différences entre deux annotateurs fictifs : le premier sera représenté par la vérité terrain, et le second sera simulé en introduisant un certain taux de désaccords d'annotation à chaque itération. Nous discutons ensuite de l'écart de similarité entre les résultats de *clustering* obtenus lorsque l'annotateur de référence atteint le seuil d'annotation partielle de son jeu de données (*accord théorique de* 90% *de* `v-measure` *sur la vérité terrain qu'il recherche*) Nous réalisons cette analyse sur des jeux de données de différentes tailles et pour plusieurs taux de désaccords insérées.

4.6.3.a Protocole expérimental

Pour résumer le protocole expérimental que nous détaillons ci-dessous, une description en pseudo-code est disponible dans l'Algorithme 4.13.

Pour information : Nous reprenons dans les grandes lignes le protocole expérimental de la précédente étude sur l'impact des erreurs d'annotation et l'intérêt de les corriger (voir Section 4.6.2. Cependant, nous utilisons ici la vérité terrain pour représenter un opérateur de référence, et nous représentons le second opérateur par les différences d'annotation introduites. De plus, comme nous voulons analyser l'impact des différences d'annotation sur des jeux de données de différentes tailles, nous allons utiliser plusieurs vérités terrains.

Nous utilisons cette fois deux vérités terrains comme références, représentant un **annotateur de référence** (*et sa propre vision métier*) :

— le jeu de données `Bank Cards (v2.0.0)` : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 1 000 questions rédigées en français et réparties en 10 classes (`perte ou vol de carte`, `carte avalée`, `commande de carte`, ...). Pour plus de détails, consulter l'Annexe A.1 ;

— le jeu de données `MLSUM FR Train Subset (v1.0.0-schild)` : ce dernier concerne les titres d'articles de journaux issus des catégories de publication les plus communes. Il est composé de 744 titres d'articles rédigés et répartis en 14 classes (*économie*, *sport*, ...). Pour plus de détails, consulter l'Annexe A.2 ;

Pour utiliser facilement plusieurs jeux de données de tailles différentes tout en maîtrisant leur contenu, nous avons donc dupliqué aléatoirement des données issues de ces jeux de référence en y insérant des fautes de frappes. La taille des jeux de données générés varie entre 1 000 à 5 000 par pas de 500. Il y a donc 9 variations de chaque jeu de références, soit 18 jeux utilisés de tailles différentes.

Données : jeux de données annotées (vérités terrains) de tailles différentes
Entrées : liste de taux de désaccords à insérer

```
pour chaque jeu de données à tester et taux de désaccords à insérer faire
    initialisation (données) : récupérer les données et la vérité terrain ;
    initialisation (contraintes) : créer une liste vide de contraintes ;
    prétraitements : supprimer le bruit dans les données avec prep.simple ;
    vectorisation : transformer les données en vecteurs avec vect.tfidf ;
    clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
    évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
    répéter
        échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
        choix des désaccords : définir les contraintes divergentes ;
        simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
        intégration corrective : ajouter les nouvelles contraintes au gestionnaire de
         contraintes, et ré-annoter les annotations en conflit ;
        clustering : regrouper les données par similarité avec clust.kmeans.cop ;
        évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
    jusqu'à annotation de toutes les contraintes possibles;
    analyse locale : afficher l'évolution de la similarité entre les clustering de la
     tentative courante et les clustering de la tentative de référence (n'ayant pas de
     différence d'annotation par rapport à la vérité terrain) ;
analyse générale : analyse des différences de résultats de clustering obtenus par taille
 de jeux de données et par taux de désaccords insérés ;
```

Résultat : discussion sur l'impact de la subjectivité de l'annotation sur la divergence des résultats de *clustering* obtenu

ALGORITHME 4.13 – *Description en pseudo-code du protocole expérimental de l'impact de la subjectivité de l'annotation sur la divergence des résultats.*

> ⚠ **Attention :** Dans le cadre de cette étude, nous faisons l'hypothèse que cette création artificielle de données n'a pas d'impact majeur sur le nombre de contraintes nécessaires pour converger vers une vérité terrain.

Sur ces jeux de données, nous exécutons une tentative complète[49] de la méthode du `Clustering Interactif` en utilisant notre paramétrage favori[50] (voir Section 4.3). À nouveau, nous allons ajouter un pourcentage de contraintes erronées à chaque itération, représentant un **autre annotateur** (*et sa propre vision métier du problème à modéliser*) :

— Le taux de désaccords insérés, variant de 0% à 25%[51] par pas de 5%, reste fixe tout au long d'une même tentative de notre méthode : nous pouvons ainsi analyser l'impact d'un taux de désaccords fixe sur les résultats au courant des itérations ;

— Les contraintes divergentes à insérer sont tirées aléatoirement parmi le lot de contraintes qui aurait été échantillonné au cours d'une tentative sans introduction de différences : ainsi, nous pouvons comparer itération par itération toutes ces simulations car elles partagent la même base de contraintes (aux valeurs de `MUST-LINK` et `CANNOT-LINK` près) ;

Puisque nous introduisons des différences d'annotations, des conflits peuvent apparaître dans le gestionnaire de contraintes. Pour rappel, un conflit est détecté dans le cas où l'ajout d'une nouvelle contrainte annotée contredit ce qui a été précédemment déduit grâce aux propriétés de transitivité des contraintes de types `MUST-LINK` et `CANNOT-LINK` (voir Figure C.1 en Annexe C.1.2). En tirant parti des conclusions de la précédente étude, nous choisissons de corriger ces désaccords dès leur détection, et supposant que le second annotateur se range à la vision de l'annotateur de référence. Pour simuler cette correction réalisée par les experts, nous recréons à chaque itération le gestionnaire de contraintes en intégrant d'abord les contraintes correctes puis les contraintes divergentes ; ainsi, les conflits ne peuvent arriver qu'à l'ajout d'une contrainte divergente, et il suffit de réévaluer la contrainte par rapport à la vérité terrain pour simuler la correction des experts.

Ainsi, il y a 6 taux de désaccords, et chacune des simulations de désaccords sera répétée 2 fois sur chaque tentative complète de la méthode pour limiter au mieux les aléas statistiques des tirages de contraintes divergentes, ce qui représente 12 simulations par tentative. Enfin, chaque tentative complète de `Clustering Interactif` est répétée 2 fois pour limiter au mieux les aléas statistiques des exécutions, soit un nombre de 24 tentatives complètes pour chacun des 18 jeux de données, ce qui représente un total de 432 tentatives à simuler au cours de cette expérience.

Nous réalisons ensuite l'analyse des différences de résultats obtenus en estimant la similarité des *clustering* entre une tentative introduisant des différences et sa tentative de référence n'en introduisant pas. Pour cela, nous procédons en trois temps :

— nous estimons d'abord, pour chaque taille de jeu de données, le nombre moyen de contraintes nécessaires aux tentatives de référence pour atteindre un score de 90% de `v-measure` par rapport à leur vérité terrain ;

49. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

50. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitements simples (`prep.simple`), vectorisation `TF-IDF` (`vect.tfidf`), *clustering* `KMeans` avec modèle `COP` (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des *clusters* différents (`sampl.closest.diff`).

51. Choix de 0% à 25% : nous utilisons ici les estimations grossières des bornes maximales d'erreurs réalisées en Section 4.6.1.

— nous estimons ensuite, pour chaque tentative introduisant des différences d'annotation, le score de `v-measure` entre son résultat de *clustering* et le résultat de *clustering* de leur tentative de référence pour le nombre de contraintes déterminé précédemment (celui nécessaire à la tentative de référence pour atteindre un score de 90% par rapport à sa vérité terrain) ;

— enfin, nous discutons des scores moyens pour les différentes tailles de jeu de données et les différents taux de désaccords introduits.

Un exemple est illustré avec la Figure 4.32.

> **Notes de l'auteur :** Il est à noter que nous aurions pu estimer ce nombre de contraintes grâce à l'Équation 4.16[52], mais comme cette estimation théorique est une moyenne qui aurait pu décaler légèrement nos résultats, nous avons préféré mesurer directement le nombre exact de contraintes nécessaires pour chaque tentative afin de ne pas avoir à analyser une double moyenne.

> **Attention :** Ces simulations étant plus lourdes que les précédentes, et considérant le manque de temps pour réaliser cette étude, nous avons dû diminuer le nombre de répétitions de nos tentatives (*une pour la génération des jeux de données, deux pour l'insertion des différences, deux pour l'exécution des tentatives complètes de la méthode*). Les résultats obtenus nous permettent tout de même de discuter des tendances générales, mais il serait intéressant de compléter a posteriori les résultats de cette étude pour en améliorer la fiabilité.

> **Rappel :** Les scripts de l'expérience, réalisés avec des *notebooks* `Python` (Van Rossum et Drake, 2009), sont disponibles dans un dossier dédié de Schild, 2022c. De plus, les jeux de données ainsi que les implémentations de notre `Clustering Interactif` sont détaillés respectivement en Annexe A et en Annexe C.

4.6.3.b Résultats obtenus

Commençons par un exemple de résultats pour un jeu de 5 000 données. La Figure 4.32 représente l'évolution moyenne de la `v-measure` du *clustering* en fonction du nombre de contraintes annotées au cours des itérations de la méthode. Sur cette figure, la courbe ayant 0% de différence d'annotation représente l'évolution moyenne des tentatives de l'opérateur de référence : celles-ci convergent pas-à-pas vers la vérité terrain (100% de `v-measure`). Les autres courbes représentent les tentatives du second opérateur et la divergence de ses *clustering* due à l'introduction de différences d'annotation. Dans un souci de lisibilité, nous avons toutefois tronqué la figure à 50 000 contraintes.

Grâce à cette figure, nous pouvons estimer à 16 250 le nombre moyen de contraintes nécessaires aux tentatives de référence pour atteindre 90% de `v-measure` par rapport à la vérité terrain (*ici : celle ayant* 5 000 *données*). Sur cette base, nous pouvons identifier la similitude moyenne des résultats de *clustering* entre chaque tentative introduisant des désaccords et leur tentative

52. Équation 4.16 : $\texttt{constraints\_needed} \propto 3.15 \cdot \texttt{dataset\_size}$

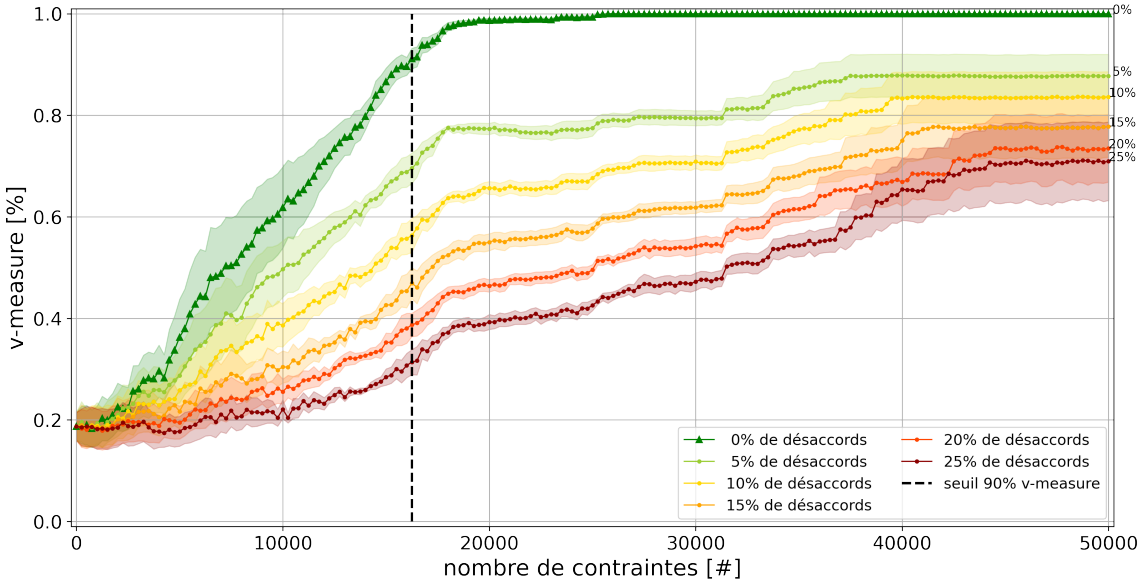

FIGURE 4.32 – *Exemple d'une évolution de similitudes moyennes (calculées en terme de* `v-measure`*) de résultats de clustering de tentatives introduisant des différences d'annotation par rapport à la vérité terrain au cours des itérations, vérité terrain ayant ici une taille de* 5 000 ***données****. Les dégradés de couleurs des courbes représentent les déclinaisons de ces évolutions en fonction des différents taux d'annotations divergentes (allant de* 0% *et* 25%*). La barre verticale indique le nombre moyen de contraintes nécessaires aux tentatives n'introduisant pas de désaccords pour obtenir un score de* 90% *de* `v-measure` *(ici :* 16 250 *contraintes).*

de référence, calculée pour un nombre fixe de 16 250 contraintes[53]. Les scores moyens de similitude sont rapportés dans la TABLE 4.13, chaque analyse pour une taille de jeu de données étant retranscrite dans une colonne dédiée.

Nous pouvons constater les points suivants :

— la similarité est bien entendu de 100% pour un taux d'incohérence de 0% (*une tentative de référence est comparée à elle-même*) ;

— à taille de jeu de données fixe, le taux de similarité diminue lorsque le taux de désaccords introduits augmente ; l'amplitude de diminution varie environ entre 35 points (*pour une taille de* 1 000 *données*) et 73 points (*pour une taille de* 4 500 *données*).

— à un taux fixe de désaccords introduits, le taux de similitude diminue lorsque la taille du jeu de données augmente ; l'amplitude de diminution varie environ entre 0 points (*pour l'introduction de* 0% *de désaccords*) et 33 points (*pour l'introduction de* 25% *de désaccords*).

4.6.3.c Discussion

L'objectif de cette étude est l'analyse de l'impact des différences de comportements inter-annotateurs sur les résultats de notre méthode, plus particulièrement l'impact de la subjectivité

53. Observation de 16 250 contraintes nécessaires : lors d'une analyse théorique utilisant l'ÉQUATION 4.16, nous aurions estimé une moyenne de $3.15 \cdot 5\,000 \propto 15\,750$ contraintes, soit un écart de 500 contraintes qui aurait introduit une légère variation dans nos résultats.

		Taille des jeux de données								
		1 000	1 500	2 000	2 500	3 000	3 500	4 000	4 500	5 000
	Contraintes annotées	4 250	6 000	7 250	8 750	10 250	11 250	12 250	13 500	16 250
Taux de désaccords simulés	0%	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)
	5%	86.16% (±3.52)	78.56% (±1.50)	80.09% (±1.27)	74.81% (±1.62)	75.06% (±1.02)	74.72% (±1.84)	71.86% (±0.91)	70.07% (±1.29)	71.03% (±3.09)
	10%	79.84% (±6.22)	66.02% (±2.00)	66.95% (±1.46)	61.13% (±1.55)	61.64% (±1.05)	60.93% (±1.92)	57.70% (±1.52)	54.00% (±1.84)	57.01% (±3.65)
	15%	73.94% (±8.48)	58.64% (±2.21)	57.13% (±1.97)	51.01% (±1.43)	48.25% (±2.14)	52.33% (±2.04)	48.52% (±1.61)	42.13% (±1.43)	47.41% (±3.27)
	20%	68.96% (±9.61)	51.92% (±2.30)	48.85% (±2.31)	42.01% (±0.96)	41.57% (±2.00)	43.37% (±1.71)	40.56% (±1.94)	34.87% (±1.15)	39.08% (±2.51)
	25%	65.05% (±10.78)	44.12% (±2.30)	40.19% (±2.01)	35.88% (±0.98)	35.31% (±1.25)	37.31% (±2.04)	32.47% (±1.46)	26.73% (±1.56)	31.90% (±2.56)

Table 4.13 – *Estimation de la **similitude moyenne** (calculée en terme de `v-measure`) des résultats de clustering des tentatives introduisant des désaccords d'annotation **par rapport aux résultats de clustering de leurs tentatives de référence**.*
Cette similitude est rapportée en fonction de la taille du jeu de données utilisé et du taux de désaccords introduits lors des tentatives. Pour chaque taille de jeu de données, les calculs sont réalisés avec un nombre de contraintes fixe, choisi comme étant le nombre de contraintes nécessaires à une tentative de référence pour atteindre une `v-measure` moyenne de 90% avec sa vérité terrain (ce nombre est rapporté en deuxième ligne).

des opérateurs manifestée par des désaccords d'annotation sur le résultat du *clustering* obtenu. Pour cela, nous avons analysé l'évolution de la similarité en simulant des désaccords d'annotation pour plusieurs tailles de jeux de données.

Sur la base des résultats rapportés dans la Table 4.13, nous pouvons constater que les résultats de *clustering* divergent rapidement si des désaccords d'annotation sont présents. En effet, l'introduction de 5% de différences fait diverger les résultats d'au moins 14 points de `v-measure`, et cette divergence est plus forte lorsque la taille du jeu de données augmente. Si nous prenons le cas extrême (25% de désaccord, taille de 4 500 données), la similitude des *clustering* obtenus n'est en moyenne plus que de 26.73% au bout de 13 500 contraintes. Ces observations dépendent bien entendu fortement des jeux de données utilisés pour réaliser ces simulations. Nous pouvons néanmoins conclure qu'en l'état, le `Clustering Interactif` est sensible à la subjectivité des annotateurs.

Une telle conclusion est en accord avec le caractère incrémental de la méthode : une différence d'annotation peut rapidement se propager et faire diverger les résultats de la méthode. Dans notre cas, nous avons optimisé l'implémentation de notre `Clustering Interactif` pour obtenir un corpus d'apprentissage pertinent avec un minimum d'annotation de contraintes. Or cette notion de pertinence témoigne d'une vision subjective de l'expert annotant les données. De ce fait, si deux annotateurs ne sont pas d'accord sur la vision à appliquer aux données lors de l'annotation, il est normal de voir leurs résultats de *clustering* diverger. Nous pourrions résumer cette situation par le fait que **ces deux experts ne recherchent pas la même vérité terrain, d'où la divergence significative de leurs résultats**. Ainsi, la sensibilité de notre méthode n'est pas la racine du problème, elle met plutôt en lumière le besoin de confronter les opinions des experts afin qu'ils puissent s'accorder.

En nous basant sur notre revue de littérature (notamment la Section 2.3.3.a), nous conseillons aux organisateurs du projet d'annotation d'**employer au moins 3 opérateurs et de les confronter aux mêmes contraintes à chaque itération** de la méthode. Ainsi, si ces derniers ont des différences d'opinion sur la modélisation du problème, celles-ci se manifesteront en observant le score d'accord inter-annotateurs. Il conviendra alors d'organiser une session de débat pour ré-annoter les désaccords et décider des adaptations à prévoir dans le guide d'annotation du projet.

> **Idées :** Pour aider à trouver un accord lors de la revue, une prise de recul peut être nécessaire. Pour ce faire, les experts pourraient par exemple **observer le graphe de contraintes** déjà annotées. En effet, le gestionnaire de contraintes gère les propriétés de transitivité entre celles-ci, créant ainsi des composants connexes de données liées par les contraintes `MUST-LINK` et distinguées par les contraintes `CANNOT-LINK` (voir Annexe C.1.2). Or le règlement d'un désaccord va nécessairement impacter ces composants connexes : en les rapprochant si le désaccord est tranché en faveur d'un `MUST-LINK`, en les éloignant sinon. Par conséquent, il serait intéressant d'aiguiller le débat pour anticiper les conséquences du consensus à trouver : ces composants sont-ils à rapprocher ? à distinguer ? ou sont-ils éventuellement à remettre en question ?
>
> Si malgré cette prise de recul, aucun accord n'a été trouvé, il est possible de ne pas caractériser cette contrainte et de **laisser l'algorithme de *clustering* trancher le débat**. En effet, la philosophie de la méthode d'annotation basée sur le `Clustering Interactif` repose sur la coopération entre l'Homme et la Machine : dans cette situation, il peut être intéressant de simplement observer quelle solution est proposée par la machine pour segmenter les données tout en respectant les autres contraintes déjà annotées.

Bien entendu, une telle approche engendre un coût supplémentaire aux estimations réalisées dans la Section 4.3 :

— d'une part, nous triplons la charge salariale à investir dans le but d'ajouter de la redondance dans les contraintes annotées ;

— d'autre part, nous introduisons un délai supplémentaire en organisant des revues de désaccords de contraintes, du moins tant que des désaccords de visions subsistent.

Néanmoins, ces surcoûts participent à l'amélioration de la stabilité de la base d'apprentissage en cours de construction. Par ailleurs, nous pouvons noter que :

— un projet d'annotation classique emploie aussi plusieurs opérateurs : ce surcoût correspond donc plutôt à un investissement supplémentaire au profit d'une fiabilisation de la qualité de ses résultats ;

— un projet d'annotation classique est déjà confronté au besoin d'organiser des sessions de revue de sa modélisation (voir l'étape *Revise* du cycle `MATTER`) : l'avantage de notre méthode réside néanmoins dans la possibilité de discuter directement des divergences d'opinions en analysant des cas d'usage métier (*évaluation d'une contrainte à l'aide de compétences métiers*) plutôt que de discuter de divergences d'opinions sur l'interprétation d'une abstraction du problème (*remise en cause d'une modélisation à l'aide de compétences analytiques*).

Notes de l'auteur : Nous concluons cette discussion par la même remarque faite à la fin de la précédente étude de robustesse : **il y a un choix à faire entre privilégier la qualité** (*ajouter de la redondance pour clarifier les désaccords d'opinion, au prix d'un surcoût d'annotation*) **ou la rapidité de conception** (*optimiser l'annotation pour une convergence en un minimum de contraintes, au risque d'introduire des incohérences dans la modélisation*).

4.6.4 Bilan concernant la robustesse du *Clustering Interactif*

Points à retenir : Au cours de cette étude de robustesse, nous avons pu voir que :

- ☑ Le `Clustering Interactif`, comme toute autre approche incrémentale, est **sensible aux erreurs et aux désaccords d'annotation** : en effet, la méthode est optimisée pour converger vers une base d'apprentissage en un minimum de contraintes, donc toute différence d'annotation peut rapidement se propager ;
- ☑ Pour **combattre les erreurs d'annotation** (*divergences intra-annotateur*), il peut être intéressant d'**ajouter de la redondance dans les annotations** : cela permet au gestionnaire de contraintes de vérifier les propriétés de transitivité des contraintes `MUST-LINK` et `CANNOT-LINK`, contribuant ainsi à la détection d'éventuelles incohérences ;
- ☑ Pour **combattre les différences d'opinions** (*désaccords inter-annotateurs*), il est nécessaire de **confronter plusieurs opérateurs aux mêmes annotations contraintes** : cela permet de débattre des désaccords d'interprétation de cas d'usages métier dissimulés derrière les différences de labellisation, et de compléter le guide d'annotation si besoin ;
- ☑ Il y a un choix à faire entre **privilégier la qualité** (*vérification des annotations et harmonisation des opinions, au prix d'un surcoût d'annotation*) **ou la rapidité de conception** (*optimisation de la convergence en un minimum d'annotations, au risque d'introduire des incohérences dans la modélisation*).

4.7 Autres hypothèses non vérifiées

Lors des études précédentes, nous avons vérifié un certain nombre d'hypothèses et avons exploré plusieurs détails pratiques pour mettre en oeuvre une méthodologie d'annotation basée sur le `Clustering Interactif`. Toutefois, certains points n'ont pas pu être étudiés en profondeur lors de ce doctorat, par manque de temps ou de moyens. Nous exposons ici un ensemble de pistes intéressantes pouvant nourrir de futurs travaux afin d'améliorer notre méthode.

4.7.1 Étude du nombre de *clusters* optimal

Un problème ouvert de la recherche lors de l'utilisation d'algorithmes de *clustering* concerne le choix du nombre de *clusters* à trouver. En effet, à part une connaissance *a priori* du nombre de thématiques présentes dans le jeu de données, il est difficile d'estimer le nombre optimal de *clusters*, d'autant plus que celui-ci peut changer en fonction de la granularité de modélisation requise pour répondre au cas d'usage.

Nous avons déjà exploré partiellement deux pistes :

— l'**exploration du graphe de contraintes** : en effet, il est possible d'estimer le nombre maximal de *clusters* grâce aux composants connexes de contraintes `MUST-LINK`, et d'estimer le nombre minimal de *clusters* grâce à la coloration du graphe de contraintes `CANNOT-LINK` ;

— les **études de pertinence** avec l'analyse des patterns linguistiques et le résumé thématique des *clusters* (cf. Section 4.4) : ces deux approches permettent de rapidement évaluer si les thématiques obtenues sont trop générales (*i.e. s'il n'y a pas assez de clusters*) ou si elles semblent trop spécifiques (*i.e. s'il y en a trop*).

Toutefois, pour aller plus loin, deux pistes potentielles pourraient être explorées :

— l'exploration brute du nombre de *clusters* par la **méthode du coude** : bien que ces approches sont plus coûteuses en temps de calcul, elles permettent d'estimer le nombre de *clusters* pour lequel la stabilité du *clustering* est la plus élevée ;

— l'utilisation d'algorithmes n'ayant pas de nombre de *clusters* en paramètres, comme des versions contraintes de `DBScan` (par exemple dans sa version `C-DBScan`, Ruiz et al., 2010) ou de la **propagation par affinité** (Givoni et Frey, 2009) : ces alternatives semblent prometteuses car elles retirent la complexité due à ce paramétrage abstrait.

> **Pour information :** L'étude de `C-DBScan` a été en partie réalisée dans le cadre d'un projet étudiants avec l'École d'Ingénieurs Télécom Physique Strasbourg (au cours de l'année 2022). Les résultats montraient que le temps de calcul était similaire à celui du `KMeans` (dans sa version `COP`). La difficulté d'utilisation résidait plutôt dans la définition du rayon de voisinage `eps` à parcourir pour établir des liens entre données. Celui-ci peut être estimé en analysant la densité vectorielle du jeu de données. Le code informatique est disponible dans Schild, 2022a[54].

54. Implémentation de `C-DBScan` : *Pull Request* en attente pour une version `0.6.0` après ajout de documentation et de tests unitaires.

4.7.2 Étude d'autres méthodes de vectorisation

Au début de ce doctorat, nous avons conclu que les algorithmes de vectorisation n'avaient pas d'impact réel sur l'efficience de notre méthodologie d'annotation. Toutefois, les modèles de langues se sont largement développés, et il est fort probable que l'utilisation d'un **modèle pré-entraîné** permette désormais d'avoir un gain de performance.

Nous pourrions par exemple tester les **architectures à base de *Transformers*** (Uszkoreit, 2017) comme `BERT` (Devlin et al., 2019) et essayer différents modèles pré-entraînés sur des données françaises pour compléter nos études réalisées dans Schild, 2022c

4.7.3 Étude d'autres méthodes d'échantillonnage

Comme nous avons pu le voir dans Section 4.6, il peut-être intéressant d'introduire un mécanisme de création de redondance dans le graphe de contraintes annotées pour identifier les erreurs d'annotation. Un tel mécanisme n'a pas encore été implémenté mais pourrait facilement être intégré aux implémentations `Python` déjà existantes (Schild, 2022a).

Pour ce faire, le parcours de graphe et la création de cycle permettraient de vérifier la présence de conflits et ainsi de **provoquer des phases de revues de contraintes** si cela est nécessaire. Une telle page de revue pourrait aussi contenir des analyses complémentaires, comme l'estimation du taux de contraintes n'ayant pas de redondance et représentant ainsi des erreurs cachées potentielles.

4.7.4 Étude de techniques de transfert d'apprentissage

Dans la Section 2.3, nous avions déjà évoqué le fait que la modélisation d'un phénomène peut être assisté par des techniques telles que la pré-annotation (Dandapat et al., 2009) ou le transfert d'apprentissage (Zhuang et al., 2021). Nous pourrions nous inspirer davantage de ces approches pour démarrer plus efficacement les premières itérations d'un `Clustering Interactif`.

Voici quelques idées inspirées de ces méthodes :

— **pré-annoter** certaines contraintes simples à l'aide de règles (*basées par exemple sur la présence de mots de vocabulaire en commun*) ou grâce à l'utilisation d'un modèle déjà disponible ;

— **introduire des données synthétiques ou empruntées** à d'autres bases d'apprentissage pour initialiser le *clustering*, et permettre ainsi ajouter d'emblée des connaissances générales dans la modélisation.

4.7.5 Étude ergonomique de l'interface d'annotation

L'application web développée au cours de ce doctorat (Schild, 2022b) permet d'essayer rapidement notre méthodologie d'annotation. Cependant, cette dernière n'a pu faire l'objet d'études poussées pour estimer la meilleure disposition des composants ou l'intérêt de certaines fonctionnalités d'annotation.

Parmi les pistes potentielles à explorer, nous avons évoqué la possibilité d'**annoter plusieurs contraintes** dans une même interface (*par exemple : annoter visuellement un mini-graphe de 4 données plutôt que d'annoter simplement une paire de données*) et le besoin de **réaliser des analyses rapides** sur les *clusters* ou sur le graphe de contraintes (voir Section 4.4 et Section 4.5). Pour aller plus loin, Bae et al., 2021 proposent d'autres listes d'interactions qui sont

possibles d'avoir avec un algorithme de *clustering*, notamment sur la manipulation de son résultat (*fusion, suppression, verrouillage, ...*) et de ses hyperparamètres (*nombre de clusters, adaptation du vocabulaire autorisé, ...*)

Toutes ces idées pourraient être l'objet de développements et d'études dédiées avec des groupes d'annotateurs différents pour voir l'impact sur les performances et les biais de conception de modèles.

Chapitre 5

Bilan et Guide d'utilisation du *Clustering Interactif*

Dans nos études, nous nous sommes intéressé aux assistants conversationnels orientés par tâches et aux méthodes de conception des bases d'apprentissage nécessaires à leur entraînement. Dans ce chapitre, nous dressons une synthèse des découvertes et des conseils d'utilisation de notre méthodologie d'annotation basée sur un `Clustering Interactif` ayant pour but d'assister les experts métiers dans la phase de modélisation des textes en intentions de dialogue.

Rappel : Nous partons du constat selon lequel la conception d'une base d'apprentissage de textes annotés en intentions est connue pour être complexe, subjective et sensible aux erreurs (voir Section 2.3). Pour limiter ces problèmes, un projet de labellisation s'organise généralement autour du cycle `MATTER` (Pustejovsky et Stubbs, 2012) durant lequel une modélisation abstraite des intentions est définie pour annoter les données ; cette modélisation est ensuite affinée ou remise en cause plusieurs fois au cours du cycle pour mieux s'adapter au projet (voir Section 2.2).

Pour modéliser et annoter les données, les experts métiers ont besoin de leurs connaissances métiers, mais aussi de compétences analytiques et techniques afin d'assurer de la qualité de la base d'apprentissage en cours de construction. Par conséquent, un projet d'annotation devient rapidement onéreux, notamment à cause :

- des formations analytiques nécessaires aux annotateurs pour intervenir dans le projet ;
- des nombreux ateliers de modélisation en mode essai-erreur nécessaires pour trouver une base d'apprentissage stable et pertinente ; et
- de la complexité engendrée par la manipulation de concepts abstraits (*intentions, entités, ...*) dans le but de représenter les connaissances des experts métiers.

Au cours de ce doctorat, et sur la base des constatations décrites ci-dessus, nous avons décidé de reconsidérer cette approche de la tâche de modélisation. Nous avons alors proposé une nouvelle méthodologie d'annotation dans le but d'impliquer les experts métiers pour leurs vraies compétences tout en leur demandant un minimum de bagages analytiques et techniques.

5.1 Présentation rapide du *Clustering Interactif*

Sur la base d'intuitions issues de la littérature (Section 3.1), nous avons décidé de centrer notre méthodologie d'annotation sur l'**annotation des similarités et des différences entre les données**. En effet, une telle stratégie semble moins complexe à appliquer, car elle ne dépend pas d'une modélisation abstraite des connaissances de l'expert, mais elle se base directement sur ses connaissances pour décrire si deux données ont ou n'ont pas un cas d'usage équivalent.

Nous mettons en oeuvre cette stratégie d'annotation par similarité au sein d'une méthodologie basée sur un `Clustering Interactif` (voir Section 3.2). Cette méthode repose sur les avantages des interactions Homme/Machine, en déléguant la conception de la base d'apprentissage à la machine (*à l'aide d'un algorithme de regroupement automatique de texte (clustering)*) et en faisant intervenir l'expert pour affiner itérativement la base d'apprentissage proposée (*en annotant des contraintes entre les données pour corriger le regroupement des textes*). **Trois étapes principales** se répètent ainsi au cours de ce **processus itératif** :

— une **sélection de données à annoter** : la machine propose un ensemble de données dont la similarité serait à confirmer (ou à infirmer) afin de corriger efficacement le regroupement automatique des données opéré à l'itération suivante ;

— une **annotation des contraintes** : l'expert caractérise chaque paire de données en répondant à la question « *est-ce que les deux données ont un cas d'usage similaire ?* », en ajoutant une contrainte `MUST-LINK` si oui (*similaires*) et en ajoutant une contrainte `CANNOT-LINK` sinon (*non similaires*) ;

— une **segmentation automatique des données** : la machine regroupe les données en fonction de leurs similarités intrinsèques et des contraintes annotées par l'expert.

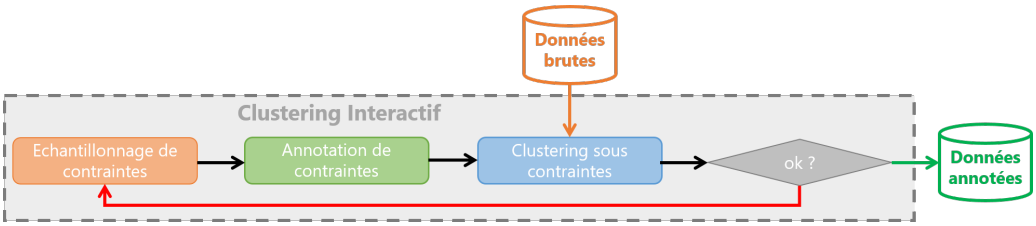

Figure 5.1 – *Schéma illustrant l'architecture du `Clustering Interactif`.*

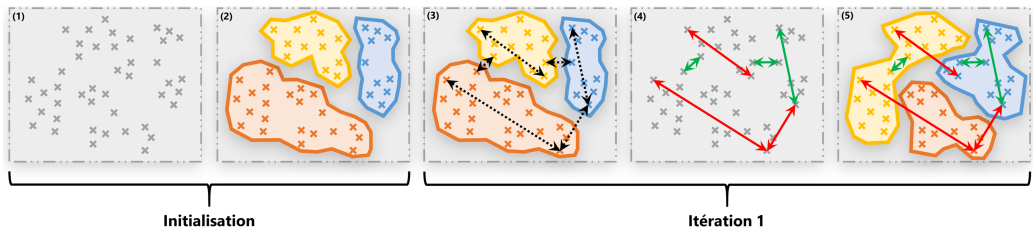

Figure 5.2 – *Exemple d'une itération de `Clustering Interactif`.*

> **Rappel :** Consulter la Section 3.2 pour plus de détails.

5.2 Avantages et limites de la méthode

Au sujet des avantages de la méthode :

- Par conception, la méthode tire parti des avantages de l'apprentissage actif et des méthodes de regroupement semi-supervisé : **l'expert n'est plus responsable de l'ensemble du travail de labellisation** ; il intervient simplement là où il offre une valeur ajoutée (*annotation des contraintes pour améliorer la pertinence de la base d'apprentissage en cours de construction*), et il délègue le reste à la machine (*regroupement automatique, sélection des contraintes intéressantes à annoter, identification des incohérences*).

- **Il n'est plus nécessaire de définir une modélisation abstraite de la connaissance d'un expert métier pour labelliser un jeu de données** : cette modélisation est construite au cours des itérations de la méthode, à l'aide des regroupements automatiques réalisés par la machine.

- Par extension, il n'est plus nécessaire de manipuler cette modélisation abstraite pouvant contenir des dizaines d'intentions de dialogue : **l'expert se contente de décrire la similarité entre deux données au cours d'une annotation binaire (`MUST-LINK` ou `CANNOT-LINK`)** pour corriger le résultat proposé par la machine.

- **Cette annotation binaire se base directement sur la ressemblance entre cas d'usage métiers** : les experts intervenant dans le projet peuvent traiter les données comme ils le feraient professionnellement au quotidien, sans avoir à manipuler ou à interpréter des concepts abstraits et potentiellement non adaptés à la situation (*intentions, ...*).

- L'implémentation de notre méthodologie d'annotation a pu être optimisée afin de converger vers une base d'apprentissage stable en un minimum de contraintes : **les coûts à engager pour cette approche semblent raisonnables et compétitifs avec l'approche d'annotation traditionnelle** (voir Section 4.3.4).

Néanmoins, il faut aussi considérer quelques limites et pistes non explorées :

- Le `Clustering Interactif` possède les défauts des approches incrémentales : ainsi, **une erreur d'annotation peut rapidement se propager et pénaliser le processus**. Par conséquent, il est important de vérifier ses annotations (*par exemple lors de session de revue avec d'autres opérateurs*), et il peut être intéressant d'introduire des contraintes redondantes afin de mieux détecter les incohérences d'annotation (voir Section 4.6.2).

- **Les différentes d'opinion entre plusieurs annotateurs concernant la caractérisation de certaines contraintes peuvent entraîner des divergences de résultat si elles ne sont pas identifiées et traitées**. De ce fait, il est primordial de confronter les points de vue des experts en leur faisant annoter les mêmes contraintes et en organisant des ateliers de revue pour débattre des différences d'annotation (voir Section 4.6.3).

- **La méthode peut souffrir d'un manque de visibilité et d'une sensation de perte de contrôle sur la modélisation en cours** : en effet, le processus peut être perçu comme répétitif (*environ* 145 *itérations à réaliser*), et le cas d'arrêt de la méthode n'est pas clairement identifiable (*il est toujours possible de réaliser un itération supplémentaire pour améliorer légèrement le résultat*). Pour ces raisons, il est important de fournir des outils d'analyse pour tenir l'opérateur informé de son avancement et de la rentabilité de chacune de ses actions (voir Sections 4.4 et 4.5).

5.3 Démarche d'annotation et d'analyse de la méthode

☑ **Annotation collaborative.** L'annotation est un acte relevant du domaine de l'interprétation et de la subjectivité. Par conséquent, **deux annotateurs peuvent avoir des divergences d'opinions lors de la caractérisation de la similarité entre certaines données**, ce qui peut introduire des incohérences dans le fonctionnement de notre méthode. Afin de limiter ces incohérences, il est nécessaire de bien définir l'objectif de l'annotation (*quel est la finalité du modèle à entraîner ?*), mais aussi d'harmoniser les points de vue des différents annotateurs.

Au cours de l'`étude de robustesse` (voir Section 4.6), **nous conseillons d'employer au moins 3 experts et de régulièrement (voire toujours) leur soumettre les mêmes contraintes à annoter**. Un tel choix semble contre-intuitif à première vue (*nous triplons le coût d'embauche alors que nous pourrions tripler le nombre de contraintes annotées*), mais ce parti pris a l'avantage de contraindre les annotateurs à discuter de leurs divergences d'opinion (*s'ils n'annotent pas de façon analogue, c'est qu'ils ne sont pas d'accord sur la manière de traiter les données*). Ainsi, même si ce choix engendre des coûts supplémentaires, il s'avère être un investissement à long terme pour garantir la qualité et la cohérence de la base d'apprentissage en cours de construction.

Les experts pourront annoter des contraintes différentes après plusieurs itérations de la méthode lorsque leur score d'accord inter-annotateurs sera considéré comme (très) `fort`.

☑ **Rentabilité d'une itération.** Comme notre méthodologie est itérative, il faut être capable de définir un cas d'arrêt (*annoter toutes les contraintes possibles étant trop ambitieux*).

Au cours de l'`étude de rentabilité` (voir Section 4.5), nous avons mis en évidence l'intérêt d'**analyser l'évolution de la différence de résultats de *clustering* entre deux itérations consécutives afin de quantifier l'intérêt d'une itération supplémentaire**. En effet, si le nouveau *clustering* reste fortement similaire au *clustering* précédent, malgré l'ajout de contraintes supplémentaires, nous pouvons en déduire que la méthode d'annotation stagne. Il peut alors être intéressant d'envisager de stopper les itérations de la méthode, d'évaluer la pertinence métier du *clustering* actuel, et de corriger manuellement les reliquats.

Pour estimer cette différence entre deux *clustering* consécutifs, nous utilisons le score de `v-measure` (voir Annexe D).

☑ **Pertinence d'un résultat.** L'analyse d'un résultat de *clustering* n'est pas une tâche aisée, et elle peut vite devenir fastidieuse si elle doit être faite à chaque itération de notre méthode.

Au cours de l'`étude de pertinence` (voir Section 4.4), nous avons proposé deux solutions prometteuses pour assister l'expert dans cette évaluation :

— Une **analyse des patterns linguistiques caractéristiques de chaque *cluster* en utilisant la `FMC`** (voir Annexe C.3) : un *cluster* peut ainsi être identifié comme pertinent s'il possède un vocabulaire caractéristique cohérent du point de vue de l'expert. Il est aussi possible de se servir de cette analyse pour identifier dans chaque texte les mots caractéristiques d'un ou plusieurs *clusters*.

— Un **résumé thématique des *clusters* par un large modèle de langage (`LLM`)** : cette approche permet d'estimer efficacement la cohérence d'un *cluster* à l'aide d'une courte description en langage naturel. Il faut toutefois être vigilant aux hallucinations du `LLM`.

Note : **Une correction manuelle est parfois nécessaire** pour valider un *cluster*.

5.4 Implémentation et paramétrages de la méthode

☑ **Développements logiciels.** Pendant ce doctorat, nous avons implémenté plusieurs algorithmes de *clustering* et d'échantillonnage (*disponibles dans* Schild, *2022a*) et nous avons intégré notre méthodologie d'annotation dans une application web (*accessible dans* Schild, *2022b*). Ces développements logiciels sont décrits en Annexe C.

☑ **Choix des paramétrages.** Au cours de nos études (voir Chapitre 4), nous avons mis en avant un **paramétrage favori** de notre méthode. Ce paramétrage, basé sur un compromis entre un maximum d'efficience et un minimum de coûts, est composé des algorithmes suivants :

- `prep.simple` : les **prétraitements simples** supprimant les minuscules, les ponctuations, les accents et les espaces blancs ;
- `vect.tfidf` : la **vectorisation** utilisant une représentation statistique du vocabulaire à l'aide d'un `TF-IDF` (Ramos, 2003) ;
- `clust.kmeans.cop` : le ***clustering* sous contraintes** utilisant l'algorithme `COP-KMeans` (Wagstaff et al., 2001) ;
- `samp.closest.diff` : l'**échantillonnage de contraintes** concernant des données issues de ***clusters* différents** et étant **les plus proches** les unes des autres.

Une approximation du temps de calcul d'une itération de `Clustering Interactif` utilisant ce paramétrage favori est estimée avec l'Équation 5.1.

$$\texttt{computation\_time}\ [s] \ \propto\ 0.17 \cdot \texttt{dataset\_size} \tag{5.1}$$

☑ **Choix d'architecture** Au lieu de réaliser les étapes du `Clustering Interactif` de manière séquentielle, il peut être intéressant d'**implémenter une architecture parallèle** de notre méthode (voir Figure 5.3). Une telle approche consiste à faire coïncider les temps de calcul et d'annotation dans le but de diminuer les temps d'attente : une approximation du nombre de contraintes déterminant l'équivalence de ces durées est estimée avec l'Équation 5.2.

$$\texttt{annotation\_batch\_size}\ [\#] \ \propto\ 0.0218 \cdot \texttt{dataset\_size} \tag{5.2}$$

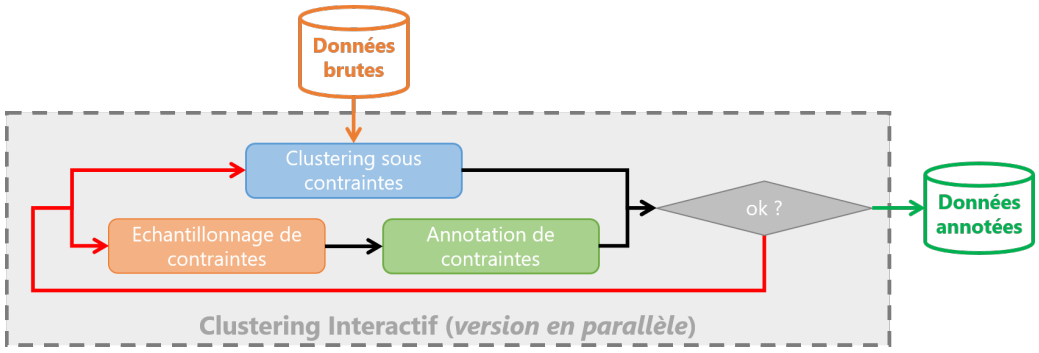

Figure 5.3 – *Schéma illustrant l'architecture du* `Clustering Interactif` *en mode parallèle.*

5.5 Estimation des coûts de la méthode

Dans nos études, et principalement au cours de la Section 4.3, nous avons pu estimer un ensemble de coûts théoriques (temporels et humains) estimés grâce aux jeux de données `Bank Cards` (Schild, 2022d) et `MLSUM` (Schild et Adler, 2023).

☑ **Vitesse d'annotation** À l'aide d'une expérience en situation réelle, nous avons estimé qu'**il faut environ 7.8 secondes pour caractériser une contrainte**. Nous avons montré que ce temps est inférieur à celui d'une annotation d'étiquettes faisant intervenir une modélisation ; l'annotation de contraintes est donc *a priori* moins complexe et plus rapide.

☑ **Nombre de contraintes nécessaires** À l'aide d'une simulation, nous avons estimé qu'**il faut environ 3.15 contraintes par donnée pour obtenir une base d'apprentissage stable** (*bien entendu, cette estimation doit varier grandement en fonction la complexité des données et la finalité du modèle à entraîner*). Nous rappelons qu'un expert métier devra confirmer la pertinence du *clustering* avant d'arrêter la méthode. Il est toutefois intéressant de noter que cette estimation du nombre de contraintes nécessaires est linéaire alors que le nombre de contraintes possibles augmente de manière quadratique avec le nombre de données.

☑ **Temps total théorique** En utilisant une architecture parallèle, nous estimons qu'**il faut environ 24.6 secondes par donnée pour obtenir une base d'apprentissage stable**. Nous pouvons aussi noter qu'une telle approche propose un nombre constant de 144.5 itérations pour annoter le nombre de contraintes requis afin d'obtenir une base d'apprentissage stable.

Une telle estimation semble compétitive avec une annotation d'étiquettes faisant intervenir une modélisation : en effet, le temps d'annotation est légèrement plus long (*à cause du nombre de contraintes à traiter*), mais nous économisons le temps nécessaire à la formation des experts ainsi qu'aux nombreuses révisions de modélisation en intentions intentions.

☑ **Coûts d'analyse et de robustesse** Il est à noter que **de nombreux coûts n'ont pas pu être estimés** (*ces derniers étaient impossibles à simuler ou à reproduire*) :

- Le **temps nécessaire à l'analyse de la pertinence et de la rentabilité** (*les gains de temps liés à nos approches pour assister ces étapes n'ont pas pu être estimés*).
- Nous avons conseillé d'**employer plusieurs experts annotant les mêmes données** : ce choix augmente le coût d'embauche, mais cet investissement garantit la cohérence des annotations et évite la conception d'une modélisation abstraite des données.
- Le **temps nécessaire aux revues d'annotations et aux débats d'opinion entre annotateurs** : de tels coûts existent aussi pour les approches traditionnelles, cependant les échanges sont ici davantage concrets car ils se basent sur les similarités de cas d'usage.

Par conséquent, **nous conseillons d'utiliser les estimations ci-dessous avec une marge d'erreur proportionnelle à la quantité et à la complexité des données à annoter**.

$$\begin{cases} \texttt{annotation\_time}\ [s] & \propto 7.8 \cdot \texttt{batch\_size} \\ \texttt{constraints\_needed}\ [\#] & \propto 3.15 \cdot \texttt{dataset\_size} \\ \texttt{total\_time}\ [s] & \propto 24.6 \cdot \texttt{dataset\_size} \\ \texttt{iterations\_needed}\ [\#] & \propto 144.5 \end{cases} \tag{5.3}$$

5.6 Conseils pour rédiger le guide d'annotation

Nous terminons ce chapitre de bilan en listant quelques conseils pour bien cadrer un projet de conception de base d'apprentissage utilisant notre méthode d'annotation.

> **Pour information :** Nous nous référons aux 7 maximes proposées par Leech, 1993 (et détaillées par Fort, 2022) que nous allons adapter pour le cas d'annotation de contraintes.

1. **La donnée initiale doit toujours pouvoir être accessible**[55] : en effet, il faut faire attention aux étapes de prétraitements et de nettoyage qui peuvent effacer des informations potentiellement importantes pour l'analyse.
2. **L'annotation d'une contrainte doit se baser sur des similarités ou des différences observables**[56] : en effet, si aucun indice ne permet de faire un choix objectif, il est préférable de s'abstenir (*voire de supprimer la donnée si elle est trop ambiguë*).
3. **Il faut documenter l'objectif de l'annotation de contraintes et expliciter clairement sur quels critères elle se base**[57] : d'une part, il est important de décrire sur quoi caractériser la similarité entre les données (*sur l'action ? sur l'objet de l'action ? sur le sentiment associé ? ...*) ; d'autre part, cette documentation doit être complétée au fur et à mesure des revues d'annotation et des résolutions d'incohérences dans la base de contraintes (*dans telle situation, la similarité sera caractérisée de telle manière*).
4. **Il est important de décrire les opérateurs réalisant les annotations**[58] : leur nombre, leur expertise du sujet traité, leur formation à la tâche d'annotation, les outils d'assistance à leur disposition, les biais potentiels au cours de l'annotation, *etc.*
5. **L'annotation est toujours une action d'interprétation**[59] : elle est donc forcément subjective et entraînera des différences d'annotations qui devront être discutées lors de revues entre opérateurs.
6. **Il est nécessaire d'être rigoureux dans l'annotation de contraintes pour ne pas introduire des incohérences ou produire des regroupements imprévus**[60] : dans le doute, il est préférable de s'abstenir de caractériser une contrainte et de laisser la machine décider du regroupement le plus adéquat au regard du reste du jeu de données.
7. **À cause de la subjectivité de la tâche, plusieurs visions peuvent être envisagées pour annoter la similarité**[61] : l'important est de pouvoir en discuter (*au moins à trois personnes*), de choisir le point de vue qui semble le plus adapté au regard de la finalité du modèle à entraîner, et de s'y tenir pour limiter les incohérences.

55. Maxime 1 : « *It should always be possible to come back to initial data.* »
56. Maxime 2 : « *Annotations should be extractable from the text.* »
57. Maxime 3 : « *The annotation procedure should be documented.* »
58. Maxime 4 : « *Mention should be made of the annotator(s) and the way annotation was made.* »
59. Maxime 5 : « *Annotation is an act of interpretation (cannot be infallible).* »
60. Maxime 6 : « *Annotation schemas should be as independent as possible on formalisms.* »
61. Maxime 7 : « *No annotation schema should consider itself a standard (it possibly becomes one).* »

Chapitre 6

Conclusion

Proposition d'une nouvelle méthode d'annotation autour d'un *Clustering Interactif*

Dans cette thèse, nous nous sommes intéressé à la tâche d'annotation, nécessaire à l'entraînement d'assistants conversationnels, et impliquant habituellement des experts du domaine à modéliser. Lors de notre revue de littérature, nous avons pu constater que l'annotation avait la réputation d'être une tâche complexe et subjective. Cette complexité provient notamment du besoin d'avoir des données représentatives du phénomène à reproduire, exerçant par conséquent une forte pression quant à la qualité de la base d'apprentissage à construire. Un projet d'annotation s'organise alors traditionnellement autour du cycle `MATTER`, une méthodologie durant laquelle la modélisation des textes en intentions est révisée plusieurs fois pour mieux s'adapter aux données du projet. Cependant, une telle organisation se révèle généralement coûteuse, entre autres à cause des nombreux ateliers de modélisation en mode essai-erreur, de la difficulté à manipuler des concepts abstraits (*intentions*, *entités*, ...), et du besoin de former les experts aux tâches d'analyse. Sur la base de ces constats, nous avons cherché une méthodologie d'annotation alternative en nous posant la question suivante : « **Comment assister la phase de modélisation de textes en intentions pour concevoir la base d'apprentissage d'un assistant conversationnel en impliquant des experts métiers pour leurs vraies compétences et en leur demandant un minimum de bagages analytiques ou techniques ?** »

En choisissant de mettre l'accent sur les connaissances réelles des experts, nous avons proposé une approche de modélisation des textes en intentions utilisant un `Clustering Interactif`. Cette méthodologie repose sur la coopération entre l'Homme et la Machine :

— d'une part, la machine réalise un *clustering* pour proposer une base initiale d'apprentissage ;

— d'autre part, l'expert métier annote des contraintes binaires entre les données dans le but d'affiner itérativement la base d'apprentissage proposée.

Une telle approche a l'avantage d'être plus instinctive, car les experts peuvent associer ou différencier les données en fonction de la similarité de leur cas d'usage, permettant ainsi de traiter les données comme ils le feraient professionnellement au quotidien. De plus, cette méthodologie définit clairement les rôles entre les parties : l'expert métier intervient uniquement pour transmettre ses connaissances métiers, et la machine se charge d'appliquer cette connaissance pour concevoir une modélisation stable et pertinente des textes en intentions.

Pour éprouver notre méthodologie d'annotation, nous avons réalisé un ensemble d'études réparties en six hypothèses : efficacité, efficience, coûts, pertinence, rentabilité et robustesse. Grâce

à ces études, nous avons pu démontrer que notre méthode diminuait sensiblement la complexité de conception d'une base d'apprentissage, réduisant notamment la nécessité de formation des experts intervenant dans un projet d'annotation. Nous mettons à disposition une implémentation technique de cette méthode (algorithmes et interface graphique associée), ainsi qu'une étude des paramètres optimaux pour obtenir une base d'apprentissage cohérente en un minimum d'annotations (hypothèse d'efficacité et d'efficience). Nous réalisons également une étude de coûts (techniques et humains) permettant de confirmer que l'utilisation d'une telle méthode est réaliste dans un cadre industriel. Les trois dernières hypothèses (pertinence, rentabilité et robustesse) permettent de vérifier le comportement et les limites de notre technique, ouvrant ainsi la discussion sur les fonctionnalités et mécanismes de suivi nécessaires à sa mise en oeuvre.

Nous dressons ensuite le bilan de notre méthodologie d'annotation sous la forme d'un guide d'utilisation synthétique. Afin que la méthode atteigne son plein potentiel, nous y fournissons un ensemble de conseils, notamment : (1) des recommandations visant à cadrer la stratégie d'annotation, (2) une aide à l'identification et à la résolution des divergences d'opinion entre annotateurs, (3) des indicateurs de rentabilité pour chaque intervention de l'expert, et (4) des méthodes d'analyse de la pertinence de la base d'apprentissage en cours de construction.

En conclusion, notre méthodologie d'annotation représente une réelle alternative aux stratégies d'annotation traditionnelles, permettant de valoriser les annotateurs pour les compétences qu'ils possèdent, et non pour les compétences dont le projet d'annotation a besoin.

Perspectives d'utilisation de notre *Clustering Interactif*

Nous avons pu montrer que cette thèse offrait une approche innovante pour concevoir une base d'apprentissage d'un assistant conversationnel, permettant d'impliquer les experts du domaine métier pour leurs vraies connaissances, tout en leur demandant un minimum de compétences analytiques et techniques. En effet, ces travaux prouvent qu'il est possible de contourner habilement la complexité d'une tâche d'annotation en reformulant judicieusement le problème exposé à l'annotateur, ouvrant ainsi la voie à des méthodes plus accessibles pour construire ces assistants.

Néanmoins, depuis 2023, la conception de *chatbot* s'est davantage tournée vers les approches génératives, celles-ci n'ayant plus besoin d'intentions pour gérer le dialogue. Mais malgré leurs performances encourageantes, ces approches manquent encore clairement de fiabilité, notamment en ce qui concerne les dérives de comportements et les risques d'hallucination ; de plus, une étape de modélisation reste nécessaire pour paramétrer certaines de leurs actions spécifiques (*allumer la lumière, lancer un appel téléphonique, faire un virement bancaire, ...*). De ce fait, nous pensons que notre méthode d'annotation reste pertinente pour entraîner des assistants orientés par tâches, pour modéliser des mécanismes de contrôle, ou encore pour assister la conception de certains assistants hybrides. D'autres part, cette méthode pourrait facilement être adaptée à d'autres usages tout en conservant l'esprit d'une tâche d'annotation à choix binaires, permettant ainsi de rester au plus proche de la connaissance métier des opérateurs.

> **Notes de l'auteur :** Parmi les techniques d'annotation récentes autour des modèles de génération de textes, nous sommes particulièrement attentifs à la méthode « `Chatbot Arena` » proposée par `LMSYS Org` (Chiang et al., 2024, voir https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard et Figure 6.1). Cette méthode consiste à comparer les réponses générées par deux modèles différents pour une même question en demandant à l'opérateur de choisir sa réponse préférée : soit la réponse de gauche est la meilleure

(*option «* `A is better` *»*), soit la réponse de droite est la meilleure (*option «* `B is better` *»*). Il est aussi possible de préciser si les deux réponses sont bonnes (*option «* `Tie` *»*) ou si aucune ne l'est (*option «* `Both are bad` *»*). Sur la base de ces annotations comparatives, un classement des modèles peut ensuite être établi en calculant leur score `elo`.

Nous avons constaté que notre `Clustering Interactif` et ce `Chatbot Arena` partagent plusieurs similitudes, bien que leurs objectifs soient différents (*la modélisation en thématiques pour le premier, le classement de modèles génératifs pour le second*). Ces similitudes incluent un accent mis sur l'annotation binaire (=/≠ *vs* 👈/👉) et sur la simplification de la tâche pour s'adapter aux compétences et aux connaissances des annotateurs. Ces points communs nous encouragent à croire qu'il sera possible d'adapter notre méthode d'annotation itérative basée sur un `Clustering Interactif` pour contribuer aux efforts de contrôle et d'alignement des modèles génératifs à venir.

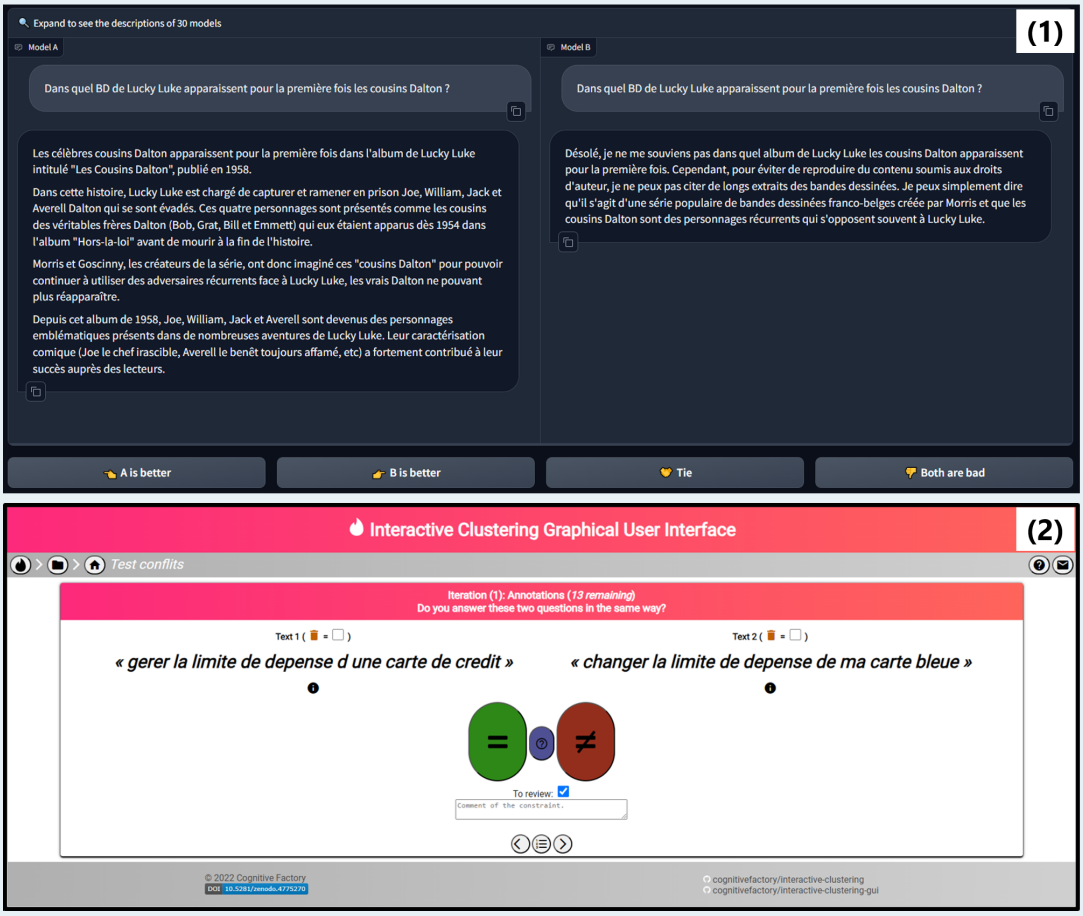

FIGURE 6.1 – *Comparaison des captures d'écran du* `Chatbot Arena` *(1) et de notre* `Clustering Interactif` *(2) : le premier concerne l'annotation de la réponse la plus adéquate parmi les réponses générées par deux modèles différents (choix binaire gauche/droite) ; le second concerne l'annotation de la similitude ou de la différence de cas d'usage entre deux questions (choix binaire égal/différent).*

Approche engagée sur l'importance de valoriser l'expertise humaine dans l'annotation

Pour conclure ce manuscrit, nous encourageons les gérants de projets d'annotation, quelle que soit la tâche d'annotation à réaliser, à repenser leur protocole de labellisation de données en considérant d'abord les vraies connaissances de leurs experts. En effet, il est tentant de concevoir une stratégie d'annotation classique, et de supposer que des annotateurs "parfaits", longuement formés ou en nombre suffisant arriveront à en supporter la complexité. Cependant, nous avons montré au cours de ce doctorat qu'en reformulant judicieusement l'objectif, il était possible de concevoir une stratégie d'annotation accessible facilement aux experts du domaine métier, faisant appel à des connaissances qu'ils appliquent professionnellement au quotidien. Par conséquent, nous pensons qu'il est de notre ressort de réexaminer l'organisation de nos projets d'annotation, afin de ne pas avoir à former des experts pour des tâches qu'ils ne maîtrisent pas, mais de les impliquer pour des tâches où ils sont déjà des experts : un tel changement de position serait alors une réelle avancée pour les annotateurs intervenant dans des projets d'apprentissage automatique.

Annexe A

Jeux de données utilisés dans nos études

Pour les différentes études réalisées au cours de ce doctorat (cf. Chapitre 4), nous avons utilisé les deux jeux de données suivants.

Sommaire

A.1 `Bank Cards` : Jeu d'entraînement en français d'assistants conversationnels traitant des demandes courantes sur les cartes bancaires

Description : Cet ensemble de données représente des exemples de demandes usuelles des clients concernant la gestion des cartes bancaires. Il peut être utilisé comme jeu d'entraînement pour un petit assistant conversationnel destiné à traiter ces demandes courantes.

Contenu : Les questions sont formulées en français. L'ensemble de données est divisé en 10 intentions (classes) dont un aperçu est disponible dans la Table A.1. Ces intentions sont construites de telle manière que toutes les questions issues d'une même intention ont la même réponse ou action. La version `1.0.0` du jeu de données contient 50 questions par intention, soit un total de 500 questions ; La version `2.0.0` du jeu de données contient 100 questions par intention, soit un total de 1 000 questions.

Intention	Définition	Exemple
`alerte_perte_vol_carte`	Affichage de la procédure de blocage d'une carte perdue ou volée.	*Comment signaler une perte de carte de paiement ?*
`carte_avalee`	Affichage de la procédure de récupération d'une carte avalée.	*Comment récupérer une carte avalée ?*
`commande_carte`	Affichage des cartes disponibles, et de la procédure de commande.	*Je souhaite changer de carte bancaire.*
`consultation_solde`	Affichage d'une synthèse des soldes bancaires du client.	*Où retrouver le solde de mon compte ?*
`couverture_assurance`	Affichage d'une synthèse des garanties d'assurances de la carte bancaire du client.	*Que couvre ma carte bancaire en cas d'hospitalisation ?*
`deblocage_carte`	Affichage de gestion du statut des cartes du client.	*Ma carte de paiement est bloquée, que faire ?*
`gestion_carte_virtuelle`	Affichage de gestion des cartes virtuelles du client.	*Comment faire pour créer une carte de paiements virtuelle ?*
`gestion_decouvert`	Affichage d'une synthèse des autorisations de découverts et de la procédure de gestion.	*Est-ce que j'ai un découvert autorisé ?*
`gestion_plafond`	Affichage de gestion des plafonds des cartes du client.	*Le plafond de ma carte est trop bas, que faire ?*
`gestion_sans_contact`	Affichage de gestion de la fonctionnalité sans contact des cartes du client.	*Je veux désactiver le sans contact sur ma carte.*

Table A.1 – *Présentation du jeu de données `Bank Cards` avec quelques exemples. La version `2.0.0` contient* 100 *questions par intention.*

Origine : Le périmètre des intentions est inspiré d'un chatbot actuellement en production. Les données ont été sélectionnées aléatoirement et reformulées manuellement pour garantir la confidentialité des utilisateurs : aucune donnée personnelle ne subsiste dans ce jeu de données. Enfin, deux réviseurs extérieurs à l'équipe de recherche (*des Data Analyst*), ayant des profils d'analystes métiers du domaine bancaire, ont validé le périmètre et le contenu de ces intentions.

Disponibilité : Le jeu de données est archivé sur la plateforme `Zenodo` et est accessible ici : Schild, 2022d.

A.2 MLSUM (The Multilingual Summarization Corpus) : Échantillon de titres d'articles de journaux en français associés à leur classification thématique

Description : C'est un ensemble d'articles de journaux avec leur titre, leur résumé et leur classification thématique. Nous l'utilisons (1) pour estimer le temps nécessaire pour annoter la similarité des titres avec des contraintes (`MUST-LINK`, `CANNOT-LINK`) et (2) pour tester la méthodologie de `Clustering Interactif` (annotation de contraintes et *clustering* sous contraintes).

Contenu : Les titres de journaux sont formulés en français. L'ensemble de données est divisé en 14 thèmes (classes) dont un aperçu est disponible dans la Table A.2. La version `1.0.0 [subset: fr+train+filtered]` contient 744 articles.

Thème	Définition	Exemple	Taille
`arts`	Actualités artistiques (spectacles, oeuvres, événements, expositions).	*La rencontre de l'art et de la gastronomie au château du Feÿ*	50
`disparitions`	Actualités nécrologiques (décès ou disparition).	*Le traducteur Jean-Pierre Carasso est mort à 73 ans*	48
`ecologie`	Actualités sur la pollution et la transition écologique.	*Comment Lyon a banni les pesticides de ses parcs et jardins*	34
`economie`	Actualités économiques, financières et boursières.	*La guerre des prix s'intensifie sur le marché du mobile en Israël*	41
`education`	Actualités liées à l'éducation et à la filière enseignante.	*Plainte de parents d'élève sur des notes jugées trop basses au bac*	62
`emploi`	Actualités liées au marché du travail et aux actions syndicales.	*Plus d'un tiers des CDI prennent fin avant la première année*	54
`immobilier`	Actualités liées au marché de l'immobilier et logements locatifs.	*Depuis la fin des années 2000, l'accession à la propriété se complique en France*	65
`meteo`	Actualités météorologiques (bulletins, catastrophes, canicule).	*L'Eure et l'est de la France balayés par les intempéries*	35
`musiques`	Actualités liées aux chanteurs, concerts et sorties d'albums.	*Opéra : Elsa Dreisig, une soprano à voix nue*	55
`police-justice`	Actualités liées aux affaires policières et aux tribunaux.	*Bygmalion : Nicolas Sarkozy directement visé*	67
`politique`	Actualités de la scène politique et législative.	*Le Sénat donne son aval à la prolongation de l'état d'urgence*	52
`sante`	Actualités sanitaires.	*Chine : un nouveau cas de grippe aviaire H7N9*	70
`sciences`	Actualités scientifiques et vulgarisation.	*L'ordinateur quantique au banc d'essai*	47
`sport`	Actualités sportives.	*F1 : Webber partira en tête à Monaco*	64

Table A.2 – *Présentation du jeu de données échantillonné à partir de `MLSUM` avec quelques exemples.*

Origine : L'ensemble de données `MLSUM` a été proposé par Scialom et al., 2020. Notre ensemble de données en est un échantillon (*sélection au hasard de* 75 *articles dans les* 14 *sujets les plus utilisés*) filtré (*conservation des articles qui ont un sujet évident par rapport à leur titre, sans leur corps*). Deux réviseurs (*une Data Scientist et moi-même*) ont travaillé sur cette tâche afin de limiter la subjectivité du filtrage : l'échantillon final contient 744 articles.

Disponibilité : Le jeu de donnée original est archivé sur `arXiv` et est accessible ici : Scialom et al., 2020. L'échantillon réalisé par nos soins est archivé sur la plateforme `Zenodo` et

est accessible ici : Schild et Adler, 2023.

Annexe B

Les assistants conversationnels (*chatbots*)

Au début de ce doctorat (`octobre 2019`), nous pouvions noter que :

— selon Costello et LoDolce, 2019, « *seuls* 4% *des clients de* `Gartner` [déclaraient] *utiliser des chatbots sur leur lieu de travail, mais* 40% [avaient] *l'intention de les mettre en oeuvre à court terme* » ; et

— selon Goasduff, 2019, « *d'ici 2022, 70% des employés* [interagiraient] *quotidiennement avec les plateformes conversationnelles* ».

Aujourd'hui (`octobre 2023`), le mot « *chatbot* » est présent sur toutes les lèvres, surtout depuis la révolution des `IA` génératives lancée par `ChatGPT` (OpenAI, 2023) :

— selon Costello et LoDolce, 2022, une entreprise sur deux aurait actuellement recours à une forme de *chatbot* pour gérer sa relation client, et « *d'ici 2027, les chatbots deviendront le principal canal de service client pour environ un quart des organisations* ».

Dans cette annexe, nous allons :

— présenter rapidement les assistants conversationnels et leurs principales utilisations (voir Section B.1) ;

— décrire les approches principales permettant de concevoir un assistant conversationnel, avec leurs avantages et leurs inconvénients (voir Section B.2) ;

— discuter du dilemme des choix de conception (voir Section B.3).

Sommaire

B.1 Présentation rapide des assistants conversationnels

Les *chatbots* sont des robots conversationnels permettant à un utilisateur d'obtenir des informations ou d'automatiser des actions à l'aide d'instructions en langage naturel.

👍 L'utilisation de tels assistants comporte **plusieurs avantages** : ces derniers permettent de réduire les coûts en automatisant certaines tâches simples et répétitives (*permettant ainsi aux opérateurs humains de se concentrer sur d'autres tâches à risques ou à forte valeur ajoutée*), ils sont réactifs et toujours accessibles (*au milieu de la nuit, les jours fériés, et même le vendredi après 16h*), et ils ont un comportement stable quelle que soit la situation (*pas de sautes d'humeur, de coups de fatigue ou d'erreurs d'inattention*).

👎 Néanmoins, ces assistants rencontrent aussi **plusieurs inconvénients** : leur compréhension limitée du langage peut introduire des erreurs (*notamment lorsque le sujet est complexe, quand il y a trop ou trop peu de contexte*), ils manquent parfois de flexibilité ou d'empathie (*nécessitant alors d'escalader la requête vers un opérateur humain*), et les tâches de conception ou de mises sous contrôle nécessitent des coûts importants (*collecte et annotation de données, création de parcours de dialogue, vérification du comportement et des dérives, ...*).

Cependant, ces assistants sont utilisés dans de nombreux domaines :

— la **relation client à distance** (*proposer une assistance 24/7, répondre aux questions fréquentes, automatiser la prise de rendez-vous, envoyer des formulaires de satisfaction, ...*) ;

— le **commerce en ligne** (*remplir des formulaires de réservation ou de rétractation, suivre une commande, assurer le service après-vente, ...*) ;

— la **domotique** (*gérer les appareils connectés, écouter de la musique, interagir avec le `GPS`, être notifié en cas d'alerte intrusion, ...*) ;

— l'**accès à l'information** (*consulter une base documentaire, favoriser l'éducation, vulgariser ou résumer des concepts, ...*) ;

— le **divertissement** (*raconter une histoire ou une blague, organiser un jeu narratif, organiser une sortie, ou simplement discuter lors d'une insomnie, ...*).

🔍 **Exemples :** Parmi les exemples connus, nous avons :

— `Alexa` (Alexa Internet, 2018) et `Google Assistant` (Google, 2016), permettant de gérer des appareils connectés ;

— L'`Assistant Virtuel SNCF`, gérant l'achat de billets pour la SNCF (SNCF, 2018) ;

— `Louis`, gérant le suivi des bagages d'Air France (Air France, 2017) ;

— `AI Dungeon`, racontant des histoires interactives et des jeux de rôles (Latitude Inc. et Oasis Tech Inc., 2019) ;

— `ChatGPT` (OpenAI, 2023) ou `BARD` (Google, 2023), permettant de discuter de presque n'importe quel sujet...

— ...

B.2 Approches de conception : *task-oriented* vs *chat-oriented*

Pour concevoir un assistant conversationnel, il faut **trois fonctionnalités** :

1. un moyen de ***comprendre la demande*** et d'en extraire les informations importantes ;
2. un moyen de ***gérer le dialogue*** et de définir la prochaine action de l'assistant ;
3. un moyen de ***répondre à l'utilisateur*** et de réaliser l'action demandée.

Il existe de nombreuses façons d'agencer et d'implémenter ces fonctionnalités, mais selon Chen et al., 2017, nous pouvons les distinguer en **deux approches principales** en fonction de l'usage de l'assistant :

1. soit l'assistant est spécialisé pour une tâche bien déterminée (*task-oriented*), dans ce cas sa conception se base traditionnellement sur une **approche symbolique** pour modéliser ses états de dialogue ;
2. soit l'assistant est axé sur la fluidité de la conversation avec l'utilisateur (*chat-oriented*), dans ce cas sa conception est plutôt basée sur une **approche numérique**, utilisant un encodeur et un décodeur pour traiter la demande en un seul jet.

Ces deux approches bien connues sont illustrées dans la Figure B.1 et seront détaillées dans les sections suivantes.

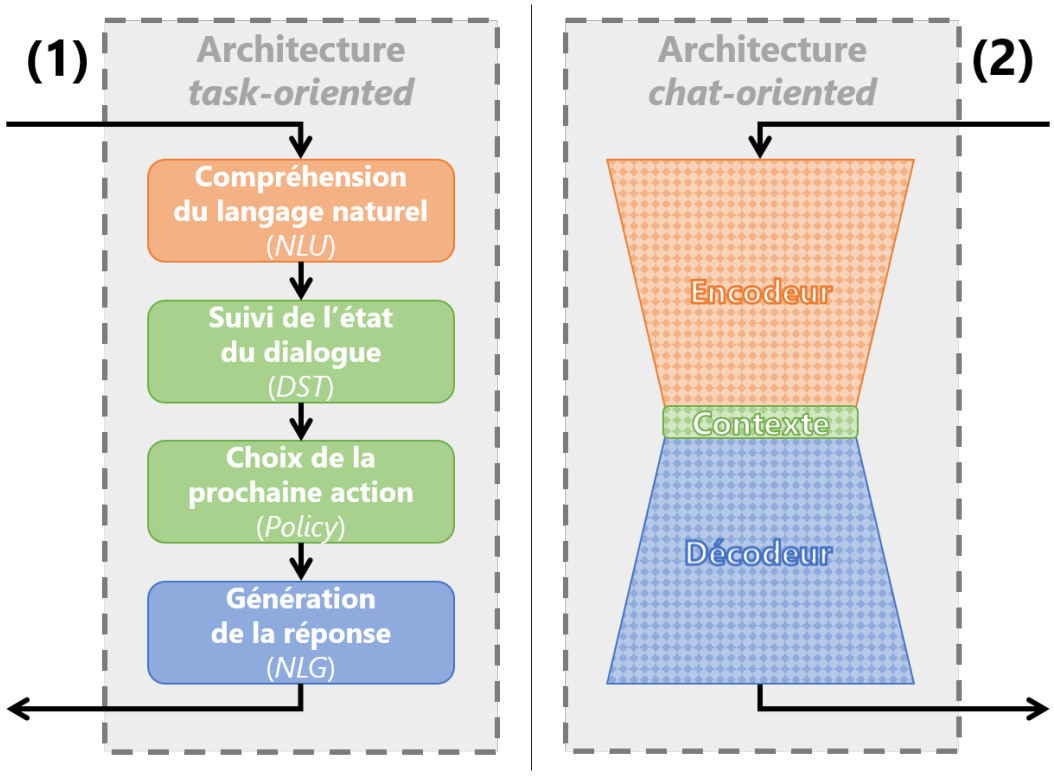

Figure B.1 – *Schéma illustrant les deux approches principales de conception d'un assistant conversationnel : **(1)** représente les **approches task-oriented** à l'aide d'une architecture manipulant des états de dialogue ; **(2)** représente les **approches chat-oriented** à l'aide d'une architecture à base de transformers, encodant et décodant numériquement le dialogue et son contexte.*

B.2.1 Approches *task-oriented*

Les **approches orientées par tâches** (*task-oriented*) considèrent la conversation comme une succession d'étapes menant à une action précise : le système est donc conçu pour collecter les informations nécessaires et interagir avec un moteur d'actions dans le but d'accomplir un objectif. Ces approches reposent généralement sur une modélisation explicite du parcours utilisateur, permettant ainsi de s'assurer de l'exécution pas à pas des actions intermédiaires de la tâche demandée.

La FIGURE B.1 **(1)** représente l'architecture traditionnellement utilisée pour concevoir ce type d'assistants. Cette architecture est composée des éléments suivants :

— le `NLU` (*Natural Language Understanding*) : ce composant utilise un ensemble de concepts pour interpréter le dialogue de manière symbolique. Cette modélisation du dialogue, généralement implémentée à l'aide de méthodes supervisées, repose sur des détections de *domaines* (*thématique générale traitée*), des détections de *sentiments* (*positif, négatif ou neutre*), des détections d'*intentions* (*action exprimée par l'utilisateur, manifestée par le verbe d'action*), ou encore des extractions d'*entités* (*ensemble de mentions textuelles pertinentes*) (voir ADAMOPOULOU et MOUSSIADES, 2020) ;

— le `DST` (*Dialogue State Tracking*) : ce composant utilise les concepts détectés précédemment par le `NLU` (*domaine*, *sentiment*, *intentions*, *entités*) pour déterminer ou mettre à jour l'état de la conversation, et ainsi définir les actions possibles à partir de l'état en cours ;

— la `PL` (*Policy Learning*) : ce composant choisit la prochaine action de l'assistant parmi celles autorisées par le `DST` pour l'état en cours. Ce choix peut être réalisé à l'aide d'un ensemble de règles, d'un apprentissage supervisé ou encore d'un apprentissage par renforcement (voir BRABRA et al., 2022) ;

— la `NLG` (*Natural Language Generation*) : ce dernier composant affiche une réponse à l'utilisateur qui soit en adéquation avec l'action choisie par la `PL`. Cette réponse peut être paramétrée à l'avance ou être générée à la volée.

Cette approche est rudimentaire mais efficace, et elle est par conséquent fréquemment utilisée pour concevoir des assistants conversationnels dans un cadre industriel.

👍 Au sujet des avantages : les différents composants de l'architecture sont simples à implémenter et à maintenir (*les différentes briques sont indépendantes et ont des fonctionnalités bien précises*) ; de nouvelles règles de dialogue peuvent facilement être ajoutées dans le système (*du moins si les processus de ces tâches sont bien documentés*) ; il est facile de contrôler le comportement de l'assistant (*il suffit de restreindre les actions possibles dans le `DST`, de figer le choix de la `PL`, ou encore de déterminer à l'avance chaque réponse de la `NLG`*) ; cette architecture simple consomme peu de ressources en production (*les modèles de détections et la gestion du dialogue peuvent tourner sur `CPU`*).

👎 Au sujet des inconvénients : les coûts initiaux peuvent être importants (*notamment en ce qui concerne la collecte et la modélisation des données d'apprentissage, ou encore la définition des parcours de dialogue pour des tâches non documentées*) ; la compréhension du langage est limitée (*certaines erreurs ou ambiguïtés de langage rendent les détections du `NLU` non adaptées ou non performantes*) ; le dialogue est très peu flexible, et le périmètre de l'assistant est réduit à ce qui est modélisé (*le dialogue peut être bloqué ou rejeté si aucune règle n'est prévue pour la demande de l'utilisateur ou pour l'état courant*).

Exemples : Pour illustrer nos propos, analysons la demande fictive suivante :

« *Je veux réserver un billet de train pour Strasbourg.* »

— le `NLU` pourrait détecter le domaine "`voyage`", l'intention "`réservation`" et les entités "*billet de train*$_{\texttt{(produit)}}$" et "*Strasbourg*$_{\texttt{(gare\_destination)}}$" ;

— le `DST` pourrait établir l'objectif de réserver un train, mais que les informations `(date_départ)` et `(gare_départ)` sont manquantes ;

— la `PL` pourrait décider de demander d'abord le complément d'information sur la gare de départ (*et demandera la prochaine information plus tard*) ;

— la `NLG` pourrait sortir une phrase de réponse prédéfinie pour demander le complément d'information à l'utilisateur.

Exemples : Nous pouvons citer les projets ou outils suivants :

— `RASA` (Bocklisch et al., 2017) et `WATSON` (Hoyt et al., 2016) sont deux moteurs de dialogue basés sur une approche symbolique et manipulant des intentions et des entités ;

— `Assistant Virtuel SNCF` (SNCF, 2018), `Google Assistant` (Google, 2016) et `Alexa` (Alexa Internet, 2018) sont des assistants connus pour être orientés par tâches (*les deux derniers pouvant même être paramétrables par l'utilisateur final pour ses usages en domotique*) ;

— Yan et al., 2012 décrivent un projet de conception d'un assistant conversationnel pour du commerce en ligne.

B.2.2 Approches *chat-oriented*

Les **approches axées sur le dialogue** (*chat-oriented* ou *non-task-oriented*) sont utilisées pour favoriser des conversations ouvertes avec les utilisateurs : le système n'est donc pas conçu pour répondre à un besoin spécifique, mais plutôt pour être en mesure de discuter avec un utilisateur sur des thématiques générales, et de rebondir à chacun de ses messages de manière fluide. Afin de capturer la vaste diversité du langage, ces approches reposent majoritairement sur les capacités du *Deep Learning*, notamment pour concevoir des modèles de langage permettant de capturer et de reproduire des séquences de phrases (voir Ni et al., 2022 et Kumar et al., 2016).

La Figure B.1 **(2)** correspond à l'architecture *transformers* (Uszkoreit, 2017), communément utilisée pour représenter ce type d'assistants. Cette architecture est composée de deux réseaux de neurones :

— un `encodeur` : ce premier réseau a pour objectif de traduire les séquences de mots du texte de l'utilisateur en une représentation numérique abstraite ;

— le `décodeur` : ce second réseau a pour objectif de traiter la représentation numérique produite par l'encodeur dans le but de générer des séquences de mots pour former la réponse ;

— entre les deux, un `vecteur de contexte` : ce vecteur permet de maintenir la continuité de la conversation entre plusieurs échanges. L'encodeur se charge de mettre à jour le contexte tandis que le décodeur l'utilise pour adapter la génération de son texte.

Cette approche est plus difficile à mettre en place qu'une approche symbolique, mais elle dispose d'un plus grand potentiel.

👍 Au sujet des avantages : par conception, le système peut s'adapter à de nombreuses situations et offrir une expérience agréable à l'utilisateur (*aucun blocage de l'état du dialogue, possibilité de personnaliser le dialogue, gestion plus simple de l'ambiguïté et des émotions, ...*) ; il n'a pas besoin d'interpréter le langage à l'aide d'une modélisation abstraite (*cette interprétation est déduite de l'immense volume de textes utilisés à l'entraînement*) ; en utilisant des larges modèles de langage, des propriétés intéressantes peuvent apparaître (*capacité de consulter un très large champ de connaissances, de résumer et traduire un texte, de résoudre un calcul, d'effectuer une tâche d'analyse ou de déduction, de générer un code informatique, ...*).

👎 Au sujet des inconvénients : les coûts d'entraînement et d'inférence du modèle sont conséquents (*besoin d'une immense quantité de données pour l'apprentissage, besoin de `GPU` et d'une infrastructure technique conséquente, ...*) ; le décodeur peut générer des contenus inexacts, erronés ou offensants (*imprécisions, fausses informations, hallucinations, réponse biaisée ou discriminatoire, ...*) ; la reproduction des données utilisées pour l'entraînement questionne le respect des droits d'auteur et expose potentiellement des données privées ou confidentielles (*capacités de reproduire des passages entiers de livres, de trouver des numéros de cartes bancaires ou des clés d'activation de logiciel issus de fuites de bases de données, ...*) ; la mise sous contrôle de l'assistant est difficile voire impossible (*manque de visibilité et d'explicabilité concernant le comportement du modèle, peu de leviers d'action directe, recours à une annotation de masse par ressenti utilisateur, ...*).

🔍 **Exemples :** Nous pouvons citer les projets ou outils suivants :

— `ChatGPT` (OpenAI, 2023), `BARD` (Google, 2023) et `LLAMA2` (Touvron et al., 2023) sont trois solutions basées sur des larges modèles de langage permettant de discuter de presque tous les domaines ainsi que de réaliser de très nombreuses tâches ;

— Kaddour et al., 2023 détaillent une longue liste d'inconvénients et de défis pour l'utilisation des larges modèles de langage.

B.3 Dilemme de conception et approches hybrides

Dans la section précédente, nous avons pu voir que deux approches de conception se distinguent : d'une part, il y a l'approche *task-oriented*, basée sur une gestion symbolique du dialogue et permettant facilement de contrôler le comportement de l'assistant ; d'autre part, il y a l'approche *chat-oriented*, basée sur un encodage/décodage numérique de la conversation et permettant une grande flexibilité dans les échanges avec l'utilisateur. Cependant, les avantages d'une approche représentent les inconvénients de l'autre : en effet, l'approche *task-oriented* est plutôt rigide, tandis que l'approche *chat-oriented* est sensible aux dérives de comportement et aux risques d'hallucinations. Nous nous trouvons donc face à un dilemme de conception : **voulons-nous privilégier la fluidité du dialogue ou le contrôle du comportement de l'assistant ?**

Ce choix cornélien questionne le **niveau d'automatisation** que nous sommes prêts à déléguer à la machine.

Notes de l'auteur : Pour estimer un **degré d'automatisation**, Sheridan et Verplank, 1978 identifient **dix niveaux** :

- de « la machine n'offre aucune assistance, l'opérateur doit choisir l'action à faire et la réaliser lui-même » (*niveau 1 : automatisation nulle*) ;
- à « la machine s'occupe de tout (analyse, choix et réalisation de l'action) sans consultation préalable de l'opérateur ni information sur le déroulement de la tâche » (*niveau 10 : automatisation totale*).

En plus de cette proposition en dix niveaux d'automatisation, Parasuraman et al., 2000 proposent une analyse selon **quatre aspects** :

- l'acquisition de l'information (*à quel point la machine collecte et sélectionne les informations pertinentes à porter à l'attention de l'opérateur ?*) ;
- l'analyse de l'information (*à quel point la machine participe à l'examen des informations dans le but de proposer les actions possibles à réaliser ?*) ;
- la prise de décision (*à quel point la machine choisit l'action à réaliser à la place de l'opérateur ?*) ;
- l'implémentation de l'action (*à quel point la machine réalise l'action et informe l'opérateur des résultats ?*).

Nous pouvons d'ailleurs faire des parallèles avec l'architecture des assistants conversationnels :

- l'acquisition de l'information correspond au `NLU` ou à l'`encodeur` ;
- l'analyse de l'information et la prise de décision correspondent aux `DST`/`PL` ou à la manipulation du `vecteur de contexte` ;
- l'implémentation de l'action correspond à la `NLG` ou au `décodeur`.

De ce fait, les choix de conception de l'assistant reviennent à définir quels composants doivent être rigides (*gestion de règles*), entraînés (*apprentissage supervisé, ...*) ou totalement délégués à la machine (*modèles de langage, ...*).

Pour résoudre ce dilemme de conception, **diverses approches hybrides peuvent être mises en oeuvre** :

— si nous voulons **plus de fluidité** dans la conversation (*pour améliorer l'ergonomie utilisateur, pour permettre de discuter de sujets plus généraux, ...*), nous intégrerons davantage de composants issus d'une approche *chat-oriented*, acceptant en échange de **déléguer** la gestion du dialogue et/ou la génération de réponses à la machine ;

— à l'inverse, si nous avons besoin d'un **contrôle plus fort** (*pour assurer la sécurité des actions réaliser, pour garantir l'image de la marque, pour éviter les scandales, ...*), nous nous tournerons davantage vers l'utilisation de composants modélisant une approche symbolique du dialogue, permettant ainsi de **limiter la machine** dans ses possibilités.

Exemples : Nous dressons une liste non exhaustive d'approches hybrides :

— la compréhension du langage naturel à l'aide de détection de concepts (`NLU`) peut être améliorée en utilisant les capacités d'un large modèle de langage (voir `RASA v3.6+` par Bocklisch et al., 2017) ou en réutilisant les représentations vectorielles produites par l'encodeur (*embeddings*) ;

— le moteur de dialogue d'une approche *task-oriented* (`DST` et `PL`) peut être entraîné par renforcement dans le but d'améliorer la prise de décision et de rendre les étapes de dialogue plus flexibles (voir Chen et al., 2017 et Brabra et al., 2022) ;

— le décodeur d'une approche *chat-oriented* peut être utilisé pour la génération de réponses d'un assistant *task-oriented* (`NLG`), permettant afin d'améliorer la qualité de ses réponses (voir Gao et al., 2018 et Chen et al., 2017) ;

— il est possible de se baser sur une tâche de recherche d'informations dans une base documentaire pour limiter les hallucinations lors de la génération de réponses (voir Shuster et al., 2021 et Y. Zhang et al., 2016).

Annexe C

Implémentations du `Clustering Interactif`

Au cours de ce doctorat, nous avons réalisé un ensemble d'implémentations en `Python` afin de mettre en oeuvre notre méthodologie de `Clustering Interactif`. Cette implémentation est répartie en trois librairies :

1. `cognitivefactory-interactive-clustering`[62] (Schild, 2022a), regroupant les gestions de données et des contraintes, les algorithmes de *clustering* et d'échantillonnage ;
2. `cognitivefactory-interactive-clustering-gui`[63] (Schild, 2022b), intégrant la logique de la méthodologie dans une application web ;
3. `cognitivefactory-features-maximization-metric`[64] (Schild, 2023), disposant d'une méthode de sélection des patterns linguistiques caractéristiques d'un jeu de données labellisées, permettant ainsi d'analyser la pertinence d'un résultat de *clustering* en fonction du vocabulaire utilisé dans chaque *cluster*.

> **Pour information :** Ces implémentations sont disponibles sur le GitHub https://github.com/cognitivefactory. Les *pipelines* d'intégration continue contiennent les étapes de formatage du code (*grâce aux librairies* `isort`[65] *et* `black`[66]), de vérification de la qualité et du typage du code (*grâce aux librairies* `flake8`[67] *et* `mypy`[68]), de la vérification des vulnérabilités et des failles de sécurité (*grâce à la librairie* `safety`[69]), de l'exécution de tests unitaires et de la vérification de la couverture du code testé (*grâce aux librairies* `pytest`[70] *et* `coverage`[71]), et de la génération de la documentation technique (*grâce à la librairie* `mkdocs`[72]).

62. https://pypi.org/project/cognitivefactory-interactive-clustering/
63. https://pypi.org/project/cognitivefactory-interactive-clustering-gui/
64. https://pypi.org/project/cognitivefactory-features-maximization-metric/
65. https://pypi.org/project/isort/
66. https://pypi.org/project/black/
67. https://pypi.org/project/flake8/
68. https://pypi.org/project/mypy/
69. https://pypi.org/project/safety/
70. https://pypi.org/project/pytest/
71. https://pypi.org/project/coverage/
72. https://pypi.org/project/mkdocs/

Dans cette annexe, nous allons détailler ces implémentations, leurs fonctionnalités et certains des choix de mises en oeuvres.

Sommaire

C.1 Implémentation de la librairie `cognitivefactory-interactive-clustering`

La librairie `cognitivefactory-interactive-clustering`[73] (Schild, 2022a) a été implémentée au cours de ce doctorat dans le but de mettre à disposition un ensemble d'algorithmes nécessaires à l'utilisation de notre méthodologie de `Clustering Interactif`. Cette librairie comporte plusieurs fonctionnalités :

— la gestion des données avec leurs prétraitements et leur vectorisation (cf. Section C.1.1) ;

— la gestion des contraintes avec le calcul des propriétés de transitivité et la détection des conflits (cf. Section C.1.2) ;

— l'exécution d'algorithmes de *clustering* sous contraintes pour proposer une segmentation des données (cf. Section C.1.3) ;

— l'exécution d'algorithmes d'échantillonnage pour sélectionner les prochaines contraintes à annoter (cf. Section C.1.4).

Nous présentons succinctement cette librairie avec certains choix d'implémentation.

> **Pour information :** La documentation technique de cette librairie est accessible par le lien suivant : https://cognitivefactory.github.io/interactive-clustering/.

Pour les sections suivantes, nous suivrons l'exemple suivant (cf. Code C.1) pour présenter nos implémentations.

```python
# Définir les données.
dict_of_texts = {
  "0": "Comment signaler un vol de carte bancaire ?",
  "1": "J'ai égaré ma carte bancaire, que faire ?",
  "2": "J'ai perdu ma carte de paiement",
  "3": "Le distributeur a avalé ma carte !",
  "4": "En retirant de l'argent, le GAB a gardé ma carte...",
  "5": "Le distributeur ne m'a pas rendu ma carte bleue.",
  # ...
  "N": "Pourquoi le sans contact ne fonctionne pas ?",
}
```

Code C.1 – *Jeu exemple pour présenter notre implémentation du `Clustering Interactif`.*

C.1.1 Gestion des données

Tout d'abord, en ce qui concerne la **manipulation de données**, nous utilisons le module `utils` de la librairie `cognitivefactory-interactive-clustering`. Les données sont stockées dans un dictionnaire `Python` afin de tracer les manipulations à l'aide d'une clé servant d'identifiant de la donnée.

73. https://pypi.org/project/cognitivefactory-interactive-clustering/

Nous avons d'une part la partie `utils.preprocessing`[74] qui permet de normaliser les données. Par défaut :

- le texte est passé en *minuscule* (de « `Bonjour` » à « `bonjour` »),
- la *ponctuation* est supprimée « `c'est-à-dire ?!` » à « `c est a dire` »),
- les *accents* sont enlevés (de « `crédit` » à « `credit` »),
- et les multiples *espaces blancs* sont convertis en un unique espace simple (de « `au  revoir` » à « `au revoir` »).

Si besoin, trois options "avancées" sont disponibles pour réaliser des prétraitements plus destructifs :

- la suppression des mots vides (*stopwords*, Nothman et al., 2018),
- la conversion des mots vers leur forme racine (*lemmatisation*, Manning et Schütze, 2000),
- et la suppression des mots en fonction de leur profondeur dans l'arbre de dépendances syntaxiques (Nivre, 2006).

Ces traitements sont réalisés en bénéficiant des fonctionnalités mises à disposition d'un modèle de langue de type SpaCy (Honnibal et Montani, 2017), avec par défaut l'utilisation du modèle `fr-core-news-md`.

Pour nos études, nous définissons quatre niveaux de prétraitements facilement identifiables :

1. L'**absence de prétraitements**, soit la conservation de la donnée brute, noté `prep.no` ;
2. Les **prétraitements simples**, correspondant au traitement de base (minuscules, ponctuations, accents, espaces blancs), notés `prep.simple` ;
3. Les **prétraitements avec lemmatisation**, correspondant au traitement de base auquel s'ajoute la conversion des mots vers leur forme racine, notés `prep.lemma` ;
4. les **prétraitements avec filtres**, correspondant au traitement de base avec l'élagage de l'arbre de dépendance syntaxique de la phrase, notés `prep.filter`.

D'autre part, la partie `utils.vectorization`[75] permet de transformer les données en une représentation exploitable pour la machine. Deux modes de vectorisation sont mis à disposition :

1. **TF-IDF** (Ramos, 2003), utilisant la fréquence d'occurrence des mots pour représenter une phrase, et noté `vect.tfidf` pour nos études ;
2. **SpaCy** (Honnibal et Montani, 2017), utilisant le modèle de langue `fr-core-news-md`, et noté `vect.frcorenewsmd`.

Vous avez un exemple d'utilisation des modules de prétraitements et de vectorisation dans Code C.2.

```
1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.utils.preprocessing
    import preprocess
```

74. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/utils/preprocessing/

75. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/utils/vectorization/

```
from cognitivefactory.interactive_clustering.utils.vectorization
    import vectorize

# Prétraitement des données.
dict_of_preprocess_texts = preprocess(
  dict_of_texts=dict_of_texts,
  apply_stopwords_deletion=False,
  apply_parsing_filter=False,
  apply_lemmatization=False,
  spacy_language_model="fr_core_news_md",
)
"""
{"0": "comment signaler un vol de carte bancaire",
 "1": "j ai egare ma carte bancaire, que faire",
 "2": "j ai perdu ma carte de paiement",
 "3": "le distributeur a avale ma carte",
 "4": "en retirant de l argent le gab a garde ma carte",
 "5": "le distributeur ne m a pas rendu ma carte bleue",
 # ...
 "N": "pourquoi le sans contact ne fonctionne pas"}
"""

# Vectorisation des données.
dict_of_vectors = vectorize(
  dict_of_texts=dict_of_preprocess_texts,
  vectorizer_type="tfidf",
)
```

CODE C.2 – *Démonstration de notre implémentation des prétraitements et de la vectorisation sur le jeu d'exemples.*

C.1.2 Gestion des contraintes

En ce qui concerne la **manipulation de contraintes**, nous utilisons le module `contraints`[76] de la librairie `cognitivefactory-interactive-clustering`.

Deux types de contraintes sont prises en charge (cf. WAGSTAFF et CARDIE, 2000) :

- les contraintes `MUST-LINK` permettant de réunir deux données,
- et les contraintes `CANNOT-LINK` permettant à l'inverse de les séparer.

Ces types de contraintes respectent les propriétés de transitivité décrites dans l'EQUATION C.1) et sont illustrées dans la FIGURE C.1 (**(1)** et **(2)**). Nous notons ainsi qu'il est possible de déduire la troisième contrainte d'un triangle de trois points si nous connaissons déjà les deux premières.

76. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/constraints/

$$(\forall A, B, C) \begin{cases} \texttt{MUST\_LINK}(A,B) \wedge \texttt{MUST\_LINK}(B,C) \Rightarrow \texttt{MUST\_LINK}(A,C) \\ \texttt{MUST\_LINK}(A,B) \wedge \texttt{CANNOT\_LINK}(B,C) \Rightarrow \texttt{CANNOT\_LINK}(A,C) \end{cases} \tag{C.1}$$

Pour respecter ces propriétés, le gestionnaire de contraintes doit calculer les transitivités à chaque ajout ou suppression de contraintes. Nous distinguerons donc une contrainte ajoutée (`added`) d'une contrainte déduite par transitivité (`inferred`).

Il se peut que la contrainte en cours d'ajout contredise les contraintes précédemment déduites : nous parlons alors d'incohérence ou de conflit (cf. Figure C.1 et Equation C.2). Dans ce cas, l'ajout de la dernière contrainte n'est pas pris en compte et le gestionnaire renvoie une erreur permettant d'identifier ce conflit. Ce conflit peut simplement venir d'une erreur d'inattention, mais peut aussi venir d'une déduction basée sur des ajouts antérieurs erronés. Sémantiquement, un conflit indique une contradiction dans la gestion des données, dû au fait que les données concernées doivent à la fois être réunies et séparées...

$$(\exists A, B, C)\ \texttt{MUST\_LINK}(A,B) \wedge \texttt{MUST\_LINK}(B,C) \wedge \texttt{CANNOT\_LINK}(A,C) \tag{C.2}$$

À partir d'une donnée D, et par application de la propriété de transitivité des `MUST-LINK`, nous appelons **composant connexe** de D l'ensemble des données D_i liées par une succession de contraintes `MUST-LINK` à D (cf. Figure C.1). Ce composant peut être vu comme un noyau de *clusters*. Il pourra être associé à d'autres noyaux par similarité pour former un *cluster* plus conséquent, ou être distingué d'autres noyaux pour former plusieurs *clusters*.

Un exemple d'utilisation du module de gestion de contraintes est consultable dans Code C.3.

```
# Import des dépendances.
from cognitivefactory.interactive_clustering.constraints.factory import managing_factory

# Création du gestionnaire de contraintes.
constraints_manager = managing_factory(
    manager="binary",
    list_of_data_IDs = list(dict_of_texts.keys()),
    # ["0", "1", "2", "3", "4", "5", ..., "N"]
)

# Ajout de contraintes.
constraints_manager.add_constraint(
    data_ID1="0",  # "Comment signaler un vol de carte bancaire ?"
    data_ID2="1",  # "J'ai égaré ma carte bancaire, que faire ?"
    constraint_type="MUST_LINK",
)
constraints_manager.add_constraint(
    data_ID1="3",  # "Le distributeur a avalé ma carte !"
    data_ID2="4",  # "En retirant de l'argent, le GAB a gardé ma carte ..."
    constraint_type="MUST_LINK",
)
constraints_manager.add_constraint(
```

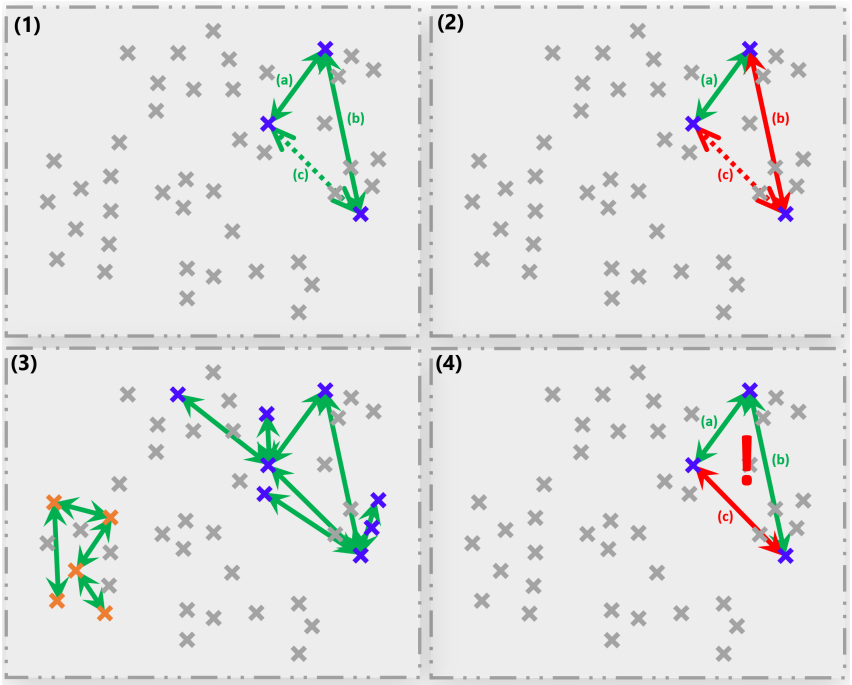

FIGURE C.1 – *Exemples des propriétés de transitivité des contraintes `MUST-LINK` (flèches vertes) et `CANNOT-LINK` (flèches rouges). (1) et (2) représentent les possibilités de déduction d'une contrainte (`(c)`) en fonction des deux autres (`(a)` et `(b)`). (3) représente deux composants connexes définis par la transitivité des contraintes `MUST-LINK`. Enfin, (4) représente un cas de conflit où une contrainte (`(c)`) ne correspond pas à sa déduction faite à partir des autres contraintes (`(a)` et `(b)`).*

```
23    data_ID1="0",  # "Comment signaler un vol de carte bancaire ?"
24    data_ID2="N",  # "Pourquoi le sans contact ne fonctionne pas ?"
25    constraint_type="CANNOT_LINK",
26 )
27 # NB: ajouter une contrainte "MUST_LINK" entre "1" et "N" lèverait
       une erreur.
28
29 # Récupération des composants connexes.
30 connected_components = constraints_manager.get_connected_components()
31 """
32 [['0', '1'],
33  ['2'],
34  ['3', '4'],
35  ['5'],
36  ['N']]
37 """
```

CODE C.3 – *Démonstration de notre implémentation de gestion des contraintes sur le jeu d'exemples.*

C.1.3 Algorithmes de *clustering* sous contraintes

En ce qui concerne le **regroupement automatique** des données par similarité, nous utilisons le module `clustering`[77] de la librairie `cognitivefactory-interactive-clustering`.

Ce module met à disposition cinq algorithmes de *clustering* sous contraintes :

1. `KMeans`, dans sa version `COP-KMeans` (Wagstaff et al., 2001), noté `clust.kmeans.cop`, et sa version `MPC-KMeans` (Khan et al., 2012), noté `clust.kmeans.mpc`) ;
2. `DBscan`, dans sa version `C-DBScan` (Ruiz et al., 2010), noté `clust.cdbscan` ;
3. **Hiérarchique** (Davidson et Ravi, 2005), avec quatre métriques de distances : `single` (noté `clust.hier.sing`), `complete` (noté `clust.hier.comp`), `average` (noté `clust.hier.avg`) et `ward` (noté `clust.hier.ward`) ;
4. **Spectral**, dans sa version `SPEC` (Kamvar et al., 2003), noté `clust.spec` ;
5. **Propagation par affinité** (Givoni et Frey, 2009), noté `clust.affprop`.

Une classe abstraite définit les prérequis des algorithmes implémentés (avoir une méthode `cluster`) et une *factory* est disponible pour instancier rapidement un objet de *clustering*. Enfin, un exemple d'utilisation de ce module est consultable dans Code C.4.

```python
# Import des dépendances.
from cognitivefactory.interactive_clustering.clustering.factory import clustering_factory

# Initialiser un objet de clustering.
clustering_model = clustering_factory(
    algorithm="kmeans",
    model="COP",
    random_seed=42,
)

# Lancer le clustering.
clustering_result = clustering_model.cluster(
    constraints_manager=constraints_manager,
    nb_clusters=2,
    vectors=dict_of_vectors,
)
"""
{"0": 0,  # "Comment signaler un vol de carte bancaire ?"
 "1": 0,  # "J'ai égaré ma carte bancaire, que faire ?"
 "2": 0,  # "J'ai perdu ma carte de paiement"
 "3": 1,  # "Le distributeur a avalé ma carte !"
 "4": 1,  # "En retirant de l'argent, le GAB a gardé ma carte..."
 "5": 1,  # "Le distributeur ne m'a pas rendu ma carte bleue."
 # ...
 "N": 1}  # "Pourquoi le sans contact ne fonctionne pas ?"
```

77. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/clustering/

```
26 """
```

CODE C.4 – *Démonstration de notre implémentation du clustering sous contraintes sur le jeu d'exemples.*

> **Pour information :** Dans le cadre d'un projet étudiant avec l'École d'Ingénieurs Télécom Physique Strasbourg (au cours de l'année 2022), les implémentations des algorithmes `MPC-KMeans`, `C-DBScan` et de propagation par affinité ont été ajoutées. Les élèves ont conclu ce projet d'extension en suggérant de se concentrer sur l'étude du C-DBScan car les deux autres algorithmes étaient soit trop instables, soit trop gourmand en temps de calcul. Les autres algorithmes (`COP-KMeans`, hiérarchique et spectral) ont été implémentés au début de ce doctorat.

C.1.4 Algorithmes d'échantillonnage de contraintes

En ce qui concerne l'**échantillonnage** de contraintes à annoter, nous utilisons le module `sampling`[78] de la librairie `cognitivefactory-interactive-clustering`.

Cet échantillonnage correspond à la sélection de paires de données. Par défaut, l'échantillonnage est purement aléatoire. Cependant, plusieurs options sont disponibles :

- une restriction sur la *distance* pouvant imposer aux données d'être les plus proches ou les plus éloignées du corpus ;
- une restriction sur le *résultat du clustering* pouvant imposer aux données d'être issues d'un même *cluster* ou de *clusters* différents,
- une restriction pour exclure les contraintes *déjà annotées*,
- et enfin une restriction pour exclure les contraintes *déjà déduites* par transitivité.

Sur cette base, nous définissons quatre niveaux d'échantillonnage facilement identifiables pour nos études :

1. Un échantillonnage **purement aléatoire**, en excluant toutes les contraintes déjà annotées ou déduites, noté `samp.random.full` ;
2. Un échantillonnage **pseudo-aléatoire** de données issues d'un **même *cluster***, en excluant toutes les contraintes déjà annotées ou déduites, noté `samp.random.same` ;
3. Un échantillonnage des données issues d'un **même *cluster*** et étant **les plus éloignées** les unes des autres, noté `samp.farthest.same` (cf. FIGURE C.2) ;
4. Un échantillonnage des données issues de ***clusters* différents** et étant **les plus proches** les unes des autres, noté `samp.closest.diff` (cf. FIGURE C.2).

Une classe abstraite définit les prérequis des algorithmes implémentés (avoir une méthode `sample`) et une *factory* est disponible pour instancier rapidement un objet d'échantillonnage. Un exemple d'utilisation de ce module est consultable dans CODE C.5.

78. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/sampling/

Figure C.2 – *Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (`samp.farthest.same`), et de données issues de clusters différents et étant les plus proches les unes des autres (`samp.closest.diff`).*

```
# Import des dépendances.
from cognitivefactory.interactive_clustering.sampling.factory import
    sampling_factory

# Initialiser un objet d'échantillonnage.
sampler = sampling_factory(
  algorithm="random",
  random_seed=42,
)

# Run sampling.
selection = sampler.sample(
  constraints_manager=constraints_manager,
  nb_to_select=2,
  clustering_result=clustering_result,  # optionnel
  vectors=dict_of_vectors,  # optionnel
)
"""
[("0", '5"),  # "Comment signaler un vol de carte bancaire ?" vs "Le
    distributeur ne m'a pas rendu ma carte bleue."
 ("0", '2"),  # "Comment signaler un vol de carte bancaire ?" vs "J'
    ai perdu ma carte de paiement"
 ("2", 'N")]  # "J'ai perdu ma carte de paiement" vs "Pourquoi le
    sans contact ne fonctionne pas ?"
"""
```

Code C.5 – *Démonstration de notre implémentation de l'échantillonnage sur le jeu d'exemples.*

C.2 Implémentation de l'application web `cognitivefactory-interactive-clustering-gui`

La librairie `cognitivefactory-interactive-clustering-gui` [79] (Schild, 2022b) a été implémentée au cours de ce doctorat dans le but d'intégrer notre méthodologie de `Clustering Interactif` au sein d'une application web. Cette application dispose de plusieurs fonctionnalités telles que :

— la gestion du projet, de ses paramétrages et de ses données (cf. Figures C.4, C.5, C.7 et C.8) ;

— la gestion et l'annotation de contraintes, ainsi que la vérification des propriétés de transitivité (cf. Figures C.9, C.10 et C.11) ;

— la gestion des étapes d'une itération et de l'exécution asynchrone des divers algorithmes (cf. Figures C.5 et C.6) ;

— quelques scripts d'analyses.

Nous présentons succinctement cette application ci-dessous à l'aide de captures d'écrans.

Pour information : La documentation technique de cette librairie est accessible au lien suivant : https://cognitivefactory.github.io/interactive-clustering-gui/.

Notes de l'auteur : L'étude d'une interface graphique et de ses fonctionnalités a été l'objet d'un premier projet étudiant avec l'École d'Ingénieurs Télécom Physique Strasbourg (au cours de l'année 2021). Lors de nos échanges, une idée consistait à s'inspirer des fonctionnalités de l'application `TINDER`[80]pour *swipe left* (respectivement *swipe right*) l'annotation d'une contrainte `MUST-LINK` (respectivement d'une contrainte `CANNOT-LINK`). Bien qu'aucune version mobile de cette application n'ait été développée, une telle fonctionnalité pourrait être envisagée afin d'améliorer le confort de l'utilisateur. Nous pouvons toutefois noter qu'un reliquat de cette discussion a mené au choix du logo (proche de celui de l'application `TINDER`), ainsi qu'au design de la page d'annotation (cf. Figure C.10).

Notes de l'auteur : Suite aux diverses études menées au cours de ce doctorat, certaines pages sont en cours de développement, notamment :

— les pages d'analyses dont le but d'intégrer les conclusions du Chapitre 4 ;

— les pages de documentation pour intégrer les discussions du Chapitre 5.

79. https://pypi.org/project/cognitivefactory-interactive-clustering-gui/
80. https://tinder.com/fr

C.2.1 Accueil et Gestion de projets

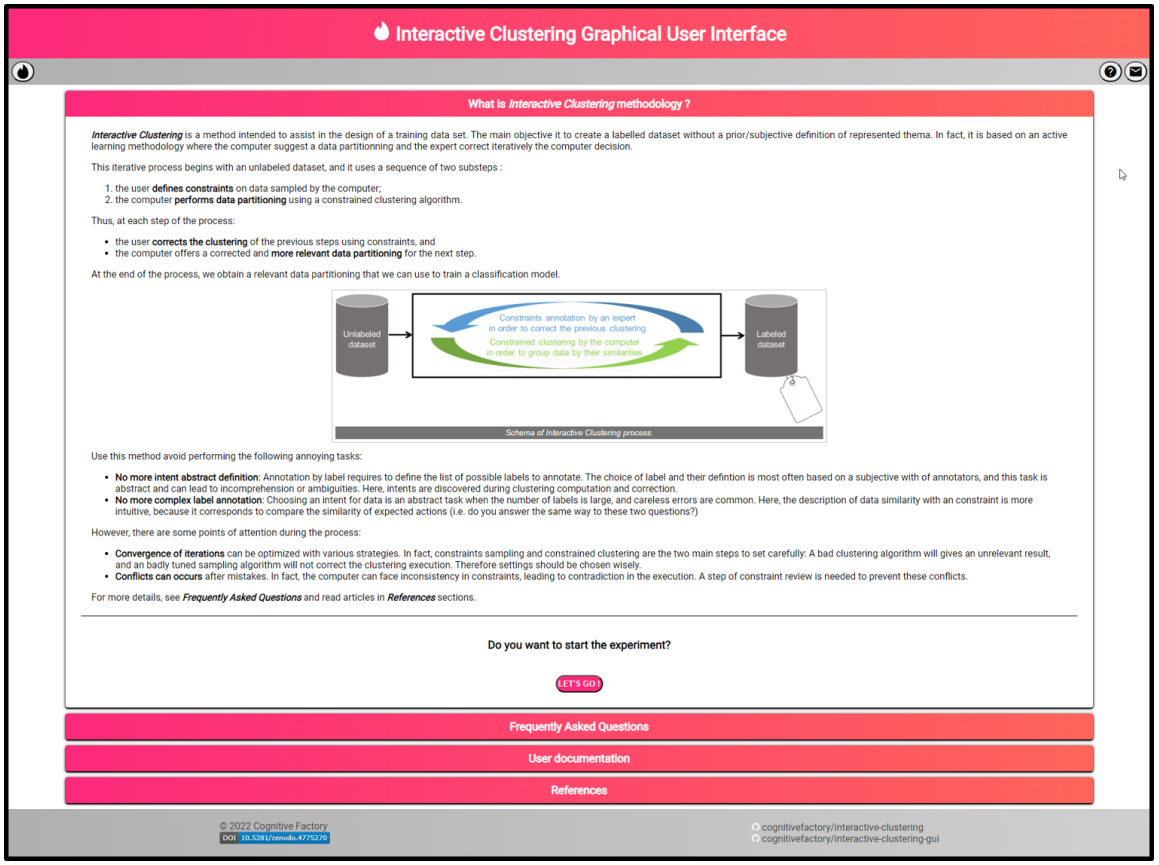

FIGURE C.3 – *Capture d'écran de l'application web implémentant notre méthodologie de Clustering Interactif : **page d'accueil de l'application**.*

Page d'accueil de l'application (FIGURE C.3). C'est la page de bienvenue de l'application. Nous y trouvons une description rapide de la méthode ainsi qu'une liste des questions fréquentes à son sujet. A terme, la documentation de la méthodologie d'annotation y sera intégrée (cf. discussions du CHAPITRE 5).

Concernant les boutons accessibles :

— Le bouton d'accueil en haut à gauche redirigera toujours sur cette page ;

— Le bouton de contact en haut à droite permet de contacter l'équipe de recherche ;

— Le bouton « `LET'S GO` » permet d'accéder à la page listant les projets d'annotation (cf. FIGURE C.4).

> **Pour information :** Dans toutes les pages suivantes, il est à noter que :
>
> — Tous les boutons peuvent être survolés pour afficher une courte description de leur action ou de leur état, ainsi que les raccourcis clavier qui permettent de les activer ;
>
> — Si besoin, tous les encadrés sont repliables pour gagner en visibilité.

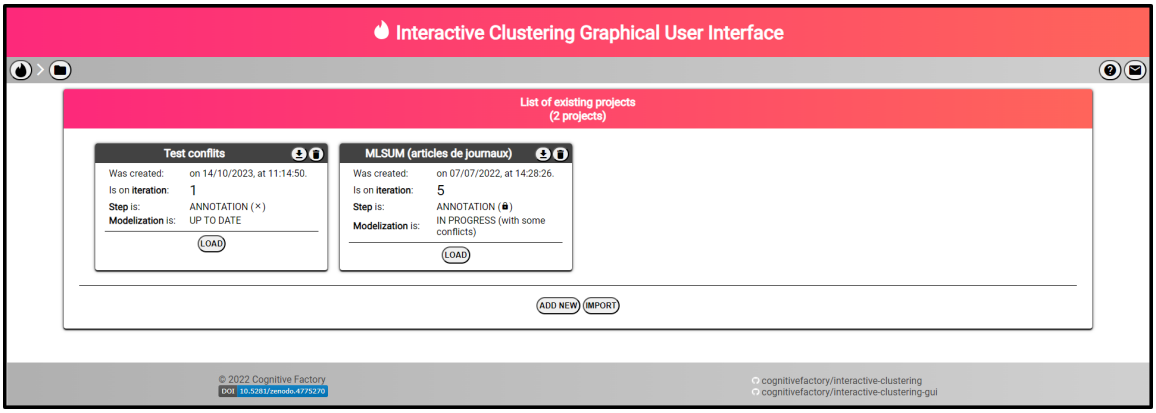

FIGURE C.4 – *Capture d'écran de l'application web implémentant notre méthodologie de Clustering Interactif : **page de gestion des projets**.*

Page de gestion des projets (FIGURE C.4). Cette page liste les projets existants sous la forme de tuiles contenant les informations importantes : nom, date de création, nombre d'itérations de la méthode, et l'état du projet (cf. FIGURE C.6).

Concernant les boutons d'action de cette page :

— Les boutons d'accueil en haut à gauche permettent de naviguer entre cette page et la page d'accueil de l'application (cf. FIGURE C.3) ;

— Il est possible de télécharger un projet au format `.zip` ou de le supprimer grâce aux boutons « 🡻 » et « 🗑 » en haut à droite de chaque tuile ;

— Pour créer un projet, le bouton « `ADD NEW` » ouvre un formulaire demandant le nom du projet et la liste des textes à annoter (fichier au format `.csv` avec séparateur '`;`') ;

— Il est aussi possible d'importer un projet contenu dans une archive `.zip` grâce au bouton « `IMPORT` » ;

— Enfin, le bouton « `LOAD` » mène à la page d'accueil du projet sélectionné (cf. FIGURE C.5).

C.2.2 Projet, Diagramme d'états et Paramétrages

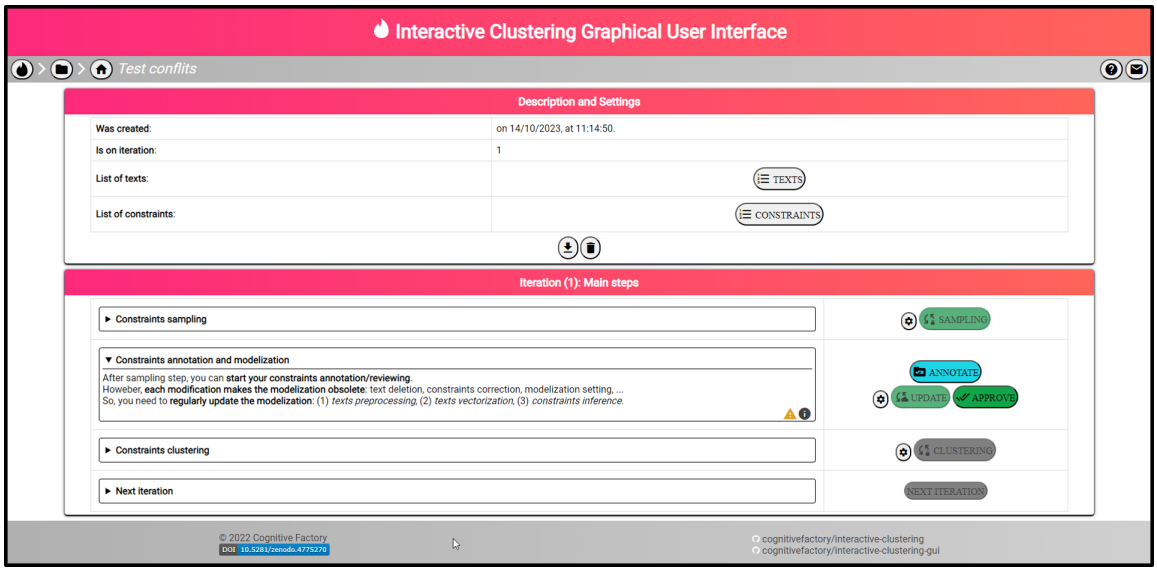

FIGURE C.5 – *Capture d'écran de l'application web implémentant notre méthodologie de `Clustering Interactif` : **page d'accueil du projet en cours**.*

Page d'accueil du projet en cours (FIGURE C.5). C'est la page principale de l'application. Elle contient en partie supérieure les informations du projet d'annotation en cours (*date de création, numéro d'itération, gestion des textes et des contraintes*), et en partie inférieure les étapes d'une itération de `Clustering Interactif` (*descriptions, boutons d'actions et de paramétrages*).

Concernant la gestion du projet (partie supérieure) :

- Les boutons d'accueil en haut à gauche permettent de naviguer entre cette page, la page de gestion des projets (cf. FIGURE C.4 et la page d'accueil de l'application (cf. FIGURE C.3) ;
- Au centre, il est possible de télécharger le projet au format `.zip` ou de le supprimer grâce aux boutons « 🡻 » et « 🗑 » ;
- Le bouton « `TEXTS` » mène vers la page d'inventaire et de gestion des textes du projet (cf. FIGURE C.8) ;
- Le bouton « `CONSTRAINTS` » mène vers la page d'inventaire et de gestion des contraintes annotées ou en cours d'annotation (cf. FIGURE C.9).

Concernant la gestion d'une itération de `Clustering Interactif` (partie inférieure), les différentes étapes sont représentées de bas en haut à l'aide d'éléments descriptifs repliables et de boutons d'actions. Nous retrouvons quatre étapes :

1. l'échantillonnage de contraintes, exécuté en tâche de fond grâce au bouton « `SAMPLING` », et dont les paramètres sont accessibles via le bouton « ⚙ » ;
2. l'annotation de contraintes, avec le bouton « `ANNOTATE` » qui redirige vers la prochaine contrainte à annoter. Cette étape contient aussi une gestion de la modélisation, c'est-à-dire une vérification des prétraitements et de la vectorisation des textes, ainsi qu'une vérification

de la cohérence des contraintes par l'absence de conflits d'annotation : le bouton « `UPDATE` » permet de recalculer cette modélisation en tâche de fond et le bouton « `APPROVE` » permet de la fixer jusqu'à la fin de l'itération en cours ;

3. le *clustering* sous contraintes, exécuté en tâche de fond grâce au bouton « `CLUSTERING` », et dont les paramètres sont accessibles via le bouton « ⚙ » ;
4. la confirmation du passage à la nouvelle itération, exécutée grâce au bouton « `NEXT ITERATION` ».

Il est à noter que :

— Les éléments de gauche sont repliables : au chargement de la page, seul l'élément de l'étape en cours est déplié ;

— La gestion de l'itération se fait à l'aide d'un diagramme d'état (cf. Figure C.6) : celui-ci se manifeste par un code couleur et l'activation/désactivation des boutons.

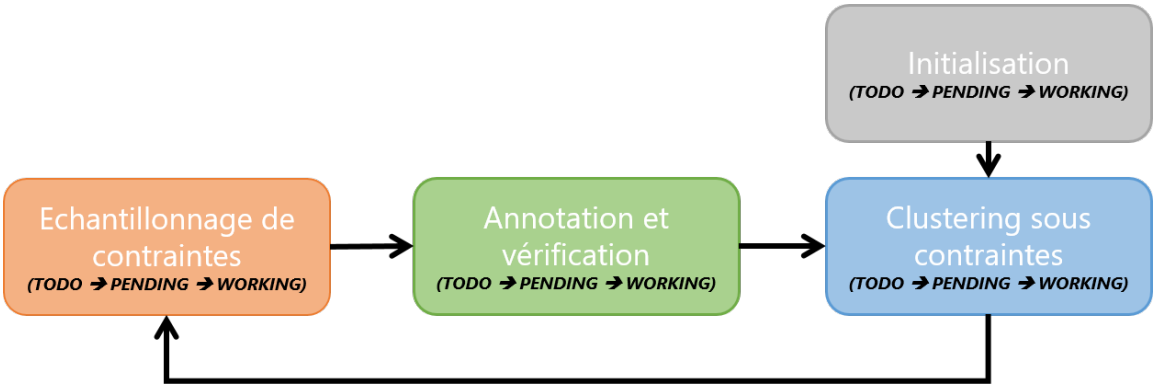

Figure C.6 – ***Diagramme d'états*** *simplifié de l'application web implémentant notre méthodologie de* `Clustering Interactif`.

Diagramme d'états de l'application et gestion des exécutions asynchrones (Figure C.6). Comme décrit en Section 3.2 et dans la Figure 3.1, une itération de *Clustering Interactif* contient trois étapes majeures : **(1)** l'échantillonnage de contraintes, **(2)** l'annotation de contraintes, et **(3)** le *clustering* sous contraintes. Ces étapes sont représentées par le diagramme d'état en Figure C.6 : ce dernier définit l'activation ou la désactivation des boutons d'action de l'application (cf. Figure C.5).

Afin de représenter l'état en cours et les actions possibles de manière pragmatique dans l'interface, un code couleur implicite est utilisé en plus de l'activation/désactivation des boutons :

— objet « grisé » et généralement désactivé : action inaccessible pour le moment ;

— objet en « vert » et activé : prochaine action à réaliser ;

— objet en « cyan » : action en cours de traitement ;

— objet en « rouge » et activé : action en erreur ou à recommencer ;

— objet en « vert grisé » et généralement désactivé : action réalisée avec succès.

D'autre part, comme certains algorithmes peuvent être lents, ces derniers sont exécutés en tâche de fond. La gestion d'état est alors affinée en quatre sous-états :

— « `TODO` » : l'action est à faire, la machine attend l'ordre de l'utilisateur ;

— « `PENDING` » : l'action a été ordonnée par l'utilisateur, mais elle n'a pas encore été prise en charge par la machine ;

— « `WORKING` » : l'action est en cours d'exécution en tâche de fond. Une barre d'avancement apparaît pour maintenir l'utilisateur informé de l'évolution de cet état ;

— *Note* : l'état « `DONE` » (action faite) n'existe pas réellement, elle est représentée par le fait que la prochaine étape ait un état « `TODO` ».

> ⚠ **Attention :** Pour une simplicité d'usage et afin d'offrir une démonstration rapide de notre méthodologie, nous avons décidé d'exécuter simplement les algorithmes en tâche de fond. Toutefois, pour favoriser les performances de l'application ainsi que sa sûreté pour une utilisation en production, nous conseillons de réimplémenter cette gestion des exécutions en privilégiant une architecture asynchrone utilisant des *workers* dédiés.

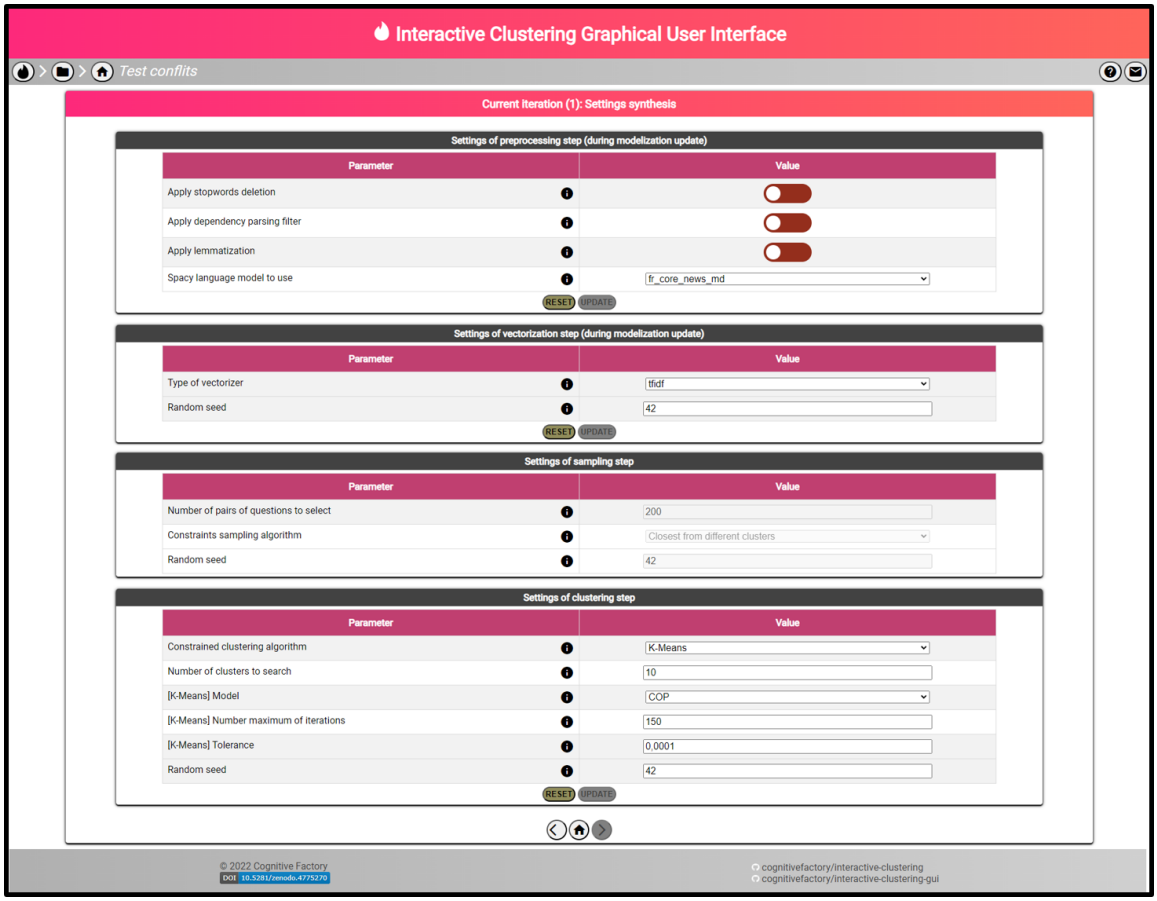

FIGURE C.7 – *Capture d'écran de l'application web implémentant notre méthodologie de `Clustering Interactif` : **page de gestion des paramètres**.*

Page de gestion des paramètres (Figure C.7). Accessible depuis les différents boutons « ⚙ », cette page liste les divers paramètres des algorithmes pour chaque itération.

— Chaque tuile représente une tâche (prétraitements, vectorisation, échantillonnage et *clustering*) : divers algorithmes et hyperparamètres son disponibles ;

— Les boutons « `UPDATE` » permettent de valider les changements, les boutons « `RESET` » rétablissent les paramètres par défaut ;

— Ces différents formulaires sont modifiables tant que l'étape n'est pas en cours d'exécution, sinon ils sont juste consultables ;

— En bas de page, il est possible de changer d'itération pour consulter les paramètres des itérations précédentes (boutons « ❮ » et « ❯ »), et de revenir vers la page d'accueil du projet (bouton « 🏠 », cf. Figure C.5).

C.2.3 Textes et Contraintes

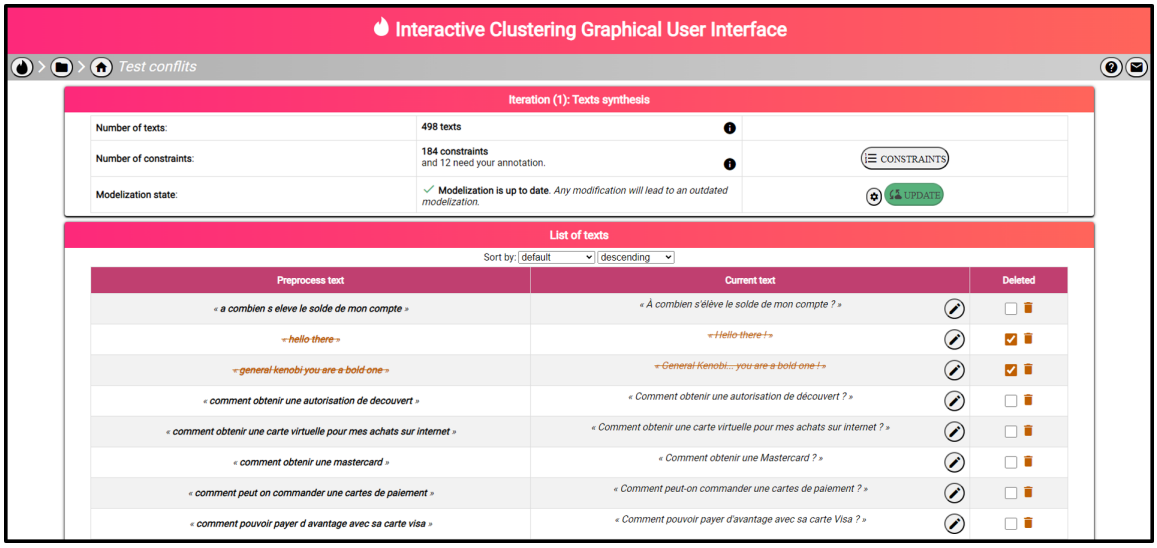

FIGURE C.8 – *Capture d'écran de l'application web implémentant notre méthodologie de `Clustering Interactif` : **page d'inventaire des textes**.*

Page d'inventaire des textes (FIGURE C.8). Cette page permet de lister les textes du projet à annoter. La page est divisée en deux : la partie supérieure donne des informations générales (*nombre de textes, nombre de contraintes à annoter, rappel de la modélisation en cours*), et la partie inférieure liste les textes dans un tableau.

Concernant les informations générales (partie supérieure) :

— Le bouton « `UPDATE` » permet de mettre à jour la modélisation lorsque des contraintes ont été ajoutées, des paramètres de prétraitements ou de vectorisation ont été mis à jour, ou si des textes ont été modifiés : cette action est exécutée en tâche de fond. La couleur de e bouton est définie par le diagramme d'état (cf. FIGURE C.6) ;

— Le bouton « `CONSTRAINTS` » mène vers la page d'inventaire et de gestion des contraintes annotées ou en cours d'annotation (cf. FIGURE C.9).

Concernant le tableau listant les textes (partie inférieure) :

— Le texte brut et sa version prétraitée sont affichés ;

— Grâce au bouton « ✎ », il est possible de corriger un texte s'il contient une faute de frappe ;

— Grâce au bouton « 🗑 », il est possible de supprimer (*ne plus prendre en compte*) un texte s'il n'est pas pertinent pour le projet : celui-ci est alors rayé en orange ;

— En haut du tableau, il est possible de trier les textes suivant différents critères (*ordre alphabétique, supprimé ou non, ...*) ;

— **Attention** : Toute action de modification (renommage, suppression) nécessite de mettre à jour la modélisation par la suite. De plus, ces actions sont désactivées si le projet n'est pas à l'étape d'annotation (cf. diagramme d'états en FIGURE C.6) ;

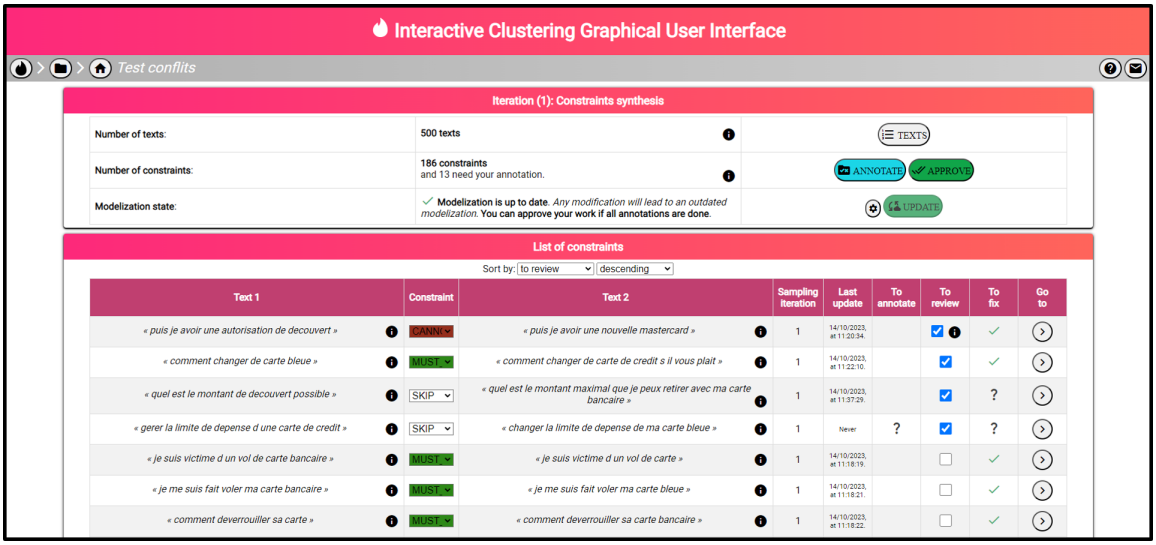

FIGURE C.9 – *Capture d'écran de l'application web implémentant notre méthodologie de Clustering Interactif : **page d'inventaire des contraintes**.*

Page d'inventaire des contraintes (FIGURE C.9). Cette page permet de lister les contraintes du projet à annoter. La page est divisée en deux : la partie supérieure donne des informations générales (*nombre de textes, nombre des contraintes à annoter, rappel de la modélisation en cours*), et la partie inférieure liste les contraintes dans un tableau.

Concernant les informations générales (partie supérieure) :

— Le bouton « `ANNOTATE` » redirige vers la prochaine contrainte à annoter (s'il en reste) ;

— Le bouton « `UPDATE` » permet de mettre à jour la modélisation si des contraintes ont été ajoutées, des paramètres de prétraitements ou de vectorisation ont été mis à jour, ou si des textes ont été modifiés : cette action est exécutée en tâche de fond. La couleur de ce bouton est définie par le diagramme d'état (cf. FIGURE C.6) ;

— Le bouton « `TEXTS` » mène vers la page d'inventaire et de gestion des données du projet (cf. FIGURE C.8).

Concernant le tableau listant les contraintes (partie inférieure) :

— Les deux textes d'une même contrainte sont affichés de part et d'autre de la valeur annotée : `MUST-LINK`, `CANNOT-LINK` ou `SKIP` (*pour une contrainte non annotée ou temporairement ignorée*) ;

— Il est possible de marquer une contrainte pour la revoir plus tard grâce à la coche « ☑ » ;

— Le bouton « › » à droite permet d'accéder à la page d'annotation de cette contrainte (cf. FIGURE C.10) ;

— Diverses informations sont disponibles à la droite du tableau : l'itération à laquelle la contrainte a été échantillonnée, sa dernière date de modification, son besoin d'annotation (« ? » *pour une contrainte encore jamais annotée*), et la présence ou non de conflits (« ✔ » *ou* « ! ») ;

— En haut du tableau, il est possible de trier les contraintes suivant différents critères (*ordre*

alphabétique, valeur d'annotation, date d'échantillonnage ou de modification, présence de conflit, ...) ;

— **Attention** : Toute action de modification de la valeur d'annotation nécessite de mettre à jour la modélisation par la suite. De plus, cette action est désactivée si le projet n'est pas à l'étape d'annotation (cf. diagramme d'états en FIGURE C.6) ;

— *Note* : Si une contrainte concerne au moins un texte qui a été supprimé (cf. FIGURE C.8), la contrainte n'apparaît pas dans ce tableau mais existe toujours dans l'application (*elle n'est plus prise en compte*).

C.2.4 Annotation et Conflits

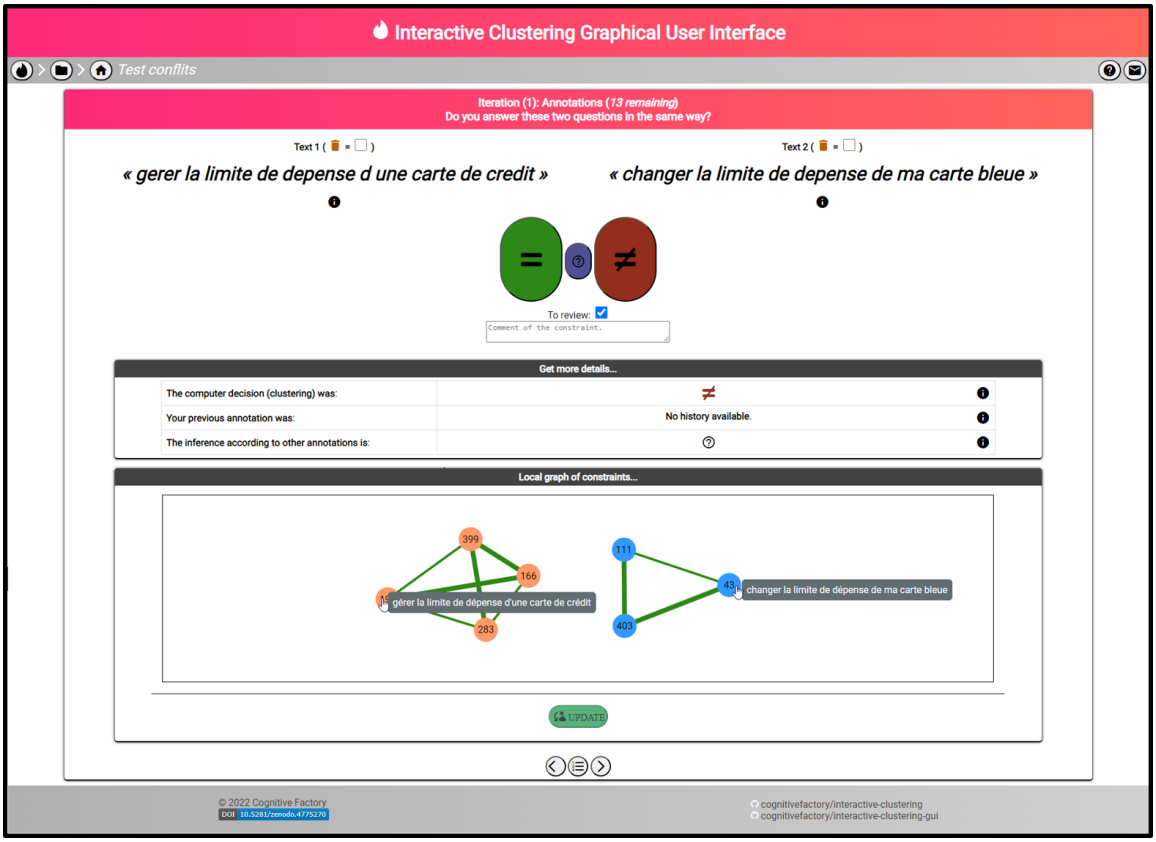

FIGURE C.10 – *Capture d'écran de l'application web implémentant notre méthodologie de Clustering Interactif : **page d'annotation d'une contrainte**.*

Page d'annotation d'une contrainte (FIGURE C.10 et FIGURE C.11). Cette page est le coeur de cette application d'annotation : la partie supérieure permet d'annoter une contrainte entre deux textes, les parties inférieures sont des détails (éléments repliés par défaut).

Concernant l'annotation de la contrainte (partie supérieure) :

— Les deux textes de la contrainte sont affichés en haut du bloc d'annotation ;

— Les deux boutons principaux d'annotation sont le « = » pour un `MUST-LINK` et le « ≠ » pour un `CANNOT-LINK`. Les raccourcis claviers de ces boutons sont respectivement « `A` » (*accept*) et « `R` » (*reject*). Il est aussi possible d'ignorer la contrainte avec le bouton « ? » (raccourci avec la barre espace) ;

— Si c'est la première fois que cette contrainte est annotée, la prochaine contrainte est automatiquement chargée lors d'un choix d'annotation (« = », « ≠ » ou « ? »). Sinon, une confirmation est demandée pour valider le changement de la valeur de la contrainte ;

— Une gestion de revue d'annotation est possible grâce à un champ de commentaire, et « ☑ » permet de marquer la contrainte pour la revoir plus tard ;

— Grâce à l'icône info-bulle, il est possible d'afficher la version non prétraitée de chaque texte De plus, grâce à la coche « 🗑 », il est possible de supprimer (*ne plus prendre en compte*)

un texte non pertinent pour le projet : la contrainte sera alors masquée de la liste des contraintes ;

— **Attention** : Toute action de modification de la valeur d'annotation nécessite de mettre à jour la modélisation par la suite. De plus, cette action est désactivée si le projet n'est pas à l'étape d'annotation (cf. diagramme d'états en FIGURE C.6) ;

Concernant les détails sur la contrainte (partie inférieure) :

— L'encadré du milieu donne quelques informations sur l'annotation : ce qu'a effectué la machine lors de la précédente étape de *clustering* (même *cluster* ou différents *clusters* ?), l'historique de ce que l'annotateur a renseigné au préalable, et la déduction faite par le gestionnaire de contraintes grâce aux propriétés de transitivité ;

— L'encadré du bas propose une visualisation du graphe de contraintes à l'aide de la librairie `d3js`[81] ;

— Les boutons en bas de page permettent de naviguer entre les pages d'annotation (boutons « ❮ » et « ❯ ») ou à revenir vers la page d'inventaire des contraintes (bouton « ☰ », cf. FIGURE C.9).

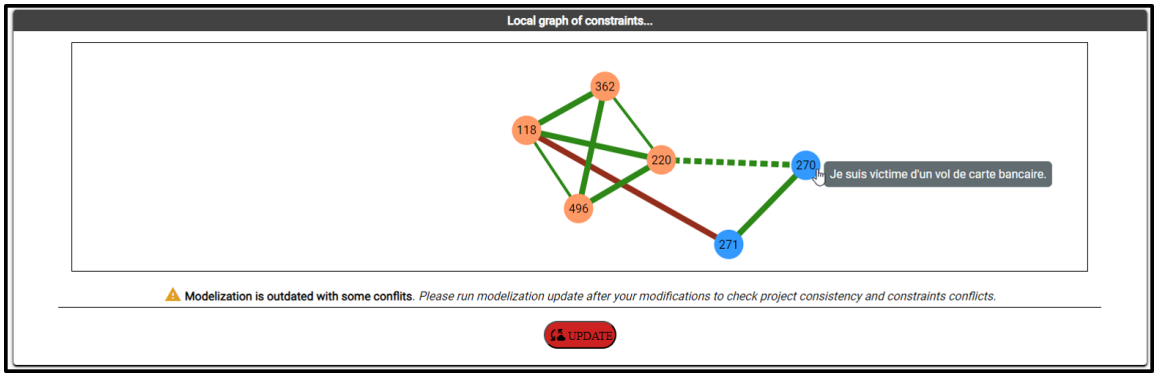

FIGURE C.11 – *Capture d'écran de l'application web implémentant notre méthodologie de Clustering Interactif : **graphe de contraintes présentant un conflit d'annotation.***

Concernant le graphe local de contraintes :

— Les cercles représentent les données : chaque numéro correspond à un identifiant, et leur textes respectifs sont visibles par un survol de souris. Ces cercles peuvent être déplacés pour une meilleur visibilité ;

— Les liens entre les cercles représentent les contraintes : en **vert** pour les `MUST-LINK`, en **rouge** pour les `CANNOT-LINK`, en **gras** pour les contraintes annotées, en écriture *fine* pour les contraintes déduites par transitivité, et en `pointillés` pour les conflits détectés. Un clic de souris sur un lien redirige vers la page d'annotation de la contrainte associée (si elle existe) ;

— Comme il peut y avoir un nombre important de contraintes dans un projet, ce graphe ne représente que la partie des contraintes impliquées dans l'annotation des deux textes de

81. https://d3js.org/

cette page : nous retrouvons ainsi les deux textes en cours d'annotation et leurs composants connexes respectifs (voir SECTION C.1.2) ;

— Dans l'exemple en FIGURE C.11, nous pouvons voir le conflit suivant : **(1)** 118 et 270 doivent être séparés car 118 est différent de 271 qui est similaire à 270, mais **(2)** 118 et 270 doivent être rapprochés car 118 est similaire à 220 qui est similaire à 270...

— **Attention** : En cas de conflit, plusieurs boutons deviennent **rouges**, et l'approbation de la modélisation est désactivée tant que le conflit n'est pas résolu.

C.3 Implémentation de l'application web `cognitivefactory-features-maximization-metric`

La librairie `cognitivefactory-features-maximization-metric`[82] (Schild, 2023) a été implémentée au cours de ce doctorat pour pouvoir utiliser la `Maximisation des traits` (*Features Maximization* notée `FMC`). Cette technique, proposée par Lamirel et al., 2017, permet de sélectionner les composantes vectorielles pertinentes (*features*) d'une base d'apprentissage. Nous allons détailler cette librairie comme suit :

— le calcul du score de *Features F-Measure* associé à cette méthode (cf. Section C.3.1) ;

— la sélection des composantes vectorielles pertinentes à l'aide du score de *Features F-Measure* (cf. Section C.3.2) ;

— l'activation des composantes vectorielles pertinentes pour chaque classe de la base d'apprentissage (cf. Section C.3.3) ;

— l'application de cette méthode à l'analyse des patterns linguistiques pertinents d'une base d'apprentissage utilisée pour de la classification de textes en intentions (cf. Section C.3.4).

> **Pour information :** La documentation technique de cette librairie est accessible au lien suivant : https://cognitivefactory.github.io/features-maximization-metric/.

Pour la suite de l'exposé, nous allons utiliser les notations suivantes :

— $\mathcal{D}$ représente l'ensemble des données d de la base d'apprentissage, et $\texttt{vecteur}[d]$ représente la description vectorielle d'une donnée $d \in \mathcal{D}$;

— $\mathcal{F}$ représente l'ensemble des composantes vectorielles f (*features*), et $\texttt{vecteur}[d][f]$ représente le poids de la composante $f \in \mathcal{F}$ du vecteur décrivant la donnée $d \in \mathcal{D}$;

— $\mathcal{C}$ représente l'ensemble des classes c possibles de la base d'apprentissage, et $\texttt{classe}[d]$ représente la classe d'une donnée $d \in \mathcal{D}$;

C.3.1 Calcul du score de *Features F-Measure*

D'abord, il faut calculer le score de *Features F-Measure* (`FM`) à partir de deux termes : la *Features Recall* (`FR`) et la *Features Predominance* (`FP`).

La ***Features Recall*** d'une composante vectorielle $f \in \mathcal{F}$ et d'une classe $c \in \mathcal{C}$, notée $\texttt{FR}[f][c]$, est un score compris entre 0 et 1 qui est obtenu par le ratio entre :

— la somme des poids des vecteurs pour la composante f et pour les données de la classe c, et

— la somme des poids des vecteurs pour la composante f et pour toutes les données.

82. https://pypi.org/project/cognitivefactory-features-maximization-metric/

$$\texttt{FR}[f][c] \;=\; \frac{\sum\limits_{\substack{d \in \mathcal{D} \\ \texttt{classe}[d]=c}} \texttt{vector}[d][f]}{\sum\limits_{c' \in \mathcal{C}} \sum\limits_{\substack{d' \in \mathcal{D} \\ \texttt{classe}[d']=c'}} \texttt{vector}[d'][f]} \tag{C.3}$$

> **Notes de l'auteur :** Ce score répond à la question : « *est-ce que la feature f permet de distinguer la classe c des autres classes c' ?* »

La ***Features Predominance*** d'une composante vectorielle $f \in \mathcal{F}$ et d'une classe $c \in \mathcal{C}$, notée $\texttt{FP}[f][c]$, est un score compris entre 0 et 1 qui est obtenu par le ratio entre :

— la somme des poids des vecteurs pour la composante f et pour les données de la classe c, et

— la somme des poids des vecteurs pour toutes les composantes et pour les données de la classe c.

$$\texttt{FP}[f][c] \;=\; \frac{\sum\limits_{\substack{d \in \mathcal{D} \\ \texttt{classe}[d]=c}} \texttt{vector}[d][f]}{\sum\limits_{f' \in \mathcal{F}} \sum\limits_{\substack{d \in \mathcal{D} \\ \texttt{classe}[d]=c}} \texttt{vector}[d][f']} \tag{C.4}$$

> **Notes de l'auteur :** Ce score répond à la question : « *est-ce que la composante f identifie mieux la classe c que les autres composantes f' ?* »

La ***Features F-Measure*** d'une composante vectorielle $f \in \mathcal{F}$ et d'une classe $c \in \mathcal{C}$, notée $\texttt{FM}[f][c]$, est un score entre 0 et 1 qui est obtenu par la moyenne harmonique entre la *Features Recall* et la *Features Predominance*.

$$\texttt{FM}[f][c] \;=\; 2 \cdot \frac{\texttt{FR}[f][c] \cdot \texttt{FP}[f][c]}{\texttt{FR}[f][c] + \texttt{FP}[f][c]} \tag{C.5}$$

> **Notes de l'auteur :** Ce score répond à la question : « *combien d'informations contient la feature f au sujet de la classe c ?* »

C.3.2 Sélection de *features* à l'aide de la `F-Measure`

L'objectif de cette seconde étape est de supprimer les composantes vectorielles qui n'apportent pas d'information de manière générale. Pour cela, un seuil de sélection est défini grâce à la moyenne globale des scores de *Features F-Measure*.

La ***Features Overall Average***, notée $\overline{\overline{\texttt{FM}}}$, est un score entre 0 et 1 qui est obtenu par la moyenne de la `Features F-Measure` pour toutes les composantes et pour toutes les classes.

$$\overline{\overline{\texttt{FM}}} = \frac{\sum_{f \in \mathcal{F}} \sum_{c \in \mathcal{C}} \texttt{FM}[f][c]}{|\mathcal{F}| \cdot |\mathcal{C}|} \tag{C.6}$$

> **Notes de l'auteur :** Ce seuil répond à la question : « *quelle est la moyenne d'information contenue dans cette représentation vectorielle ?* »

La **sélection de *features*** se fait en comparant les valeurs de `Features F-Measures` à cette moyenne globale : si une composante $f \in \mathcal{F}$ a un score FM[f][c'] supérieur à la moyenne $\overline{\overline{\texttt{FM}}}$ pour au moins une classe $c' \in \mathcal{C}$, alors la composante f est sélectionnée ; sinon, elle est supprimée.

$$\begin{aligned} \mathcal{F}_{\texttt{SELECTED}} &:= \left\{ f \in \mathcal{F} \;\middle|\; (\exists c' \in \mathcal{C})\ \texttt{FM}[f][c'] \geq \overline{\overline{\texttt{FM}}} \right\} \\ \mathcal{F}_{\texttt{DELETED}} &:= \left\{ f \in \mathcal{F} \;\middle|\; (\forall c' \in \mathcal{C})\ \texttt{FM}[f][c'] < \overline{\overline{\texttt{FM}}} \right\} \end{aligned} \tag{C.7}$$

> **Notes de l'auteur :** Cette sélection répond à la question : « *quelles sont les features qui apportent plus d'information que la moyenne d'information contenue dans cette représentation vectorielle ?* »

C.3.3 Activation des *features* à l'aide de la `F-Measure`

L'objectif de cette dernière étape est de vérifier l'activation des composantes vectorielles pour chaque classe $c \in \mathcal{C}$. Pour cela, un seuil d'activation est défini pour chaque *feature* $f \in \mathcal{F}$ grâce à la moyenne locale des scores de *Features F-Measure*.

La ***Features Marginal Average*** d'une composante sélectionnée $f \in \mathcal{F}_{\texttt{SELECTED}}$, notée $\overline{\texttt{FM}[f]}$, est un score entre 0 et 1 qui est obtenu par la moyenne locale de la `Features F-Measure` pour la composante f et pour toutes les classes.

$$\overline{\texttt{FM}[f]} = \frac{\sum_{c' \in \mathcal{C}} \texttt{FM}[f][c']}{|\mathcal{F}_{\texttt{SELECTED}}|} \tag{C.8}$$

> **Notes de l'auteur :** Ce seuil répond à la question : « *quelle est la moyenne d'information contenue par la feature f dans cette représentation vectorielle ?* »

L'**activation de *features*** se fait en comparant les valeurs de `Features F-Measures` à cette moyenne locale : si une composante sélectionnée $f \in \mathcal{F}_{\texttt{SELECTED}}$ a un score FM[f][c] supérieur à la moyenne locale $\overline{\texttt{FM}[f]}$ pour une classe $c \in \mathcal{C}$, alors cette composante f est activée pour cette classe c.

$$\begin{cases} f \text{ active } c & \text{si } \texttt{FM}[f][c] \geq \overline{\texttt{FM}[f]} \\ f \text{ n'active pas } c & \text{sinon} \end{cases} \tag{C.9}$$

> **Notes de l'auteur :** Cette activation répond à la question : « *pour quelle(s) classe(s) la feature f est pertinente ?* »

C.3.4 Application à l'analyse de la classification de textes

Plaçons-nous dans le cas d'une classification de textes en intentions :

— $\mathcal{D}$ représente l'ensemble des textes de la base d'apprentissage ;

— $\mathcal{F}$ représente l'ensemble des composantes vectorielles permettant de décrire les textes ;

— $\mathcal{C}$ représente l'ensemble des intentions possibles de la base d'apprentissage.

Si nous utilisons une représentation vectorielle basée sur une description statistique du vocabulaire (comme `Bag of Words` (Harris, 1954) ou `TF-IDF` (Ramos, 2003)), nous pouvons alors déduire quels sont les mots du vocabulaire qui décrivent le mieux chaque intention grâce à la `Maximisation des traits`. Il est possible de compléter l'analyse en considérant que :

— un pattern linguistique f est caractéristique d'une classe c s'il ne s'active que pour cette classe. La liste des patterns linguistiques représentatifs d'une classe c se notera $\mathcal{F}_{\texttt{ACTIVATED}}[c]$;

— un pattern linguistique f est ambigu s'il s'active pour plusieurs classes.

$$\mathcal{F}_{\texttt{ACTIVATED}}[c] \ := \ \left\{ \ f \in \mathcal{F}_{\texttt{SELECTED}} \ \middle| \ \texttt{FM}[f][c] \geq \overline{\texttt{FM}[f]} \ \right\} \tag{C.10}$$

> **Notes de l'auteur :** Nous utilisons cette technique dans la Section 4.4.2 pour déterminer la pertinence sémantique d'un *cluster* :
>
> — en vérifiant la valeur métier du vocabulaire caractéristique du *cluster* ;
>
> — en vérifiant que les textes de ce *cluster* contiennent peu de mots des vocabulaires caractéristiques des autres *clusters*.

Annexe D

Évaluation d'un *clustering* à l'aide de la `v-measure`

Au cours de nos études, nous avons besoin de comparer nos résultats de *clustering* à une vérité terrain représentant la cible à atteindre au cours des itérations. Nous utilisons alors la `v-measure` (Rosenberg et Hirschberg, 2007) pour discuter de la proximité du résultat d'une itération avec sa cible.

> **Notes de l'auteur :** Dans cette annexe, nous allons décrire la `v-measure` de manière formelle puis détailler son fonctionnement grâce à quelques exemples. Si la première section est très (*voir trop*) abstraite, consulter directement la deuxième section : les exemples permettront d'illustrer les intuitions principales.

Sommaire

D.1 Définition de la `v-measure`

Pour ces définitions, nous allons utiliser les notations suivantes :

— $\mathcal{D}$ représente l'ensemble des données d à segmenter ;

— $\mathcal{C}$ représente l'ensemble des classes c de la vérité terrain (*référence à atteindre*) ;

— $\mathcal{K}$ représente l'ensemble des *clusters* k de la segmentation (*clustering proposé par la machine pour une itération donnée*).

La `v-measure` entre un *clustering* $\mathcal{K}$ et la vérité terrain $\mathcal{C}$ (servant de référence) est un score entropique calculé à partir de deux termes : l'homogénéité (*homogeneity*) et la complétude (*completeness*). Pour calculer ces termes, nous avons besoin d'introduire les concepts d'entropie intrinsèque et d'entropie conditionnelle.

L'**entropie intrinsèque** d'une segmentation $\mathcal{X}$ des données $\mathcal{D}$, notée $H(\mathcal{X})$, est définie comme l'opposé de la somme des termes $p(x) \cdot log\ p(x)$, où $p(x)$ représente la proportion d'éléments de $\mathcal{D}$ contenus dans le groupe $x \in \mathcal{X}$.

$$H(\mathcal{X}) \ = \ -\sum_{x \in \mathcal{X}} \frac{|x|}{|\mathcal{D}|} \cdot \texttt{log}\Big(\frac{|x|}{|\mathcal{D}|}\Big) \tag{D.1}$$

L'**entropie conditionnelle** d'une segmentation $\mathcal{X}$ par rapport à une autre segmentation $\mathcal{Y}$, notée $H(\mathcal{X}|\mathcal{Y})$, est un score compris entre 0 et $H(\mathcal{X})$, qui est obtenu par l'opposé des sommes des termes $p(x \cap y) \cdot log\ p(x \cap y|y)$, où $p(x \cap y)$ représente la proportion d'éléments de $\mathcal{D}$ contenus dans l'intersection $x \cap y$, et $p(x \cap y|y)$ représente la proportion d'éléments de y contenus dans l'intersection $x \cap y$.

$$H(\mathcal{X}|\mathcal{Y}) \ = \ -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \frac{|x \cap y|}{|\mathcal{D}|} \cdot \texttt{log}\Big(\frac{|x \cap y|}{|y|}\Big) \tag{D.2}$$

> **Notes de l'auteur :** La notion d'entropie est difficile à appréhender, mais nous proposons toutefois les pistes d'interprétation suivantes :
>
> — l'entropie intrinsèque $H(\mathcal{X})$ représente le "**désordre**" de la segmentation $\mathcal{X}$: en effet, si les données seront réparties dans un grand nombre de groupes ou dans des groupes déséquilibrés, alors il y aura plus de termes $p(x) \cdot logp(x)$, donc l'entropie sera plus grande ;
>
> — l'entropie conditionnelle $H(\mathcal{X}|\mathcal{Y})$ représente à quel point une segmentation $\mathcal{X}$ **ne respecte pas une autre segmentation** $\mathcal{Y}$: en effet, si $\mathcal{X}$ respecte $\mathcal{Y}$, alors les intersections non nulles entre $\mathcal{X}$ et $\mathcal{Y}$ ressembleront davantage aux groupes de $\mathcal{Y}$ et les termes $log\ p(x \cap y|y)$ seront davantage nuls, donc l'entropie conditionnelle sera plus faible.

L'**homogénéité** d'un *clustering* $\mathcal{K}$ par rapport à la vérité terrain $\mathcal{C}$ (servant de référence), notée $\texttt{homogeneity}(C|K)$, est un score entre 0 et 1, qui est obtenu par 1 moins le ratio entre :

— l'entropie conditionnelle de la vérité terrain $\mathcal{C}$ par rapport au *clustering* $\mathcal{K}$, et

— l'entropie intrinsèque de la vérité terrain $\mathcal{C}$.

$$\texttt{homogeneity}(\mathcal{C}|\mathcal{K}) \;=\; 1 - \frac{H(\mathcal{C}|\mathcal{K})}{H(\mathcal{C})} \tag{D.3}$$

La **complétude** d'un *clustering* $\mathcal{K}$ par rapport à la vérité terrain $\mathcal{C}$ (servant de référence), notée $\texttt{completeness}(C|K)$, est un score entre 0 et 1, qui est obtenu par 1 moins le ratio entre :

— l'entropie conditionnelle du *clustering* $\mathcal{K}$ par rapport à la vérité terrain $\mathcal{C}$, et

— l'entropie intrinsèque du *clustering* $\mathcal{K}$.

$$\texttt{completeness}(\mathcal{C}|\mathcal{K}) \;=\; 1 - \frac{H(\mathcal{K}|\mathcal{C})}{H(\mathcal{K})} \tag{D.4}$$

Notes de l'auteur : Nous pouvons interpréter ces termes de la manière suivante :

— l'homogénéité $\texttt{homogeneity}(\mathcal{C}|\mathcal{K})$ représente à quel point **chaque *cluster* k contient uniquement des données provenant d'une même et une seule classe** c : en effet, l'homogénéité est maximale lorsque l'entropie conditionnelle $H(\mathcal{C}|\mathcal{K})$ est minimale, et cette entropie conditionnelle est nulle si chaque *cluster* k ne contient que des données d'une même classe c ($log\frac{|k|}{|k|} = log(1) = 0$) ;

— la complétude $\texttt{completeness}(\mathcal{C}|\mathcal{K})$ représente à quel point **chaque classe c est contenue entièrement dans un même et unique *cluster* k** : en effet, la complétude est maximale lorsque l'entropie conditionnelle $H(\mathcal{K}|\mathcal{C})$ est minimale, et cette entropie conditionnelle est nulle si chaque classe c ne contient que des données d'un même *cluster* k ;

— ces termes sont liés : $\texttt{homogeneity}(\mathcal{C}|\mathcal{K}) = \texttt{completeness}(\mathcal{K}|\mathcal{C})$ ($log\frac{|c|}{|c|} = log(1) = 0$).

La **V-Measure** d'un *clustering* $\mathcal{K}$ par rapport à la vérité terrain $\mathcal{C}$ (servant de référence), notée $\texttt{v-measure}(C|K)$, est un score entre 0 et 1, qui est obtenu par la moyenne harmonique entre l'homogénéité et la complétude.

$$\texttt{v-measure}(C|K) \;=\; 2 \cdot \frac{\texttt{homogeneity}(\mathcal{C}|\mathcal{K}) \cdot \texttt{completeness}(\mathcal{C}|\mathcal{K})}{\texttt{homogeneity}(\mathcal{C}|\mathcal{K}) + \texttt{completeness}(\mathcal{C}|\mathcal{K})} \tag{D.5}$$

Notes de l'auteur : Ce score de `v-measure` est le compromis entre l'homogénéité et la complétude. Pour être maximal, il faut que chaque *cluster* ne contienne que des données d'une même classe et qu'en échange chaque classe ne soit contenue que dans un seul *cluster*. Autrement dit : la `v-measure` est maximale si chaque *cluster* correspond a une classe et si chaque classe correspond a un *cluster*.

Ce score est symétrique : $\texttt{v-measure}(C|K) = \texttt{v-measure}(K|C)$.

D.2 Quelques exemples de calcul avec la `v-measure`

Prenons quelques exemples pour illustrer les calculs d'homogénéité, de complétude et de `v-measure`. Pour cela, nous considérons le jeu d'exemples de 10 objets, représentés en FIGURE D.1 **(1)**, que nous voulons regrouper par forme (`carrés`, `ronds`, `triangles`), comme dans la FIGURE D.1 **(2)**.

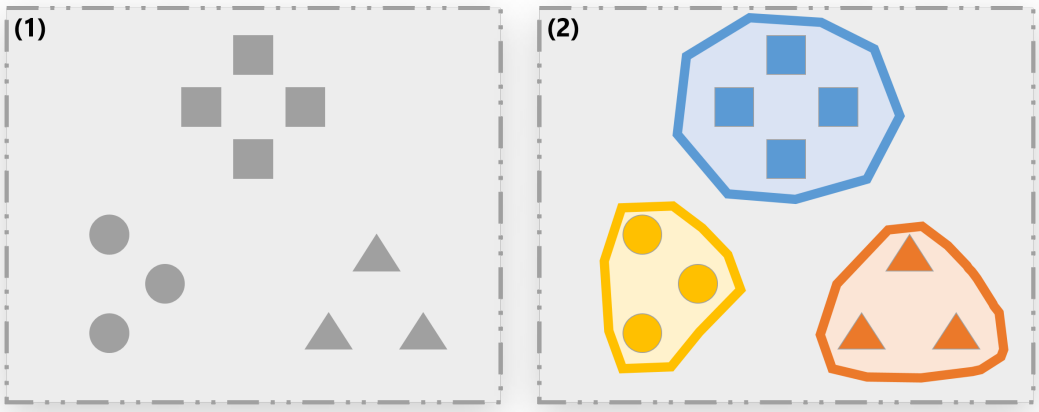

FIGURE D.1 – *Présentation du jeu d'exemples : **(1)** représente 10 données non regroupées ; (2) représente ces 10 données regroupées en 3 clusters en fonction de formes : `carrés`, `rondse t triangles`.*

La **segmentation parfaite** de ces objets suivant leur forme est visible en FIGURE D.1 **(2)**. En effet, nous pouvons voir que :

— ce *clustering* est **homogène** (*homogeneity* = 1.00) : il n'y a que des `carrés` dans le *cluster* **bleu**, il n'y a que des `ronds` dans le *cluster* **jaune** et il n'y a que des `triangles` dans le *cluster* **orange** ;

— ce *clustering* est **complet** (*completeness* = 1.00) : tous les `carrés` sont dans le *cluster* **bleu**, tous les `ronds` sont dans le *cluster* **jaune** et tous les `triangles` sont dans le *cluster* **orange** ;

— le *clustering* **bleu**/**jaune**/**orange** correspond donc parfaitement à sa vérité terrain `carrés`/`ronds`/`triangles` (`v-measure` = 1.00).

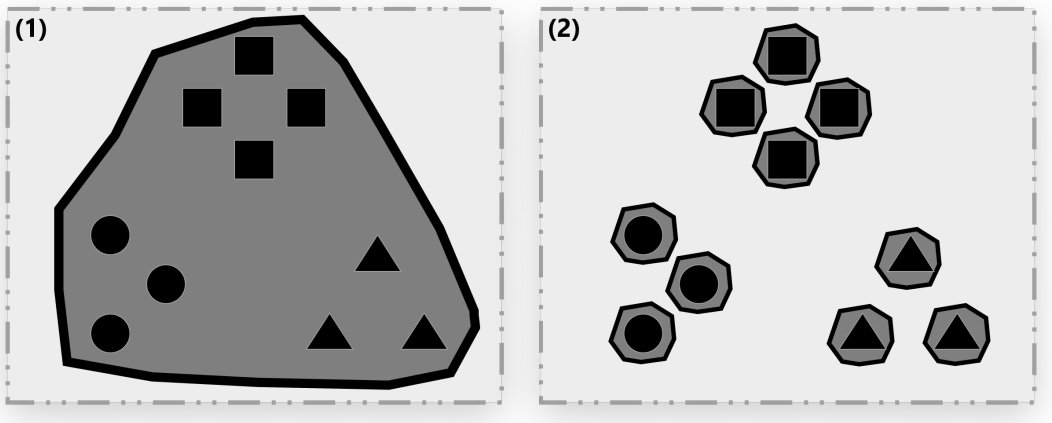

FIGURE D.2 – *Exemples de* `clustering` *avec des cas extrêmes : (1) représente un regroupement en un unique cluster contenant toutes les données ; (2) représente un regroupement en* 10 *clusters où chaque cluster contient une seule donnée.*

Prenons maintenant les deux cas extrêmes de la FIGURE D.2.

— **(1)** représente une segmentation où toutes les données sont regroupées dans un seul *cluster* :

  — ce *clustering* est **complet** (*completeness* = 1.00) : tous les `carrés` sont dans cet unique *cluster*, tous les `ronds` sont dans cet unique *cluster* et tous les `triangles` sont dans cet unique *cluster* ;

  — ce *clustering* n'est **pas homogène** (*homogeneity* = 0.00) : cet unique *cluster* contient à la fois des `carrés`, des `ronds` et des `triangles` ;

  — la `v-measure` finale est de 0.00, confirmant que ce *clustering* simpliste n'est pas performant car il n'est pas homogène.

— **(2)** représente une segmentation où chaque donnée a son propre *cluster* :

  — ce *clustering* est **homogène** (*homogeneity* = 1.00) : comme chaque *cluster* ne contient qu'une seule donnée, donc chaque *cluster* ne contient que des données de la même forme ;

  — ce *clustering* n'est pas **complet** (*completeness* = 0.00) : tous les `carrés` ne sont pas dans un même *cluster*, tous les `ronds` ne sont pas dans un même *cluster* et tous les `triangles` ne sont pas dans un même *cluster* ;

  — la `v-measure` finale est de 0.00, confirmant que ce *clustering* simpliste n'est pas performant car il n'est pas complet.

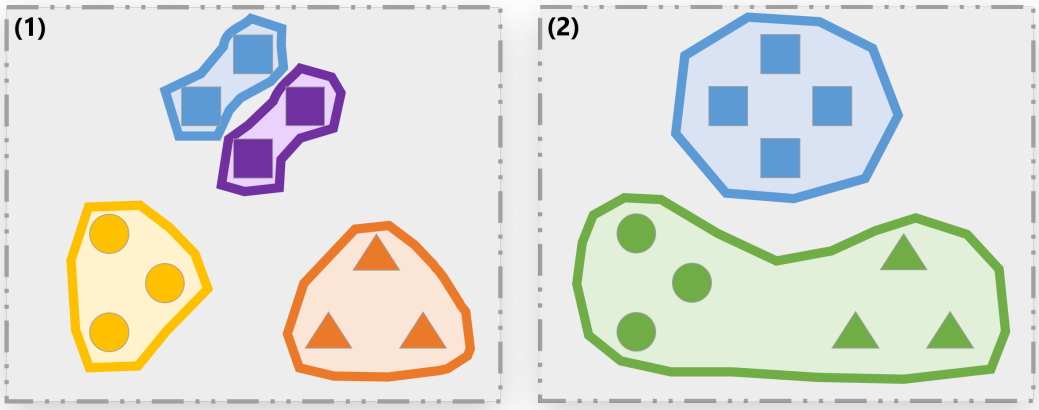

Figure D.3 – *Exemples de* `clustering` *avec des cas simples : (1) représente un regroupement en 4 clusters où les* `ronds` *et les* `triangles` *sont correctement regroupés, mais où les* `carrés` *sont séparés. (2) représente un regroupement en 2 clusters où les* `carrés` *sont correctement regroupés, mais où les* `ronds` *et les* `triangles` *ont été rassemblés.*

Examinons enfin les deux cas simples de la Figure D.3.

— **(1)** représente une segmentation presque parfaite où les `carrés` sont séparés :

  — ce *clustering* est **homogène** (*homogeneity* = 1.00) : il n'y a que des `carrés` dans le *cluster* bleu, il n'y a que des `carrés` dans le *cluster* violet, il n'y a que des `carrés` dans le *cluster* jaune et il n'y a que des `carrés` dans le *cluster* orange ;

  — ce *clustering* n'est **pas assez complet** (*completeness* = 0.78) : tous les `ronds` sont dans le *cluster* jaune et tous les `triangles` sont dans le *cluster* orange, mais les `carrés` sont séparés dans les *clusters* bleu et violet ;

  — le score de `v-measure` final n'est de 0.89, pénalisé par le manque de complétude des *clusters* bleu et violet.

— **(2)** représente une segmentation presque parfaite où les `ronds` et les `triangles` ont été rassemblés :

  — ce *clustering* est **complet** (*completeness* = 1.00) : tous les `carrés` sont dans le *cluster* bleu, tous les `ronds` sont dans le *cluster* vert et tous les `triangles` sont dans le *cluster* vert ;

  — ce *clustering* n'est **pas assez homogène** (*completeness* = 0.62) : il n'y a que des `carrés` dans le *cluster* bleu, mais il y a à la fois des `ronds` et des `triangles` dans le *cluster* vert ;

  — le score de `v-measure` final n'est de 0.76, pénalisé par le manque d'homogénéité du *cluster* vert.

Bibliographie

Adamopoulou, E., & Moussiades, L. (2020). An Overview of Chatbot Technology. In I. Maglogiannis, L. Iliadis & E. Pimenidis (Éd.), *Artificial Intelligence Applications and Innovations* (p. 373-383). Springer International Publishing. https://doi.org/10/ghj8

Agarwal, P., Alam, M. A., & Biswas, R. (2011). Issues,Challenges and Tools of Clustering Algorithms. *IJCSI International Journal of Computer Science Issues*, *8*(3). https://doi.org/10.48550/ARXIV.1110.2610

Air France. (2017). *Louis*. https://corporate.airfrance.com/fr/communiques-presse/air-france-presente-louis-son-chatbot-intelligent

Aized Amin Soofi & Arshad Awan. (2017). Classification Techniques in Machine Learning : Applications and Issues. *J. Basic Appl. Sci.*, *13*, 459-465. https://doi.org/10.6000/1927-5129.2017.13.76

Alammar, J., & Grefenstette, E. (2022). *Cohere Sandbox*. https://github.com/cohere-ai/sandbox-topically

Alasadi, S. A., & Bhaya, W. S. (2017). Review of Data Preprocessing Techniques in Data Mining. *Journal of Engineering and Applied Sciences*, *12*(16), 4102-4107. https://www.academia.edu/download/54509277/4102-4107.pdf

Alexa Internet. (2018). Keyword Research, Competitor Analysis, & Website Ranking | Alexa. https://www.alexa.com

Anderson, J. R. (2013, novembre 19). *The Architecture of Cognition* (0e éd.). Psychology Press. Récupérée juin 7, 2023, à partir de https://www.taylorfrancis.com/books/9781317759539

Artstein, R., & Poesio, M. (2008). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, *34*(4), 555-596. https://doi.org/10.1162/coli.07-034-R2

Asher, N., Nasr, A., & Perrotin, R. (2017). Manuel d'annotation en actes de dialogue pour le corpus Datcha.

Audacity Team. (2000). *Audacity : Free Audio Editor and Recorder*. https://www.audacityteam.org/

Awel, M. A., & Abidi, A. I. (2019). Review on Optical Character Recognition. *06*(06).

Bae, J., Helldin, T., Riveiro, M., Nowaczyk, S., Bouguelia, M.-R., & Falkman, G. (2021). Interactive Clustering : A Comprehensive Review. *ACM Comput. Surv.*, *53*(1), 1-39. https://doi.org/10.1145/3340960

Baledent, A. (2023, décembre 1). *De la complexité de l'annotation manuelle : méthodologie, biais et recommandations*. https://theses.hal.science/tel-04011353

Bayerl, P. S., & Paul, K. I. (2011). What Determines Inter-Coder Agreement in Manual Annotations ? A Meta-Analytic Investigation. *Computational Linguistics*, *37*(4), 699-725. https://doi.org/10.1162/COLI_a_00074

Berchmans, D., & Kumar, S. S. (2014). Optical Character Recognition : An Overview and an Insight. *2014 International Conference on Control, Instrumentation, Communication and*

Computational Technologies (ICCICCT), 1361-1365. https://doi.org/10.1109/ICCICCT.2014.6993174

Biber, D. (1993). Representativeness in Corpus Design. *Literary and Linguistic Computing*, *8*(4). https://doi.org/10.1093/llc/8.4.243

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of machine Learning research*, (3), 993-1022.

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa : Open Source Language Understanding and Dialogue Management. *ArXiv preprint*. Récupérée octobre 23, 2020, à partir de http://arxiv.org/abs/1712.05181

Böhmová, A., Hajič, J., Hajičová, E., & Hladká, B. (2003). The Prague Dependency Treebank. In A. Abeillé (Éd.), *Treebanks* (p. 103-127, T. 20). Springer Netherlands. Récupérée septembre 21, 2023, à partir de http://link.springer.com/10.1007/978-94-010-0201-1_7

Brabra, H., Baez, M., Benatallah, B., Gaaloul, W., Bouguelia, S., & Zamanirad, S. (2022). Dialogue Management in Conversational Systems : A Review of Approaches, Challenges, and Opportunities. *IEEE Trans. Cogn. Dev. Syst.*, *14*(3), 783-798. https://doi.org/10.1109/TCDS.2021.3086565

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language Models Are Few-Shot Learners. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2005.14165

Brysbaert, M. (2019). How many words do we read per minute ? A review and meta-analysis of reading rate. *Journal of Memory and Language*, *109*, 104047. https://doi.org/10.1016/j.jml.2019.104047

Callison-Burch, C., & Dredze, M. (2010). Creating Speech and Language Data with Amazon's Mechanical Turk. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 1-12. https://aclanthology.org/W10-0701

Chen, H., Liu, X., Yin, D., & Tang, J. (2017). A Survey on Dialogue Systems : Recent Advances and New Frontiers. *SIGKDD Explor. Newsl.*, *19*(2), 25-35. https://doi.org/10.1145/3166054.3166058

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., & Stoica, I. (2024). Chatbot Arena : An Open Platform for Evaluating LLMs by Human Preference. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2403.04132

Clemmensen, L. H., & Kjaersgaard, R. D. (2022). Data Representativity for Machine Learning and AI Systems. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2203.04706

Collins, W. (2017, avril). Chapter 7 : Overfitting. In *Algorithms To Live By : The Computer Science of Human Decisions* (p. 149-168).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Mach Learn*, *20*(3), 273-297. https://doi.org/10.1007/BF00994018

Costello, K., & LoDolce, M. (2019). Gartner Top Technologies and Trends Driving the Digital Workplace [newspaper]. *Gartner, Inc.* Récupérée octobre 23, 2020, à partir de https://www.gartner.com/smarterwithgartner/top-10-technologies-driving-the-digital-workplace/

Costello, K., & LoDolce, M. (2022). Gartner Predicts Chatbots Will Become a Primary Customer Service Channel Within Five Years [newspaper]. *Gartner, Inc.* Récupérée octobre

9, 2023, à partir de https://www.gartner.com/en/newsroom/press-releases/2022-07-27-gartner-predicts-chatbots-will-become-a-primary-customer-service-channel-within-five-years

Creative commons. (2013). *CC BY-NC 4.0 LEGAL CODE - Attribution-NonCommercial 4.0 International*. Récupérée septembre 29, 2023, à partir de https://creativecommons.org/licenses/by-nc/4.0/legalcode.en

CVAT.Ai Corporation. (2019, octobre 17). *Computer Vision Annotation Tool (CVAT)*. Récupérée septembre 27, 2023, à partir de https://www.cvat.ai/

Dagan, I., Glickman, O., & Magnini, B. (2005). The PASCAL Recognising Textual Entailment Challenge. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, *3944*, 177-190. https://doi.org/10.1007/11736790_9

Dandapat, S., Biswas, P., Choudhury, M., & Bali, K. (2009). Complex Linguistic Annotation – No Easy Way out ! A Case from Bangla and Hindi POS Labeling Tasks. *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, 10-18. https://aclanthology.org/W09-3002

DataBird. (2023, juillet). *Les 10 métiers data les plus recherchés en 2023*. DataBird. Récupérée septembre 26, 2023, à partir de https://www.data-bird.co/blog/metiers-data

Davidson, I., & Ravi, S. S. (2005). Agglomerative Hierarchical Clustering with Constraints : Theoretical and Empirical Results (A. M. Jorge, L. Torgo, P. Brazdil, R. Camacho & J. Gama, Éd.). *Knowledge Discovery in Databases : PKDD 2005*, *3721*, 59-70. Récupérée octobre 22, 2020, à partir de http://link.springer.com/10.1007/11564126_11

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv preprint*. Récupérée juin 10, 2020, à partir de http://arxiv.org/abs/1810.04805

Diamond, I., Cox, D. R., & Snell, E. J. (1990). Analysis of Binary Data. 2nd Edn. *Applied Statistics*, *39*(2), 260. https://doi.org/10.2307/2347766

Dipper, S., Gotze, M., & Skopeteas, S. (2004). Towards User-Adaptive Annotation Guidelines. *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, 23-30. https://aclanthology.org/W04-1904

Dzieza, J. (2023). AI is a lot of work [newspaper]. *New York Magazine : Artificial Intelligence*. Récupérée septembre 29, 2023, à partir de https://nymag.com/intelligencer/article/ai-artificial-intelligence-humans-technology-business-factory.html

Edwards, A. W. F. (1992). *Likelihood* (Expanded ed). Johns Hopkins Univ. Press.

Elkosantini, S., & Gien, D. (2009). Integration of human behavioural aspects in a dynamic model for a manufacturing system. *International Journal of Production Research*, *47*(10), 2601-2623. https://doi.org/10.1080/00207540701663490

Ester, M., Kriegel, H.-P., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.

Falke, T., Ribeiro, L. F. R., Utama, P. A., Dagan, I., & Gurevych, I. (2019). Ranking Generated Summaries by Correctness : An Interesting but Challenging Application for Natural Language Inference. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2214-2220. https://doi.org/10.18653/v1/P19-1213

Finlayson, M. A., & Erjavec, T. (2016). Overview of Annotation Creation : Processes & Tools. *ArXiv preprint*. Récupérée juin 14, 2021, à partir de http://arxiv.org/abs/1602.05753

Fiumara, J., Cieri, C., Wright, J., & Liberman, M. (2020). LanguageARC : Developing language resources through citizen linguistics. *Proceedings of the LREC 2020 Workshop*

on "Citizen Linguistics in Language Resource Development", 1-6. https://aclanthology.org/2020.cllrd-1.1

Fort, K. (2017). Experts Ou (Foule de) Non-Experts ? La Question de l'expertise Des Annotateurs Vue de La Myriadisation (Crowdsourcing). *corela*. https://doi.org/10.4000/corela.4835

Fort, K. (2022, mars 31). *Manual Annotation : What is it ? How to do it (properly) ?* (École thématique d'été). https://members.loria.fr/KFort/files/fichiers_cours/AnnotationQuality.pdf

Fort, K., Ehrmann, M., & Nazarenko, A. (2009). Vers Une Méthodologie d'annotation Des Entités Nommées En Corpus ? *Traitement Automatique Des Langues Naturelles 2009*. https://hal.science/hal-00402321

Fort, K., Nazarenko, A., & Rosset, S. (2012). Modeling the complexity of manual annotation tasks : a grid of analysis. *Proceedings of COLING 2012*, 895-910. https://hal.science/hal-00769631

Fort, K., & Sagot, B. (2010). Influence of Pre-Annotation on POS-Tagged Corpus Development. *Proceedings of the Fourth Linguistic Annotation Workshop*, 56-63. https://aclanthology.org/W10-1807

Gao, J., Galley, M., & Li, L. (2018). Neural Approaches to Conversational AI. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics : Tutorial Abstracts*, 2-7. https://doi.org/10.18653/v1/P18-5002

Garside, R., Leech, G. N., & McEnery, T. (Éd.). (1997). *Corpus annotation : linguistic information from computer text corpora*. Longman.

Girden, E. (1992). *ANOVA*. SAGE Publications, Inc. Récupérée juillet 6, 2023, à partir de https://methods.sagepub.com/book/anova

Givoni, I. E., & Frey, B. J. (2009). Semi-Supervised Affinity Propagation with Instance-Level Constraints.

Goasduff, L. (2019). Chatbots Will Appeal to Modern Workers [newspaper]. *Gartner, Inc.* Récupérée octobre 23, 2020, à partir de https://www.gartner.com/smarterwithgartner/chatbots-will-appeal-to-modern-workers/

Google. (2016). *Google Assistant, Your Own Personal Google*. https://assistant.google.com/

Google. (2023, juillet 13). *Bard - Chat Based AI Tool from Google*. https://bard.google.com/chat

Goyal, A., Gupta, V., & Kumar, M. (2018). Recent Named Entity Recognition and Classification techniques : A systematic review. *Computer Science Review*, *29*, 21-43. https://doi.org/10.1016/j.cosrev.2018.06.001

Guillaume, B., Fort, K., & Lefèbvre, N. (2016). Crowdsourcing Complex Language Resources : Playing to Annotate Dependency Syntax. *International Conference on Computational Linguistics (COLING)*. https://inria.hal.science/hal-01378980

Gut, U., & Bayerl, P. S. (2004). Measuring the Reliability of Manual Annotations of Speech Corpora. https://api.semanticscholar.org/CorpusID:27970161

Harris, Z. S. (1954). Distributional Structure. WORD, *10*(2-3), 146-162. https://doi.org/10.1080/00437956.1954.11659520

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index) : Results of Empirical and Theoretical Research. In P. A. Hancock & N. Meshkati (Éd.), *Human Mental Workload* (p. 139-183, T. 52). North-Holland. https://www.sciencedirect.com/science/article/pii/S0166411508623869

Honnibal, M., & Montani, I. (2017). spaCy 2 : Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing.

Howe, J. (2008). *Crowdsourcing : how the power of the crowd is driving the future of business* (Random House Books). RH Business Books.

Hoyt, R. E., Snider, D., Thompson, C., & Mantravadi, S. (2016). IBM Watson Analytics : Automating Visualization, Descriptive, and Predictive Statistics. *JMIR Public Health Surveill*, *2*(2). https://doi.org/10.2196/publichealth.5810

Huang, J., Shao, H., & Chang, K. C.-C. (2022). Are Large Pre-Trained Language Models Leaking Your Personal Information ? *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2205.12628

Hugging Face. (2016). *Hugging Face - the AI Community Building the Future.* https://huggingface.co/datasets

Iman, M., Arabnia, H. R., & Rasheed, K. (2023). A Review of Deep Transfer Learning and Recent Advancements. *Technologies*, *11*(2), 40. https://doi.org/10.3390/technologies11020040

International Organization for Standardization. (2007, février 16). *Codes for the Representation of Names of Languages – Part 3 : Alpha-3 Code for Comprehensive Coverage of Languages.* https://www.iso.org/standard/39534.html

Isoz, V. (2017, décembre 20). *Découvrir les métiers de la data science*. LinkedIn Learning. Récupérée septembre 26, 2023, à partir de https://fr.linkedin.com/learning/decouvrir-les-metiers-de-la-data-science/decouvrir-la-data-science

Jaipuria, N., Zhang, X., Bhasin, R., Arafa, M., Chakravarty, P., Shrivastava, S., Manglani, S., & Murali, V. N. (2020). Deflating Dataset Bias Using Synthetic Data Augmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Jones, G., Hocine, M., Salomon, J., Dab, W., & Temime, L. (2015). Demographic and occupational predictors of stress and fatigue in French intensive-care registered nurses and nurses' aides : A cross-sectional study. *International Journal of Nursing Studies*, *52*(1), 250-259. https://doi.org/10.1016/j.ijnurstu.2014.07.015

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., & McHardy, R. (2023). Challenges and Applications of Large Language Models. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2307.10169

Kahneman, D. (2011). *Thinking, Fast and Slow.* Farrar, Straus and Giroux.

Kamvar, S. D., Klein, D., & Manning, C. D. (2003). Spectral Learning. *Proceedings of the international joint conference on artificial intelligence*, 561-566.

Keung, P., Lu, Y., Szarvas, G., & Smith, N. A. (2020). The Multilingual Amazon Reviews Corpus. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4563-4568. https://doi.org/10.18653/v1/2020.emnlp-main.369

Khan, M. A., Tamim, I., Ahmed, E., & Awal, M. A. (2012). Multiple Parameter Based Clustering (MPC) : Prospective Analysis for Effective Clustering in Wireless Sensor Network (WSN) Using K-Means Algorithm. *WSN*, *04*(01), 18-24. https://doi.org/10.4236/wsn.2012.41003

Kirch, W. (2008). Pearson's Correlation Coefficient. In *Encyclopedia of Public Health* (p. 1090-1091). Springer Netherlands. Récupérée juillet 6, 2023, à partir de https://link.springer.com/10.1007/978-1-4020-5614-7_2569

Klie, J.-C., Bugert, M., Boullosa, B., de Castilho, R. E., & Gurevych, I. (2018). The INCEpTION platform : Machine-assisted and knowledge-oriented interactive annotation. *Proceedings of the 27th International Conference on Computational Linguistics : System Demonstrations*, 5-9. https://inception-project.github.io/

Kothadiya, D., Pise, N., & Bedekar, M. (2020). Different Methods Review for Speech to Text and Text to Speech Conversion. *IJCA*, *175*(20), 9-12. https://doi.org/10.5120/ijca2020920727

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning : a review of classification and combining techniques. *Artif Intell Rev*, *26*(3), 159-190. https://doi.org/10.1007/s10462-007-9052-3

Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. *WIREs Data Min & Knowl*, *1*(3), 231-240. https://doi.org/10.1002/widm.30

Krippendorff, K. (2004). *Content analysis : an introduction to its methodology* (2nd ed). Sage.

Kruskal, W., & Mosteller, F. (1979a). Representative Sampling, I : Non-Scientific Literature. *International Statistical Review / Revue Internationale de Statistique*, *47*(1), 13. https://doi.org/10.2307/1403202

Kruskal, W., & Mosteller, F. (1979b). Representative Sampling, II : Scientific Literature, Excluding Statistics. *International Statistical Review / Revue Internationale de Statistique*, *47*(2), 111. https://doi.org/10.2307/1402564

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., & Socher, R. (2016). Ask Me Anything : Dynamic Memory Networks for Natural Language Processing. *International Conference on Machine Learning*, *48*, 1378-1387. Récupérée décembre 28, 2018, à partir de http://arxiv.org/abs/1506.07285

Lamirel, J.-C., Cuxac, P., & Hajlaoui, K. (2017). A Novel Approach to Feature Selection Based on Quality Estimation Metrics. In F. Guillet, B. Pinaud & G. Venturini (Éd.), *Advances in Knowledge Discovery and Management* (p. 121-140, T. 665). Springer International Publishing. Récupérée novembre 23, 2018, à partir de http://link.springer.com/10.1007/978-3-319-45763-5_7

Lampert, T., Dao, T.-B.-H., Lafabregue, B., Serrette, N., Forestier, G., Crémilleux, B., Vrain, C., & Gançarski, P. (2018). Constrained distance based clustering for time-series : a comparative and experimental study. *Data Min Knowl Disc*, *32*(6), 1663-1707. https://doi.org/10/gfbpj8

Lampert, T., Lafabregue, B., & Gancarski, P. (2019). Constrained Distance based K-Means Clustering for Satellite Image Time-Series. *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2419-2422. https://doi.org/10/ggx3tj

Landis, J. R., & Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, *33*(1), 159. https://doi.org/10.2307/2529310

Latitude Inc. & Oasis Tech Inc. (2019). *AI Dungeon*. https://play.aidungeon.com/

Lee, K., & Sengupta, S. (2022). *Introducing the Ai Research Supercluster - Meta's Cutting-Edge Ai Supercomputer for Ai Research*. Meta AI. https://ai.meta.com/blog/ai-rsc/

Leech, G. (1993). Corpus Annotation Schemes. *Literary and Linguistic Computing*, *8*(4), 275-281. https://doi.org/10.1093/llc/8.4.275

Leech, G. (2004). Adding linguistic annotation. In M. Wynne (Éd.), *Developing linguistic corpora : a guide to good practice* (Oxbow Books, p. 17-29). AHDS : Literature, Languages, and Linguistics. http://ahds.ac.uk/creating/guides/linguistic-corpora/chapter2.htm

Les Echos. (2023). IA : L'auteur de "Game of Thrones" et d'autres écrivains portent plainte contre le créateur de ChatGPT [newspaper]. *Les Echos : Tech-Médias*. Récupérée septembre 29, 2023, à partir de https://www.lesechos.fr/tech-medias/intelligence-artificielle/ia-lauteur-de-game-of-thrones-et-dautres-ecrivains-portent-plainte-contre-le-createur-de-chatgpt-1980235

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART : Denoising Sequence-to-Sequence Pre-training for

Natural Language Generation, Translation, and Comprehension. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.1910.13461

Li, J., Sun, A., Han, J., & Li, C. (2022). A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.*, *34*(1), 50-70. https://doi.org/10.1109/TKDE.2020.2981314

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2014). Microsoft COCO : Common Objects in Context. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.1405.0312

Loignon, S. (2023). IA : Les médias français s'organisent face à la collecte de données par les robots [newspaper]. *Les Echos : Tech-Médias*. Récupérée octobre 4, 2023, à partir de https://www.lesechos.fr/tech-medias/medias/ia-les-medias-francais-sorganisent-face-a-la-collecte-de-donnees-par-les-robots-1973079

Maalouf, M. (2011). Logistic regression in data analysis : an overview. *IJDATS*, *3*(3), 281. https://doi.org/10.1504/IJDATS.2011.041335

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, *1*(14), 281-297.

Maharana, K., Mondal, S., & Nemade, B. (2022). A review : Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, *3*(1), 91-99. https://doi.org/10.1016/j.gltp.2022.04.020

Manning, C. D., & Schütze, H. (2000). *Foundations of statistical natural language processing* (2e éd. avec des corrections). MIT Press.

McCowan, I., Moore, D., Dines, J., Gatica-Perez, D., Flynn, M., Wellner, P., & Bourlard, H. (2005, mars). *On the Use of Information Retrieval Measures for Speech Recognition Evaluation* (IDIAP-RR 04-73). IDIAP Research Institute. Martigny, Switzerland.

Microsoft Corporation. (2018). *Microsoft Excel*. https://office.microsoft.com/excel

Miller, G. A., & Charles, W. G. (1991). Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, *6*(1), 1-28. https://doi.org/10.1080/01690969108406936

Montani, I., & Honnibal, M. (2017, décembre 18). *Prodigy : A Modern and Scriptable Annotation Tool for Creating Training Data for Machine Learning Models*. https://prodi.gy/

Morris & Goscinny, R. (1950). *Rodeo*. Dupuis.

Morris & Goscinny, R. (1952). *Sous le ciel de l'ouest*. Dupuis.

Morris & Goscinny, R. (1958). *Les Cousins Dalton*. Dupuis.

Mu, Z., Yang, X., & Dong, Y. (2021). Review of End-to-End Speech Synthesis Technology Based on Deep Learning. *ArXiv preprint*. https://doi.org/10.48550/arXiv.2104.09995

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering : An overview. *Wiley Interdisc. Rew. : Data Mining and Knowledge Discovery*, *2*, 86-97. https://doi.org/10.1002/widm.53

Nédellec, C., Bessieres, P., Bossy, R., & Kotoujansky, A. (2006). Annotation Guidelines for Machine Learning-Based Named Entity Recognition in Microbiology.

Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, *135*(3), 370. https://doi.org/10.2307/2344614

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On Spectral Clustering : Analysis and an Algorithm. In T. G. Dietterich, S. Becker & Z. Ghahramani (Éd.), *Advances in Neural Information Processing Systems 14* (p. 849-856). MIT Press. Récupérée octobre

22, 2020, à partir de http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf

Ni, J., Young, T., Pandelea, V., Xue, F., & Cambria, E. (2022). Recent Advances in Deep Learning Based Dialogue Systems : A Systematic Survey. *ArXiv preprint*. Récupérée juillet 26, 2023, à partir de http://arxiv.org/abs/2105.04387

Nicoletti, L., & Bass, D. (2023). Generative AI takes stereotypes and bias from bad to worse [newspaper]. *Bloomberg.com*. Récupérée octobre 2, 2023, à partir de https://www.bloomberg.com/graphics/2023-generative-ai-bias/

Nivre, J. (2006). *Inductive Dependency Parsing* (T. 34). Springer Netherlands. Récupérée juillet 6, 2023, à partir de http://link.springer.com/10.1007/1-4020-4889-0

Nothman, J., Qin, H., & Yurchak, R. (2018). Stop Word Lists in Free Open-source Software Packages. *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, 7-12. https://doi.org/10.18653/v1/W18-2502

O'Neill, M., & Connor, M. (2023). Amplifying Limitations, Harms and Risks of Large Language Models. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2307.04821

OpenAI. (2023). *ChatGPT*. https://chat.openai.com

Parasuraman, R., Sheridan, T., & Wickens, C. (2000). A Model for Types and Levels of Human Interaction with Automation. *IEEE Trans. Syst., Man, Cybern. A*, *30*(3), 286-297. https://doi.org/10.1109/3468.844354

Parnami, A., & Lee, M. (2022). Learning from Few Examples : A Summary of Approaches to Few-Shot Learning. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2203.04291

Perrigo, B., & Zorthian, J. (2023). Exclusive : OpenAI used Kenyan workers on less than $2 per hour to make ChatGPT less toxic [newspaper]. *Time : Business, Technology*. Récupérée septembre 29, 2023, à partir de https://time.com/6247678/openai-chatgpt-kenya-workers/

Perrotin, R., Nasr, A., & Auguste, J. (2018). Annotation En Actes de Dialogue Pour Les Conversations d'Assistance En Ligne. *25e Conférence Sur Le Traitement Automatique Des Langues Naturelles (TALN)*. https://hal.science/hal-01943345

Perry, T. (2021). LightTag : Text Annotation Platform. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, 20-27. https://aclanthology.org/2021.emnlp-demo.3

Pradhan, S. S., Loper, E., Dligach, D., & Palmer, M. (2007). SemEval-2007 task 17 : English lexical sample, SRL and all words. *Proceedings of the 4th International Workshop on Semantic Evaluations - SemEval '07*, 87-92. https://doi.org/10.3115/1621474.1621490

Proposal for a Regulation of the European Parliament and the Council Laying down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts (2021, avril 21). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206

Purves, D., & Brannon, E. M. (Éd.). (2013). *Principles of cognitive neuroscience* (2. ed). Sinauer.

Pustejovsky, J., & Stubbs, A. (2012, octobre 10). *Natural language annotation for machine learning*. O'Reilly Media, Inc. https://api.semanticscholar.org/CorpusID:60457717

R Core Team. (2017). *R : A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, *1*(8), 9.

Radovilsky, Z., Hegde, V., & Acharya, A. (2018). Skills Requirements of Business Data Analytics and Data Science Jobs : A Comparative Analysis. *16*(1).

Rajbahadur, G. K., Tuck, E., Zi, L., Lin, D., Chen, B., Ming, Z., Jiang & German, D. M. (2022). Can I use this publicly available dataset to build commercial AI software ? – A Case Study on Publicly Available Image Datasets. *ArXiv preprint*. Récupérée septembre 29, 2023, à partir de http://arxiv.org/abs/2111.02374

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2102.12092

Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the first instructional conference on machine learning*.

Raschka, S., & Mirjalili, V. (2019). *Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow 2* (Third edition). Packt.

Re3data.Org. (2013). *Zenodo*. https://doi.org/10.17616/R3QP53

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation) (2016, mai 4). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679

Roach, J. (2023). *How Microsoft's Bet on Azure Unlocked an AI Revolution*. Microsoft. https://news.microsoft.com/

Rosenberg, A., & Hirschberg, J. (2007). V-Measure : A Conditional Entropy-Based External Cluster Evaluation Measure.

Rowe, N. (2023). "It's destroyed me completely" : Kenyan moderators decry toll of training of AI models [newspaper]. *The Guardian : Artificial Intelligence*. Récupérée septembre 29, 2023, à partir de https://www.theguardian.com/technology/2023/aug/02/ai-chatbot-training-human-toll-content-moderator-meta-openai

Ruiz, C., Spiliopoulou, M., & Menasalvas, E. (2010). Density-based semi-supervised clustering. *Data Min Knowl Disc*, *21*(3), 345-370. https://doi.org/10.1007/s10618-009-0157-y

Sagot, B., Fort, K., Adda, G., Mariani, J., & Lang, B. (2011). Un Turc Mécanique Pour Les Ressources Linguistiques : Critique de La Myriadisation Du Travail Parcellisé. *TALN'2011 - Traitement Automatique Des Langues Naturelles*. https://inria.hal.science/inria-00617067

Sasaki, Y. (2007). The truth of the F-measure.

Schild, E. (2022a, août 22). *Cognitivefactory/Interactive-Clustering*. Récupérée février 13, 2023, à partir de https://doi.org/10.5281/zenodo.4775251

Schild, E. (2022b, septembre 1). *Cognitivefactory/Interactive-Clustering-Gui*. Récupérée février 13, 2023, à partir de https://doi.org/10.5281/zenodo.4775270

Schild, E. (2022c, novembre 5). *Cognitivefactory/Interactive-Clustering-Comparative-Study*. Récupérée février 13, 2023, à partir de https://doi.org/10.5281/zenodo.5648255

Schild, E. (2022d, novembre 9). *French trainset for chatbots dealing with usual requests on bank cards*. Zenodo. https://doi.org/10.5281/zenodo.4769949

Schild, E. (2023, février 16). *Cognitivefactory/Features-Maximization-Metric*. Récupérée février 16, 2023, à partir de https://doi.org/10.5281/zenodo.7646382

Schild, E., & Adler, M. (2023, octobre 2). *Subset of 'MLSUM : The Multilingual Summarization Corpus' for constraints annotation experiment* (Version 1.0.0 [subset : fr+train+filtered]). Zenodo. https://doi.org/10.5281/ZENODO.8399301

Schild, E., Durantin, G., & Lamirel, J.-C. (2021). Concevoir un assistant conversationnel de manière itérative et semi-supervisée avec le clustering interactif. *TextMine 2021 (TM'2021) - En conjonction avec EGC 2021*, 11-14. https://hal.inria.fr/hal-03133060

Schild, E., Durantin, G., Lamirel, J.-C., & Miconi, F. (2021). Conception itérative et semi-supervisée d'assistants conversationnels par regroupement interactif des questions. *RNTI E-37*. Récupérée juin 14, 2021, à partir de https://hal.inria.fr/hal-03133007

Schild, E., Durantin, G., Lamirel, J.-C., & Miconi, F. (2022). Iterative and Semi-Supervised Design of Chatbots Using Interactive Clustering. *International Journal of Data Warehousing and Mining (IJDWM)*, *18*(2), 1-19. https://doi.org/10.4018/IJDWM.298007

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., & Staiano, J. (2020, avril 30). *MLSUM : The Multilingual Summarization Corpus* (arXiv :2004.14900). arXiv. Récupérée juin 7, 2023, à partir de http://arxiv.org/abs/2004.14900

Seabold, S., & Perktold, J. (2010). Statsmodels : Econometric and Statistical Modeling with Python, 92-96. https://doi.org/10.25080/Majora-92bf1922-011

Settles, B. (2010). Active Learning Literature Survey, 67.

Sheridan, T. B., & Verplank, W. L. (1978, juillet 15). *Human and Computer Control of Undersea Teleoperators*. Defense Technical Information Center. Fort Belvoir, VA. Récupérée septembre 1, 2023, à partir de http://www.dtic.mil/docs/citations/ADA057655

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *J Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Text Data Augmentation for Deep Learning. *J Big Data*, *8*(1), 101. https://doi.org/10.1186/s40537-021-00492-0

Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval Augmentation Reduces Hallucination in Conversation. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2104.07567

Sinclair, J. (2004). Corpus and Text : Basic Principles. In M. Wynne (Éd.), *Developing Linguistic Corpora : A Guide to Good Practice* (Oxbow Books, p. 1-16). AHDS : Literature, Languages, and Linguistics. http://ahds.ac.uk/creating/guides/linguistic-corpora/chapter1.htm

SNCF. (2018). *Agent Virtuel SNCF*. https://bot.assistant.sncf/index.html

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. (2008). Cheap and Fast - But is it Good ? Evaluating Non-Expert Annotations for Natural Language Tasks. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 254-263.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11-21. https://doi.org/10.1108/eb026526

Sperandio, J.-C. (1978). The Regulation of Working Methods as a Function of Work-load among Air Traffic Controllers. *Ergonomics*, *21*(3), 195-202. https://doi.org/10.1080/00140137808931713

Sperandio, J.-C. (1987). *L'ergonomie Du Travail Mental*. FeniXX.

Steinbach, M., Ertöz, L., & Kumar, V. (2004). The Challenges of Clustering High Dimensional Data. In L. T. Wille (Éd.), *New Directions in Statistical Physics* (p. 273-309). Springer Berlin Heidelberg. Récupérée octobre 16, 2023, à partir de http://link.springer.com/10.1007/978-3-662-08968-2_16

Stubbs, A. C. (2013). *A Methodology for Using Professional Knowledge in Corpus* [thèse de doct., Brandeis University].

Team Datascientest. (2022, septembre). *Les métiers de la data : mieux comprendre leurs différences*. Datascientest.com. Récupérée septembre 26, 2023, à partir de https://datascientest.com/les-metiers-de-la-data

Thorndike, R. L. (1953). Who belongs in the family ? *Psychometrika*, *18*(4), 267-276. https://doi.org/10.1007/BF02289263

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., ... Scialom, T. (2023). Llama 2 : Open Foundation and Fine-Tuned Chat Models. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.2307.09288

Tukey, J. W. (1949). Comparing Individual Means in the Analysis of Variance. *Biometrics*, *5*(2), 99. https://doi.org/10.2307/3001913

Uszkoreit, J. (2017, août 31). *Transformer : A Novel Neural Network Architecture for Language Understanding*. Google AI Blog. Récupérée juin 10, 2020, à partir de http://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Valette, M. (2016). Analyse statistique des données textuelles et traitement automatique des langues. Une étude comparée. *International conference on statistical analysis of textual data (JADT2016)*, *2*, 697-706. https://inalco.hal.science/hal-01335084

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual* (CreateSpace).

Von Ahn, L. (2006). Games with a Purpose. *Computer*, *39*(6), 92-94. https://doi.org/10.1109/MC.2006.196

Voormann, H., & Gut, U. (2008). Agile Corpus Creation. https://api.semanticscholar.org/CorpusID:56885448

Wagstaff, K., & Cardie, C. (2000). Clustering with Instance-level Constraints. *Proceedings of the Seventeenth International Conference on Machine Learning*, 1103-1110.

Wagstaff, K., Cardie, C., Rogers, S., & Schrödl, S. (2001). Constrained K-means Clustering with Background Knowledge, 577-584.

Wallach, D., & Goffinet, B. (1987). Mean Squared Error of Prediction in Models for Studying Ecological and Agronomic Systems. *Biometrics*, 561-573.

Wynne, M. (Éd.). (2004). *Developing linguistic corpora : a guide to good practice* (Oxbow Books). AHDS : Literature, Languages, and Linguistics. http://www.ahds.ac.uk/creating/guides/linguistic-corpora/index.htm

Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, *2*, 165-193.

Yan, Z., Duan, N., Chen, P., Zhou, M., Zhou, J., & Li, Z. (2012). Building Task-Oriented Dialogue Systems for Online Shopping. *AAAI*, *31*(1). https://doi.org/10.1609/aaai.v31i1.11182

Yang, T.-N., & Wang, S.-D. (2004). Competitive algorithms for the clustering of noisy data. *Fuzzy Sets and Systems*, *141*(2), 281-299. https://doi.org/10.1016/S0165-0114(02)00525-0

Zdaniuk, B. (2014). Ordinary Least-Squares (OLS) Model. In A. C. Michalos (Éd.), *Encyclopedia of Quality of Life and Well-Being Research* (p. 4515-4517). Springer Netherlands. Récupérée septembre 15, 2023, à partir de http://link.springer.com/10.1007/978-94-007-0753-5_2008

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2019). PEGASUS : Pre-training with Extracted Gap-sentences for Abstractive Summarization. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.1912.08777

Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., McDonnell, T., Nguyen, A. T., Xu, D., Wallace, B. C., & Lease, M. (2016). Neural Information Retrieval : A Literature Review. *ArXiv preprint*. https://doi.org/10.48550/ARXIV.1611.06792

Zhou, Z.-H. (2021). *Machine learning* (S. Liu, Trad.). Springer. https://books.google.fr/books?id=Zd5hywEACAAJ

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A Comprehensive Survey on Transfer Learning. *Proc. IEEE*, *109*(1), 43-76. https://doi.org/10.1109/JPROC.2020.3004555

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste de codes informatiques