Analyse automatique par transitions pour l'identification des expressions polylexicales

Hazem Al Saied

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Analyse automatique par transitions pour l'identification des expressions polylexicales

THÈSE

présentée et soutenue publiquement le 20 décembre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Hazem Al Saied

Composition du jury

Rapporteurs : Emmanuel Morin - Professeur, Université de Nantes - Nantes
Sophie Rosset - Directrice de recherches, CNRS, LIMSI - Orsay

Examinateurs : Benoît Sagot - Chargé de recherches, INRIA - Paris
Christophe Cerisara - Chargé de recherches, CNRS, LORIA - Nancy

Encadrants : Marie Candito - Maître de conférences, Université Paris Diderot - Paris
Mathieu Constant - Professeur, Université de Lorraine - Nancy

Laboratoire d'Analyse et Traitement Informatique de la Langue Française (ATILF) - UMR 7118

Mis en page avec la classe thesul.

Remerciements

Remerciements tout d'abord à mon amour, Yafa, mon petit, Jad, mes parents, mes amis et tous ceux qui ont été proches.

Remerciements aux directeurs de ma thèse, Marie et Mathieu, pour leur bonne volonté, leurs disponibilités, leur engagement, leur investissement et leurs conseils. Grâce à eux, il m'a été possible de développer de nombreuses compétences utiles à ma future carrière.

Remerciements à tous les membres du jury, à savoir Emmanuel Morin, Sophie Rosset, Benoît Sagot et Christophe Cerisara, pour leur présence et leurs remarques et critiques pertinentes sur mon travail.

Remerciements aux collègues de l'ATILF et du LORIA, avec qui j'ai eu de nombreuses discussions enrichissantes, d'un point de vue personnel comme professionnel.

Résumé

Cette thèse porte sur l'identification des expressions polylexicales, abordée au moyen d'une analyse par transitions. Une expression polylexicale (EP) est une construction linguistique composée de plusieurs éléments dont la combinaison montre une irrégularité à un ou plusieurs niveaux linguistiques : morphologique (*grand-mère*), syntaxique (*bel et bien*), sémantique (*carte bleue*) ou pragmatique (*à table*), qui confère à l'expression un statut d'unité lexicale. La tâche d'identification d'EPs consiste à annoter en contexte les occurrences d'EPs dans des textes, i.e à détecter les ensembles de tokens formant de telles occurrences. Par exemple, dans la phrase *Cette série ne vit finalement jamais le jour.*, les tokens *vit*, *le* et *jour* seraient marqués comme formant une occurrence de l'EP *voir le jour*. La tâche est formellement complexe, dans la mesure où une EP peut être discontinue et un token peut appartenir à plusieurs EPs.

L'analyse par transitions est une approche célèbre qui construit une sortie structurée à partir d'une séquence d'éléments , en appliquant une séquence d'actions (appelées « transitions ») choisies parmi un ensemble prédéfini, pour construire incrémentalement la structure de sortie. L'utilisation de l'apprentissage automatique permet d'optimiser un classifieur pour réaliser le choix de la transition à opérer à chaque étape, et ainsi choisir au final une structure de sortie parmi un ensemble en général très grand de structures possibles.

Dans cette thèse, nous proposons un système par transitions dédié à l'identification des EPs au sein de phrases représentées comme des séquences de tokens, et étudions diverses architectures pour le classifieur qui sélectionne les transitions à appliquer, permettant de construire l'analyse de la phrase. La première variante de notre système utilise un classifieur linéaire de type machine à vecteur support (SVM) . Les variantes suivantes utilisent des modèles neuronaux : un simple perceptron multicouche (MLP), puis des variantes intégrant une ou plusieurs couches récurrentes. Le scénario privilégié est une identification d'EPs n'utilisant pas d'informations syntaxiques, alors même que l'on sait les deux tâches liées. Nous étudions ensuite une approche par apprentissage multitâche, réalisant et mettant à profit conjointement l'étiquetage morpho-syntaxique, l'identification des EPs par transitions et l'analyse syntaxique en dépendances par transitions.

La thèse comporte une partie expérimentale importante. Nous avons d'une part étudié quelles techniques de ré-échantillonnage des données permettent une bonne stabilité de l'apprentissage malgré des initialisations aléatoires. D'autre part nous avons proposé une méthode de réglage des hyperparamètres de nos modèles par analyse de tendances au sein d'une recherche aléatoire de combinaison d'hyperparamètres. Nous produisons des systèmes avec la contrainte d'utiliser le même hyperparamétrage pour différentes langues. Nous utilisons en effet de manière privilégiée les données des deux compétitions internationales PARSEME (1.0 et 1.1), contenant des annotations d'EPs verbales pour 18 et 20 langues.

Nos variantes produisent de très bons résultats, et notamment les scores d'état de l'art pour de nombreuses langues des jeux de données PARSEME 1.0 et 1.1. L'une des variantes s'est classée première pour la plupart des langues lors de la campagne PARSEME 1.0. Comparées aux autres méthodes de la littérature, nos variantes MLP et SVM montrent une bonne performance en particulier pour les EPs discontinues ou les EPs correspondant à des variantes d'occurrences vues à l'apprentissage. Mais comme les autres systèmes, nos modèles ont des performances faibles

sur les EPs non vues à l'apprentissage. Nous constatons cela dit que les variantes récurrentes et notre approche multitâche ont des performances globales peu compétitives, mais prometteuses pour ce qui est des EPs inconnues. Ceci suggère que l'accent doit être mis sur la découverte d'EPs utilisant des données non annotées en grosse quantité.

Mots-clés: Expressions polylexicales, Identification des expressions polylexicales, Analyse par transitions, Modèles linéaires, Modèles neuronaux, Réglage d'hyperparamètres de tendances, Classification avec données déséquilibrées

Abstract

This thesis focuses on the identification of multi-word expressions, addressed through a transition-based system. A multi-word expression (MWE) is a linguistic construct composed of several elements whose combination shows irregularity at one or more linguistic levels : morphological (*in short* cf. **in shorter*), syntactic (*by and large*), semantic (*green card*) or pragmatic (*go to the bathroom*).

Identifying MWEs in context amounts to annotating the occurrences of MWEs in texts, i.e. to detecting sets of tokens forming such occurrences. For example, in the sentence *This has nothing to do with the book*, the tokens *has*, *to*, *do* and *with* would be marked as forming an occurrence of the MWE *have to do with*. The task is formally complex, as an MWE can be discontinuous and a token can belong to several MWEs.

Transition-based analysis is a famous NLP technique to build a structured output from a sequence of elements, applying a sequence of actions (called « transitions ») chosen from a predefined set, to incrementally build the output structure. The use of machine learning makes it possible to optimize a classifier to choose the transition to apply at each stage, and thus to choose an output structure among a generally very large set of possible structures.

In this thesis, we propose a transition system dedicated to MWE identification within sentences represented as token sequences, and we study various architectures for the classifier which selects the transitions to apply to build the sentence analysis. The first variant of our system uses a linear support vector machine (SVM) classifier. The following variants use neural models : a simple multilayer perceptron (MLP), followed by variants integrating one or more recurrent layers. The preferred scenario is an identification of MWEs without the use of syntactic information, even though we know the two related tasks. We further study a multitasking approach, which jointly performs and take mutual advantage of morphosyntactic tagging, transition-based MWE identification and dependency parsing.

The thesis comprises an important experimental part. Firstly, we studied which resampling techniques allow good learning stability despite random initializations. Secondly, we proposed a method for tuning the hyperparameters of our models by trend analysis within a random search for a hyperparameter combination. We produce systems with the constraint of using the same hyperparameter combination for different languages. We use data from the two PARSEME international competitions (1.0 and 1.1), which contain verbal MWE annotations for 18 and 20 languages.

Our variants produce very good results, including state-of-the-art scores for many languages in the PARSEME 1.0 and 1.1 datasets. One of the variants ranked first for most languages in the PARSEME 1.0 shared task. Compared to other methods in the literature, our MLP and SVM variants show good performance, especially for discontinuous MWEs or MWEs corresponding to variants of occurrence seen in the training set. But like other systems, our models have poor performance on MWEs that are were not seen at learning time. We note that recurrent variants and our multitasking approach have overall performance that is not very competitive, but promising for these unknown MWEs. This suggests that the focus should be on the discovery of MWEs using non-annotated data in large quantities.

Keywords: Multiword expression, Multiword expression identification, Transition-based analysis, Linear models, Neuronal models, Trend-based hyperparameter tuning, Unbalanced dataset classification

À Jad et Yafa !
À ma mère et mon père !
À tout ce qui est beau et bienveillant à la fois !

Table des matières

II Méthode d'analyse automatique par transitions pour l'identification des EPs

1
Description du système

2
Modèle linéaire

Conclusion

Introduction

Introduction

1 Contexte

Cette thèse porte sur l'identification des expressions polylexicales, traitée via une analyse par transitions, une technique initialement développée pour l'analyse syntaxique.

Il existe une vaste littérature sur la notion d'expression polylexicale, tant en linguistique qu'en Traitement Automatique des Langues (TAL), qui donne lieu à des définitions qui ne recoupent pas toujours exactement les mêmes phénomènes linguistiques. Dans ce travail, nous adoptons la définition des jeux de données que nous utilisons pour nos expériences, à savoir une définition opératoire reposant sur des critères linguistiques formels (Savary et al., 2017; Ramisch et al., 2018). En première approximation, une Expression Polylexicale (EP) est une unité lexicale composée de plusieurs lexèmes, qui comprend une idiosyncrasie morphologique (par exemple dans *prendre la porte*, la variation en nombre de *porte* provoque un changement de sens inattendu), syntaxique (*bel et bien*), sémantique (*mettre les voiles*) et/ou pragmatique (*à table*).
On inclut également à cette définition grossière les cas d'unités typographiquement séparées, mais n'apparaissant que dans le contexte d'une EP (comme *fur* dans *au fur et à mesure*). Une EP constitue en général également une irrégularité statistique, i.e. la probabilité de coocurrence de ses composants est en général supérieure à celle attendue, mais dans les données annotées en EPs que nous utilisons l'irrégularité statistique à elle seule (caractéristique définitoire d'une collocation) n'est pas un critère suffisant d'EP.

Les EPs occupent une place importante en linguistique et en TAL, tout d'abord du fait de leur idiosyncrasie caractéristique : celle-ci implique un traitement particulier des EPs, que ce soit dans une théorie linguistique ou bien dans un système de TAL. Cela se manifeste en général par le fait que certaines variations (lexicales, morphologiques ou syntaxiques) qui seraient normalement possibles étant donnés les composants de l'EP sont impossibles, ou provoquent un changement de sens inattendu. Par ailleurs, les EPs sont un phénomène fréquent au niveau du lexique : Jackendoff (1997) estime que le nombre d'EPs dans le lexique d'un locuteur est comparable au nombre de mots simples et l'étude de Fellbaum (1999) montre que 41 % des entrées de WordNet 1.7, une ressource lexicale de référence en anglais, sont polylexicales.

La tâche d'identification des EPs consiste à repérer en contexte quels ensembles de composants forment une EP. Les deux caractéristiques précédemment citées (irrégularité et fréquence) expliquent à la fois l'intérêt du traitement des EPs en TAL et sa difficulté inhérente, cf. l'irrégularité se prête par définition mal à un traitement systématique.

Une autre difficulté est qu'une occurrence des composants d'une EP peut selon le contexte avoir une lecture idiomatique ou une lecture littérale (différence que l'on trouve entre par exemple *Il a fini par* ***prendre la porte****, en colère.*, et *Puis-je prendre la porte entreposée dans la cave ?*).

Cette ambiguïté représente une caractéristique clé des EPs. Par ailleurs, les EPs peuvent avoir une certaine variabilité. Si les EPs grammaticales sont souvent totalement invariables et continues, la variabilité morphologique, voire syntaxique, caractérise de nombreuses EPs, et se traduit parfois par une discontinuité des composants de l'EP, en particulier pour les EPs verbales (*prendre*$_1$ *en*$_1$ *compte*$_1$ *ces données* vs. *prendre*$_1$ *ces données en*$_1$ *compte*$_1$). Une complication formelle supplémentaire est qu'une EP peut être enchâssée dans une autre EP (*Il*$_{1,2}$ *fait*$_{1,2}$ *l'*$_{1,2}$ *objet*$_{1,2}$ *d'une évaluation*$_2$). Deux EPs peuvent se chevaucher (*Accomplir*$_{1,2}$ *de multiples tâches*$_1$ *et démarches*$_2$) [1].

L'identification des EPs est motivée par l'idiosyncrasie qui les caractérise. D'une part, l'idiosyncrasie sémantique nécessite un traitement particulier dans le cadre de toute tâche incluant une part sémantique. En traduction automatique par exemple, la traduction littérale de la plupart des EPs est en général problématique et l'identification des EPs non et semi-compositionnelles peut faciliter leur traduction et améliorer les scores de la traduction automatique statistique (Diab et Bhutada, 2009) et même neuronale (Rikters et Bojar, 2017). D'autre part, l'idiosyncrasie morphologique et/ou syntaxique des EPs peut perturber les tâches d'étiquetage morphosyntaxique et d'analyse syntaxique (Constant et al., 2017).

Jusqu'à relativement récemment, le domaine de l'identification des EP a été dominé par des approches à base de règles, des approches de classification binaire des candidats ainsi que des approches d'étiquetage de séquences. Les méthodes à base de règles consistent à créer un ensemble de règles symboliques reposant le plus souvent sur un lexique, et à les appliquer sur le texte en cours de traitement pour annoter ses EPs. Les règles peuvent être créées manuellement, ce qui sollicite des connaissances linguistiques et un travail importants, ou extraites automatiquement. La simplicité de ces méthodes et leur capacité de faire face à la discontinuité, à la variabilité morphologique et aux autres caractéristiques des EPs ne masquent cependant pas leur coût de développement et leur dépendance aux langues. Les règles sont par ailleurs rarement contextuelles, et ces méthodes ont tendance à ne pas gérer l'ambiguïté entre occurrence idiomatique versus littérale.

Les méthodes de classification binaire des candidats réduisent le problème d'identification à un problème de classification des occurrences d'EPs entre idiomatiques et littérales. Les méthodes de cette approche supposent l'existence d'EPs candidates préannotées (souvent automatiquement). Alors que cette approche fournit des performances compétitives, elle exige des ressources spécialisées et elle est dépendante d'une autre tâche pour extraire les EPs candidates.

Les méthodes d'étiquetage séquentiel occupent une bonne partie de la littérature d'identification, puisque les premières méthodes d'identification avaient tendance à inclure la tâche d'identification dans d'autres tâches basées sur l'étiquetage séquentiel pour faire face à la rareté des ressources annotées en EPs. Suite à l'apparition récente de jeux de données dédiés aux EPs (Schneider et al., 2016; Savary et al., 2017), l'utilisation de cette approche s'est orientée vers l'identification des EPs exclusivement et les travaux se sont multipliés. Certaines méthodes se servent de l'ordre linéaire de la phrase en associant à une séquence de tokens une unique séquence d'étiquettes. D'autres profitent de la structure syntaxique pour construire leurs séquences et les étiqueter. Des modèles statistiques ou neuronaux apprennent, sur ces données, à prédire la séquence d'étiquettes à partir de la séquence d'entrée.

1. Nous utilisons plusieurs notations pour les exemples contenant des EPs. Si l'on ne cite que l'EP, on utilise simplement des italiques (par exemple *parce que*). Si on cite une occurrence d'une seule EP au sein d'une portion plus large, les composants de l'EP sont notés en gras. Si on cite une phrase comprenant plusieurs occurrences d'EPs, on utilise un indiçage des composants pour repérer les ensembles de composants formant chaque EP.

L'apparition des jeux de données PARSEME, à l'occasion d'une compétition internationale sur l'identification des EPs verbales (Savary et al., 2017; Ramisch et al., 2018) forme un tournant majeur de ce domaine. Ces données comprennent des phrases annotatées pour les EPs verbales [2], pour une vingtaine de langues, couvrant différentes familles linguistiques. Comparées aux autres catégories, les EPs verbales subissent plus de discontinuités et de variations morphologiques et syntaxiques, puisque la tête verbale peut en général être fléchie et peut recevoir des modifieurs. En outre, certaines catégories d'EPs verbales, en particulier les constructions avec verbe support (*prendre décision*), autorisent des variations syntaxiques plus complexes (extraction, coordination, passivation, etc.). Ainsi, ces données représentent une belle opportunité pour le développement, l'expérimentation et l'évaluation des méthodes d'identification.

2 Objectifs et méthodologie

Dans notre travail de thèse, nous avons concentré nos efforts sur une approche très utilisée en traitement automatique des langues, l'analyse automatique par transitions. L'objectif principal de la thèse est d'étudier en profondeur l'application de cette approche à l'identification automatique des EPs. Nous partons de l'hypothèse que cette approche, couramment utilisée pour l'analyse syntaxique en dépendances, est capable de relever de nombreux défis de la tâche d'identification. En particulier, dans cette thèse, nous voulons montrer que la méthode proposée est capable de modéliser l'idiosyncrasie des EPs et de bien gérer la variabilité, la discontinuité, l'ambiguïté et l'enchâssement qui peuvent caractériser les EPs, tout en gérant les EPs non vues dans le corpus d'apprentissage de manière relativement satisfaisante. La méthode a aussi l'avantage de pouvoir être générique et indépendante de la langue traitée, et peu exigeante en ce qui concerne l'expertise linguistique.

L'analyse automatique par transitions est connue pour sa capacité à décomposer des problèmes de prédiction de structure en une séquence de problèmes de prédiction locale. Un système par transitions prend en entrée une séquence d'éléments et construit une sortie structurée à partir de ceux-ci, par exemple un arbre de dépendances syntaxiques à partir d'une séquence de tokens. Cette construction se fait en appliquant une séquence d'actions, dites « transitions », choisies dans un ensemble prédéfini, pour construire incrémentalement la structure de sortie. Lors de l'analyse de l'entrée, chaque transition est fournie par un oracle ou prédite par un classifieur, d'après l'état courant du système, appelé une « configuration ».

Dans les systèmes par transitions à pile, la configuration est représentée par un tampon d'éléments (restants à traiter), une pile d'éléments en cours de traitement et un ensemble des structures construites ou en cours de construction. Nous faisons l'hypothèse que le contexte structuré et ciblé, présent sur les configurations d'une méthode d'analyse par transitions, peut permettre à cette méthode de répondre aux défis de variabilité et de discontinuité. Nous montrons en particulier que le cadre formel de l'analyse par transitions offre un terrain de jeu flexible pour produire un système avec une puissance expressive plus élevée que celle des méthodes d'étiquetage séquentiel. Profitant de l'apparition des jeux de données PARSEME, nous proposons une méthode d'analyse automatique par transitions, dont l'ensemble des transitions est capable de gérer la discontinuité et certains cas d'enchâssement.

Le choix des transitions à appliquer étant délégué à un classifieur, nous nous intéressons

2. Une EP verbale est une EP dont la forme canonique a pour tête syntaxique un verbe, et dont la distribution est celle d'un verbe, d'un syntagme verbal ou d'une phrase (Candito et al., 2017). Elles sont plus rares que les EPs grammaticales, mais tout aussi voire plus cruciales à traiter pour des applications sémantiques.

ensuite à différents modèles linéaires et non linéaires de classification que nous intégrons dans notre système d'identification. Dans un premier temps, nous nous sommes intéressé aux modèles linéaires de type machine à vecteur support (SVM), qui permettent d'obtenir d'excellents résultats sur la tâche d'identification des expressions verbales dans une vingtaine de langue. Dans un deuxième temps, nous nous sommes intéressé aux réseaux de neurones représentant des modèles non linéaires de classification, qui montrent des performances remarquables sur d'autres tâches du TAL. Nous avons en particulier étudié les réseaux utilisés par les méthodes d'analyse syntaxique en dépendances par transitions, dans l'objectif de proposer des architectures neuronales les plus pertinentes possibles pour l'identification avec une analyse par transitions. Les réseaux que nous étudions se nourrissent de certains éléments constitutifs de la configuration pour prédire la prochaine transition à appliquer. Nous nous plaçons pour la majorité des expériences dans un scénario où l'on dispose d'informations morphologiques sur ces éléments (lemme et étiquettes morphosyntaxiques) mais pas d'informations syntaxiques. L'analyse morphologique et l'analyse syntaxique étant cependant liées à l'identification d'EPs, nous avons également abordé en fin de thèse leur réalisation conjointe.

Suivant les traces de Chen et Manning (2014), de Weiss et al. (2015) et d'Andor et al. (2016), nous commençons par étudier un simple perceptron multicouche (MLP) avec une seule couche cachée. Malgré sa compétitivité pour les systèmes d'analyse syntaxique en dépendances, le MLP se révèle dans un premier temps peu robuste et non compétitif pour notre tâche, notamment à cause de la relative rareté des EPs verbales dans les données PARSEME (cf. un ratio d'une EP verbale annotée tous les 75 tokens environ, dans les jeux de données PARSEME 1.0) qui produit une distribution très déséquilibrée des transitions dans le jeu d'apprentissage. Nous avons alors exploité des techniques de ré-échantillonnage visant à équilibrer la répartition des classes (i.e. nos transitions), car elles sont réputées être efficaces dans un tel cas de figure (Chawla, 2009). Certaines techniques se sont révélées cruciales, notamment la stratégie de sous-échantillonnage consistant à éliminer les phrases sans EPs dans les jeux d'entraînement et la stratégie de sur-échantillonnage aléatoire des classes minoritaires pour obtenir une distribution uniforme. Ces techniques ont permis de stabiliser l'apprentissage du modèle et de le rendre compétitif. Nous avons par ailleurs mis au point une méthode robuste translingue de réglage des combinaisons d'hyperparamètres, qui analyse les « tendances » des valeurs d'hyperparamètres au sein d'une recherche aléatoire. Ces différentes stratégies ont permis d'obtenir des scores globaux dépassant l'état-de-l'art sur les données PARSEME pour l'identification des expressions verbales.

Bien que globalement performant, le MLP se caractérise de manière très décevante par son incapacité quasi totale à identifier dans les corpus d'évaluation les EPs qui n'ont pas été préalablement vues, exactement ou avec certaines variations, dans les données d'apprentissage. Afin d'augmenter la capacité de généralisation de notre système, nous proposons des variantes contextualisées de notre modèle neuronal. Dans un premier temps, nous proposons une variante permettant d'augmenter le contexte des éléments ciblés dans la configuration. Puis, nous proposons d'intégrer des couches récurrentes qui permettent de modéliser des structures complexes telles que les séquences et les arbres en conservant une bonne part de leurs traits structurels (Goldberg, 2017).

Enfin, nous étudions une approche par apprentissage multitâche, réalisant et mettant à profit conjointement l'étiquetage morphosyntaxique, l'identification des EPs par transitions et l'analyse syntaxique en dépendances par transitions. L'idée est que le système d'identification s'appuie sur deux tâches auxiliaires qui se sont déjà révélées complémentaires dans la littérature de l'identification (Constant et Tellier, 2012; Candito et Constant, 2014; Nasr et al., 2015). Nous avons utilisé l'approche proposée par Zhang et Weiss (2016) qui consiste à empiler des modules neuronaux dédiés à des tâches diverses (pour eux, étiquetage morphosyntaxique et analyse syntaxique) sur

une couche partagée de contextualisation des éléments en entrée. Les paramètres de cette couche étant partagés entre les différents modules, la mise à jour des paramètres (incluant ceux de la couche partagée) lors de l'apprentissage d'une tâche impactera forcément les autres. Bien que globalement moins performantes que notre simple perceptron multicouche, toutes ces variantes se sont révélées prometteuses en termes d'identification des EPs inconnues, l'une des faiblesses du MLP.

3 Plan de la thèse

Cette thèse s'organise en deux parties, en plus d'une introduction générale et d'une conclusion. La première partie comprend quatre chapitres, alors que la deuxième est composée de six chapitres.

La première partie intitulée **État de l'art** présente, comme son nom l'indique, une vue d'ensemble de la littérature sur les EPs et la tâche d'identification, ainsi qu'une introduction aux concepts et techniques fondamentaux utilisés dans cette thèse.

— Le premier chapitre porte sur la définition d'une EP, ses propriétés linguistiques telles que l'idiosyncrasie, la variabilité et l'ambiguïté ainsi que ses propriétés formelles : discontinuité, enchâssement et chevauchement. Le chapitre aborde également les catégories linguistiques des EPs des jeux de données exploités.

— Dans le deuxième chapitre, nous définissons la tâche d'identification des EPs et détaillons ses motivations et les défis qu'elle représente. Ce chapitre présente également les jeux de données, leurs schémas d'annotation, leurs formats et les mesures d'évaluation que nous utilisons dans nos expérimentations. Nous détaillons ici également deux approches d'identification non centrales pour notre travail : l'identification à base de règles et l'identification par classification binaire de candidats (préannotés ou identifiés automatiquement).

— Le troisième chapitre est consacré aux méthodes d'étiquetage de séquences, très utilisées en TAL et pour l'identification d'EPs. Il commence par présenter les méthodes d'étiquetage séquentiel qui intègrent l'identification au sein d'une autre tâche. Ensuite, le chapitre présente des exemples de deux types de méthodes d'étiquetage séquentiel consacrées à l'identification exclusivement : celles qui se basent sur l'ordre linéaire des tokens de la phrase et celles qui profitent de la structure syntaxique et de leur connectivité.

— Dans le quatrième chapitre, nous présentons les différents concepts et techniques qui seront au coeur de la thèse, avec tout d'abord l'analyse automatique par transitions que nous avons intensément utilisée dans cette thèse. Nous montrons notamment un exemple de tels systèmes pour l'analyse syntaxique en dépendances, ainsi que les réseaux de neurones utilisés pour ce genre de systèmes. Nous introduisons enfin le principe de l'apprentissage multitâche en donnant quelques exemples d'architectures neuronales multitâches.

La seconde partie intitulée **Méthode d'analyse automatique par transitions pour l'identification des EPs** décrit notre méthode d'identification, nos modèles et nos expéri-

mentations.

— Le premier chapitre décrit formellement notre système par transitions et les différents ensembles de transitions qu'utilise ce système. Il définit également l'oracle et l'algorithme d'analyse et caractérise la puissance expressive du système.

— Le deuxième chapitre présente le modèle linéaire que nous avons utilisé ainsi que les patrons de traits utilisés. Le chapitre présente également les expérimentations menées avec ce modèle sur les données PARSEME 1.0, ainsi que leur cadre expérimental. Il compare sa performance avec la performance d'un système de base et de systèmes existants. Puis, il analyse l'impact des différents patrons de traits sur la performance. Nous avons participé à la première compétition internationale PARSEME d'identification des EPs verbales sur 18 langues, avec une première variante linéaire de notre méthode, qui a produit les scores de l'état de l'art pour la plupart des langues concernées (Al Saied et al., 2017).

— Dans le troisième chapitre, nous présentons un réseau de neurones basique pour remplacer notre modèle linéaire. Nous commençons le chapitre avec une description des couches lexicales, responsable de la production des plongements lexicaux et des plongements d'étiquettes morphosyntaxiques, en entrée du modèle. Ensuite, nous définissions le modèle neuronal MLP et les techniques de ré-échantillonnage utilisées pour la stabilisation de l'apprentissage. Les autres sections rendent compte des expérimentations du modèle ainsi que des analyses de l'impact du ré-échantillonnage et de la fréquence des EPs des jeux d'évaluation vues dans le corpus d'apprentissage. Nous analysons également l'impact de la taille du corpus et du type de vocabulaire utilisé. La dernière section présente une structure, empilant nos deux modèles, linéaire et neuronale.

— Dans le quatrième chapitre, nous présentons des modèles MLP intégrant davantage de contexte, à l'aide d'éléments ciblés supplémentaires ou de couches récurrentes. Le chapitre présente également un modèle plus évolué, dit KG-2016, utilisant un MLP classique alimenté par les éléments ciblés de la configuration plongés dans le contexte de la phrase en entrée. Les autres sections offrent une vue sélective des expérimentations que nous avons réalisées avec les différentes variantes de notre système.

— Le cinquième chapitre présente l'architecture multitâche que nous proposons pour les tâches d'identification, d'étiquetage morphosyntaxique et de l'analyse syntaxique en dépendances. Ce chapitre décrit également les expérimentations menées pour cette variante.

— Le sixième chapitre décrit l'utilisation de notre système pour une étude dans le domaine des neurosciences. Nous décrivons aussi un système récent de la littérature s'appuyant sur notre travail. Enfin, nous présentons les détails techniques de la mise en œuvre de nos différentes variantes.

Enfin, la conclusion générale revient sur les principaux résultats et constats produits au cours de cette thèse et présente des perspectives possibles.

Première partie

Identification des EPs : un état de l'art

Chapitre 1

Expressions polylexicales (EPs)

Ce chapitre introduit la notion d'expressions polylexicales. Il commence par donner une définition des EPs, puis par distinguer ces éléments des autres constructions linguistiques relativement liées telles que les collocations et les termes (section 1.1). La section 1.2 présente les propriétés linguistiques et formelles principales de ces constructions. La section 1.3 aborde les typologies proposées des EPs et les différentes catégories linguistiques des EPs des jeux de données exploités par notre méthode. Enfin, la section 1.4 présente les critères d'annotation suivis par les jeux de données exploités dans cette thèse.

Notons que nous nous basons principalement sur les travaux de Sag et al. (2002); Baldwin et Kim (2010) et Constant et al. (2017) pour aborder les différents aspects linguistiques et computationnels de ces constructions.

1.1 EPs : Définition

La notion d'EPs (expressions polylexicales) [3] est connue pour être difficile à cerner, comme l'est d'une manière plus générale la notion de mot (Baldwin et Kim, 2010). Elle est pourtant cruciale en linguistique et en TAL (Traitement Automatique des Langues), car particulièrement fréquente à l'échelle du lexique : Jackendoff (1997) estime que le nombre d'EPs dans le lexique d'un locuteur est comparable au nombre de mots simples et l'étude Fellbaum (1999) montre que 41 % des entrées de WordNet 1.7 sont des multimots. Les données de la compétition internationale PARSEME 1.0, sur l'identification des EPs verbales, ont un ratio d'une EP verbale annotée tous les 75 tokens environ.

Une définition purement statistique de la notion d'EP est celle de Carpuat et Diab (2010), qui considèrent comme EPs les séquences qui coexistent statistiquement plus que le hasard. Une définition distributionnelle, approximative, est celle d'une séquence de "mots" qui se comporte comme une seule unité à un certain niveau de l'analyse linguistique (Calzolari et al., 2002). Comme certaines EPs (les continues) sont parfois considérées comme des mots, on peut reformuler la définition en « une séquence d'éléments qui se comportent chacun comme un mot dans d'autres contextes, mais qui lorsqu'utilisés ensemble peuvent former une unité lexicale ». Il est cependant peu satisfaisant que cette reformulation de type « mot avec espaces » s'appuie sur les conventions

3. Nous avons choisi d'utiliser des acronymes parfois anglais et parfois français. Nous gardons en particulier les acronymes anglais pour les termes du domaine de l'apprentissage automatique (CRF, MLP, LSTM, ..). Pour les termes linguistiques, nous utilisons en général les acronymes français, sauf par exemple POS pour étiquettes morphosyntaxique. Dans tous les cas, la première mention d'un acronyme est accompagnée de sa forme non abrégée, ainsi que de sa traduction en français s'il s'agit d'un acronyme anglais.

typographiques. D'une part elle s'applique mal aux langues sans séparateurs de mots, comme la plupart des langues asiatiques. D'autre part certains éléments classiquement considérés comme EPs contiennent des symboles spéciaux mais pas d'espaces, au moins dans certaines variantes. C'est le cas par exemple des variantes *c'est à dire* ou *c'est-à-dire*. De plus, certains éléments séparés par des espaces ne sont pas des mots, car jamais présents en dehors de l'EP (comme par exemple *fur* dans *au fur et à mesure*). Enfin, le terme EP peut également être utilisé pour une séquence discontinue de mots, qui ne peut pas du coup être considérée comme ayant le même statut qu'un seul mot.

Sag et al. (2002) se basent, quant à eux, sur le caractère imprévisible du comportement morphologique, syntaxique et sémantique des EPs, qu'ils définissent comme des interprétations idiosyncrasiques qui transcendent les frontières de mot, mettant ainsi l'accent sur l'aspect idiosyncrasique de la sémantique des EPs.

Dans leur article de référence, Baldwin et Kim (2010) tentent de réunir tous ces aspects en définissant l'EP comme un élément lexical qui : **(a)** peut être décomposé en plusieurs lexèmes ; et **(b)** affiche une idiosyncrasie lexicale, syntaxique, sémantique, pragmatique et/ou statistique. Nous suivons les pas de Constant et al. (2017) en adoptant cette définition, qui offre un cadre flexible pour aborder l'identification de ces entités, en capturant tous les aspects mentionnés par les définitions précédentes, et en insistant sur les différents niveaux linguistiques sur lesquels les EPs peuvent avoir un impact. Nous terminons cette section de définition par les liens des EPs avec deux concepts liés, les collocations et les termes.

1.1.1 EPs versus collocations

La notion de collocation est liée à celle d'EP, les collocations étant, selon les auteurs, distinctes ou bien incluses au sein des EPs. Ainsi par exemple Evert (2005) définit une collocation comme une combinaison de mots dont les propriétés sémantiques et/ou syntaxiques ne peuvent pas être entièrement prédites à partir de celles de ses composants, et qui doivent donc être répertoriées dans un lexique, ce qui du coup se lit comme un synonyme d'EP. Dans un sens plus étroit, elles sont comprises comme des paires de mots semi-compositionnels, avec un élément « libre » (la base) et un autre élément déterminé par la base (le colocataire). Mais généralement les collocations sont définies sur une base au moins en partie statistique et sont placées quelque part dans la zone grise entre les idiomes fixes et les combinaisons libres. Ainsi Baldwin et Kim (2010) les définissent comme des « EPs statistiquement idiomatiques » (*statistically idiomatic MWE* en anglais), c'est-à-dire pour lesquelles la ou une des idiosyncrasies est d'avoir une fréquence en corpus plus importante qu'attendu.

Dans toute la suite, nous suivrons la terminologie du projet PARSEME [4], étant donné que nous utilisons les jeux de données issues des compétitions internationales PARSEME 1.0 et PARSEME 1.1 pour la majeure partie de nos expérimentations (données dont la typologie est décrite infra, section 1.3.1, et qui sont plus détaillées sections 2.3.1 et 2.3.2). Le guide d'annotation PARSEME 1.1 [5] oppose clairement les EPs aux collocations, ces dernières étant comprises comme des « combinaisons de mots dont l'idiosyncrasie est purement statistique. En d'autres termes, les tokens dans les collocations ont tendance à coïncider plus souvent que prévu par hasard, mais ils ne montrent aucune idiosyncrasie substantielle sur le plan orthographique, morphologique, syntaxique et (plus particulièrement) sémantique. »

4. `https://typo.uni-konstanz.de/parseme/`
5. `https://parsemefr.lif.univ-mrs.fr/parseme-st-guidelines/1.1/`

1.1.2 EPs versus termes

Les termes techniques sont également une notion qui recoupe partiellement celle d'EP. Le mot « terme » s'applique aux mots simples ou aux EPs désignant des concepts spécifiques à un domaine, en général technique. Il a donc une portée plus large que les EPs car il couvre des mots simples techniques, et plus étroite dans le sens où les EPs non techniques ne sont pas des termes. En particulier, les EPs grammaticales (i.e. de catégorie morphosyntaxique fermée) ne sont a priori pas des termes. Dans les données PARSEME que nous utilisons le plus souvent, les EPs verbales sont rarement techniques, et ne sont pas des termes. Dans le FTB, que nous utilisons également, qui contient tout type d'EP, la notion de terme n'a pas été utilisée, les mêmes critères linguistiques s'appliquant, quel que soit le champ sémantique de l'EP (voir infra section 2.3.3 pour les critères utilisés dans le FTB).

1.2 Propriétés linguistiques et formelles des EPs

Dans cette section, nous allons discuter des différentes propriétés caractéristiques des EPs, avec tout d'abord la caractéristique définitoire qu'est l'idiosyncrasie, puis des caractéristiques interprétatives (l'ambiguïté), morphosyntaxiques (la variabilité), ainsi que les propriétés formelles des EPs (la discontinuité, l'enchâssement et le chevauchement).

1.2.1 Idiosyncrasie

L'idiosyncrasie est l'une des caractéristiques de base des EPs et fait référence à l'impossibilité dans le cas d'une EP à admettre une variation (lexicale, syntaxique, sémantique) qui serait normalement possible étant donnés les composants de l'EP et sa structure.

Idiosyncrasie lexicale Elle se traduit par l'occurrence d'un ou de plusieurs composants d'une EP qui n'appartiennent pas au lexique conventionnel (comme *f*ur dans *au fur et à mesure*). On parle également de mot « cranberry » dans ce cas. Ce type d'idiosyncrasie entraîne forcément une idiosyncrasie syntaxique et sémantique en raison de l'absence de connaissances lexicales associées directement aux parties de l'EP (Baldwin et Kim, 2010).
On pourrait également parler d'idiosyncrasie lexicale (d'un autre type), dans le cas d'une invariabilité lexicale inattendue : l'impossibilité ou le changement de sens inattendu en cas de remplacement d'un composant par un synonyme ou antonyme est un critère fréquemment utilisé (utilisé en particulier par le FTB, et dans le guide PARSEME sous le titre « lexical inflexibility »).

Idiosyncrasie syntaxique Cette idiosyncrasie concerne deux cas :

— D'une part une irrégularité « interne », où la séquence des composants ne forme pas une structure syntaxique habituelle, comme par exemple l'adverbe *bel et bien*, composé d'une coordination irrégulière entre un adjectif et un adverbe ;

— d'autre part une irrégularité « externe », où l'on peut bien repérer une structure syntaxique régulière dans la séquence des composants de l'EP, mais où la distribution de l'EP n'est pas celle attendue pour cette structure. Par exemple *fleur bleue* a une structure régulière, mais sa distribution attendue est celle d'un nom, alors qu'il a une distribution d'adjectif

Idiosyncrasie sémantique (non compositionnalité) On parle d'idiosyncrasie sémantique ou non compositionnalité sémantique lorsque le sens d'une EP ne peut pas être déduit totalement

de ses parties (Sag et al., 2002). Baldwin et Kim (2010) soulignent l'importance de la « figuration » dans la formation d'une EP, avec des cas de métaphore lexicalisée (*hurler avec les loups : se ranger à l'opinion du plus grand nombre ou des plus forts*), d'hyperbole lexicalisée (*mourir de soif*) ou d'expression incluant une métonymie (*Vivre de sa plume : Avoir pour profession écrivain*).

L'idiosyncrasie sémantique peut avoir divers degrés. On peut avoir compositionnalité pratiquement totale, par exemple avec *traduction automatique*, semi-compositionnalité comme pour *vin rouge* ou non compositionnalité par exemple dans *fleur bleue (sentimental, romanesque)*.

Par définition les EPs non compositionnelles et semi-compositionnelles ne peuvent en général pas être traduites littéralement, et posent ainsi des défis en traduction automatique. À noter cependant que les EPs les plus répandues sont généralement perçues comme moins idiomatiques sur le plan sémantique (Keysar et Bly, 1995).

Idiosyncrasie pragmatique On parle d'idiosyncrasie pragmatique lorsqu'une EP est associée à un contexte extralinguistique particulier, comme par exemple *à table*. De part l'impact du contexte extralinguistique, ces EPs sont particulièrement ambiguës pour des outils de TAL, entre lecture idiomatique et littérale.

Idiosyncrasie statistique L'idiosyncrasie statistique correspond à une fréquence de cooccurrence arbitrairement importante (Constant et al., 2017), et est une caractéristique fréquente des EPs. Par exemple, l'adjectif *grand* accompagne le nom *cru*, pour former la collocation *de grand cru,*, plus fréquemment que d'autres adjectifs tels qu'*éminent* ou *prestigieux*. L'idiosyncrasie de fréquence peut porter également sur l'ordre des mots (par exemple *noir et blanc*). À noter cependant que dans les données que nous avons utilisées, l'idiosyncrasie statistique n'est jamais un critère suffisant d'annotation. Pour le guide d'annotation PARSEME cité supra, le terme collocation est réservé à des combinaisons fréquentes mais non EP .

1.2.2 Ambiguïté

Une même séquence peut selon le contexte avoir une lecture idiomatique (i.e. comme EP) versus une lecture littérale. Par exemple l'EP *prendre la porte* signifie *quitter une pièce* dans *Il a fini par prendre la porte, en colère.*, alors qu'elle a un sens littéral dans *Puis-je prendre la porte entreposée dans la cave ?*. On peut également avoir des occurrences « accidentelles », où les composants d'une EP n'ont aucun lien syntaxique, comme par exemple la séquence *bien que* qui n'est pas EP dans *J'aime bien que mes collègues me suivent dans mes projets fous.*.

1.2.3 Variabilité

Si les EPs grammaticales peuvent souvent être totalement invariables et continues (comme par exemple *parce que*, qui n'admet que l'élision du *que* comme variation), ce qui facilite leur identification, ce n'est pas le cas d'autres EPs, en particulier verbales et nominales, qui admettent une variation morphologique, voire syntaxique. Par exemple, pour la très grande majorité, le verbe des EPs verbales peut se fléchir de manière régulière. Des variations syntaxiques régulières peuvent être possibles (par exemple *rendre hommage* peut être utilisé au passif personnel (*Un vibrant hommage a été rendu...*) ou impersonnel (*il a été rendu un vibrant hommage...*). La régularité syntaxique est totale, en particulier pour les Constructions à Verbe Support (CVSs), où le nom peut être relativisé, questionné, le verbe peut apparaître comme une participiale modifiant le nom (*les décision prises hier*), etc.

1.2.4 Propriétés formelles : discontinuité, enchâssement et chevauchement

La variabilité citée supra peut se traduire par des discontinuités au sein d'une EP. Elle peut intervenir en particulier en cas d'ajout de modifieurs de composants de l'EP, par exemple un adverbe pour les EPs verbales, un modifieur du nom pour une CVS. Les Constructions de Verbes à Particule (CVPs) sont particulièrement sujettes à discontinuités [6]. À noter qu'au sein d'une discontinuité, on peut avoir des EPs, comme dans l'exemple anglais *Take*$_1$ *the fact that I did'nt give*$_2$ *up*$_2$ *into*$_1$ *account*$_1$*. (lit. Prenez le fait que je n'ai pas donné haut en compte ; Prenez en compte que je n'ai pas abandonné).*

Nous parlerons d'enchâssement d'une EP $_1$ dans une autre EP $_2$ lorsque tous les tokens d'EP_1 appartiennent à EP $_2$. Par exemple, dans la phrase *Je ne voudrais pas faire un faux pas*, *faux pas* est enchâssée dans la construction à verbe support *faire faux pas.*

Enfin, on parlera de chevauchement lorsque deux EPs partagent certains tokens, mais aucune n'est totalement contenue dans l'autre. Le phénomène peut apparaître en particulier dans le cas de coordination, lorsqu'un des tokens est partagé et non répété, en particulier pour les CVSs (*prendre*$_{1,2}$ *les décisions*$_1$ *et mesures*$_2$ *qui s'imposent*).

1.3 Typologies et catégories d'EPs

Il existe plusieurs manières de catégoriser les EPs. Sag et al. (2002) réunit la plupart des catégories d'EPs en une seule typologie arborescente, que nous reproduisons dans la figure 1.1. Cette classification se focalise sur les caractéristiques statistiques, syntaxiques et sémantiques des EPs. La distinction première sépare les EPs institutionnalisées et les EPs lexicalisées. **Les EPs institutionnalisées** ne sont idiosyncrasiques que sur le plan statistique (i.e. cela rejoint le terme de « collocation » au sens du guide PARSEME cité supra, section 1.1.1). À noter que cette dichotomie est clairement schématique, puisque les EPs lexicalisées ont également en général une idiosyncrasie statistique.

Les EPs lexicalisées, quant à elles, présentent une idiosyncrasie lexicale, syntaxique, sémantique ou/et pragmatique. Elles sont distinguées selon leur flexibilité syntaxique, en EPs fixes, EPs semi-fixes et et EPs syntaxiquement flexibles. **Les expressions fixes** sont complètement lexicalisées et n'admettent ni variation morphosyntaxique ni modification interne, ou contiennent des composants n'apparaissant pas indépendamment de l'expression (Sag et al., 2002). Par exemple, *ad hoc* et *de facto* sont des EPs issues du latin, dont les composants n'appartiennent même pas au dictionnaire. **Les expressions semi-fixes** admettent quant à elles certaines variations, comme la flexion de certains composants, la sélection de déterminants, mais n'admettent pas de variation dans l'ordre de leurs composants. Par exemple, l'expression *Prendre la porte* est une EP régulière sur le plan syntaxique et sémantique, le verbe se fléchit de manière régulière, en revanche l'expression perd son sens idiomatique si son complément, *la porte*, est au pluriel.

Un autre angle d'approche consiste à classer les EPs d'après leur catégorie morphosyntaxique, elle-même déterminée d'après la distribution de l'EP. Tout en reprenant la typologie à la Sag et al. (2002), Baldwin et Kim (2010) font une description des propriétés typiques des EPs pour trois classes morphosyntaxiques principales : les EPs nominales, verbales et prépositionnelles. Le guide d'annotation du FTB Abeillé et Clément (2003) fournit également des exemples typiques

6. Dans la section décrivant plus précisément les données PARSEME 1.0 et PARSEME 1.1, sections 2.3.1 et 2.3.2, nous reproduisons des tableaux de (Savary et al., 2017) donnant la fréquence des discontinuités selon les types d'EPs verbales et les langues.

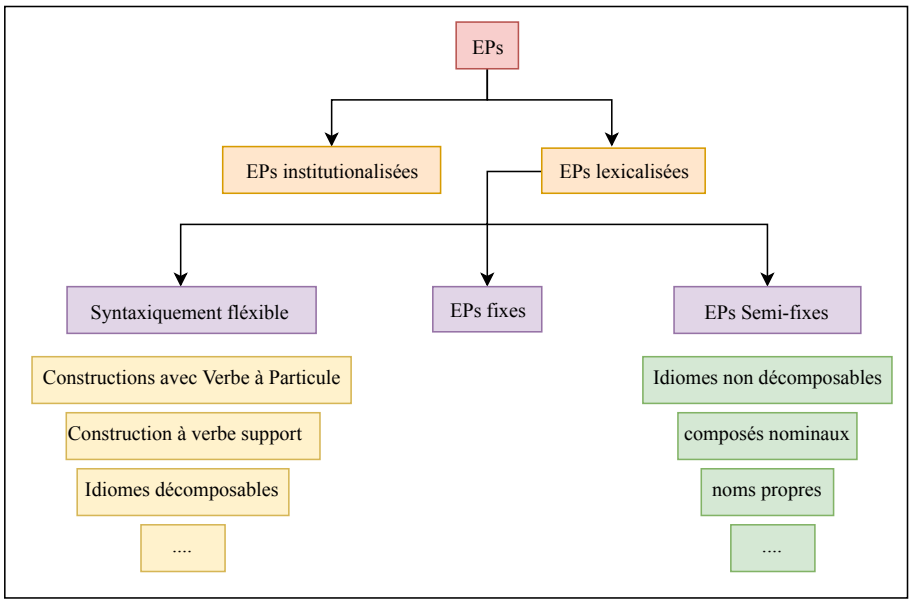

FIGURE 1.1 – **EPs - classification de Sag et al. (2002)** La classification des EPs selon Sag et al. (2002)

pour les différentes catégories morphosyntaxiques d'EPs. À noter cependant qu'il s'agit de caractéristiques typiques mais non définitoires. L'approche du projet PARSEME est en cela différente : les différentes catégories d'EPs verbales y sont définies de manière opératoire, et l'approche est validée pour les 20 langues couvertes par le projet.

Nous résumons infra d'abord la typologie précise des EPs verbales du projet PARSEME, puis donnons des exemples d'EPs pour les différentes autres catégories morphosyntaxiques.

1.3.1 Les EPs verbales dans les données PARSEME

Dans ce projet, une EP verbale est une EP dont la forme canonique a pour tête syntaxique un verbe, et dont la distribution est celle d'un verbe, d'un syntagme verbal ou d'une phrase (Candito et al., 2017). Une typologie des EPs verbales a été définie (légèrement modifiée entre les campagnes PARSEME 1.0 et PARSEME 1.1), et les différents types retenus sont associés à des arbres de décision précis fondés sur des tests linguistiques. Par exemple, une construction doit être annotée comme idiome verbal si (1) le verbe de la construction n'a pas un et un seul dépendant syntaxique lexicalisé, comme en *prendre le taureau par les cornes* ou (2) le verbe de l'expression a un et un seul dépendant syntaxique lexicalisé, mais ne satisfait pas les critères des constructions de verbe à support.

Nous détaillons maintenant cette typologie d'EPs verbales, dans sa version la plus à jour (i.e. celle de la campagne PARSEME 1.1) [7].

Constructions avec verbe à particule (CVP) Les Constructions avec Verbe à Particule (CVPs) consistent en un verbe et une ou plusieurs particules, telles que *break down* (lit. *casser bas ; tomber en panne*) [8]. Une condition suffisante est si la même éventualité ne peut pas être

7. La majorité des exemples de cette section sont issus du guide d'annotations de PARSEME 1.1. Les informations détaillées sur ces jeux de données sont fournies sections 2.3.1, 2.3.2.

8. Les significations que nous fournissons pour les expressions sont issues du dictionnaire Larousse pour les

évoquée avec le verbe sans la particule (par exemple *to do somebody in* (lit. *faire quelqu'un dedans ; tuer quelqu'un*) n'implique pas *to do somebody* (lit. *faire quelqu'un*). À l'inverse, les cas où la particule a un sens spatial ou directionnel n'entrent pas dans cette catégorie, alors qu'en font partie les cas où le même type de particule perd son sens spatial ou directionnel : Sag et al. (2002) mentionnent l'exemple de *eat up* (lit. *manger vers-le-haut, manger entièrement*), où la particule *up* a un sens d'achèvement.

Les CVPs sont omniprésentes dans les langues germaniques telles que l'anglais, l'allemand, le suédois, bien qu'elles ne soient pas limitées à cette famille de langues, mais sont peu fréquentes en langues romanes et slaves ou en farsi et grec, par exemple. Pour certaines langues germaniques et également pour le hongrois, les CVPs peuvent selon leur contexte syntaxique être orthographiées en deux tokens ou bien un seul token sans séparateur typographique, auquel cas le token est tout de même annoté comme EP (cas que nous appelons TP (Token polylexical), cf. section 2.3.1.0).

Les CVPs transitives peuvent accepter un groupe nominal placé soit entre le verbe et la particule *pull yourself up* (**s'arrêter**) ou bien après la particule *I'll take down the Web site* (*Je vais noter le site*). Les adverbes peuvent souvent être insérés entre le verbe et la particule *he was staring stubbornly up* (*Il matait obstinément*) en particulier dans le cas intransitif. Deux cas de CVP sont considérés dans la typologie PARSEME 1.1, selon que l'ajout de la particule provoque un changement de sens « allant significativement au-delà de l'ajout de la particule » [9] (**CVP.full**), ou bien au contraire, l'ajout de la particule donne une expression dont le sens est partiellement prédictible (**CVP.semi**). À noter que les cas de particules à sens spatial conservé ne sont pas considérés comme CVP.

Constructions à verbe support (CVS) Dans le guide d'annotation PARSEME, les constructions à verbe support (CVS) sont des expressions formées d'un verbe v et d'un nom n (lui-même simple ou composé), qui est un dépendant syntaxique (direct ou indirect) du verbe (dans la forme canonique de l'expression) [10]. Le nom doit évoquer une éventualité (au sens évènement (décision, visite) ou état (peur, courage)), et doit être prédicatif, au sens d'avoir au moins un argument sémantique. Une CVS peut être réduite en un syntagme nominal sans le verbe, dans lequel un argument syntaxique du verbe v (en général le sujet) devient un dépendant de n (*Luc* ***prend*** *une* ***décision*** *difficile* → *la* ***décision*** *difficile de Luc*).

Deux cas de CVSs sont considérés dans la typologie PARSEME 1.1 :

— **CVS.full** : le verbe est « léger », en ce sens où sa contribution sémantique à l'expression correspond uniquement à porter les marques flexionnelles habituelles pour les verbes (personne, nombre, temps, mode et aspect). En particulier le sujet du verbe est un argument sémantique du nom **n**. À noter que cela peut provenir du fait que le verbe n'a pas son sens habituel (par exemple dans *rendre visite*, *rendre* n'a pas son sens habituel, ou bien que le verbe a intrinsèquement un sens très général, qui se trouve redondant par rapport au sens du n (comme par exemple *commettre un crime*). Ces CVSs sont idiosyncrasiques en cela qu'il peut être difficile de prédire quel verbe support se combine à un nom donné (Abeille, 1988).

— **CVS.cause** : le verbe a un sens causatif : le sujet du **v** est la cause ou la source de l'éventualité exprimée par n, les compléments du v autres que le nom n sont des arguments sémantiques du n (comme par exemple dans *X donner envie à Y de Z*).

expressions françaises et du dictionnaire anglais-français de Cambridge pour les expressions anglaises.

9. Cf. le guide PARSEME 1.1 `https://parsemefr.lis-lab.fr/parseme-st-guidelines/1.1/?page=050_Cross-lingual_tests/050_Verb-particle_constructions__LB_VPC_RB_`.

10. Pour le Hindi, ont été incluses comme CVS des expressions formées d'un verbe et d'un adjectif.

Les CVSs, qui se manifestent dans la plupart des langues, sont sujettes à une variabilité syntaxique complète, comme la passivisation *la décision a été prise*, l'extraction *Quelles décisions avez-vous prises ?*) et la modification interne *elle a pris une bonne décision*).

Verbes intrinsèquement réfléchis (VIRs) Les verbes réflexifs sont formés d'un verbe associé à un clitique ou pronom réfléchi (c'est-à-dire un clitique ou pronom qui peut être utilisé avec un sens réflexif : se référer au sujet du verbe, comme *me* dans *Je me regarde*, ou *herself* dans *She talks to herself*. L'important, cela dit, est que justement dans le cas des VIRs, le réfléchi n'a pas un sens réflexif. Plus précisément, sont annotés comme VIRs dans les données PARSEME, les cas de combinaison v+réfléchi (dans l'ordre pertinent pour la langue en question) où (a) le v ne peut pas être employé sans le réfléchi (*s'efforcer*), ou (b) les versions avec ou sans réfléchi ont des sens et/ou des cadres de sous-catégorisation clairement différents (par exemple *X agir (sur Y)* versus *il s'agir de Y*).

Constructions à plusieurs verbes (CPVs) Les Constructions à Plusieurs Verbes (CPVs) sont composées d'une séquence de deux verbes adjacents (dans un ordre dépendant de la langue), un verbe dominant et un verbe dépendant, comme *laisser tomber*. Le comportement de ces expressions est très hétérogène d'une langue à l'autre. Cependant, les deux verbes de l'expression partagent le même sujet et ne peuvent porter qu'un seul temps, aspect et polarité.

Verbes intrinsèquement adpositionnels (VIAs) Les Verbes Intrinsèquement Adpositionnels (VIAs) (terme plus général que les verbes prépositionnels) se composent d'un verbe ou d'une EP verbale et d'une adposition, caractérisés par le fait que **(i)** l'objet de la préposition n'est pas lexicalisé, **(ii)** l'absence de l'adposition est impossible (par exemple en anglais *rely* doit forcément être accompagné de *on*), ou **(iii)** elle modifie de manière significative le sens du verbe.

Idiomes verbaux (ID) Enfin la catégorie des Idiomes Verbaux (IDs) regroupe les autres cas d'EP verbale. Il peut s'agir d'EP formé d'un verbe et plus d'un dépendant plein, que le tout forme une phrase (*Rome ne s'est pas faite en un jour*) ou pas (*prendre le taureau par les cornes*), de cas à structure syntaxique irrégulière (*court-circuiter*), d'expressions dont la structure n'est pas couverte par les autres catégories (par exemple *lire en diagonale*) ou enfin d'expressions formées d'un verbe et un dépendant lexicalisé mais qui ne satisfont pas les critères des constructions à verbe support (CVS). Par exemple *X porter atteinte à Y* est classé ID et pas CVS, car on ne peut pas former un syntagme nominal de tête *atteinte* comportant les arguments X et Y.

Les tests pour identifier un ID (versus une expression compositionnelle) sont centrés sur la notion d'invariabilité inattendue, qu'il s'agisse d'invariabilité lexicale, morphologique ou syntaxique. Il est à noter que ces tests n'abordent pas de manière frontale la question de la compositionnalité sémantique, mais sont dans la mesure du possible des tests formels plutôt que sémantiques, tels qu'une invariabilité morphologique ou syntaxique inattendue. Il en résulte que les EPs dans PARSEME peuvent avoir différents degrés d'opacité sémantique. Les CVSs ne peuvent par définition pas être totalement opaques (cf. le nom doit avoir un de ses sens habituels). Un ID comme *cacher son jeu* a, quant à lui, un sens lié au sens de *cacher*, alors que *ne pas payer de mine* est en synchronie totalement opaque.

1.4 Critères d'identification d'EPs

Des critères d'identification d'EPs ont été définis dans la littérature, notamment Gross (1996) pour le français. Certains critères ont été mis à l'épreuve d'une annotation effective. Comme dit supra, les types définis dans le projet PARSEME s'accompagnent d'arbres de décision précis [11], que nous ne pouvons pas résumer au risque de les dénaturer. Nous résumons en revanche les critères définis pour l'annotation du FTB (corpus que nous utilisons pour certaines de nos expériences), et ceux du projet PARSEME-FR (émanation française du projet PARSEME) [12]. Dans les deux cas, les critères sont génériques et pas définis catégorie par catégorie.

1.4.1 Les critères d'EPs dans le FTB

Le FTB a longtemps été le seul corpus en français annoté pour les EPs. Les critères d'annotation des EPs sont relativement succincts dans le guide d'annotation (Abeillé et Clément, 2003), mais offre un résumé des critères typiques, notamment développé par Gross (1996). Un point important est que ces critères ne sont des conditions ni nécessaires ni suffisantes, ce qui laisse une marge importante d'interprétation aux annotateurs [13].

- Critères morphologiques ou graphiques :
  - Un des éléments n'apparaît que dans l'expression (*au **fur** et à mesure*).
  - Le genre ou le nombre de l'EP n'est pas celui attendu (*un peau-rouge*).
  - La séquence s'écrit avec un tiret ou une apostrophe (*court-circuiter*).
- Critères morphosyntaxiques :
  - L'EP a la même distribution qu'une catégorie terminale (par exemple *en vertu de* a la distribution d'une préposition). À noter que ce critère n'est pas du tout opératoire, cf. par exemple n'importe quel syntagme nominal va avoir la distribution d'un nom propre, ce qui n'en fait pas un composé pour autant.
  - La catégorie syntaxique de l'EP est inattendue d'après les composants de l'EP (*fleur bleue*).
  - La séquence de catégories des composants de l'EP n'est pas une séquence régulière (par exemple *à la va-vite* comprend une succession irrégulière d'un déterminant suivi d'un verbe).
  - Il n'y a pas ou peu d'insertion possible.
  - L'accord morphologique n'est pas respecté (*une grand-mère* mais **une grand**e**-mère*)
- Critères sémantiques :
  - La sémantique de l'EP n'est pas la composition sémantique du sens de ses composants (*carte bleue*).
  - On ne peut pas remplacer un constituant par un synonyme ou un antonyme (***grand** cru*).
  - L'EP peut correspondre à une seule unité dans d'autres langues (*pomme de terre* traduit par *potato* en anglais).

11. `https://parsemefr.lif.univ-mrs.fr/parseme-st-guidelines/1.1/?page=home`
12. Le corpus annoté avec ces critères est le corpus Sequoia, mais il n'a pas été livré assez tôt pour que nous puissions l'utiliser
13. Nous n'avons pas connaissance d'une évaluation chiffrée de la qualité d'annotation des EPs dans ce corpus.

Les EPs nominales sont les plus fréquentes (29.5 %). Ont été annotées comme EPs les « entités nommées » de personne, organisation et lieu, les titres d'œuvre, mais également des noms communs composés comme *garde d'enfants* ou *mot-valise*. Ont été annotées comme EP tous les nombres écrits en chiffres ou en lettres, les adjectifs numéraux et cardinaux comprenant plusieurs composants. La plupart des nombres écrits en chiffres ont la catégorie déterminant. En dehors de ceux-ci, les catégories prépondérantes sont les adverbes composés et les prépositions complexes, se terminant en général par les prépositions *de* ou *à*, comme par exemple *par rapport à*. Les EPs de structure préposition + SN [14] sont catégorisés adverbe ou adjectif selon leur distribution (par exemple *en effet* versus *de qualité*).

1.4.2 Les critères d'EPs dans le projet PARSEME-FR

Dans le cadre du projet PARSEME-FR, un guide d'annotation en EPs pour le français [15] a été réalisé, dans le but de réaliser des corpus d'évaluation annotés en EPs. Les critères sont cette fois-ci entendus comme des critères suffisants, une EP annotée pouvant ainsi satisfaire un nombre variable de critères. Ce choix vise à capturer le fait que le statut d'EP est plus ou moins net et une catégorisation binaire est trop schématique.

Le guide d'annotation commence par un premier arbre de décision permettant d'aiguiller vers un guide spécifique pour les entités nommées, versus un guide pour les autres EPs. Les « entités nommées », terme consacré en TAL, correspondent en fait plutôt à des noms d'entités. La distinction mise en avant concerne le mode d'établissement de la convention de nommage entre un nom (une expression linguistique) et une entité du monde (cognitif) des locuteurs : la convention doit ou pas être ré-établie pour chaque nouvelle entité. Par exemple pour un nom propre comme *Sarah Delavigne*, pour pouvoir user de ce nom pour une nouvelle personne, il faut apprendre ou établir une convention de nommage entre le nom et la personne, alors que pour pouvoir appeler la même personne *le bébé d'Inès*, il suffit d'en connaître les caractéristiques.

Le guide pour les EPs non entité nommée établit une liste précise de critères suffisants, que nous résumons ici [16] :

- La séquence contient un mot qui n'existe que dans ce contexte (*à l'instar de, au fur et à mesure, parce que*).
- Critère sémantique :
  - La tête syntaxique de l'expression n'est pas hyperonyme de l'expression (un *cordon bleu* n'est pas un cordon).
  - On ne peut pas trouver une relation de prédicativité (*arme blanche/#une arme qui est blanche*).
- Irrégularité syntaxique :
  - La structure syntaxique interne est irrégulière (séquence de catégories irrégulière).
  - La distribution de la séquence n'est pas celle attendue d'après la structure interne (par exemple *suite à*).
- Figement lexical :

14. Le symbole SN correspond à un syntagme nominal.

15. `https://gitlab.lis-lab.fr/PARSEME-FR/PARSEME-FR-public/wikis/Guide-annotation-PARSEME_FR-chapeau`

16. Pour tous les tests, « l'impossibilité » signifie soit que l'on obtient une séquence inacceptable, soit que la modification produit un changement de sens inattendu.

  - Un élément plein ne peut être remplacé par aucun voisin sémantique (remplacement impossible ou provoquant un changement de sens inattendu, par exemple *eau/#boisson de vie*)
  - Le déterminant est figé (*garde du corps*), y compris le cas sans déterminant
  - Possibilité d'absence de déterminant (à considérer hormis les cas où l'absence de déterminant est régulière, comme par exemple après la préposition *en*) *à terme, à domicile.*
- Figement morphosyntaxique :
  - On ne peut pas changer les traits morphosyntaxiques (*de dernière/*s minute/*s*).
  - On ne peut pas insérer de modifieur - (*(*très) bien que, étoile (*très) filante*).
  - Impossibilité de certaines variations syntaxiques normalement possibles (par exemple Nom Adj $\longrightarrow$ #Nom de (Det) N-adj).

1.5 Résumé

Dans ce chapitre, nous avons passé en revue différents aspects linguistiques de notre objet d'étude, les expressions polylexicales (EPs) : leurs définitions, leurs propriétés linguistiques et formelles, les typologies et catégories d'EPs, ainsi que les critères d'identification.

Nous avons commencé par donner plusieurs définitions proposées dans la littérature, en insistant sur celle que nous avons adoptée. Cette dernière, directement liée aux données que nous utilisons dans nos travaux, caractérise une EP comme une unité lexicale qui peut être décomposée en plusieurs lexèmes et qui affiche de l'idiosyncrasie à un ou plusieurs niveaux linguistiques. L'idiosyncrasie fait référence à l'impossibilité dans le cas d'une EP à admettre une variation (lexicale, syntaxique, sémantique) qui serait normalement possible étant donnés les composants de l'EP et sa structure. Notre définition exclut les séquences ayant uniquement une idiosyncrasie statistique, c'est-à-dire les collocations.

Nous présentons ensuite les différentes idiosyncrasies qui caractérisent les EPs ainsi que leurs propriétés formelles, leur variabilité et leur ambiguïté. Alors que l'idiosyncrasie lexicale se traduit par la présence d'un ou de plusieurs composants d'une EP qui n'appartiennent pas au lexique conventionnel, l'idiosyncrasie syntaxique se manifeste par la formation d'une structure syntaxique inhabituelle par les composants de l'EP ou lorsqu'elle a une distribution inattendue. L'idiosyncrasie sémantique caractérise les EPs dont le sens ne peut pas être déduit totalement de leurs composants. Le chapitre présente également les propriétés formelles des EPs : leur variabilité morphosyntaxique, leur discontinuité que peut entraîner leur flexibilité syntaxique, leur ambiguïté, ou l'enchâssement ou le chevauchement de leurs occurrences, etc.

Nous présentons ensuite différentes classifications d'EPs présentes dans la littérature : la typologie de Sag et al. (2002), et la classification en catégories morphosyntaxiques de Baldwin et Kim (2010). Les expressions verbales faisant l'objet d'une attention particulière dans nos travaux, nous donnons également les différentes typologies des expressions verbales des jeux de données PARSEME 1.0 et PARSEME 1.1.

Enfin, nous détaillons les critères d'identification utilisés pour différentes ressources annotées en EPs. Nous nous concentrons en particulier sur l'annotation des EPs du French Treebank et sur la campagne d'annotation du projet PARSEME-FR. Il existe des critères pour les différents niveaux linguistiques (morphologiques, morphosyntaxiques, syntaxiques et sémantiques), et ils se fondent très souvent sur le figement des composants des EPs.

Chapitre 2

Identification des EPs

Ce chapitre aborde les EPs du point de vue du TAL et porte sur la tâche d'identification des EPs, ses motivations et ses défis. Il décrit également les ressources linguistiques les plus répandues dans la littérature de l'identification des EPs. Nous nous intéressons plus particulièrement aux ressources que nous exploitons dans nos travaux.

La section 2.1 présente une définition formelle de l'identification et fait la distinction entre cette tâche et la tâche d'extraction des EPs. La section 2.2 expose les différentes motivations de cette tâche alors que la section 2.3 rend compte des jeux de données exploités dans le travail de la thèse et par la plupart des méthodes de la littérature. La section 2.4 présente les formats des jeux de données et les schémas les plus utilisés. La section 2.5 aborde les différents défis qu'une méthode d'identification doit affronter. La section 2.6 présente les métriques d'évaluation les plus répandues dans la littérature.

Enfin, ce chapitre présente deux approches d'identification des EPs. La section 2.7 introduit les méthodes d'identification à base de règles, tandis que la section 2.8 présente d'autres méthodes, associées à la découverte des EPs, basées sur la classification binaire des candidats.

2.1 Définition de la tâche

La tâche d'identification d'EPs consiste à annoter les occurrences d'EPs au sein de phrases (Baldwin et Kim, 2010; Constant et al., 2017). Plus précisément, à partir d'une phrase préalablement segmentée en « tokens » (nous préciserons cette notion infra section 2.4.2), la tâche consiste à détecter les ensembles de tokens formant des occurrences d'EPs. Par exemple, dans la phrase *Cette série ne vit finalement jamais le jour*, l'ensemble de tokens *vit*, *le* et *jour* seraient marqués comme formant une occurrence de l'EP *voir le jour*.

L'identification peut s'accompagner d'une catégorisation des EPs reconnues : l'EP *voir le jour* serait catégorisée « idiome verbal » (ID) selon le jeu de catégories d'EPs défini dans le guide d'annotation PARSEME (cf. section 1.3.1). En comparaison à d'autres tâches de TAL supposant d'identifier des groupes de tokens (comme la reconnaissance d'entités nommées ou la segmentation syntaxique (« chunking » en anglais), les EPs à identifier sont potentiellement discontinues, contrairement aux entités nommées ou aux chunks, ce qui ajoute une complexité formelle.

2.1.1 Identification d'occurrences vs extraction de types d'EPs

Constant et al. (2017) distinguent la tâche d'identification des EPs de celle d'extraction

d'EPs. La première consiste à identifier toutes les occurrences d'EPs dans un texte, alors que la seconde consiste à rechercher de nouvelles entrées lexicales polylexicales dans des corpus et à les intégrer dans des structures spéciales, comme des lexiques, pour une utilisation ultérieure. Alors que l'identification entraîne l'annotation du texte en cours, l'extraction produit une liste d'entrées lexicales d'EP.

L'évaluation de l'identification d'EPs utilise les métriques classiques de précision, rappel et F-mesure (voir section 2.6), alors qu'en extraction, on ne dispose en général pas de la liste exhaustive des EPs extrayables d'un grand corpus donné. On utilise alors plutôt un score de précision sur les EPs extraites (validées manuellement), ou bien une évaluation extrinsèque (Constant et al., 2017).

En termes d'approche technique, les méthodes non supervisées dominent la littérature de l'extraction d'EPs alors qu'en l'état actuel, l'identification connaît ses meilleures performances grâce aux méthodes supervisées. Une phase d'extraction automatique d'EPs est parfois utilisée pour la tâche d'identification. En particulier, des lexiques d'EPs issus de procédures automatiques d'extraction peuvent être utilisés dans des systèmes d'identification supervisés sous forme de traits (Hashimoto et al., 2006; Diab et Bhutada, 2009; Constant et al., 2012; Gharbieh et al., 2017). D'autres méthodes d'identification s'appuient sur des méthodes d'extraction, qui proposent un ensemble d'EPs candidates pour la phrase en cours, et ensuite filtrent ces candidats en analysant leurs contextes (Legrand et Collobert, 2016; Pasquer et al., 2018) (voir section 2.8).

2.2 Motivations

L'identification des EPs est motivée par l'idiosyncrasie qui les caractérise.
D'une part l'idiosyncrasie sémantique nécessite un traitement particulier dans le cadre de toute tâche incluant une part sémantique. Par exemple, il faut bien repérer d'une manière ou d'une autre que *cordon bleu* ne s'interprète pas comme la combinaison régulière de *cordon* avec *bleu* dans une tâche de type compréhension automatique du langage naturel. En traduction automatique, la traduction littérale de la plupart des EPs est en général problématique, du fait des différents types d'idiosyncrasie, qui peuvent représenter une complication sérieuse pour les modèles statistiques utilisant l'alignement de mots (Baldwin et Kim, 2010). L'identification des EPs non et semi-compositionnelles peut faciliter leur traduction et améliorer les scores de la traduction automatique statistique (Diab et Bhutada, 2009). Même si cela est moins standard pour les modèles neuronaux actuels, nous pouvons citer (Rikters et Bojar, 2017) comme exemple d'intégration bénéfique d'EPs en traduction automatique neuronale.

D'autre part, même sans aller jusqu'à une analyse sémantique, l'idiosyncrasie morphologique et/ou syntaxique des EPs peut perturber les tâches d'étiquetage morphosyntaxique et d'analyse syntaxique. Par exemple dans *un fusil longue portée*, la séquence de catégories nom+adjectif+nom peut être mal prédite car inhabituelle. Ou bien dans *Il est parti il y a 10 minutes*, sans l'information que *il y a* n'est pas verbal, un analyseur syntaxique pourrait repérer deux clauses à tort. Les résultats d'identification d'EPs et analyse syntaxique automatique sont cependant mitigés, les erreurs d'identification contrebalançant le gain pour l'analyse syntaxique (Constant et Tellier, 2012; Candito et Constant, 2014). En revanche inversement, Nasr et al. (2015) ont montré que l'analyse syntaxique peut contribuer à l'amélioration de la performance de l'identification de leur modèle joint (voir section 3.1.4 pour une description de leur système).

Langue	entraînement			Test			Annotations	
	Phrases	EPs	Tokens	Phrases	EPs	Tokens	Morpho	Synt
BG	6.9	1.9	157.6	1.9	0.5	42.5	-	-
CS	44.0	12.9	740.5	5.5	1.7	92.7	+	-
DE	6.3	2.4	120.8	1.2	0.5	24.0	+	+
EL	5.2	1.5	142.3	3.6	0.5	83.9	+	+
ES	2.5	0.7	102.2	2.1	0.5	57.8	+	-
FA	2.7	2.7	46.5	0.5	0.5	8.7	+	-
FR	17.9	4.5	450.2	1.7	4.5	35.7	+	+
HE	4.7	1.3	99.8	2.3	0.5	47.6	-	-
HU	3.6	3.0	87.8	0.7	0.5	20.4	+	+
IT	15.7	2.0	387.3	1.3	0.5	40.5	+	+
LT	12.2	0.4	209.6	2.7	0.1	46.6	-	-
MT	6.0	0.8	141.1	4.6	0.5	111.2	+	-
PL	11.6	3.1	191.2	2.0	0.5	29.7	+	+
PT	19.6	3.4	359.3	2.6	0.5	54.7	+	+
RO	45.5	4.0	778.7	6.0	0.5	100.8	+	-
SL	8.9	1.8	183.3	2.5	0.5	52.6	+	+
SV	0.2	0.1	03.4	1.6	0.2	26.1	+	+
TR	16.7	6.2	334.9	1.3	0.5	27.2	+	+

TABLE 2.1 – **PARSEME 1.0 :** Le nombre, en milliers (arrondis), de **phrases**, d'**EPs** et de **tokens** dans les jeux d'**entraînement** et de **test** des langues de PARSEME 1.0, ainsi que la disponibilité des annotations **morpho**logiques (étiquettes morphosyntaxiques et lemmes) et les annotations **synt**axiques (arbre de dépendances). Les langues sont représentées par leur code ISO 639-1.

2.3 Jeux de données

Cette section présente les jeux de données utilisés dans nos expérimentations : les jeux de données multilingues PARSEME 1.0 et PARSEME 1.1, le French TreeBank et les données DiMSUM pour l'anglais. Ces jeux de données ont clairement contribué au développement du domaine de l'identification des EPs et sont très fréquemment cités dans la littérature de ce domaine, et donc dans notre section d'état de l'art.

2.3.1 Jeux de données de PARSEME 1.0

Les jeux de données PARSEME 1.0 ont été produits dans le cadre de la première compétition internationale pour l'identification des EPs verbales (Savary et al., 2017). Cette campagne a été organisée par le projet de PARSing and Multiword Expressions (PARSEME), une action européenne COST qui a permis de constituer un réseau scientifique interdisciplinaire dédié au traitement des EPs en lien avec l'analyse syntaxique. Ce réseau rassemble des experts interdisciplinaires de 31 pays, travaillant sur 30 langues et 6 dialectes de 10 familles linguistiques.

Les données consistent principalement en des phrases tokenisées et annotées en EPs, pour 18 langues, mais avec de fortes disparités de taille de données entre les différentes langues. Le tableau 2.1 donne pour chaque langue les nombres de phrases, d'EPs d'entraînement et de test (en milliers). Une EP y est définie simplement comme un ensemble de tokens, contigus ou pas

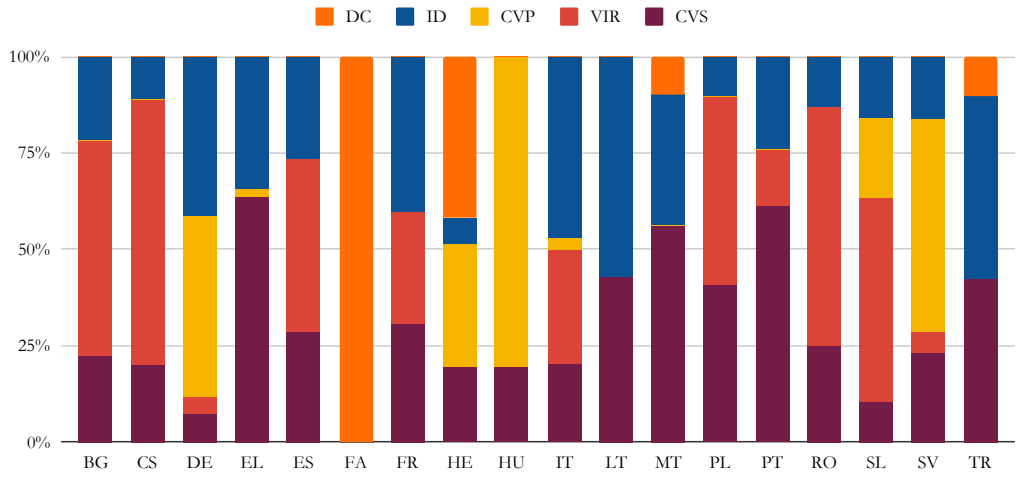

FIGURE 2.1 – **PARSEME 1.0- catégories :** La distribution des catégories dans les jeux d'entraînement de PARSEME 1.0.

(comme dans *Cette série ne* **vit** *finalement jamais* **le jour**.).

Chaque EP annotée est associée à l'une des catégories suivantes : constructions avec verbe à particule (CVP) ; constructions à verbe support (CVS) ; expressions verbales idiomatiques (ID) ; verbes intrinsèquement réfléchis (VIR) et une catégorie pour les EPs verbales d'autres catégories (DC). Nous donnons à la figure 2.1 la répartition des EPs dans les différentes catégories pour chaque langue.

Longueur des EPs : de 1 à n tokens

Une caractéristique de ces données est qu'un token à lui tout seul peut constituer une EP (cf. le terme TP introduit à la section 1.3.1). En effet, les corpus annotés en EPs dans le cadre de PARSEME couvrent 18 langues (20 pour la 2ème campagne), et sont souvent issus de corpus préexistants, pour la plupart préalablement annotés en morphosyntaxe voire en syntaxe. Ils comprennent ainsi des choix variés concernant la segmentation en tokens, qui peut suivre des critères purement typographiques, ou refléter des choix linguistiques de segmentation en mots. Par exemple selon les cas, le mot *aujourd'hui* peut constituer deux tokens (si l'on segmente sur une base typographique pure) ou un seul. Les consignes d'annotation PARSEME laissent ainsi la liberté d'annoter un unique token comme EP, dès lors que l'on peut envisager un autre choix de segmentation qui donnerait plusieurs tokens. Cela donne ainsi lieu à des EPs contenant un seul token, que nous appellerons désormais Token Polylexical (TP). On peut également avoir des TPs dans le cas d'un token qui alterne avec une version à plusieurs composants séparés. C'est en particulier le cas par exemple des verbes avec particule en allemand qui peuvent être séparés en deux tokens (par exemple dans *Die Regierung* ***teilte*** *nur* ***mit****, ... :La police a seulement annoncé, ...*) ou apparaître en un seul token, qui est alors annoté comme EP (de type TP) (*... , wie die Polizei am Sonntag* ***mitteilte*** *: d'après ce que la police a communiqué dimanche*).

La table 2.2 donne entre autres les longueurs moyennes des EPs de PARSEME 1.0 pour chaque langue (colonne 2) et la proportion des TPs. Pour la plupart des langues, les TPs sont inexistants ou extrêmement marginaux, à l'exception des données de l'allemand où ils sont massivement présents (30 % des EPs annotées) et du hongrois (75 %). Pour chaque langue, on a une

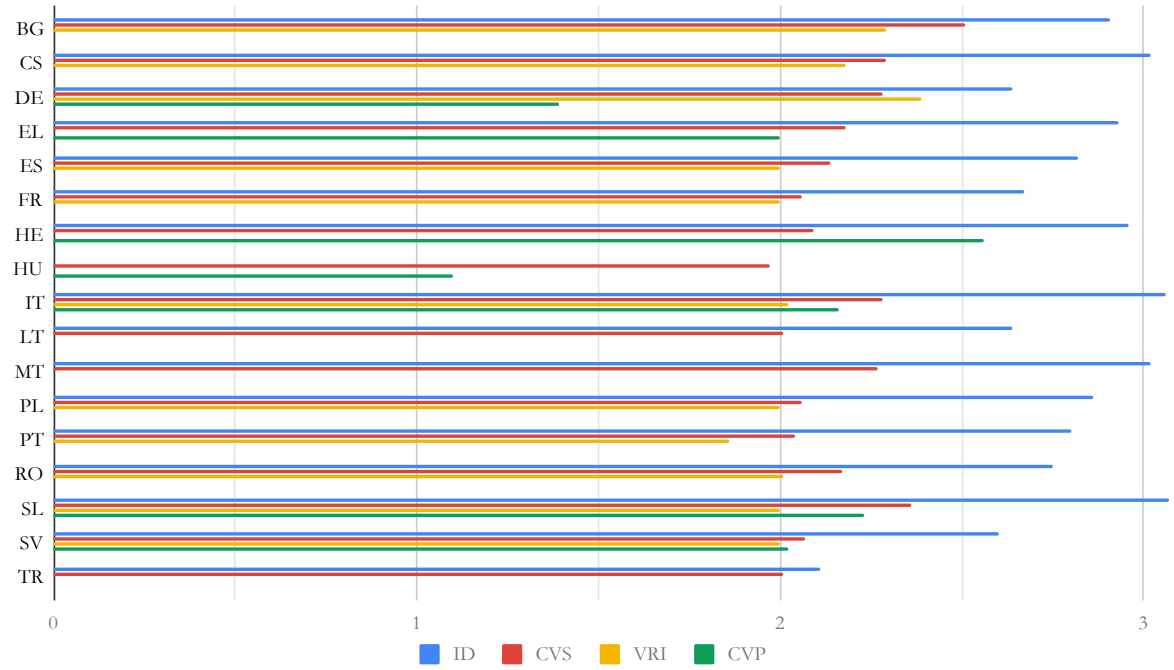

FIGURE 2.2 – **PARSEME 1.0 : Longueurs des EPs :** La moyenne de la longueur des EPs dans les différentes catégories des données de PARSEME 1.0. Source : Savary et al. (2018)

longueur moyenne (incluant les TPs) un peu supérieure à 2. Certaines langues comme l'hébreu, le maltais, l'italien dépassent 2,5 de longueur moyenne, avec une déviation absolue moyenne (DAM) importante, ce qui montre une certaine disparité dans les longueurs d'EPs.

Les longueurs typiques des EPs diffèrent selon les catégories, comme le montre la figure 2.2, qui fournit la longueur moyenne des EPs (en nombre de composants) pour chaque catégorie.

Discontinuités, enchâssements et chevauchements

Les EPs peuvent être discontinues. On donne dans la partie droite de la table 2.2 les répartitions par taille de la discontinuité. Pour l'allemand et le slovène, les EPs continues sont minoritaires (35, 7 % et 44, 4 % d'EPs continues). Elles sont à égale proportion pour le turc, mais majoritaires pour toutes les autres langues. Au sein des EPs discontinues, l'allemand et le slovène sont également distingués, avec des discontinuités plus longues. La figure 2.3 donne la longueur moyenne des discontinuités, par catégorie d'EPs (en nombre total de tokens intervenant entre les composants d'une EP mais n'y appartenant pas), ce qui permet de constater qu'en allemand ce sont les constructions avec verbes à particule (CVPs) qui donnent de longues discontinuités, alors que pour le slovène cela est plus réparti. Globalement, les constructions à verbe support (CVSs) sont sans surprise un peu plus discontinues, cf. on a souvent ne serait-ce qu'un déterminant entre le verbe support et le nom.

Par ailleurs, une EP peut être enchâssée dans une autre plus longue, et deux EPs peuvent se chevaucher (voir section 1.2.4), mais cela se produit de façon très marginale : les EPs enchâssées représentent 3 % des EPs des jeux de données allemand et hébreu et 1 % (ou moins) des EPs des autres langues. Le chevauchement concerne 5,5 % des EPs en hébreu, 2,3 % des EPs en portugais, 3,6 % en slovène et moins de 1 % pour les autres langues.

	Longueur			Longueur des discontinuités (hors TP)						
	Moy	DAM	TP %	Moy	DAM	0%	1%	2%	3%	>3%
BG	2.45	0.63		0.64	1.05	82.1	10.7	1.7	1.3	4.2
CS	2.3	0.46		1.35	1.53	51.5	18.3	11.4	7.3	11.4
DE	2.02	0.61	29.2	2.96	2.94	35.7	16.3	9.2	8.2	30.5
EL	2.45	0.61		0.94	1.08	57.4	25.7	8.2	3.3	5.4
ES	2.24	0.39		0.47	0.66	69.9	21.7	4.4	1.9	2.1
FA	2.16	0.27		0.42	0.7	82.9	7.5	3.8	2.2	3.7
FR	2.29	0.44		0.65	0.8	61.9	25.0	7.5	2.8	2.8
HE	2.71	0.75		0.47	0.74	78.9	10.1	4.2	3.4	3.5
HU	1.27	0.39	73.5	1.01	1.29	63.7	22.4	4.3	1.9	7.7
IT	2.58	0.64		0.28	0.46	80.9	14.2	2.9	1.1	0.8
LT	2.35	0.53		0.72	0.94	64.9	19.7	9.0	2.2	4.2
MT	2.64	0.68	0.9	0.34	0.53	77.0	16.1	4.3	1.6	1.0
PL	2.11	0.2		0.53	0.77	73.3	14.9	6.2	2.9	2.8
PT	2.19	0.37	2.2	0.67	0.78	58.3	30.1	6.6	2.4	2.6
RO	2.15	0.25		0.55	0.72	64.7	17.1	17.2	0.8	0.3
SL	2.27	0.43	0.8	1.47	1.54	44.4	25.1	12.5	6.7	11.4
SV	2.14	0.25		0.38	0.59	78.6	12.5	5.4	1.8	1.8
TR	2.06	0.11		0.57	0.57	49.4	47.0	2.6	0.5	0.5

TABLE 2.2 – **PARSEME 1.0- Longueurs et discontinuités :** Longueurs et discontinuités des EPs en nombre de composants d'EPs, dans les corpus d'entraînement de PARSEME 1.0. Les occurrences d'EPs sont comptées et pas les types. Col. 2–3 : moyenne et déviation absolue moyenne (DAM) pour la longueur des occurrences d'EPs. Col. 4 : proportion de TPs. Col. 5–6 : moyenne et DAM pour la longueur des discontinuités. Col. 7–11 : proportions de la continuité (0) et des discontinuités (1, 2, 3, >3). Source : Savary et al. (2017).

On a beaucoup de cas d'enchâssements en allemand, les TPs y étant plus facilement enchâssables qu'une EP à plusieurs tokens, mais ceci ne se vérifie pas pour le hongrois qui a beaucoup de TPs mais peu d'enchâssements.

Annotations supplémentaires distribuées pour la compétition internationale

Selon les langues, les données fournies pour la compétition PARSEME 1.0 sont plus ou moins riches :

— Pour 4 langues (bulgare, espagnol, hébreu et lituanien), ne sont fournies que les annotations d'EPs.

— Pour 4 autres (tchèque, maltais, roumain et slovène), sont également fournies des annotations morphologiques (avec pour chaque token la catégorie morphosyntaxique et le lemme, plus d'éventuels traits morphologiques).

— Enfin pour les 10 autres langues, on dispose des annotations morphologiques et des annotations syntaxiques, sous la forme d'un arbre de dépendances (au format CoNLL-U, cf. infra section 2.4.2.0).

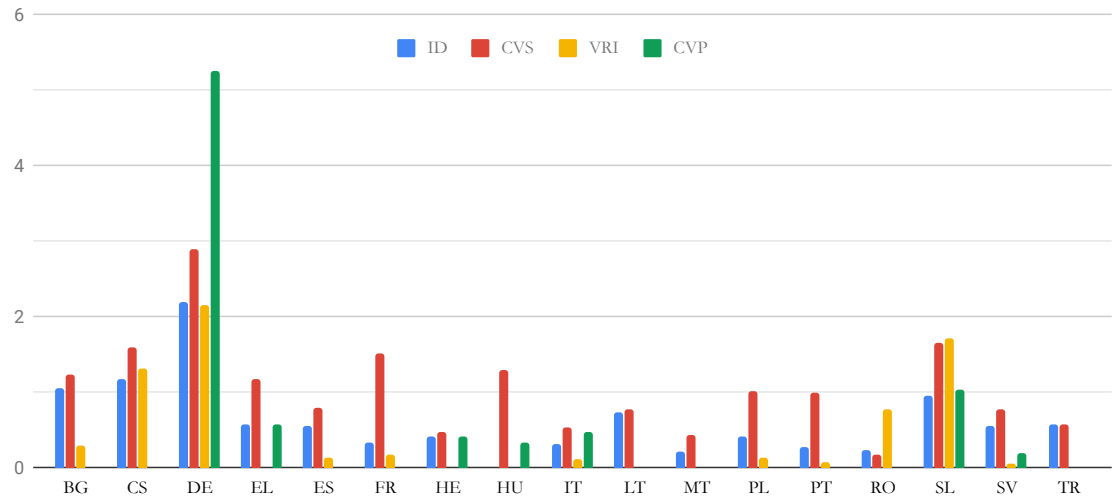

FIGURE 2.3 – **PARSEME 1.0 : Trous :** La moyenne de la longueur des trous des EPs dans les différentes catégories des langues du jeu de données PARSEME 1.0 (nombre total de tokens apparaissant entre les composants d'une même EP mais n'y appartenant pas). Source : Savary et al. (2018)

Langue	Entraînement			Dev			Test		
	phrases	EPs	Tokens	phrases	EPs	Tokens	phrases	EPs	Tokens
BG	17.8	5.4	399.2	2.0	0.7	42.0	1.8	0.7	39.2
DE	6.7	2.8	130.2	1.2	0.5	22.1	1.1	0.5	20.5
EL	4.4	1.4	122.5	2.6	0.5	64.8	1.3	0.5	35.9
EN	3.5	0.3	53.2	-	-	-	4.0	0.5	71.0
ES	2.8	1.7	96.5	0.7	0.5	26.2	2.0	0.5	58.9
EU	8.3	2.8	117.2	1.5	0.5	21.6	1.4	0.5	19.0
FA	2.8	2.5	45.2	0.5	0.5	8.9	0.4	0.5	7.5
FR	17.2	4.6	420.8	2.2	0.6	54.7	1.6	0.5	38.4
HE	12.1	1.2	237.5	3.4	0.5	65.8	3.2	0.5	65.7
HI	0.9	0.5	17.9	-	-	-	0.8	0.5	17.6
HR	2.3	1.5	53.5	0.8	0.5	19.6	0.7	0.5	16.4
HU	4.8	6.2	120.0	0.6	0.8	15.6	0.8	0.8	20.8
IT	13.6	3.3	342.3	0.9	0.5	31.2	1.3	0.5	35.6
LT	4.9	0.3	90.1	-	-	-	6.2	0.5	118.4
PL	13.1	4.1	220.4	1.8	0.5	26.0	1.3	0.5	27.7
PT	22.0	4.4	473.4	3.1	0.6	64.1	2.8	0.6	58.6
RO	42.7	4.7	782.0	7.1	0.6	118.7	6.9	0.6	115.0
SL	9.6	2.4	201.9	2.0	0.5	38.1	2.0	0.5	40.5
TR	16.7	6.1	334.9	1.3	0.5	27.2	0.6	0.5	14.4

TABLE 2.3 – **PARSEME 1.1 :** Le nombre de **phrases**, d'**EPs** et de **tokens** dans les jeux d'**entraînement**, de **développement** et de **test** pour les langues de PARSEME 1.1. Les langues sont représentées par leur code ISO 639-1 et toutes les valeurs du tableau sont mises à l'échelle et arrondies (1/1000).

Langue	Famille	DEV	corpus	Lemme	UPOS	XPOS	Syntaxe
ES	Romane	+	+	A	A	N.A	A
FR	Romane	+	+	N.D	M	N.A	M
IT	Romane	+	-	A	A	N.D	A
PT	Romane	+	+	Mult	Mult	N.D	Mult
RO	Romane	+	-	UDP	UDP	UDP	UDP
BG	Slave	+	-	UDP	UDP	UDP	UDP
HR	Slave	+	-	A	A	N.A	A
LT	Slave	-	-	A	A	N.D	N.A
PL	Slave	+	+	Mult	Mult	Mult	Mult
SL	Slave	+	-	M	N.A	M	M
DE	Germanique	+	+	UDP	Mult	UDP/TT	Mult
EN	Germanique	-	+	Mult	Mult	Mult	Mult
HE	Sémitique	+	+	UDP	UDP	UDP	UDP
EL	Hellénique	+	+	A	UDP	UDP	UDP
FA	Indo-iraniennes	+	-	M	M	M	N.D
HI	Indo-iraniennes	-	-	UDP	UDP	N.A	UDP
HU	Ouraliennes	+	-	M	M	N.A	M
TR	Turque	+	-	ITU	ITU	N.D	ITU
EU	Autre	+	+	A	Mult	N.A	Mult

TABLE 2.4 – **PARSEME 1.1 : Annotations supplémentaires** pour chaque langue dans les données PARSEME 1.1. Le tableau indique, pour chaque langue, sa famille linguistique (**Famille**), si elle dispose d'un jeu de développement (**DEV**), si les données viennent de plusieurs corpus (+) ou d'un seul corpus (**-**). Les autres colonnes (Lemme, XPOS, UPOS, Syntaxe), indiquent si les annotations de ce type sont non disponibles (**N.A**), si elles existent mais leurs sources ne sont par définies dans la description du jeu de données (**N.D**) ; si elles sont manuelles (M) ou automatiques (**A**) ou si elles sont mélangées (**Mult**). Si l'outil utilisé lors de l'annotation automatique est indiqué, nous le fournissons en remplaçant le (A) par le nom de l'outil : **UDP**ipe (`https://www.ufal.mff.cuni.cz/udpipe/`) ; **ITU** NLP Pipeline (`http://tools.nlp.itu.edu.tr/`) ; **T**ree **T**agger (`https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`).

2.3.2 Jeux de données de PARSEME 1.1

La deuxième compétition PARSEME 1.1 a eu lieu en 2018 (Ramisch et al., 2018), et pour cette occasion de nouvelles données ont été fournies (cf. tableau 2.4). Le guide d'annotation a été modifié légèrement, l'ensemble des catégories a été un peu élargi. Les données sont en grande majorité les mêmes corpus, avec révision manuelle concernant les changements du guide d'annotation. Pour être incluses pour cette deuxième campagne, les données devaient cependant comprendre un nouveau corpus de test. Certaines langues ne sont plus incluses faute d'avoir pu fournir de nouveau test. D'autres langues font leur apparition, comme le basque (EU) ou le Hindi (HI), pour un total de 20 langues couvertes. Comme pour PARSEME 1.0, les tailles varient beaucoup selon les langues. Le tableau 2.3 fournit les nombres de phrases, d'EPs et de tokens dans les jeux d'entraînement, de développement et de test.

Une différence avec PARSEME 1.0 est que chaque langue comprend des annotations morphologiques et syntaxiques (et ces annotations sont fournies dans un seul fichier tabulé, avec les annotations d'EPs, cf. infra le format CUPT section 2.4.2.0). Toutes les langues ont un fi-

Dataset	FTB			DiMSUM		
	Phrases	EPs	Tokens	Phrases	EPs	Tokens
Entraînement	14.8	23.7	443.1	04.8	04.2	73.8
Développement	01.2	02.1	38.8	-	-	-
Test	02.5	04.0	75.2	01.0	00.8	16.5

TABLE 2.5 – **FTB et DiMSUM :** Le nombre (en milliers arrondis) de **phrases**, d'**EPs** et de **tokens** dans les jeux d'**entraînement**, de **développement** et de **test** pour le français (FTB) et l'anglais (DiMSUM).

chier d'entraînement et de test, et toutes sauf l'anglais, l'hindi et le lituanien ont un fichier de développement.

La définition formelle de ce qu'est une EP (un ensemble de un ou plusieurs tokens) est la même que dans les données PARSEME 1.0.

Là encore, chaque EP annotée est associée à une catégorie. Le jeu de catégories a été légèrement revu et élargi entre les deux campagnes PARSEME, et nous avons décrit ces catégories les plus à jour au chapitre 1, section 1.3.1.

2.3.3 French treebank (FTB)

Le French Treebank (FTB) (Abeillé et al., 2003) est un corpus arboré initialement annoté en constituants, comprenant des annotations d'EPs, pratiquement toutes continues. Nous utilisons la version intégrée aux données de la compétition internationale « Statistical Parsing of Morphologically Rich Languages » (SPMRL) de Seddah et al. (2013). Ces données sont fournies dans une version en constituants et une version automatiquement convertie en dépendances. Les deux versions reportent donc les annotations d'EPs de la ressource originale (sauf les quelques EPs discontinues). Il s'agit des premières données rassemblant sous un format utilisable en analyse syntaxique à la fois le statut d'EP et la représentation syntaxique. Dans le FTB SPMRL, une EP est définie comme une séquence continue de tokens et a une étiquette morphosyntaxique. En ce qui concerne les arbres en constituants, l'étiquette morphosyntaxique de l'EP est un nœud interne de l'arbre, dominant la séquence des tokens de l'EP. Par exemple pour *Paul mange une pomme de terre*, on aura un nœud de catégorie NC (pour nom commun) pour *pomme*, mais également pour le sous-arbre pour *pomme de terre*, soit en format parenthésé : (Sint (NP (NC Paul)) (VN (V mange)) (NP (DET une) (**NC** (NC pomme) (P de) (NC terre)))).

Dans les arbres en dépendances, les EPs sont repérées grâce à une étiquette de dépendance spécifique (*dep_ cpd*), et ont une structure plate : le premier token d'une EP est considéré comme la tête, et tous les autres composants y sont rattachés avec un arc étiqueté *dep_ cpd*.

Les EPs de ce jeu de données sont exclusivement continues et un composant ne peut appartenir qu'à une seule EP (pas d'enchâssement ni de chevauchement donc).
Le FTB ne comprend pas beaucoup d'EPs verbales, et une EP peut être un nom propre composé, un nom commun composé, un déterminant composé, un adverbe composé, un adjectif composé ou même un pronom composé (voir le tableau 2.5 pour le nombre des phrases, tokens et EPs du FTB et la figure 2.4 pour les proportions de chaque catégorie morphosyntaxiques d'EPs).

2.3.4 Detecting Minimal Semantic Units and their Meanings (DiMSUM)

La campagne internationale DiMSUM a pour but de prédire les unités sémantiques minimales de phrases en anglais, et d'attribuer des classes sémantiques à certaines de ces unités (Schneider

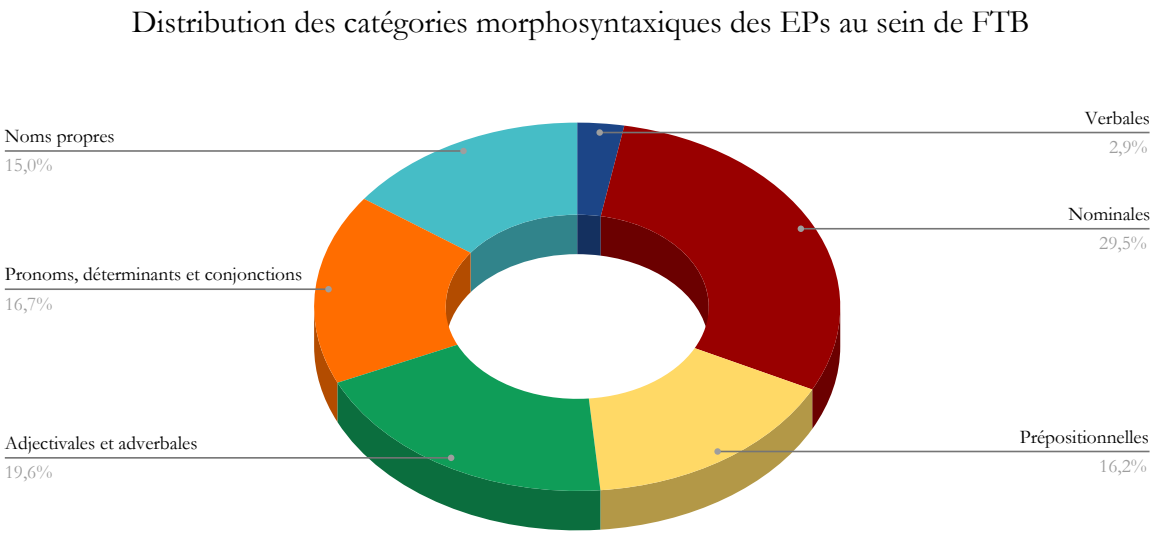

FIGURE 2.4 – **Catégories de FTB :** La distribution des catégories morphosyntaxiques des EPs au sein du jeu d'entraînement du FTB SPMRL (i.e. valeur du trait MWE_head des EPs).

et al., 2016). Cette campagne fournit un jeu de données dont les EPs sont annotées, puisqu'elles représentent des unités sémantiques.
Le corpus d'entraînement comprend des phrases de deux domaines : des critiques en ligne (i.e. « reviews ») et des tweets. Les données de test incluent en outre des « TED talks » comme troisième domaine.

Les EPs sont réparties en plusieurs catégories telles que les expressions verbales *find out (découvrir)*, les expressions idiomatiques *shoot the breeze (papoter)*, les expressions prépositionnelles *in time (à temps)*, les entités nommées *(Philip Pullman)*, entre autres (Schneider et al., 2014b). Le tableau 2.5 fournit également le nombre de phrases, tokens et EPs de ce jeu de données. Les tokens d'une EP ne doivent pas nécessairement être contigus ($made_1$ *a* $carry_2$ out_2 $order_1$). De plus, les EPs annotées sont divisées en deux catégories : des collocations statistiques (mais compositionnelles) sont considérées comme des « EPs faibles », et se distinguent des « EPs fortes », non ou semi-compositionnelles.

Les données DiMSUM utilisent un schéma d'annotation de type IOB-iob (cf. infra section 2.4.1). Chaque ligne correspondant à un token est composée de sept champs représentant l'indice du token, sa forme fléchie, son lemme, son étiquette morphosyntaxique, l'étiquette IOB-iob d'EP, l'indice du premier token de l'EP dont le token fait partie (si c'était le cas), un champ pour indiquer le niveau de compositionnalité de l'EP et l'identifiant de la phrase.

2.4 Annotations : schémas et formats

Nous détaillons dans cette section les différents schémas et formats utilisés pour représenter les EPs au sein de divers jeux de données. Nous nous limitons aux schémas et formats principaux utilisés par les travaux les plus significatifs dans la littérature d'identification des EPs, et dans

Shéma	La	décision	de	mettre	cet	homme	à	l'	écart	a	été	prise
IOB	O	O	O	B	O	O	I	I	I	O	O	O
IOB-iob	O	B	O	b	o	o	i	i	i	O	O	I

FIGURE 2.5 – **Schémas IOB et IOB-iob - exemple :** Les représentations IOB et IOB-iob de la phrase *La* ***décision***$_1$ *de* ***mettre***$_2$ *cet homme* ***à***$_2$ ***l'écart***$_2$ *a été* ***prise***$_1$ contenant deux EPs (*décision prise* et *mettre à l'écart*, cette dernière apparaissant au sein de la discontinuité entre *décision* et *prise*. Le schéma IOB fourni ne permet pas de gérer ce cas, et n'annote ici que *mettre à l'écart*, contrairement à IOB-iob.

les données sur lesquelles s'appuient nos expérimentations.

2.4.1 Schémas IOB et IOB-iob

Les schémas dits IOB (« Inside Outside Begin ») ou IOB-iob sont couramment utilisés par les méthodes d'étiquetage de séquences pour de multiples tâches de TAL comme la segmentation syntaxique (chunking) et la reconnaissance des entités nommées. Le schéma IOB a été proposé par Ramshaw et Marcus (1999) avec l'objectif de représenter les groupes nominaux non récursifs. Il s'agit d'un schéma d'étiquetage séquentiel, associant une unique étiquette à chaque token de la phrase. Alors que I indique si le token est à l'intérieur du groupe nominal, O indique que le token est à l'extérieur de celui-ci et B marque l'élément le plus à gauche du groupe. Les méthodes d'identification des EPs à base d'étiquetage de séquences s'inspirent de ce schéma pour représenter les phrases des corpus d'apprentissage. Le token peut être soit le premier token d'une EP (il est alors étiqueté B), soit un token complémentaire de l'EP (I), soit à l'extérieur de l'EP (O). Ce schéma permet de représenter des EPs continues et discontinues, mais n'est pas capable de représenter les cas d'enchâssement, de chevauchement, ni même d'imbrication (un cas particulier d'enchâssement, où l'EP enchâssante a des composantes à gauche et à droite de l'EP enchâssée).

Pour augmenter le pouvoir expressif de ce schéma, Schneider et al. (2014a) proposent une modification afin de l'adapter à l'imbrication. Le schéma IOB-iob est composé de six étiquettes : B, I, O, b, i et o. Les étiquettes majuscules sont similaires à celles du schéma IOB standard. Les étiquettes en minuscules ont des significations similaires, mais pour les EPs imbriquées : b est utilisé pour le premier composant de l'EP imbriquée, i pour ses composants suivants et o pour les éventuels tokens qui n'appartiennent pas à l'EP imbriquée (ni enchâssante) et se produisent dans l'une de ses discontinuités (comme *cet homme* dans la figure 2.5). Ce schéma ne permet de représenter l'imbrication qu'à un seul niveau. Comme son ascendant, ce schéma n'est pas capable de représenter les scénarios d'enchâssement ou de chevauchement.

Il existe d'autres variantes du schéma IOB permettant de s'adapter à divers cadres expérimentaux. Klyueva et al. (2017) donne à ce schéma la capacité de catégoriser les EPs en remplaçant la étiquette B par la catégorie linguistique de l'EP. Diab et Bhutada (2009) se base sur ce schéma pour distinguer les occurrences littérales des occurrences idiomatiques des CVSs en proposant cinq différentes étiquettes : *BL* (début d'une occurrence littérale), *IL* (à l'intérieur d'une occurrence littérale), *BI* (début d'une occurrence idiomatique), *II* (à l'intérieur d'une occurrence idiomatique), *O* (à l'extérieur d'une occurrence d'EP). À noter que le jeu de données DiMSUM utilise le schéma IOB-iob.

Position	Forme fléchie	Lemme	POS	...
# sentid : Europar.550_00166				
# sentence-text : Nous devrions prendre cela au sérieux.				
1	Nous	il	PRON	...
2	devrions	devoir	VERB	...
3	prendre	prendre	VERB	...
4	cela	cela	PRON	...
5-6	au	_	_	...
5	à	à	PREP	...
6	le	le	DET	...
7	sérieux	sérieux	NOUN	...
8	.	.	PUNCT	...

FIGURE 2.6 – Les premières colonnes pour le format CoNLL-U pour la phrase *Nous devrions* ***prendre*** *cela* ***au sérieux.***, comportant le traitement de l'amalgame *au*.

2.4.2 Schéma PARSEME

Dans les annotations PARSEME, une EP correspond simplement à un ensemble de tokens d'une phrase (éventuellement réduit à un seul token), l'ensemble étant associé à un identifiant unique au sein de la phrase. Un token peut appartenir à un nombre quelconque d'EPs. On obtient ainsi un pouvoir expressif maximal. La seule restriction est que le schéma se base sur la tokenisation d'entrée de la phrase, et ne peut pas annoter comme composant une sous-partie d'un token. Le schéma prévoit également un typage des EPs.

Les données sont fournies dans deux formats, un format tabulé simple (PARSEME-TSV), et un format couplé à l'annotation syntaxique en dépendances (format CUPT). Avant de les détailler, nous faisons un point terminologique concernant token et mot.

Token versus mot dans les annotations de type CoNLL-U

Les formats des données PARSEME utilisent les principes de tokenisation du projet Universal Dependencies [17], avec en particulier un format qui permet qu'un token correspondant à un amalgame soit défait en plusieurs « mots », comme par exemple l'amalgame *aux* pouvant être défait en deux « mots » *à* et *les*. La terminologie du projet « Universal Dependencies » utilise le terme « multi-word token » pour l'amalgame, et le terme « word » pour les mots qui en sont extraits, ainsi que les mots non issus d'un amalgame. Dans les annotations au format CoNLL-U, utilisé pour les « Universal Dependencies », on a une ligne par amalgame ou mot (voir à la figure 2.6, par exemple la ligne pour *au* et la ligne pour le *à* qui en est extrait). Seuls les mots ont un numéro, les amalgames sont fournis pour mémoire du texte original, avec une indication de la plage de mots auxquels ils correspondent, mais ils ne peuvent pas recevoir d'annotations (morphologiques, syntaxiques, ou dans le cas de PARSEME des annotations d'EPs).

Dans toute la suite, nous préférons garder le terme **« token »** pour une unité APRÈS décomposition des amalgames. Ainsi dans la figure 2.6, on a selon cette terminologie 8 tokens numérotés de 1 à 8, y compris les tokens *à* et *le* issu de l'amalgame *au*. Celui-ci n'a pas d'étiquette morphosyntaxique et n'entre pas dans l'arbre de dépendances syntaxiques. On indique seulement la plage de tokens qu'il couvre (ici la plage 5-6).

17. `https://universaldependencies.org/format.html`

Position	Forme fléchie	Booléen (nsp/_)	EP
# sentid : Europar.550_00166			
# sentence-text : Nous devrions prendre cela au sérieux.			
1	Nous	_	_
2	devrions	_	_
3	prendre	_	1 :ID
4	cela	_	_
5-6	au	_	_
5	à	_	1
6	le	_	1
7	sérieux	nsp	1
8	.	_	_

FIGURE 2.7 – **Format PARSEME-TSV - exemple :** Un exemple au format PARSEME-TSV pour la phrase *Nous devrions* ***prendre*** *cela* ***au sérieux***. où *prendre au sérieux* est annoté comme un idiome verbal. Les deux premières lignes indiquent l'identifiant et le texte de la phrase. Le symbole *« nsp »* de la troisième colonne indique que le token n'est pas suivi par un espace dans le texte original.

Format PARSEME-TSV

Le format PARSEME-TSV a été utilisé dans la cadre de la première campagne PARSEME (campagne 1.0, (Savary et al., 2017)). C'est un format tabulé, avec chaque phrase séparée par une ligne vide, puis un token ou un amalgame par ligne. Chaque token est représenté par une ligne de quatre champs : la position du token dans la phrase, sa forme fléchie, le caractère ('_') si le token est suivi par un espace dans la phrase, sinon, il contient *« nsp »*. Le dernier champ est dédié à l'appartenance à aucune, une ou plusieurs EPs. Le champ contient le caractère ('_') si le token ne fait pas partie d'une EP verbale. Sinon, ce champ contient une liste de codes d'EPs, séparés par des points-virgules. Le premier token d'une EP est repéré par un identifiant entier, suivi par une étiquette de la catégorie de l'EP. Le champ ne contient que l'identifiant de l'EP pour les autres tokens de l'EP (voir figure 2.7).

Comme indiqué section 2.3.1.0, pour certaines des langues des données PARSEME 1.0, le fichier PARSEME-TSV est accompagné d'un fichier CoNLL-U qui fournit l'annotation morphologique et l'arbre de dépendances syntaxiques.

Le fichier CoNll-U contient les phrases tokenisées de manière exactement parallèle au fichier PARSEME-TSV. Chaque token t est associé à un ensemble d'informations linguistiques : sa position dans la phrase courante, sa forme fléchie, son lemme, son étiquette morphosyntaxique universelle (Petrov et al., 2011), son étiquette morphosyntaxique, ses traits morphologiques, l'indice du mot gouverneur syntaxique de t et le label de la dépendance syntaxique entre le gouverneur et t, ainsi qu'un champ pour des informations diverses.

Format CUPT

Le format CUPT est une fusion des deux formats CoNLL-U et PARSEME-TSV que nous venons de décrire. Il s'agit d'un format tabulé comprenant les 10 colonnes du CoNLL-U, et les colonnes 3 et 4 du PARSEME-TSV, modulo quelques modifications mineures comme le remplacement du caractère (_) par un autre caractère (*) dans la dernière colonne pour indiquer que

Position	Forme fléchie	Lemme	EMS_u	...	EP
1	Cette	ce	DET	...	*
2	photo	photo	NOUN	...	*
3	a	avoir	AUX	...	*
4	subi	subir	VERB	...	1 :LVC.full ;2 :LVC.full
5	des	un	DET	...	*
6	retouches	retouche	NOUN	...	1
7	et	et	CONJ	...	*
8	des	un	DET	...	
9	recadrages	recadrage	NOUN	;..	2
10	.	.	PUNCT	;..	*

FIGURE 2.8 – **Format CUPT :** Un exemple au format *CUPT* des annotations de la phrase *Cette photo a* ***subi***$_{1,2}$ *des* ***retouches***$_1$ *et des* ***recadrages***$_2$.. Les colonnes sont les 10 colonnes du CoNLL-U (on ne montre ici que le rang, la forme fléchie, le lemme, la POS). La dernière colonne fournit les annotations d'EPs, au même format que le format PARSEME-TSV.

le token n'appartient à aucune EP. La figure 2.8 montre un exemple de phrase annotée dans ce format.

2.5 Défis

En tant que tâche à part entière, l'identification d'EPs comporte de nombreux défis, en particulier liés aux caractéristiques des EPs que sont l'ambiguïté, la discontinuité, l'enchâssement et le chevauchement, ainsi que leur variabilité (cf. section 1.2).

2.5.1 Ambiguïté

Comme pour de nombreuses tâches du TAL, l'ambiguïté est l'un des défis importants de l'identification d'EPs, puisqu'une méthode d'identification doit être capable de distinguer les occurrences littérales des occurrences idiomatiques d'une EP, afin que des tâches aval utilisant la sémantique puissent distinguer construction compositionnelle versus idiosyncrasique du sens. Par exemple, la distinction des occurrences idiomatiques et littérales de l'EP *les carottes sont cuites* est indispensable pour les modèles de traduction automatique.
Le degré d'ambiguïté est bien sûr variable d'une EP à une autre, et d'une langue à une autre. On peut fournir une mesure globale du degré d'ambiguïté d'un corpus annoté en EPs : il faut pour chaque EP annotée, rechercher les occurrences qui n'ont pas été annotées, et correspondent donc à des occurrences littérales. La recherche peut se faire de manière plus ou moins exacte (on peut rechercher exactement la même séquence de formes fléchies, voire les mêmes formes fléchies exactement dans les mêmes relations syntaxiques, ou bien à l'autre extrême, les mêmes lemmes que l'EP, dans tout ordre). Il peut être ensuite intéressant d'étudier le comportement d'un système d'identification face aux occurrences littérales, en comptant comme réponse correcte le fait de ne pas reconnaître l'occurrence. Dans leur article, Pasquer et al. (2018) projettent des patrons utilisant les lemmes et les annotations morphosyntaxiques des EPs des jeux d'entraînement de PARSEME 1.1 pour annoter les occurrences littérales. Ce travail révèle une proportion d'occurrences littérales variable selon les langues [18] avec en tout 51 % d'occurrences littérales, ce qui

18. 96 % des occurrences du turc sont littérales alors que elles ne représentent que 24 % seulement en lituanien.

démontre l'importance de l'ambiguïté lecture littérale versus lecture idiomatique.

Notez que les étiquettes morphosyntaxiques et les informations syntaxiques ainsi que l'analyse sémantique peuvent aider à désambiguïser certains cas ambigus. Par exemple, dans *Je cherche à savoir qui a écrit ce livre*, l'analyse syntaxique indique sans ambiguïté que *à savoir* n'est pas EP ici, alors qu'elle l'est dans *Il a volé tout l'argent qu'il a trouvé, à savoir 300 euros*. Ces informations ne sont pas utilisables dans le cas d'architectures séquentielles où l'identification d'EPs précède l'analyse syntaxique, mais on peut ici citer le cas d'architectures jointes où l'analyse syntaxique aide l'identification (Nasr et al., 2015; Constant et Nivre, 2016). Ce type d'architecture sera détaillé plus loin dans l'état-de-l'art.

2.5.2 Discontinuité

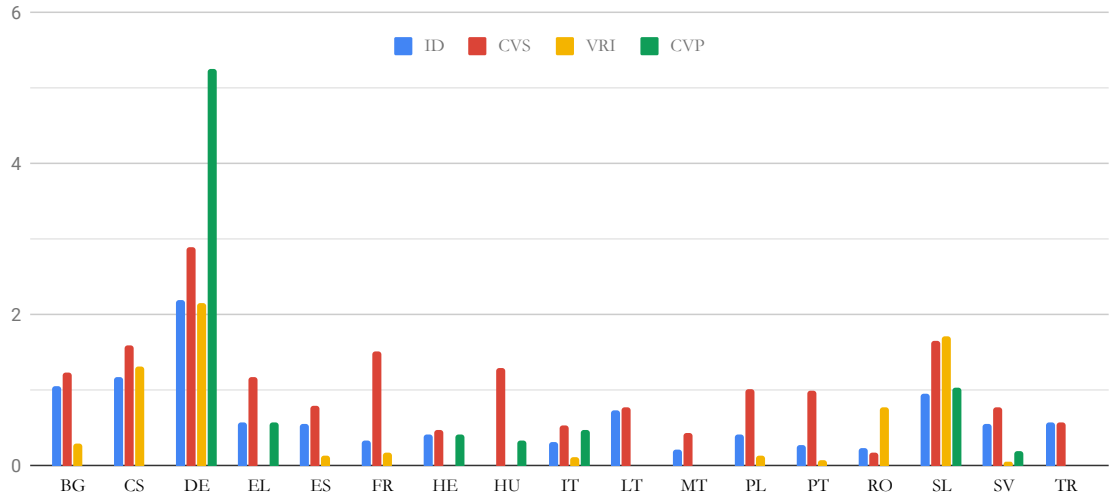

FIGURE 2.9 – **PARSEME 1.0 : Trous :** La moyenne de la longueur des trous des EPs pour les différentes catégories des langues des jeux d'entraînement des données PARSEME 1.0. Source : Savary et al. (2018).

La discontinuité des EPs rend leur identification plus difficile, la tâche étant de manière évidente plus complexe qu'une recherche de séquences continues (ne serait-ce qu'en termes combinatoires). En pratique, la discontinuité est relativement rare pour les EPs autres que verbales. Par exemple dans le « French Treebank » version 1.0 (voir section 2.3.3), on dénombre 39 EPs discontinues sur les plus de 30 000 annotées [19]. En revanche, dans les données PARSEME ciblant les EPs verbales, la discontinuité est loin d'être marginale. À noter cependant que les EPs tendent à être connectées syntaxiquement, y compris les discontinues.

Moreau et al. (2018b) mesurent en effet les proportions des EPs verbales séquentiellement continues versus discontinues et syntaxiquement connectées versus déconnectées, pour toutes les langues de PARSEME 1.1. Alors qu'une EP est considérée comme séquentiellement continue si tous ses tokens sont contigus, elle est connectée par dépendances syntaxiques si chaque token dépend d'un autre token de l'EP, à l'exception de la tête de l'ensemble de l'EP, qui dépend d'un token en dehors de l'EP. Nous reprenons leur visualisation graphique à la figure 2.10, qui montre tout d'abord que la majorité des EPs verbales est à la fois séquentiellement continue et connec-

19. À noter que les constructions à verbe support, source de discontinuités, ne sont pas annotées dans le FTB.

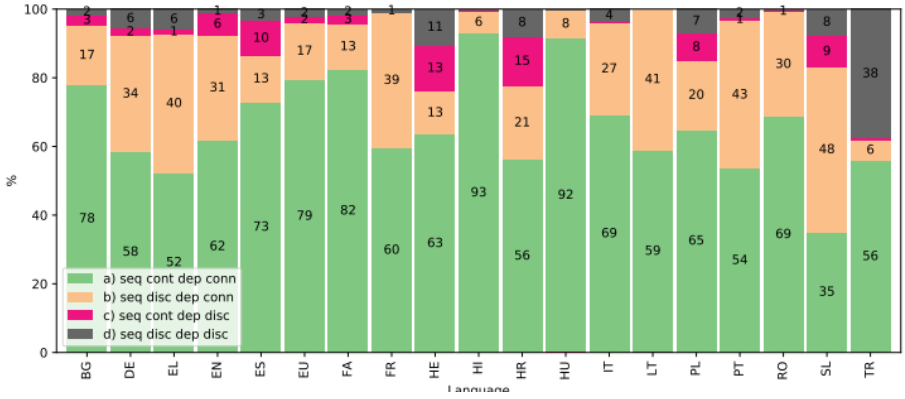

FIGURE 2.10 – **PARSEME 1.1- distribution des EPs verbales :** Proportions des EPs verbales **(a)** séquentiellement continues et connectées par dépendances (seq cont dep conn), **(b)** séquentiellement discontinues, mais connectées par dépendances (seq disc dep conn), **(c)** séquentiellement continues mais déconnectées par dépendances (seq cont disque) et **(d)** séquentiellement discontinues et déconnectées par dépendances (seq disc dep disc), pour chaque langue des jeux de données de PARSEME 1.1. Source : Moreau et al. (2018b).

tée par dépendances pour la plupart des langues (en vert). En outre, les EPs séquentiellement discontinues sont également majoritairement connectées par dépendances (en orange) pour la plupart des langues. Par exemple, l'EP *Mettre l'accent* est séquentiellement discontinue dans la phrase *L'accent est également mis sur les deux roues ..*, mais syntaxiquement connectée. Moreau et al. (2018b) en déduisent que la connectivité syntaxique est une caractéristique exploitable en théorie pour l'identification des EPs verbales, y compris discontinues.

On remarque cependant que les discontinuités sont relativement courtes pour la plupart des langues des campagnes PARSEME, comme en témoigne la figure 2.9, qui fournit la longueur moyenne des « trous » cumulés pour les EPs dans les données PARSEME 1.0. Les discontinuités les plus longues sont globalement celles des CVSs, et l'allemand se distingue avec des longueurs moyennes supérieures à 2 pour les IDs, CVSs et VIRs, et supérieures à 5 pour les CVPs.

2.5.3 Enchâssement et chevauchement

L'enchâssement et le chevauchement (cf. section 1.2.4) représentent d'autres scénarios qui peuvent compliquer l'identification des EPs. Ces deux phénomènes posent un vrai défi pour les méthodes d'identification d'EPs, puisque beaucoup de schémas utilisés dans les méthodes d'étiquetage de séquences n'ont pas le pouvoir expressif pour représenter ces deux phénomènes (voir section 2.4). Le problème ne peut donc même pas être abordé avec de telles méthodes. Alors que l'enchâssement est pris en considération par des méthodes de prédiction structurelle gloutonne par transitions (qui seront abordées chapitre 4), le chevauchement représente un autre défi pour ces méthodes ou d'autres, comme l'identification via étiquetage des séquences (abordée chapitre 3).

2.5.4 Variabilité

Les EPs grammaticales sont souvent totalement invariables et à l'inverse, les EPs verbales contiennent dans leur grande majorité un verbe tête qui se fléchit de manière régulière, apportant ainsi de la variabilité. Elles admettent également une plus grande variabilité syntaxique.

Sys	Phrase								P		R		F	
	Le	premier	ministre	a	fait	plusieurs	faux	pas	eval-cmpts	eval-eps	eval-cmpts	eval-eps	eval-cmpts	eval-eps
G	*	1_{NP}	1	*	2_{CVS}	*	$2;3_{NP}$	2;3	-	-	-	-	-	-
S_1	1	1	1	*	*	*	*	2	3/4	0/2	3/7	0/3	.55	.0
S_2	*	1_{NP}	1	*	2_{CVS}	2	2	2	5/6	1/2	5/7	1/3	.77	.40
S_3	*	*	1	*	2	*	3	2;3	5/5	1/3	5/7	1/3	.83	.33

TABLE 2.6 – **Métriques d'évaluation - exemple :** Alors que la première ligne représente les annotations gold G pour la phrase *Le* $\boldsymbol{premier}_1$ $\boldsymbol{ministre}_1$ *a* $\boldsymbol{fait}_2$ *plusieurs* $\boldsymbol{faux}_{2,3}$ $\boldsymbol{pas}_{2,3}$, les autres trois lignes représentent les annotations prédites par trois systèmes d'identification. Les six dernières colonnes fournissent les scores de précision P, de rappel R et de F-mesure F, en **eval-cmpts** et **eval-eps** pour chaque système. Notez que les annotations gold et celles du système S_2 comprennent les catégories linguistiques des EPs identifiées.

Néanmoins, les analyses morphologique et syntaxique peuvent aider à limiter l'impact de cette variabilité, en réduisant les mots à leurs formes canoniques et en bénéficiant de l'arbre syntaxique pour mieux identifier les variations de structure. C'est bien sûr au prix d'un manque de précision, car on a souvent des EPs où seuls certains composants sont variables. Par exemple dans *la cour des grands*, seul *cour* est variable.

2.6 Mesures d'évaluation

On évalue la performance des systèmes d'identification d'EPs avec les mesures traditionnelles telles que la précision, le rappel et la F-mesure, en comparant les EPs prédites par le système et les EPs dans la référence. Cette comparaison peut être calculée soit au niveau des composants des EPs soit au niveau des EPs prises dans leur globalité.

2.6.1 Évaluation au niveau des EPs (eval-eps)

La précision au niveau d'EP $\mathbf{P_{eval-eps}}$ est le nombre d'EPs correctement identifiées du jeu d'évaluation $|\mathbf{EP_{ci}}|$, divisé par le nombre d'EPs identifiées $|\mathbf{EP_i}|$ (correctement ou non). Une EP est correctement identifiée si l'ensemble des tokens de l'EP prédite correspond strictement à l'ensemble des tokens de l'EP attendue.

$$\mathbf{P_{eval-eps}} = \frac{|\mathbf{EP_{ci}}|}{|\mathbf{EP_i}|} \tag{2.1}$$

Le rappel au niveau d'EP $\mathbf{R_{eval-eps}}$ est le nombre d'EPs correctement identifiées du jeu d'évaluation $|\mathbf{EP_{ci}}|$, divisé par le nombre d'EPs attendues $|\mathbf{EP_g}|$.

$$\mathbf{R_{eval-eps}} = \frac{|\mathbf{EP_{ci}}|}{|\mathbf{EP_g}|} \tag{2.2}$$

La F-mesure **F** combine la précision et le rappel pour produire une moyenne harmonique qui incarne le score de la performance d'un système d'identification. Ce score est calculé pour

eval-eps et eval-cmpts de la même manière :

$$\mathbf{F_{eval-eps}} = \frac{\mathbf{2} * (\mathbf{P_{eval-eps}} * \mathbf{R_{eval-eps}})}{(\mathbf{P_{eval-eps}} + \mathbf{R_{eval-eps}})} \tag{2.3}$$

2.6.2 Évaluation au niveau des composants (eval-cmpts)

L'évaluation au niveau des composants considère pour chaque mot uniquement la décision binaire appartient-il ou pas à au moins une EP. La précision en eval-cmpts $\mathbf{P_{eval-cmpts}}$ est le nombre de tokens correctement annotés par le système d'identification en tant que composants d'EP, divisé par le nombre total de tokens annotés par le système en tant que composants d'EP.

$$\mathbf{P_{eval-cmpts}} = \frac{|\mathbf{T_{ci}}|}{|\mathbf{T_i}|} \tag{2.4}$$

Le rappel $\mathbf{R_{eval-cmpts}}$ se calcule de la même manière, en divisant le nombre de tokens correctement identifiés $|\mathbf{T_{ci}}|$ par le nombre attendu de tokens annotés comme composants des EPs.

$$\mathbf{R_{eval-cmpts}} = \frac{|\mathbf{T_{ci}}|}{|\mathbf{T_g}|} \tag{2.5}$$

La F-mesure associée se calcule de la manière suivante :

$$\mathbf{F_{eval-cmpts}} = \frac{\mathbf{2} * (\mathbf{P_{eval-cmpts}} * \mathbf{R_{eval-cmpts}})}{(\mathbf{P_{eval-cmpts}} + \mathbf{R_{eval-cmpts}})} \tag{2.6}$$

La table 2.6 présente un exemple pratique du calcul de ces scores pour une phrase de huit tokens et trois systèmes d'identification.

2.6.3 Évaluation de la catégorisation

Pour mesurer la performance de la catégorisation d'un système d'identification d'EPs, nous utilisons les mêmes mesures $(\mathbf{P}, \mathbf{R}, \mathbf{F})$ de l'identification. Ces scores peuvent être calculés au niveau de l'EP elle-même ou au niveau des composants, et sont calculés de la même manière pour les scores d'identification. Par exemple, la précision (pour une catégorie particulière $\mathbf{C}$) $\mathbf{P_{eval-eps(C)}}$ d'un système d'identification est le nombre d'EPs correctement identifiées (avec la catégorie correcte) $|\mathbf{EP(C)_{ci}}|$ divisé par le nombre d'EPs identifiées $|\mathbf{EP(C)_i}|$.

2.6.4 Discussion

Nous venons de présenter les mesures que nous avons adoptées pour évaluer la performance de nos systèmes et la comparer avec d'autres systèmes. Ces mesures sont les mesures adoptées pour les deux éditions de la campagne internationale pour l'identification des EPs verbales (Savary et al., 2017; Ramisch et al., 2018). Schneider et al. (2016) présente une autre approche pour calculer les mesures de précision, de rappel et de F-mesure en considérant que l'EP est un ensemble de liens entre ses tokens, et se basant sur le nombre de liens au lieu du nombre de tokens. La précision au niveau d'EP, par exemple, est la proportion de liens prédits dont les mots appartiennent à la même EP dans les annotations attendues.

La métrique eval-cmpts ne fait pas l'unanimité dans la communauté de l'identification, puisqu'elles ne prennent pas en compte «l'importance» des tokens au sein l'EP. Par exemple, l'annotation d'un verbe comme *prendre*, qui peut représenter une tête syntaxique pour d'autres tokens, n'a pas la même valeur que l'identification d'un simple déterminant comme *le* dans *prendre le*

relais puisque l'annotation erronée du verbe peut nuire à l'analyse syntaxique et même celle de la sémantique de la phrase. Constant et al. (2017) propose d'utiliser des schémas de pondération pour développer les métriques basées sur le token et pénaliser les systèmes qui échouent à identifier les tokens clés des EPs.

Enfin, comme habituellement en TAL, lorsque l'identification des EPs est effectuée avec d'autres tâches de TAL (telles que l'analyse syntaxique, la traduction automatique ou l'analyse sémantique), son évaluation peut nécessiter d'autres mesures ou être intégrée aux mesures utilisées pour ces tâches (Constant et al., 2017).

2.7 Méthodes d'identification à base de règles

Les méthodes à base de règles consistent à créer un ensemble de règles symboliques et à les appliquer sur le texte en cours de traitement pour annoter ses EPs. Les règles peuvent être créées manuellement, ce qui sollicite des connaissances linguistiques et un travail importants, ou extraites automatiquement. Les règles sont généralement utilisées pour identifier les occurrences des EPs dans le texte brut ou dans une version prétraitée du corpus cible. L'identification peut être appliquée à l'aide d'expressions régulières et de transducteurs finis. Les méthodes à base de règles représentent une solution simple pour l'identification des EPs, avec la capacité de faire face aux défis de la discontinuité, de la variabilité et de l'ambiguïté. Ces méthodes sont également capables de traiter les TPs, les EPs enchâssées et même avec chevauchement. Cependant, ces méthodes sont dépendantes de la langue, lentes à développer et nécessitent une connaissance linguistique considérable.

2.7.1 Projection de lexique d'EPs

La méthode la plus simple consiste à utiliser un lexique d'EPs et à le projeter sur le corpus cible. La projection peut utiliser des informations de lemmes et étiquettes morphosyntaxiques pour gérer la variabilité, et une heuristique de sélection de l'EP la plus longue en cas de chevauchement. Carpuat et Diab (2010) utilisent cette méthode d'identification dans le cadre d'un travail explorant plusieurs stratégies d'intégration de l'identification des EPs au sein de systèmes de traduction automatique. Les scores de traduction automatique confirment que l'identification peut augmenter l'efficacité d'une telle tâche. Pourtant, cette méthode a tendance à produire un bon rappel mais au prix d'une faible précision, l'ambiguïté entre occurrence littérale et idiomatique n'étant pas traitée.

Pour enrichir les lexiques et générer des variantes potentielles des EPs existantes, Forcada et al. (2011) définissent un paradigme de flexion. Cela permet à leur plateforme de traduction à base de règles d'identifier des occurrences variantes des EPs stockées dans les lexiques.

2.7.2 Transducteurs finis

Gross (1987) a présenté les transducteurs finis comme une technique efficace pour l'analyse lexicale des mots des langues naturelles, y compris les EPs.

Un transducteur fini est un automate à états finis qui produit une sortie en lisant une séquence en entrée. Un transducteur fini est constitué d'un nombre fini d'états liés par des transitions étiquetées avec une paire (entrée/sortie). Le transducteur fini part d'un état initial et passe par des états différents en fonction de l'élément lu en entrée et de la transition choisie, tout en produisant la sortie associée à la transition parcourue. Les méthodes d'identification qui utilisent cette technique utilisent la plupart du temps les transducteurs finis pour représenter des lexiques d'EPs

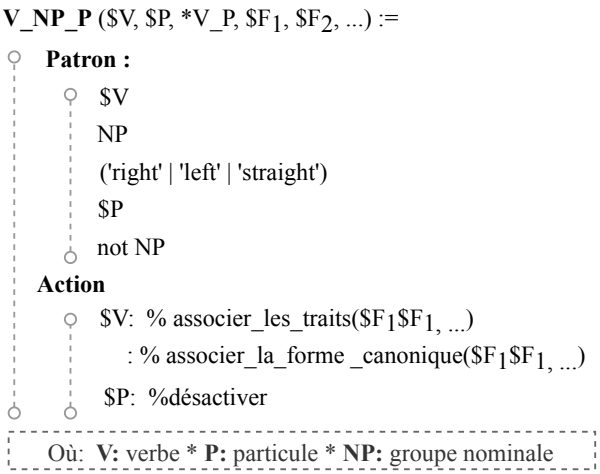

FIGURE 2.11 – **Macro d'identification des verbes à particule** Une macro définie pour identifier des verbes à particule comme *take off (enlever)*. Le patron de la macro identifie l'expression si un verbe $\$V$ est suivi par un groupe nominal NP, un adverbe (facultatif) de l'ensemble {« right », « back », « straight »}, une particule, et à condition qu'il ne soit pas suivi par un groupe nominal. Les instructions « associer_les_traits » et « associer_la_forme_canonique » extraient des traits à la suite de l'identification et en y basant pour servir l'étape suivante, celle de l'analyse syntaxique. L'action « désactiver » marque la particule comme faisant partie du verbe à particule. Source : Li et al. (2003).

pour définir les transitions de leurs transducteurs finis. Les transducteurs finis offrent un cadre générique et simple, permettant de gérer la variabilité, la discontinuité et l'ambiguïté (Constant et al., 2017) ainsi que le traitement de l'enchâssement et du chevauchement.

Chanod et Tapanainen (1996) présentent l'une des premières méthodes d'identification à base de règles dans le cadre d'un analyseur lexical, visant à alimenter un analyseur syntaxique. Cet analyseur lexical incluant un transducteur fini qui code une grande variété d'EPs. Ce transducteur est en réalité un simple lexique d'EPs et d'expressions régulières pour les ponctuations et les expressions numériques et sert à repérer les occurrences d'EPs continues, en privilégiant les plus longues en cas d'ambiguïté. L'analyseur ne peut donc pas distinguer occurrences idiomatiques et occurrences littérales.

Afin de gérer la variabilité lexicale et syntaxique des EPs, Breidt et al. (1996) utilisent des grammaires locales [20], écrites comme des expressions régulières pour un transducteur fini. Pour réduire à la fois l'ambiguïté et la variabilité, Hashimoto et al. (2016) divisent les idiomes japonais en trois classes : (1) les idiomes ni variables ni ambiguës, (2) les idiomes variables, mais non ambigus, (3) les idiomes variables et ambigus. Des dictionnaires de contraintes lexicales à base d'étiquettes morphosyntaxiques et de la structure interne des idiomes sont ensuite construits. Ces dictionnaires permettent au système de détecter les formes admissibles de variabilité des occurrences des deuxième et troisième classes et de désambiguïser les occurrences idiomatiques

20. Une grammaire locale est un ensemble de contraintes syntaxiques locales ciblés sur des éléments lexicaux obéissant à des règles spécifiques (comme par exemple le système des clitiques verbaux en français, comme *ne, le, lui*) (Mohri, 2005).

des occurrences littérales de la troisième classe.

Li et al. (2003) identifient les verbes à particule anglais en utilisant une méthode de reconnaissance par patrons à base d'un formalisme appelé lexique expert (« Expert Lexicon »). Le module d'identification est positionné, dans un système en cascade (« pipeline »), entre un module de segmentation syntaxique et un module d'analyse syntaxique dans l'objectif d'intégrer les interactions morphosyntaxiques. Le lexique expert est un formalisme basé sur un index, pouvant associer des règles de reconnaissance par patrons à des entrées lexicales. Par exemple, Li et al. (2003) utilisent la macro de la figure 2.11 pour identifier les verbes à particule tels que *take (the coat) off* (*Retirer (le manteau)*). Cependant, le codage de telles macros nécessite des connaissances linguistiques et un corpus de développement représentatif pour le débogage des règles. La méthode donne de très bons résultats, en comparaison avec la performance des méthodes à base de tables de correspondances sur la même catégorie.

Ramisch (2015) présente une boite à outils générique (« MWE-Toolkit »), qui permet de définir des règles d'identification et d'extraction des EPs continues et discontinues, enchâssées ou avec chevauchement grâce à des heuristiques paramétrables. La paramétrabilité de cette boite à outils représente sa contribution principale, puisque les paramètres permettent de fournir des heuristiques très flexibles en contrôlant la taille minimale et maximale de trous potentiels pour les expressions discontinues, les étiquettes morphosyntaxiques du voisinage, les variantes locales, etc.

2.8 Identification par classification binaire de candidats

Certaines méthodes s'inspirent des méthodes de désambiguïsation lexicale, puisqu'elles réduisent le problème d'identification à un problème de classification des occurrences d'EPs entre idiomatiques et littérales.
Pour expérimenter leurs modèles, certaines méthodes de cette approche se basent sur la littérature d'extraction pour repérer les occurrences, i.e les candidats. D'autres utilisent des ressources dont les candidats sont préannotés. Les modèles de classification varient, puisqu'il existe des modèles supervisés et non supervisés, linéaires et neuronaux.

2.8.1 Méthodes supervisées pour la classification de candidats préannotés

Ces méthodes ne s'occupent pas de l'extraction des occurrences d'EPs et partent soit de données où les candidats sont déjà repérés, soit ont réalisé leur propre extraction de candidats.

Hashimoto et Kawahara (2008) ont construit un corpus japonais où les occurrences des idiomes sont annotées et catégorisées comme littérales ou idiomatiques. Pour classifier ces occurrences, Hashimoto et Kawahara (2008) utilisent une méthode standard de désambiguïsation lexicale via un classifieur de type machine à vecteurs de support (« Support Vector Machine » (SVM)) exploitant les patrons de traits classiques dans la littérature de la désambiguïsation lexicale, ainsi que d'autres patrons spécifiques aux idiomes. Les patrons de traits classiques incluent les formes fléchies, les lemmes, les étiquettes morphosyntaxiques, les hyperonymes, les têtes et les dépendants syntaxiques des tokens du contexte de l'occurrence.

Les patrons de traits spécifiques aux idiomes essaient d'apprendre les régularités et les rigidités morphologiques et/ou syntaxiques, qui peuvent aider la classification. Ces traits indiquent par exemple si les constituants de l'idiome sont adjacents, si un composant nominal de l'idiome

a un modifieur, si un suffixe passif ou causatif est attaché à un composant verbal de l'idiome, etc.

Katz et Giesbrecht (2006) confirment empiriquement que le contexte linguistique local peut fournir des indices adéquats pour classifier les occurrences non compositionnelles d'EPs. Ils utilisent ensuite l'analyse sémantique latente pour produire un modèle sémantique distributionnel, à la fois pour les tokens simples et les occurrences d'EPs, littérales ou idiomatiques. Plus précisément, Katz et Giesbrecht (2006) produisent une représentation vectorielle de l'occurrence en sommant les vecteurs de ses composants et la compare au vecteur de l'occurrence comme une seule unité, en présumant que les occurrences non compositionnelles se distingueront par un faible score de similarité.

Boukobza et Rappoport (2009) utilisent des traits de surface pour entraîner un SVM binaire et distinguer les occurrences littérales des occurrences idiomatiques. Les traits de surface concernent aussi bien la forme canonique de l'EP que l'occurrence de l'EP en contexte, avec des informations telles que :

— le nombre de mots entre les tokens de l'EP les plus à gauche et les plus à droite. Par exemple, il existe trois tokens entre les deux extrémités de l'EP *Lire en diagonale* dans la phrase *J'ai lu cet article en diagonale* ;

— une liste des longueurs des trous entre chaque paire de tokens de l'expression en fonction de leur ordre dans la forme canonique. Par exemple, il existe un trou de longueur 2 entre *lire* et *en* de l'exemple précédent ;

— une valeur booléenne indiquant si l'ordre des mots de l'expression dans la phrase correspond à celui de la forme canonique, ce qui est vrai pour le même exemple ;

— la proportion de tokens de l'EP qui étant une variante flexionnelle par rapport à la forme canonique. Le verbe *lire* représente la seule variante flexionnelle dans l'occurrence *lu en diagonale* et le trait aura la valeur de $1/3$;

— si les étiquettes morphosyntaxiques de l'occurrence sont identiques à celles de la forme canonique, ce qui est vrai dans notre exemple.

Ces traits « surfaciques » se sont révélés plus efficaces que des traits utilisant des informations syntaxiques, issus du sous-arbre minimal contenant les tokens de l'EP, tels que le nombre d'arêtes dans le sous-arbre minimal, une liste des distances des nœuds de l'EP, par rapport à la racine, une liste des relations des descendants et frères entre chaque paire de nœuds de l'EP, etc.

2.8.2 Méthodes non supervisées pour la classification de candidats préannotés

Les méthodes non supervisées ne bénéficient pas de telles annotations pour construire leurs modèles. Elles se basent donc sur des hypothèses (linguistiques) pour distinguer les occurrences littérales des occurrences idiomatiques

Cook et al. (2007) obtiennent des scores compétitifs avec ceux de Katz et Giesbrecht (2006) en développant des méthodes non supervisées pour la classification des occurrences de constructions verbe-nom, où le nom est l'objet direct du verbe. Leur approche suppose que, dans la plupart des cas, les occurrences idiomatiques d'une EP ont tendance à se produire dans un petit

nombre de formes canoniques et que les occurrences littérales sont moins limitées syntaxiquement. Ils construisent ainsi une base de patrons de formes canoniques résumant les manifestations syntaxiques d'une expression idiomatique.

L'une des méthodes expérimentées détermine les formes canoniques d'une expression comme étant celles dont la fréquence est beaucoup plus élevée que la fréquence moyenne de toutes ses formes. D'autres méthodes supposent que les occurrences littérales et idiomatiques d'un idiome se produiront généralement dans des contextes différents. Ils enrichissent donc leur base de patrons syntaxiques par des informations de contexte local, en construisant des vecteurs de cooccurrence pour les formes canoniques littérales et d'autres vecteurs pour les formes canoniques idiomatiques.

Li et Sporleder (2010) présentent un autre modèle non supervisé compétitif pour la classification des occurrences littérales et idiomatiques en utilisant des traits sémantiques du contexte. Cette méthode est basée sur l'idée que l'occurrence figurée présente des liens moins sémantiques et cohérents avec le contexte que l'occurrence littérale. Cette méthode suppose que les deux classes d'occurrences sont générées par deux gaussiennes différentes et utilise le modèle de mélange gaussien (« gaussian mixture models » en anglais), appris grâce à l'algorithme de maximisation d'espérance, afin de modéliser les données en deux clusters dont la distribution de chacun est gaussienne. Le choix de ce modèle vient de sa capacité à généraliser et à s'adapter aux occurrences figurées ainsi que les occurrences littérales (Li et Sporleder, 2010).
Le modèle utilise une mesure de distance sémantique (Normalized Google Distance) pour modéliser le degré d'association sémantique (i.e. relatedness) et représente les données par des traits de lien sémantique, telles que le degré d'association moyen entre l'expression cible et les tokens de son contexte, le degré d'association sémantique moyen au sein du contexte de l'expression cible, un vecteur creux représentant le score du lien entre les composants de l'expression cible et le contexte, etc.

2.8.3 Méthodes supervisées pour la classification des candidats automatiques

Une autre approche consiste à faire une identification en deux temps, avec d'abord l'extraction d'occurrences candidates, suivie d'une classification binaire de celles-ci pour décider s'il s'agit effectivement d'une occurrence d'EP ou pas.

Pasquer et al. (2018) se concentrent sur l'identification des EPs verbales déjà vues dans un corpus d'entraînement, pour identifier toute manifestation de ces EPs, et apprendre un modèle stochastique capable de traiter la variabilité et l'ambiguïté. La méthode commence par analyser le corpus d'apprentissage pour extraire les patrons les plus fréquents des EPs verbales annotées. Les patrons s'appuient sur les étiquettes morphosyntaxiques des EPs du corpus d'entraînement (ex. *NOUN* et *VERB* pour l'EP *decision made*) et ses lemmes (ex. *decision* et *make*).

Dans un deuxième temps, la méthode extrait un ensemble de candidats contenant des exemples positifs (occurrences idiomatiques) et négatifs (occurrences littérales) d'EPs. Alors que les instances idiomatiques sont déjà annotées dans le jeu de données d'apprentissage, les patrons extraits dans la première étape sont utilisés pour extraire les occurrences littérales (non annotées).

À la fin, une fonction d'extraction de traits encode le jeu d'entraînement créé pour entraîner un modèle stochastique (classifieur naïf bayésien) à différencier l'occurrence idiomatique de l'occurrence littérale. Cette méthode, indépendante de la langue et expérimentée sur 19 langues, n'arrive pas à produire des scores compétitifs avec ceux produits dans le cadre de PARSEME 1.1 bien qu'il soit compétitif sur les EPs déjà vues dans le corpus d'apprentissage.

Représentation de candidats par vecteurs denses

Une des grandes différences entre les modèles linéaires et les réseaux de neurones dans le domaine de TAL est l'utilisation des vecteurs denses pour représenter les éléments d'entrées. Cette représentation est caractérisée par son pouvoir de généraliser et d'intégrer les interactions entre les traits, puisque les méthodes de génération de ces représentations peuvent produire des vecteurs similaires pour des éléments similaires d'un point de vue donné. Par exemple, le vecteur dense pour *chien* peut être similaire au vecteur appris pour *chat* lors de l'utilisation d'une méthode qui favorise l'aspect sémantique.
Par ailleurs, les réseaux de neurones nécessitent la minimisation du nombre de paramètres pour réduire le temps d'entraînement. Par conséquent, il vaut mieux éviter l'utilisation des vecteurs one-hot ou des vecteurs creux et représenter les traits dans des espaces vectoriels denses.

Ces vecteurs denses associent chaque ensemble de traits de base à un espace de faible dimensionnalité et associent chaque trait à un point de cet espace. Cette représentation se traduit par un vecteur de paramètres qui peuvent intégrer les réseaux de neurones et être l'objet de la propagation d'activation (Goldberg, 2016).

La génération de vecteurs denses aléatoires consiste à indexer les traits et à associer chaque index à un vecteur généré aléatoirement. Un mécanisme de table de correspondance est alors chargé de transformer les indices en vecteurs. Cependant, il existe plusieurs approches pour générer des vecteurs de plongement lexical de mots. Ils peuvent être générés aléatoirement ou par des méthodes de préentraînement supervisées ou non supervisées. *Word2vec* est l'une des méthodes courantes non supervisées qui apprend les plongements lexicaux en utilisant le contexte ; pour prédire un mot cible (sac continu de mots) ; ou en utilisant un mot pour prédire un contexte cible (skip-gramme) (Goldberg et Levy, 2014). *FastText* est un autre exemple couramment utilisé dans la littérature d'identification d'EPs (Zampieri et al., 2019) et dans d'autres domaines où la morphologie compte (Bojanowski et al., 2016). Cette méthode représente les mots sous forme de n-grammes de caractères, puis additionne les vecteurs pour obtenir une représentation complète de chaque mot, en considérant que les informations au niveau du caractère (y compris les affixes et les règles grammaticales) peuvent aider le modèle à développer des représentations sémantiques plus généralisées et ainsi mieux représenter les mots rares.

Legrand et Collobert (2016) bénéficient des plongements lexicaux préentraînés pour représenter les n-grammes par des vecteurs denses. Leur réseau de neurones reçoit les plongements lexicaux des tokens des n-grammes extraits pour les associer à une couche dense. De cette façon et en utilisant un réseau de neurones pour chaque longueur des n-grammes, la sortie de la couche dense, dont la dimension est identique pour tous les réseaux utilisés, produit une représentation continue de taille fixe pour des n-grammes de longueurs variées. Legrand et Collobert (2016) utilisent ensuite ces représentations denses pour entraîner un classifieur, qui va déterminer, au temps d'analyse, si le n-gramme en cours de traitement est une EP ou non. Notez que le réseau exploite également des plongements des étiquettes morphosyntaxiques pour produire la représentation dense de chaque n-gramme.

La méthode suppose que chaque token de la phrase doit appartenir à un seul n-gramme, c'est-à-dire, à une seule EP. Pour maintenir cette limitation, Legrand et Collobert (2016) crée un graphe structuré des n-grammes extraits et utilise l'algorithme de Viterbi [21] pour trouver le meilleur chemin du treillis produit. Cette méthode dépasse les scores de l'état de l'art obtenues sur les données du FTB (Seddah et al., 2013).

21. L'algorithme de Viterbi est un algorithme de programmation dynamique, couramment utilisé en TAL, permettant de trouver la séquence la plus probable dans un espace d'observations (Forney, 1973).

Alors que cette méthode a été expérimentée sur un jeu de données dont toutes les EPs sont continues, elle peut théoriquement traiter les EPs discontinues en extrayant les skip-grammes. Cependant, l'hypothèse de départ de cette méthode l'empêche de traiter les scénarios d'enchâssement et de chevauchement et peut entraîner une baisse importante de performance pour certaines langues comme l'allemand et le slovène.

2.9 Résumé

Dans ce chapitre, nous avons commencé par définir la tâche d'identification d'EPs, comme le repérage d'occurrences d'EPs dans du texte, par opposition à l'extraction d'EPs dans de gros corpus. Nous avons présenté les motivations sous-jacentes à l'identification d'EPs : d'une part l'idiosyncrasie inhérente aux EPs fait que, non traitées spécifiquement, les EPs peuvent perturber les autres tâches d'analyse de phrase. D'autre part, une idiosyncrasie formelle est en général le signe d'une irrégularité dans la composition sémantique, et doit donc être prise en compte pour toute tâche aval intégrant une partie d'analyse sémantique.

Nous avons ensuite présenté les jeux de données annotés en EPs utilisés dans notre travail. On retiendra en particulier les données PARSEME, focalisées sur les EPs verbales, et disponibles pour de nombreuses langues (18 pour la première version des données, PARSEME 1.0, et 20 pour la seconde PARSEME 1.1). Les caractéristiques des EPs verbales annotées est une discontinuité fréquente (avec une proportion d'EPs verbales discontinues variables selon les langues, mais de plus de 50% pour certaines), et une variabilité morphologique et syntaxique, du fait que la syntaxe des EPs verbales est en général régulière et donc sujette à variation. D'autres cas formels comme un enchâssement complet d'une EP dans une autre, ou un chevauchement de deux EPs, partageant certains tokens, sont finalement assez marginales dans les données PARSEME.
Les deux autres jeux de données utilisés dans quelques unes de nos expérimentations sont d'une part le French Treebank, qui comprend des articles en français du journal *Le Monde*, où (entre autres) des EPs de tous types mais presqu'exclusivement continues sont annotées, y compris des entités nommées, et d'autre part, les données DiMSUM, qui sont des tweets et critiques en ligne en anglais, annotées en EPs. Dans DiMSUM, les combinaisons ayant une idiosyncrasie seulement statistique (les collocations) sont également annotées, comme « EPs faibles », distinguées des « EPs fortes ».

Une fois les données présentées, nous avons détaillé les défis posés pour la tâche d'identification d'EPs. Outre la variabilité, la discontinuité, l'enchâssement et le chevauchement, la tâche est compliquée par l'ambiguïté en contexte entre une lecture idiomatique d'une séquence (qui est alors effectivement une EP), versus une lecture littérale ou versus une co-occurrence accidentelle des composants de l'EP.

Nous avons également détaillé dans ce chapitre les détails permettant une expérimentation rigoureuse : formats d'annotation et métriques d'évaluation de la tâche d'identification, soit au niveau des EPs complètes (eval-eps) soit au niveau des composants d'EPs (eval-cmpts).

Nous avons terminé ce chapitre par un début d'état de l'art de l'identification d'EPs, avec deux techniques non centrales à notre travail : (i) l'identification déterministe d'EPs, par projection d'un lexique d'EPs sur des textes (éventuellement compilé en transducteur), avec le problème notable du non traitement de l'ambiguïté de lecture littérale versus idiomatique ; et (ii) l'identification d'EPs en deux phases, avec repérage manuel ou automatique de candidats EPs en contexte, suivi d'une classification binaire des candidats en EPs effectives ou pas. Ce cadre formel se prête bien à l'utilisation de techniques actuelles en TAL, à savoir l'utilisation de représentations denses et contextualisées des éléments à classifier (ici les candidats EPs).

Chapitre 3

Identification d'EPs comme un étiquetage de séquences

Ce chapitre est dédié aux méthodes d'identification des EPs qui se basent sur un étiquetage séquentiel. Nous commencerons par des travaux utilisant cette technique pour des tâches de TAL autres mais qui intègrent l'identification des EPs (section 3.1). La section 3.2 s'intéresse aux méthodes d'étiquetage de séquences pour l'identification des EPs en profitant de l'ordre linéaire des tokens de la phrase et/ou de sa structure syntaxique. La section 3.3 est consacrée à une méthode atypique et innovante, nommée TRAVERSAL, qui parcourt la structure syntaxique de la phrase pour étiqueter ses tokens et constituer ensuite les EPs à partir des étiquettes produites. Enfin, la section 3.4 récapitule les points essentiels du chapitre.

3.1 Identification des EPs intégrée à d'autres tâches de TAL

Compte tenu de la rareté des ressources annotées en EPs avant l'apparition des jeux de données de PARSEME, l'identification a connu un début timide. L'identification des EPs était au départ le plus souvent incluse dans d'autres tâches du TAL, en utilisant des annotations qui incluent entre autres l'information concernant les EPs. Ces méthodes se basent souvent sur l'étiquetage de séquences, utilisée dans de nombreuses tâches du TAL. Cette section présente plusieurs méthodes dédiées à différentes tâches, et montre, pour celles-ci, comment elles ont intégré l'identification des EPs.

3.1.1 Super-étiquetage

Blunsom et Baldwin (2006a) proposent une méthode de super-étiquetage [22] à base de Champ aléatoire conditionnel (Conditional Random Field (CRF)), et l'applique à la tâche d'apprentissage de nouvelles unités lexicales pour les grammaires HPSG (Grammaire syntagmatique guidée par les têtes) en anglais et en japonais, où les tokens des phrases sont accompagnés par des étiquettes de leurs types lexicaux. Le super-étiqueteur considère les EPs continues comme des unités lexicales incluant des espaces et les identifie en conséquence. Les EPs continues sont décrites comme une seule entrée lexicale incluant des espaces, et possèdent les mêmes attributs

22. Le super-étiquetage (supertagging) est la tâche de prédire des super-étiquettes de constituants syntaxiques pour un formalisme linguistique tel que la grammaire combinatoire catégorielle. Son objectif est d'améliorer l'efficacité de l'analyse syntaxique globale en prévoyant des super-étiquettes prédites localement (Rama et al., 2014).

que les entrées lexicales simples. Les EPs discontinues sont traitées différemment puisque leurs composants sont définis comme des entrées lexicales distinctes, avec des annotations pour indiquer leur association. Le modèle CRF linéaire est optimisé grâce à des traits indépendants de la langue et issus du token et de son contexte, tels que des unigrammes et des bigrammes, ainsi que des traits lexicaux tels que les préfixes, les suffixes.

3.1.2 Étiquetage morphosyntaxique

Constant et Sigogne (2011) utilisent également un modèle à base de CRF pour constituer leur étiqueteur morphosyntaxique intégrant l'identification d'EPs. L'intégration de la tâche d'étiquetage morphosyntaxique et d'identification des EPs est réalisée en utilisant un schéma IOB (voir section 2.4.1), dont les étiquettes combinent l'étiquette morphosyntaxique de l'EP (ainsi que les tokens non appartenant aux EPs) avec l'étiquette IOB permettant l'identification des EPs. Constant et Sigogne (2011) donne l'exemple de *pouvoir d'achat*, dont l'étiquette morphosyntaxique est *NC*, ce qui produit l'étiquetage *« Le/D+O pouvoir/NC+B d'/NC+I achat/NC+I a/V+O augmenté/V+O »*.

La méthode commence par réaliser une analyse lexicale préliminaire de la phrase, en bénéficiant de lexiques morphologiques à grande couverture et de ressources d'EPs, afin de produire un transducteur fini acyclique représentant les séquences d'étiquetage candidates. Le modèle CRF doit ensuite sélectionner la séquence la plus probable. Ce modèle utilise des traits concernant le token à étiqueter, ainsi que des tokens de son contexte linéaire, ainsi que des traits basés sur des ressources externes. Les traits sur le token en cours de traitement sont d'une part des informations orthographiques (la forme fléchie, la forme minusculisée, des suffixes et préfixes (de différentes tailles), si la forme du token contient un trait d'union, un chiffre, si elle est en majuscule, en minuscule, etc.) et d'autre part des informations issues des lexiques : si le token est annoté comme appartenant à une EP dans les lexiques externes, et sa « classe d'ambiguïté », c'est-à-dire la liste des étiquettes morphosyntaxiques possibles pour ce token. Les traits incluent également les formes fléchies dans une fenêtre autour du token courant, ainsi que des bigrammes de formes fléchies dans cette fenêtre, et enfin la catégorie du token précédent (cf. l'inférence est réalisée par un CRF linéaire).

Le système proposé atteint des performances dépassant l'état de l'art sur le French TreeBank. Ce travail montre en particulier l'efficacité de l'intégration des traits basés sur des ressources externes et confirme que l'identification des EPs peut être bénéfique à l'étiquetage morphosyntaxique.

3.1.3 Analyse syntaxique en constituants

Constant et al. (2012) évaluent deux stratégies empiriques pour intégrer l'identification des EPs dans l'analyse syntaxique en constituants. La première stratégie est de commencer par identifier les EPs (grâce à un système du même type que celui de la section précédente), de fusionner en un seul token les EPs ainsi prédites, puis de procéder à l'analyse syntaxique. La seconde stratégie consiste à intégrer l'identification des EPs au sein de la grammaire apprise sur le jeu d'entraînement pour directement prédire un arbre en constituants qui inclut les EPs. Un modèle de type entropie maximale (« MaxEnt ») est ensuite utilisé pour reclasser les arbres de sortie et choisir le meilleur arbre syntaxique.

Le modèle exploite des traits basés sur les tokens (unigrammes et bigrammes), ainsi que leurs étiquettes morphosyntaxiques. D'autres traits indiquent par exemple les occurrences potentielles d'EPs du texte se trouvant dans un lexique externe, ce qui permet au classifieur de faire plus

ou moins confiance au lexique suivant le contexte d'occurrence. Avec la première stratégie (pré-regroupement des EPs prédites par CRF), les scores d'identification d'EPs et le score d'analyse syntagmatique non étiquetée sont améliorés. Avec la seconde stratégie, l'amélioration concerne également le score d'analyse syntagmatique étiquetée.

3.1.4 Analyse syntaxique en dépendances

Le travail de Constant et al. (2013) concerne cette fois l'intégration de l'identification d'EPs et de l'analyse syntaxique en dépendances. Ces auteurs mettent à profit une version du FTB où les arbres syntaxiques en dépendances intègrent des annotations d'EPs (Seddah et al., 2013) au moyen d'une étiquette de dépendances spéciale « `dep_cpd` »[23].

Constant et al. (2013) comparent deux types d'architecture : la première est une architecture séquentielle où l'identification précède une analyse syntaxique avec EPs prédites fusionnées en un seul token. La seconde architecture est simplement l'application directe d'un analyseur en dépendances, la prédiction des étiquettes de dépendances `dep_cpd` étant interprétée comme de l'identification d'EPs. Une dernière étape concerne la prédiction d'une étiquette morphosyntaxique pour chaque EP prédite, au moyen d'un modèle à maximum d'entropie, qui utilise un ensemble de traits lexicaux de l'EP, de ses composants et de son contexte syntaxique et lexical.

Dans le cadre de la campagne internationale SPMRL 2013 pour l'analyse en dépendances de langues à morphologie riche (Seddah et al., 2013), les auteurs ont concouru pour les données du français. L'architecture séquentielle arrive nettement en tête par rapport à des analyseurs syntaxiques simples utilisant une analyse jointe. Ceci est compréhensible car cette architecture utilise des informations issues de lexiques externes d'EPs, ce que ne fait pas l'architecture jointe. Cette architecture produit un score F de 80,8 pour l'identification des EPs.

Sur le même jeu de données, Candito et Constant (2014) réalisent également la double tâche identification d'EPs et analyse en dépendances. Ils proposent une approche mettant à profit que certaines EPs ont une structure syntaxique régulière, tout en ayant une idiosyncrasie sémantique. Or le statut d'EP est représenté dans les données avec une structure plate (comme par exemple la représentation de l'EP *abus de biens sociaux* dans le haut de la figure 3.1). Candito et Constant (2014) font l'hypothèse que découpler la régularité syntaxique de la régularité sémantique, pour certaines EPs, peut aider à l'analyse syntaxique en renforçant la cohérence des annotations syntaxiques. Ils commencent par étiqueter les EPs du FTB en syntaxiquement régulière ou pas (via des patrons morphosyntaxiques). Ils transforment ensuite automatiquement la représentation des EPs syntaxiquement régulières. Ainsi par exemple 3.1, *abus de biens sociaux* récupère une structure régulière, et son statut d'EP est disponible via des traits. En revanche, *en vain*, irrégulier, garde sa structure plate.

Les auteurs testent ensuite plusieurs architectures (séquentielles et jointes), certaines faisant un traitement différent pour les EPs syntaxiquement régulières et les autres. Une architecture complètement jointe se base sur une représentation comme au bas de la figure 3.1, mais où le statut d'EP des EPs syntaxiquement régulières est injecté comme suffixe des étiquettes de dépendances.

C'est l'architecture complètement jointe intégrant des traits issus d'un identifieur d'EPs qui obtient les meilleurs résultats de performance comparable à celle des travaux précédents sur le même jeu de données. Ces résultats montrent donc que ce type de représentation n'améliore

23. Le token le plus à gauche d'une EP annotée est considéré comme la tête, et tous les autres composants y sont rattachés avec une dépendance étiquetée `dep_cpd`. Il s'agit de la première représentation intégrant EPs et syntaxe en dépendances.

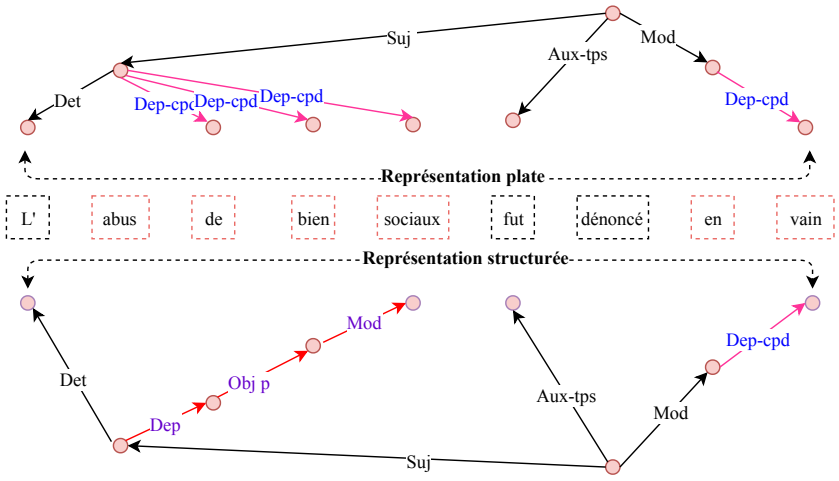

FIGURE 3.1 – **Représentations de Candito et Constant (2014) versus celle de FTB :** Un exemple montrant la différence entre la représentation plate d'EPs dans le jeu de données FTB et la représentation proposée par Candito et Constant (2014). Cet exemple comprend deux EPs : *Abus de bien sociaux* et *En vain* qui correspondent respectivement à une EP syntaxiquement régulière et une EP irrégulière. Alors que la première représentation associe tous les tokens d'une EP à son premier token avec la étiquette *dep_ cpd*, la représentation de Candito et Constant (2014), hiérarchise l'EP syntaxiquement régulière avec des relations syntaxiques explicites. Les EPs irrégulières gardent leur représentation plate dans les deux cas. Source : Candito et Constant (2014).

pas en soi l'identification d'EPs ni l'analyse syntaxique, mais les auteurs soulignent que pour un même niveau de performance, la sortie de leur analyseur est plus riche, car fournissant la structure interne de certaines EPs, ce qui peut être utile pour celles ayant un certain degré de compositionnalité sémantique.

Nasr et al. (2015) proposent un modèle d'analyse en dépendances et identification d'EPs mais focalisé sur certaines EPs irrégulières, telles que les constructions formées d'un adverbe suivi de *que* (comme *bien que*) ou celles composées de *de* suivi d'un déterminant (e.g., *de la* est codé comme EP dans *Il boit de la bière.* mais pas dans *Il parle de la bière*). De la même manière que supra, les EPs ont une représentation plate avec une étiquette de dépendances spécifique (mais cette fois-ci c'est le *que* qui est considéré comme la tête). L'originalité de ce travail tient à l'incorporation sous forme de traits d'informations sur la sous-catégorisation syntaxique des verbes, issus d'un lexique syntaxique (plus précisément si un verbe sous-catégorise ou pas une complétive en *que*, et/ou un complément prépositionnel en *de*). L'intuition est de pouvoir par exemple repérer que *bien que* est EP dans *Il mange bien qu'il n'ait pas faim*, mais pas dans *Il sait bien qu'il n'a pas faim*, grâce au trait que *manger* ne sous-catégorise pas de complétive, contrairement à *savoir*. D'après Nasr et al. (2015), ce modèle arrive à reconnaître 81,79% des conjonctions *ADV + que* avec une précision de 91,57% et 82,74% des déterminants *de + DET* avec une précision de 86,70%.

3.1.5 Étiquetage morphosyntaxique et analyse en dépendances

USzeged est un système d'identification d'EPs verbales, proposé par Simkó et al. (2017) et expérimenté sur les jeux de données de PARSEME 1.0. Inspiré de Vincze et al. (2013), USzeged se sert de l'analyse syntaxique en dépendances et de l'étiquetage morphosyntaxique pour identifier les EPs. La différence avec les systèmes cités dans la section précédente est qu'il s'agit ici de trouver des EPs contenant plusieurs tokens mais également des EPs à un seul token (TP), pour lesquels c'est une étiquette morphosyntaxique spéciale qui signale le caractère EP.

Pour identifier les EPs à plusieurs tokens, le système remplace le libellé des relations de dépendance par la catégorie linguistique de l'EP. Par exemple, l'annotation de l'EP *Prendre décision* dans la phrase *prends ta décision.* remplace la relation d'objet entre *prendre* et *décision* et la remplace par *CVS*. Les relations syntaxiques entre les tokens des EPs composées de plus de deux tokens subissent, toutes, le même remplacement. Notez que bien que cette méthode réduise la quantité des informations syntaxiques véhiculées, elle n'affecte pas la structure des relations de dépendances pour les EPs à tokens multiples. D'autre part, le remplacement se passe au niveau des étiquettes morphosyntaxiques pour les TPs, c'est-à-dire, le TP *court-circuiter* sera annoté comme idiome verbal (ID).

À la suite de ce traitement, un taggeur morphosyntaxique et un analyseur de dépendances (Bohnet, 2010) sont appris et utilisés pour prédire les étiquettes morphosyntaxiques et les dépendances syntaxiques. La dernière étape consiste à extraire les EPs à partir des annotations produites en considérant que chaque catégorie produite dans les étiquettes morphosyntaxiques représente un TP et que chaque série, connectée, de libellés catégorielles dans l'arbre syntaxique de dépendances est une EP à tokens multiples. Par conséquent, les EPs à tokens multiples non connectées syntaxiquement ne peuvent pas être identifiées par cette méthode.

Le système a été expérimenté sur neuf langues de PARSEME 1.0 et a produit un score F de 25,4 (versus un score F de 52,0 pour les meilleurs systèmes sur ces langues). Par contre, ce système a produit les meilleurs scores sur l'allemand et le hongrois, qui connaissent une proportion considérable de TPs.

3.2 Méthodes de type étiquetage de séquences

De nombreux travaux ramènent la tâche d'identification d'EPs à un étiquetage séquentiel. Il s'agit d'associer à une séquence de tokens une unique séquence d'étiquettes de type IOB ou IOB-iob (cf. 2.4.1), encodant de manière non ambiguë où sont les EPs identifiées. Toutes les techniques usuelles de classification séquentielle par apprentissage supervisé peuvent être utilisées. Cette architecture s'est révélée efficace et performante pour l'identification d'EPs. En revanche, les schémas traditionnels utilisés dans ce genre de méthodes n'ont pas le pouvoir expressif pour représenter les EPs enchâssées ou en chevauchement, ni donc de les identifier.

Nous présentons dans cette section un certain nombre de ces méthodes en nous concentrant sur les jeux d'étiquettes, les traits et les modèles exploités.

3.2.1 Méthodes par apprentissage « classique »

Diab et Bhutada (2009) s'intéressent à la classification d'occurrences de constructions comportant un verbe et un nom, qui peuvent avoir une interprétation tantôt littérale tantôt idiomatique. Pour cela, ils utilisent un système d'apprentissage supervisé classique (système d'étiquetage de séquences YamCha [24] en intégrant un schéma IOB particulier.

24. YamCha (Yet Another Multipurpose CHunk Annotator) est étiqueteur séquentiel à base de SVM, utilisant

Celui-ci utilise cinq étiquettes : *BL* (début d'une occurrence littérale), *IL* (à l'intérieur d'une occurrence littérale), *BI* (début d'une occurrence idiomatique), *II* (à l'intérieur d'une occurrence idiomatique), *O* (à l'extérieur d'une occurrence).

Les traits utilisés pour représenter une position dans la phrase couvrent des fenêtres de contexte, des attributs linguistiques du token à cette position tels que les trois derniers caractères, l'étiquette morphosyntaxique, la forme fléchie, le lemme, etc. Cette approche a permis de dépasser le score de l'état de l'art (un score F de 84,6% correspondant à un score F de 90,0% pour l'identification des occurrences idiomatiques et de 62,0% pour les occurrences littérales.) sur l'ensemble de données utilisé Cook et al. (2008) [25].

Le système ADAPT de Maldonado et al. (2017) est l'un des premiers modèles à base de CRF pour les EPs verbales. Ce système utilise des traits s'appuyant entre autres sur les dépendances syntaxiques pour apprendre un modèle CRF classique (à l'aide d'une librairie CRF++) [26].

Le travail de Maldonado et al. (2017) explore l'impact de nombreux patrons de traits sur la performance d'un modèle CRF classique. ADAPT a été paramétré et évalué pour plusieurs jeux de traits sur plusieurs langues des jeux de données PARSEME 1.0 (une langue par famille linguistique). Les jeux de traits les plus importants sont les unigrammes et bigrammes à base de traits linguistiques basiques, tels que les formes fléchies, les lemmes, les étiquettes morphosyntaxiques, la tête syntaxique, la relation syntaxique du token en cours de traitement avec son contexte. Ce système a obtenu des scores très compétitifs dans la campagne internationale PARSEME 1.0 sur la plupart des langues et a réalisé le premier score eval-cmpts pour trois langues (français, polonais et suédois).

Inspirés par Schütze (1998) et Maldonado-Guerra et Emms (2011), Moreau et al. (2018a) étendent le travail de Maldonado et al. (2017) en ajoutant un module sémantique de reclassement (reranker). Ce module sert à corriger une partie des prédictions erronées produites par le module CRF. Moreau et al. (2018a) s'inspirent des travaux de la sémantique distributionnelle Schütze (1998); Maldonado-Guerra et Emms (2011) pour calculer des vecteurs de contexte pour les EPs, en comptant les fréquences des mots qui se produisent au voisinage de l'EP dans un grand corpus externe. Un modèle d'arbres de régression est ensuite appris sur ces traits pour reclasser les dix EPs candidates les plus probables.

Le module de reclassement permet une augmentation de 11,9% du score F_{avg}(eval-eps) pour les langues de PARSEME 1.1, mais beaucoup moins (1,1%) pour F_{avg}(eval-cmpts). Cela signifie que le module de reclassement réussit à discriminer parmi des analyses proches en termes d'eval-cmpts, mais qui diffèrent en termes d'identification d'EPs entières.

3.2.2 Méthodes à base de réseaux de neurones

Gharbieh et al. (2017) expérimentent et évaluent plusieurs réseaux de neurones (un réseau à propagation avant, un réseau récurrent et un réseau convolutionnel), sur le jeu de données DiMSUM, pour un système d'identification d'EPs par étiquetage séquentiel avec schéma IOB. Ce système transforme les représentations vectorielles creuses de Schneider et al. (2014a) en vecteurs denses pour les adapter aux entrées attendues de la part des réseaux de neurones.

des noyaux polynomiaux de degré 2. `http://chasen.org/~taku/software/yamcha/`.

25. Le jeu de données Cook et al. (2008) comprend 2920 occurrences de constructions formées d'un verbe et d'un nom, tirées du British National Corpus (BNC) et annotées manuellement

26. CRF ++ est une implémentation open source et extensible de CRF pour l'étiquetage de données séquentielles. Source : CRF++ : Yet Another CRF toolkit(`https://taku910.github.io/crfpp/`)

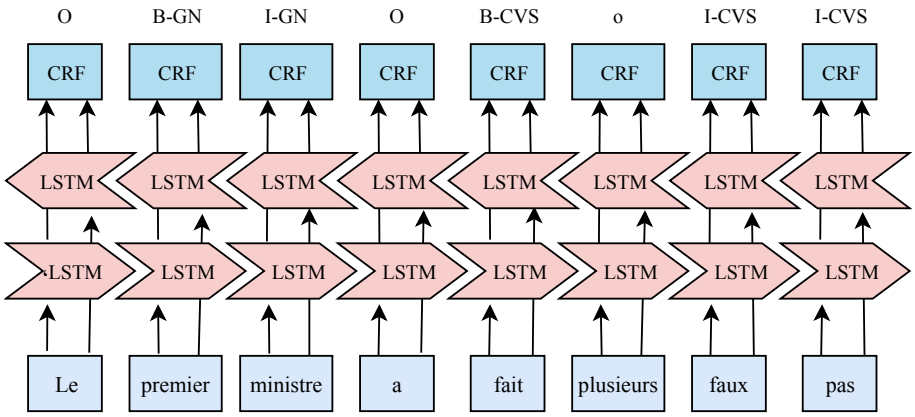

FIGURE 3.2 – **Système Deep-GBT :** Le réseau de neurones bidirectionnel du système proposé par Berk et al. (2018). Notez que chaque token de l'entrée est représenté par plusieurs plongements ainsi que des vecteurs de valeurs binaires et discrètes , représentant divers traits du token. Les vecteurs binaires et discrètes sont utilisés pour représenter les espaces vectoriels de faible dimension comme les informations orthographiques. Source : Berk et al. (2018).

Le réseau à propagation avant comprend une seule couche cachée et est alimenté par des plongements lexicaux préentraînés des tokens et des lemmes de la séquence d'entrée, ainsi que par des vecteurs binaires (« one-hot ») pour les étiquettes morphosyntaxiques. Le réseau reçoit également des vecteurs binaires précisant des traits morphologiques et typographiques pour indiquer l'existence de chiffres ou de caractères spéciaux au sein des tokens de la séquence. Par ailleurs, le réseau profite des traits issus de multiples lexiques d'EPs, utilisés par Schneider et al. (2014a), indiquant si les tokens et les lemmes font potentiellement partie d'une EP et selon quel lexique. Ils utilisent également le moyennage des plongements lexicaux de tous les tokens de la phrase abordée pour alimenter le réseau en traits contextuels.

En ce qui concerne le réseau récurrent, composé d'une seule couche récurrente, le token correspondant à chaque stade temporel est représenté à l'aide des mêmes traits, à l'exception de ceux qui représentent le contexte de l'EP, étant donné que ce type d'informations est capturé par la couche récurrente.

Le réseau convolutionnel représente chaque token par le même ensemble de traits que ceux du réseau de propagation avant. Plusieurs filtres sont ensuite appliqués sur ces ensembles de traits pour extraire différents traits locaux à travers l'utilisation de différentes tailles de fenêtres. De plus, une opération de « maxpooling » est exécutée sur ces valeurs pour alimenter la couche cachée qui produit la sortie prédite. Ce réseau convolutionnel a été expérimenté avec divers nombres de couches cachées. Après un réglage fin des hyperparamètres, les réseaux ont été évalués sur le jeu de test de DiMSUM. Les résultats montrent que le réseau convolutionnel dépasse l'état de l'art précédent (59,96 versus 58,69 pour Kirilin et al. (2016)) et qu'en outre le réseau récurrent généralise sur les EPs inconnues de manière plus efficace que les autres architectures.

Klyueva et al. (2017) présentent l'un des premiers réseaux de neurones pour l'identification des EPs verbales. Ils abordent le problème de l'identification comme un problème d'étiquetage de séquences, utilisant un schéma IOB-iob où l'étiquette (B), indiquant la tête d'une EP, est remplacée par la catégorie de l'EP. Le réseau neuronal est composé d'une seule couche récurrente bidirectionnelle avec des unités GRU (Gated Recurrent Units), connectée à une couche dense

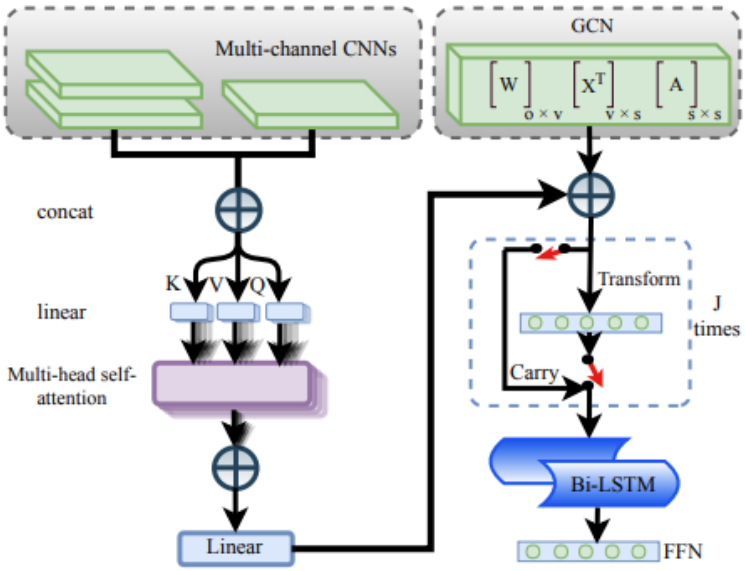

FIGURE 3.3 – **Réseau de neurones de Rohanian et al. (2019) :** La structure neuronale proposée par Rohanian et al. (2019), qui réunit une structure à base de convolution de graphes (à droite en haut) et une autre à base d'attention autonome à multiples têtes (à gauche) par un mécanisme de synchronisation (à droite en bas). Source : Rohanian et al. (2019).

avec une fonction d'activation softmax renvoyant une distribution de probabilités des étiquettes de sortie possibles.

Chaque mot en entrée est représenté comme la concaténation des plongements, initialisés de manière aléatoire, de la forme fléchie du token, de son lemme et de son étiquette morphosyntaxique. L'article fournit les valeurs des hyperparamètres sans parler d'un véritable réglage de ces valeurs. Ce modèle a l'avantage d'une structure récurrente, simple, indépendante de la langue, permettant l'identification et la catégorisation des EPs, mais sa performance est très modeste.

Deep-GBT est un système hybride CRF-LSTM bidirectionnel, proposé par Berk et al. (2018). Ce réseau neuronal est constitué des couches suivantes : une couche d'entrée pour les tokens, leurs étiquettes morphosyntaxiques, et les étiquettes de leurs relations syntaxiques avec leurs dépendants ; une couche de plongements qui associe les valeurs discrètes d'entrée avec leurs représentations continues ; une couche LSTM (Long short-term memory) bidirectionnelle recevant les représentations continues et une couche CRF qui reçoit la sortie du LSTM et qui permet de prédire les étiquettes de la séquence (voir 3.2). Notez que ce réseau utilise les vecteurs préentraînés de FastText (Grave et al., 2018).

Deep-GBT adopte le schéma IOB-iob pour transformer les phrases annotées en séquences d'étiquettes. Ce schéma permet de représenter l'imbrication d'une EP au sein d'une autre EP (voir section 2.4.1). Le système s'est classé premier pour deux langues (bulgare et allemand), et obtient une F-mesure très compétitive pour le polonais, mais perd sa compétitivité pour les autres langues de la campagne internationale PARSEME 1.1.

Inspiré par le travail de Ma et Hovy (2016) sur la reconnaissance des entités nommées, le réseau de Taslimipoor et Rohanian (2018) est composé d'un module convolutionnel, de couches récurrentes et d'une couche CRF au sommet. Le module convolutionnel est composé de deux

couches, avec une taille de fenêtre de 2 pour l'une et de 3 pour l'autre, qui extraient les bigrammes et les trigrammes les plus intéressants de la phrase. Les traits extraits sont ensuite concaténés et transmis aux couches LSTM bidirectionnelles. Les couches récurrentes transmettent leur sortie à une couche CRF afin de prédire les étiquettes de sortie en utilisant une optimisation globale sur la séquence. Cette couche est facultative et peut être remplacée par une couche softmax traditionnelle.

Outre des plongements lexicaux non modifiables [27], le réseau incorpore des représentations one-hot des étiquettes morphosyntaxiques et des traits typographiques et morphologiques supplémentaires, déterminant si le token commence par une majuscule, ne comprend que des majuscules, etc.

Le système bat tous les systèmes participants à la campagne internationale PARSEME 1.1 dans les open et closed tracks (F_{avg}(eval-eps) = 58,09) et dépasse les modèles état de l'art pour toutes les langues cibles en termes de généralisation (i.e. identification des EPs inconnues). Les plongements lexicaux préentraînés jouent un rôle clé dans la performance de ce système, où la F-mesure (eval-eps) perd 10 points (sur DE, EN, ES, FA, FR) avec des plongements lexicaux initialisés de manière aléatoire. Par ailleurs, l'utilisation de la couche CRF ne rapporte pas une amélioration significative du modèle.

Alors que Taslimipoor et Rohanian (2018) utilisaient un module convolutionnel standard comme détecteurs de n-grammes séquentiels, Rohanian et al. (2019) utilisent un module convolutionnel pour les graphes (GCN) pour pallier la modélisation inadéquate des EPs discontinues, en exploitant les arbres syntaxiques de dépendances au lieu des séquences linéaires. Ce module utilise une procédure efficace de propagation, basée sur une approximation au premier ordre des convolutions spectrales sur les graphes (Kipf et Welling, 2016).

Les auteurs proposent une seconde architecture où un module convolutionnel est connecté à un module d'autoattention (« Self-attention ») à multiples têtes (Vaswani et al., 2017) pour tenir compte de la discontinuité des EPs et associer des tokens non contigus de la séquence, indépendamment de leurs distances (Vaswani et al., 2017).

Enfin, une troisième architecture, illustrée dans la figure 3.3, réunit les deux précédentes avec un mécanisme de « porte » (« gating mechanism »), qui utilise des couches « autoroutes » à propagation avant (« feed-forward highway layers ») (Srivastava et al., 2015). Cette architecture combinée dépasse nettement l'état de l'art de l'identification d'EPs des données PARSEME 1.1, pour les quatre langues ciblées dans ce travail (allemand, anglais, français, farsi).

3.3 TRAVERSAL : Système atypique d'étiquetage global au sein d'un arbre syntaxique

Waszczuk (2018) propose l'un des systèmes les plus innovants de la campagne internationale PARSEME 1.1. Son système, nommé TRAVERSAL, remplace les séquences lexicales qu'utilisent les méthodes d'étiquetage de séquences par les arbres de dépendances distribués dans les données de la campagne PARSEME 1.1. TRAVERSAL repose sur l'hypothèse que les EPs sont majori-

27. Pour apprendre la représentation de tokens, Taslimipoor et Rohanian (2018) utilisent des plongements lexicaux préentraînés. La méthode de production de ces vecteurs repose sur l'hypothèse selon laquelle les informations au niveau du caractère, contenues dans les n-grammes de caractères, aident le modèle à développer des représentations distributionnelles plus généralisées, ce qui permet de mieux représenter les mots rares. Les plongements lexicaux sont ainsi appris en représentant les mots sous forme de n-grammes de caractères, puis en sommant les vecteurs pour obtenir une représentation complète de chaque mot.

tairement connectées par dépendances syntaxiques. La tâche d'identification est ramenée à une tâche **(i)** d'étiquetage des nœuds de l'arbre syntaxique en `EP` ou `non EP`, suivie d'une tâche **(ii)** de « segmentation en EPs » permettant de constituer les EPs à partir des étiquettes `EP` ou `non-EP` posées sur chaque token.

La tâche (i) exploite la structure syntaxique d'entrée. Un parcours d'arbre est défini, des nœuds vers leur frère gauche, et du premier fils vers son nœud père. Mais chaque nœud de l'arbre syntaxique est remplacé par quatre nœuds (ci-après super-nœud pour plus de clarté), correspondant aux quatre combinaisons pour le booléen « ce super-nœud est `EP` ou `non-EP` », et « son gouverneur syntaxique est `EP` ou `non-EP` », définissant ainsi un hypergraphe. Un étiquetage complet `EP` ou `non-EP` correspond à un parcours dans l'hypergraphe, avec un choix de super-nœud pour chaque nœud. Le système construit initialement l'hypergraphe avec tous les super-nœuds, puis tous les parcours valides (c'est-à-dire les parcours entre super-nœuds qui garantissent une étiquette unique pour un nœud au sein du parcours), construit une représentation vectorielle des arcs des parcours. Chaque parcours est représenté par la somme des vecteurs de ses arcs. Le parcours optimal est sélectionné au moyen d'un classifieur de type régression logistique multiclasse. La représentation vectorielle d'un arc utilise différentes combinaisons d'attributs des deux super-nœuds n et m de l'arc (les attributs linguistiques (forme fléchie, lemme, POS) de n et m , leur étiquette `EP` ou `non-EP`, le label de la dépendance syntaxique entrante de dans l'arbre syntaxique.

Étant donné que la méthode ne prédit pas le début et la fin des EPs, Waszczuk (2018) applique la sous-tâche (ii) de segmentation d'EPs, qui considère que tous les nœuds étiquetés `EP` en (i) et adjacents dans l'arbre syntaxique appartiennent à la même occurrence d'EP.

En se basant sur les annotations syntaxiques des données PARSEME 1.1, TRAVERSAL s'est classé premier de la « closed track » de la campagne internationale PARSEME 1.1, selon les deux mesures d'évaluation officielles (F_{avg}(eval-eps) = 54,0 et F_{avg}(eval-cmpts) = 59,7). L'utilisation des arbres de dépendances renforce la capacité du système en matière de traitement des EPs discontinues, puisque le système s'est également particulièrement bien comporté pour les EPs discontinues.

3.4 Résumé

Nous avons commencé dans ce chapitre d'état de l'art par résumer les travaux intégrant l'identification d'EPs au sein d'autres types d'étiquetage, en particulier l'étiquetage morphosyntaxique. Nous avons cité la technique de "super-étiquetage" de Blunsom et Baldwin (2006b), revenant à combiner analyse lexicale et désambiguisation morphosyntaxique. Les travaux de Constant et Sigogne (2011) utilise un étiquetage séquentiel de type CRF, avec un schéma d'étiquetage combinant étiquette morphosyntaxique (POS) et statut d'EP.
De là, nous avons cité les travaux qui combinent identification d'EPs et analyse syntaxique automatique, caractérisés par l'utilisation d'une représentation syntaxique intégrant le marquage des EPs, permettant d'utiliser ensuite toute technique d'analyse syntaxique pour réaliser conjointement les deux tâches, que ce soit en analyse en constituants (Constant et al., 2012) ou en dépendances (Constant et al., 2013; Candito et Constant, 2014; Nasr et al., 2015; Simkó et al., 2017).

Nous sommes ensuite revenu à la tâche d'identification d'EPs traitée isolément, et avons passé en revue des travaux traitant l'identification d'EPs comme une tâche d'étiquetage de séquences, au moyen de jeu d'étiquettes de type IOB indiquant si un token débute ou est au sein d'une EP, ou bien n'appartient par à une EP. Les travaux utilisant cette approche sont nombreux, ils se dis-

tinguent par le jeu exact d'étiquettes utilisées (permettant ou pas un enchâssement, permettant ou pas une catégorisation des EPs). Nous avons cité des travaux "pré-neuronaux", comme (Diab et Bhutada, 2009; Maldonado et al., 2017) utilisant une approche de type CRF, et des travaux plus récents utilisant une architecture neuronale, comme (Berk et al., 2018) utilisant une couche récurrente bi-LSTM, utilisant un extracteur de traits de type réseau convolutionnel (CNN), soit sur n-grammes séquentiels (Taslimipoor et Rohanian, 2018) soit un module convolutionnel pour les graphes (GCN) exploitant des arbres de dépendances, ainsi qu'un mécanisme d'auto-attention (Rohanian et al., 2019). Ces deux derniers travaux obtiennent des résultats très compétitifs : le système SHOMA de Taslimipoor et Rohanian (2018) est classé premier de PARSEME 1.1 pour l'open track, et surtout la version de la même équipe (Rohanian et al., 2019) a des résultats très nettement supérieurs pour les 4 langues sur lesquelles ce travail se focalise.

Enfin nous avons terminé ce chapitre par le système TRAVERSAL de Waszczuk (2018), un système innovant réalisant l'identification d'EPs à partir d'un arbre de dépendances syntaxiques, avec une première étape revenant à un étiquetage binaire des tokens en "EP" ou "non EP" suivi d'une identification d'EPs comme tout sous-arbre syntaxique de tokens étiqueté EP. L'originalité de l'approche concerne la première étape, où la classification binaire résulte d'une recherche de parcours optimal au sein d'un hypergraphe dérivé de l'arbre de dépendances syntaxiques. Le système s'est classé premier dans la closed track de PARSEME 1.1.

Nous allons dans le chapitre suivant continuer notre état de l'art, en consacrant un chapitre spécifique à l'approche par transitions pour l'identification d'EPs, qui est celle que nous avons privilégiée dans tout notre travail.

Chapitre 4

Identification d'EPs et systèmes par transitions, apprentissage multitâche

Nous présentons dans ce chapitre un état de l'art de deux approches techniques que nous avons amplement utilisées dans nos travaux de thèse : l'analyse automatique par transitions, et l'apprentissage multitâche.

Les systèmes par transitions sont connus pour leur capacité à décomposer les problèmes de prédiction de structure en une séquence de problèmes de prédictions locales. Ils sont principalement utilisés dans des systèmes d'analyse syntaxique, en particulier en dépendances, et ils sont au cœur de notre approche technique pour l'identification d'EPs. C'est pourquoi nous présenterons dans une première section les concepts et les mécanismes principaux des systèmes par transitions à base de pile (section 4.1). Nous présenterons ensuite des systèmes par transitions plus élaborés permettant de réaliser conjointement l'analyse syntaxique ainsi que l'identification d'EPs (section 4.2). Nous consacrerons la section 4.3 à l'utilisation de réseaux de neurones pour les classifieurs prédisant les transitions.

Enfin, étant donné que nous proposons une architecture multitâche pour l'apprentissage des tâches d'identification des EPs, d'étiquetage morphosyntaxique et de l'analyse syntaxique en dépendances, nous présentons, dans la section 4.4, les concepts principaux de l'apprentissage multitâche, ses mécanismes d'apprentissage et des architectures multitâches visant l'identification des EPs et celles qui ont inspiré notre travail. La section 4.5 présente un résumé du chapitre.

4.1 Systèmes par transitions

Les systèmes par transitions sont couramment utilisés pour l'analyse syntaxique et d'autres problèmes de prédiction de structures. Pour présenter les concepts principaux d'un système par transitions, nous allons nous baser sur l'article de Nivre (2008) qui détaille un cadre formel et les algorithmes d'analyse déterministe pour les systèmes à piles et ceux à listes. Nous nous concentrerons ici sur les systèmes à pile, puis détaillerons le système de transitions arc-standard qui est un des plus célèbres pour l'analyse en dépendances projectives. Notons que toutes les formalisations que nous allons donner dans cette section sont issues du système Arc-standard (voir section 4.1.2).

Un système par transitions prend en entrée une séquence d'éléments et construit une sortie structurée à partir de ceux-ci, comme par exemple un arbre. Cette construction se fait en appliquant une séquence d'actions (traditionnellement appelées « transitions ») choisies parmi un ensemble prédéfini d'actions possibles, pour construire incrémentalement la structure de sortie.

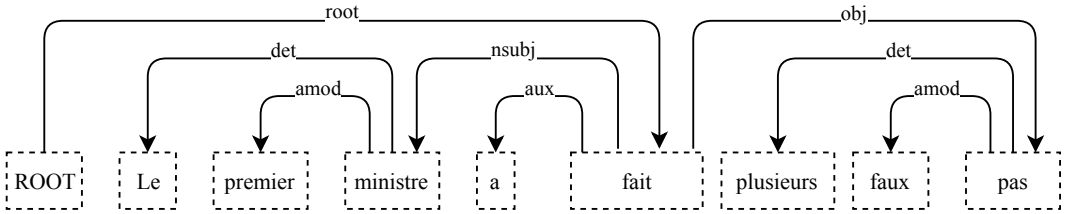

FIGURE 4.1 – **Arbre de dépendances - exemple :** L'arbre de dépendances pour la phrase *Le* ***premier***$_1$ ***ministre***$_1$ *a* ***fait***$_2$ *plusieurs* ***faux***$_{2,3}$ ***pas***$_{2,3}$ en utilisant le schéma d'annotation de dépendances universelles (« Universal dependencies »). Notez que *root* est un nœud artificiel représentant la tête de toute la phrase et que les arêtes sont orientées de la tête syntaxique vers le dépendant et que *det, nsubj, nmod, ..* représentent les types de relations syntaxiques pouvant lier les deux tokens des arêtes. Source : Nivre (2008).

Lors de l'analyse d'une phrase, chaque transition est fournie par un oracle ou prédite par un classifieur, d'après l'état courant du système, appelé une « configuration ».

Un analyseur par transitions se compose de deux modules : d'une part le jeu de transitions et leurs conditions d'application, d'autre part un appareil pour décider pour une configuration courante donnée quelle transition appliquer.

Nous décrivons infra tout d'abord la formalisation d'un arbre syntaxique, puis la définition générale d'un système par transitions à pile pour l'analyse syntaxique en dépendances, ainsi que l'algorithme d'analyse. Nous détaillons ensuite le jeu de transitions « arc-standard ». Puis, nous passons aux modules permettant de choisir la transition à appliquer à une configuration courante : en phase de prédiction d'arbres de dépendances, il s'agit d'un classifieur supervisé, en phase d'apprentissage de ce classifieur, il s'agit d'un oracle.

4.1.1 Formalisation d'un arbre de dépendances

La structure syntaxique d'une phrase peut être modélisée par un arbre de dépendances, qui représente chaque mot et ses dépendants syntaxiques grâce à des arêtes orientées et étiquetées (voir figure 4.1). Par commodité, il est usuel en analyse syntaxique d'ajouter un mot artificiel *ROOT* servant de racine de l'arbre (et permettant ainsi qu'il n'y ait pas une seule racine lexicale, ce qui correspond alors à une analyse partielle). Plus formellement, étant donné un ensemble $\mathbf{L} = \{\mathbf{l_1}, ..., \mathbf{l_{|L|}}\}$ d'étiquettes de dépendances, un arbre de dépendances pour une phrase $\mathbf{P} = (\mathbf{m_0}, \mathbf{m_1}, ..., \mathbf{m_n})$ est un graphe orienté étiqueté $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, où

1. $\mathbf{V} = \{\mathbf{0}, \mathbf{1}, ..., \mathbf{n}\}$ est un ensemble de nœuds correspondant à la position linéaire d'un mot dans la phrase (y compris $ROOT = 0$) ;
2. $\mathbf{A} \subseteq \mathbf{V} \times \mathbf{L} \times \mathbf{V}$ est un ensemble de triplets ordonnés $\{(\mathbf{i}, \mathbf{l}, \mathbf{j})\}$, où $\mathbf{i}$ est la tête, $\mathbf{j}$ le dépendant et $\mathbf{l}$ l'étiquette de dépendances, tel que tout nœud $\mathbf{i} > \mathbf{0}$ a exactement un gouverneur (i.e. apparaît dans exactement un triplet).

4.1.2 Systèmes par transitions pour l'analyse en dépendances

Définition Un système par transitions pour l'analyse syntaxique en dépendances est un quadruplet $\mathbf{S} = (\mathbf{C}, \mathbf{T}, \mathbf{F_i}, \mathbf{C_t})$, où

1. $\mathbf{C}$ est l'ensemble des configurations, chacune contenant un tampon $\mathbf{B}$ de nœuds (restants à traiter), une pile $\mathbf{S}$ de nœuds en cours de traitement et un ensemble $\mathbf{A}$ d'arcs de dépendances ;

Transition	$\text{Configuration}_{avant}$		$\text{Configuration}_{apres}$	Précondition
SHIFT	$(S, m\|B, A)$	$\rightarrow$	$(S\|m, B, A)$	
LEFT ARC	$(S\|s_1, s_0, B, A)$	$\rightarrow$	$(S\|s_0, B, A \cup \{(s_0, l, s_1)\})$	$\|S\| >= 2$ and $s_1 \neq Root$
RIGHT ARC	$(S\|s_1, s_0, B, A)$	$\rightarrow$	$(S\|s_1, B, A \cup \{(s_1, l, s_0)\})$	$\|S\| >= 2$

FIGURE 4.2 – **Arc-standard - ensemble de transitions :** Ensemble de transitions définissant le système arc-standard (nom de la transition, configuration d'entrée, configuration après l'application de la transition, conditions d'applicabilité de la transition

.

2. $\mathbf{T}$ est l'ensemble des transitions possibles, chacune d'elles étant une fonction (partielle) $t : C \rightarrow C$;
3. $\mathbf{F_i}$ est une fonction d'initialisation associant une phrase $\mathbf{P} = (\mathbf{m_0}, \mathbf{m_1}, ..., \mathbf{m_n})$ à une unique configuration initiale $\mathbf{c_i} = (\mathbf{B} = [\mathbf{1}, ..., \mathbf{n}], \mathbf{S} = [\mathbf{ROOT}], \{\})$;
4. $\mathbf{C_t}$ est l'ensemble de configurations terminales $\mathbf{C_t} = \{(\mathbf{B} = [\,], \mathbf{S} = [\mathbf{ROOT}], \mathbf{A})\}$.

L'ensemble des transitions possibles est donc prédéfini, et opère des manipulations sur le tampon, la pile et/ou l'ensemble d'arcs de la configuration d'entrée.

Construction de l'analyse syntaxique d'une phrase L'algorithme d'analyse d'une phrase préalablement segmentée en tokens $\mathbf{P} = (\mathbf{m_0}, \mathbf{m_1}, ..., \mathbf{m_n})$ commence par générer la configuration initiale $\mathbf{c_0} = \mathbf{F_i}(\mathbf{P})$. Une boucle doit ensuite d'une part choisir la prochaine transition à appliquer (grâce à un oracle ou bien un classifieur), et appliquer effectivement la transition choisie à la configuration courante, pour obtenir une nouvelle configuration. La boucle termine dès lors que l'on atteint une configuration terminale.

Ainsi formellement, une analyse de $\mathbf{P}$ au moyen d'un tel système est une séquence $\mathbf{C_{0:k}} = (\mathbf{c_0}, \mathbf{c_1}, ..., \mathbf{c_k})$ où :

1. $\mathbf{c_0} = \mathbf{F_i}(\mathbf{P})$;
2. $\mathbf{c_k} \in \mathbf{C_t}$;
3. $\mathbf{c_i} = \mathbf{t}(\mathbf{c_{i-1}}) : \mathbf{i} : (\mathbf{1} \leq \mathbf{i} \leq \mathbf{k})$ et $\mathbf{t} \in \mathbf{T}$.

L'application des transitions ajoute des arcs de dépendances, ce qui construit l'analyse en dépendances de la phrase au fur et à mesure. Au final, l'ensemble des arcs de la configuration finale est interprété comme étant la représentation syntaxique de la phrase d'entrée. À noter que selon les jeux de transitions, on a la garantie ou pas que l'ensemble des arcs de la configuration terminale est bien un arbre connecté. À noter également qu'en général, la séquence de transitions permettant d'aboutir à un arbre donné n'est pas forcément unique.

Arc-standard : un système de transitions célèbre pour l'analyse en dépendances De nombreux jeux de transitions ont été définis dans la littérature pour produire des arbres de dépendances (voire pour certains des graphes plus généraux), chacun ayant des propriétés différentes. Nous ne donnons ici qu'un des plus célèbres, le système arc-standard, sur lequel nous avons fondé la plupart de nos systèmes d'identification d'EPs.

La figure 4.2 détaille l'ensemble de transitions définissant le système arc-standard pour l'analyse syntaxique en dépendances (Nivre, 2004a) [28]. Chaque transition est définie par un ensemble

28. Nous utilisons une notation avec un | pour désigner une structure au sein de la pile ou du tampon : $S|m$

Trans.		$C = (S,$	$B,$	$A)$
$F_i(S)$	$\rightarrow$	[Root]	[le, premier, ..]	[ ]
Shift	$\rightarrow$	[Root, le]	[premier, ..]	A
Shift	$\rightarrow$	[Root, le, premier]	[ministre, ..]	A
Shift	$\rightarrow$	[Root, le, premier, ministre]	[a, ..]	A
Left arc	$\rightarrow$	[Root, le, ministre]	[a, ..]	A $\cup(ministre, amod, premier)$
Left arc	$\rightarrow$	[Root, ministre]	[a, ..]	A $\cup(ministre, det, le)$
Shift	$\rightarrow$	[Root, ministre, a]	[fait, ..]	A
Shift	$\rightarrow$	[Root, ministre, a, fait]	[plusieurs, ..]	A
Left arc	$\rightarrow$	[Root, ministre, fait]	[plusieurs, ..]	A $\cup(fait, aux, a)$
Left arc	$\rightarrow$	[Root, fait]	[plusieurs, ..]	A $\cup(fait, nsubj, ministre)$
Shift	$\rightarrow$	[Root, fait, plusieurs]	[faux, ..]	A
Shift	$\rightarrow$	[Root, fait, plusieurs, faux]	[pas]	A
Shift	$\rightarrow$	[Root, fait, plusieurs, faux, pas]	[ ]	A
Left arc	$\rightarrow$	[Root, fait, plusieurs, pas]	[ ]	A $\cup(pas, amod, faux)$
Left arc	$\rightarrow$	[Root, fait, pas]	[ ]	A $\cup(pas, det, plusieurs)$
Right arc	$\rightarrow$	[Root, fait]	[ ]	A $\cup(fait, obj, pas)$
Right arc	$\rightarrow$	[Root]	[ ]	A $\cup(Root, root, fait)$

FIGURE 4.3 – **Arc-standard - exemple :**Une séquence de transitions (avec les configurations intermédiaires) permettant de produire l'arbre de la figure 4.1 pour la phrase $S =$ *Le* ***premier***$_1$ ***ministre***$_1$ *a* ***fait***$_2$ *plusieurs* ***faux***$_{2,3}$ ***pas***$_{2,3}$ avec le système arc-standard.

de conditions d'applicabilité et par un ensemble de modifications à apporter sur la configuration d'entrée. La transition LEFT ARC ajoute un arc où le premier token de la pile est la tête du deuxième. Elle élimine le deuxième token de la pile. Cette transition ne s'applique que si la pile a au moins deux tokens, le deuxième n'étant pas *ROOT*. La transition RIGHT ARC ajoute un arc où le deuxième token de la pile est la tête du premier. Elle élimine le premier token de la pile. La transition SHIFT déplace le premier token du tampon au-dessus de la pile.

Nivre (2008) montre d'une part que ce jeu de transitions (avec utilisation du nœud fictif ROOT) produit nécessairement un arbre de dépendances dit projectif [29], et d'autre part qu'arc-standard est « complet » : pour tout arbre projectif de racine ROOT il existe au moins une séquence de transitions arc-standard permettant de le produire. La figure 4.3 illustre le fonctionnement du système arc-standard et la construction graduelle de l'ensemble des arcs de dépendances, pour former l'arbre de dépendances de la figure 4.1.

Choix de la transition à appliquer La particularité des systèmes par transitions pour l'analyse syntaxique est l'ambiguïté quant à la transition à appliquer pour une configuration donnée. Depuis le début des années 2000, avec en particulier les travaux de Yamada et Matsumoto (2003) et Nivre (2003), le choix de la transition est réalisé par un classifieur appris de manière en général supervisée à partir d'un corpus annoté en arbres de dépendances. On distingue alors deux modes de choix de la transition à appliquer à la configuration courante pour l'utilisation du système par transitions :

— lors de la prédiction de l'arbre de dépendances d'une phrase, le choix est en général sim-

désigne une pile constituée de S plus m en tête de pile, alors que $m|B$ désigne un buffer constitué de l'élément m suivi d'une séquence (éventuellement vide) B.

29. Un arbre de dépendances est projectif, si tout nœud a une projection correspondant à une séquence continue de mots, la projection d'un nœud n étant l'ensemble des nœuds atteignables par dépendance directe ou indirecte à partir de n.

Entrées : Une phrase P segmentée en tokens ;
Un arbre de référence G ;
Une fonction oracle o
$c_0 \leftarrow F_i(P)$
$i \leftarrow 0$
$paires\_gold \leftarrow [\,]$
tant que $c_i \notin C_t$ **faire**
  $t_i \leftarrow o(c_i, G)$
  Ajouter (c_i, t_i) aux $paires\_gold$ $c_{i+1} \leftarrow t_i(c_i)$
  $i \leftarrow i + 1$
fin
retourner $paires\_gold$

FIGURE 4.4 – L'algorithme de génération des données d'entraînement, qui transforme une phrase d'entraînement en une séquence de paires (configuration, transition).

plement la transition de meilleur score (l'algorithme est dit « glouton » dans ce cas) ; une recherche en faisceau peut éventuellement conserver plusieurs séquences de transitions ;

— lors de l'apprentissage du classifieur, on applique la transition fournie par un « oracle », qui utilise l'arbre de dépendances de référence (« gold ») pour la phrase.

Oracle L'apprentissage supervisé du classifieur de transitions suppose de disposer de couples associant une configuration à la transition « gold » à appliquer à cette configuration de manière à arriver en fin de processus à l'arbre de dépendances gold. Pour chaque phrase et arbre de dépendances ($\mathbf{P}$, G) du corpus arboré d'apprentissage, l'oracle statique fournit de manière déterministe une séquence unique de $\mathbf{r}$ paires $(\mathbf{c_i}, \mathbf{t_i})$ avec $\mathbf{c_0} = \mathbf{F_i}(\mathbf{P})$ et telle que la configuration finale $\mathbf{t_r}(\mathbf{c_r})$ donne l'arbre de dépendances gold $\mathbf{G}$.

Formellement un oracle statique est une fonction prenant une configuration et un arbre de dépendances gold en entrée, et renvoyant une transition, $\mathbf{o}(\mathbf{c_i}, \mathbf{G}) = \mathbf{t_i}$. L'algorithme de génération des données d'entraînement transformant une phrase et un arbre gold en une séquence de paires gold (configuration, transition) est fourni à la figure 4.4. En pratique, pour un jeu de transitions donné, on peut définir divers oracles au moyen d'un ordre de priorité sur les transitions applicables à une configuration donnée, en contraignant qu'un arc ajouté par une transition soit effectivement un arc présent dans l'arbre gold $\mathbf{G}$. Par exemple pour le jeu Arc-standard, un oracle naturel définit $\mathbf{o}(\mathbf{c})$ avec les heuristiques suivantes :

— LEFT ARC est à appliquer dès lors que l'arc qu'elle ajouterait fait partie de l'arbre gold.

— RIGHT ARC est à appliquer dès lors que l'arc $(\mathbf{h}, \mathbf{l}, \mathbf{d})$ qu'elle ajouterait fait partie de l'arbre gold, et qu'en outre le dépendant $\mathbf{d}$ a déjà tous ses nœuds fils.

— À défaut, appliquer SHIFT.

L'oracle que nous venons de présenter est dit « statique » (Goldberg et Nivre, 2013), c'est-à-dire d'une part déterministe (il est indépendant des prédictions faites par le classifieur en cours d'apprentissage) et également « incomplet » car il n'est pas forcément défini pour n'importe quelle configuration, mais seulement pour celles obtenues en suivant toujours les transitions prédites par l'oracle justement. Or en phase de prédiction, le classifieur recevra en entrée des configurations potentiellement erronées du fait d'erreurs précédentes, situation à laquelle il n'aura pas

Input :
a tokenized sentence P,
a score_transitions function $c \leftarrow F_i(P)$
tant que $c \notin C_t$ **faire**
  $\hat{t} \leftarrow argmax_t score\_transitions(c, t)$
  $c \leftarrow \hat{t}(c)$
fin
retourner $arcs(c)$

FIGURE 4.5 – L'algorithme glouton d'analyse par transitions

été confrontées en phase d'apprentissage. Goldberg et Nivre (2013) proposent des oracles dits « dynamiques » permettant d'une part d'adapter le choix en fonction des prédictions courantes du classifieur, et d'autre part d'explorer les chemins alternatifs et non optimaux pendant l'entraînement de manière dynamique, ce qui rend les conditions d'entraînement plus similaires à celles de la phase de prédiction et permet d'améliorer la précision de l'analyse.

Extraction de traits et entraînement du classifieur De manière classique, l'apprentissage du classifieur devant prédire une transition à partir d'une configuration se fait en définissant en amont une représentation vectorielle pour une configuration. Avant la généralisation de l'utilisation de réseaux de neurones, la représentation vectorielle d'une configuration utilisait des traits de type vecteurs creux, concaténant des informations contextuelles riches concernant les deux ou trois premiers éléments du tampon et de la pile, y compris des informations sur la structure syntaxique partiellement déjà construite. Par exemple Zhang et Nivre (2011a) utilisent des traits comme « un vecteur one-hot pour l'étiquette morphosyntaxique du dépendant le plus à gauche de S_0 » (avec S_0 le premier élément de la pile).

Les premiers systèmes neuronaux pour analyseur à transitions utilisaient quant à eux des représentations denses pour les attributs linguistiques de ces mêmes types d'éléments (comme par exemple dans Chen et Manning (2014)). Nous renvoyons à la section 4.3 pour l'architecture et les traits utilisés par les réseaux neuronaux pour les systèmes par transitions.

Algorithme glouton d'analyse En supposant disposer d'un classifieur donnant, pour une configuration donnée, un score à toutes les transitions applicables, l'algorithme le plus simple pour effectivement prédire un arbre de dépendances est l'algorithme glouton, qui choisit et applique à chaque configuration la transition de plus haut score, jusqu'à atteindre une configuration terminale (voir 4.5). Cet algorithme simple est linéaire en la taille de la phrase, mais souffre bien sûr de propagation d'erreurs, le choix d'une transition étant très local à la configuration courante. Une recherche par faisceau peut conserver à chaque étape **n** séquences de transitions et choisir au final la meilleure parmi les **n** analyses complètes. Une optimisation globale sur la séquence de transitions est en général computationnellement impraticable, mais des propositions pour utiliser un espace plus grand de recherche des séquences de transitions ont été proposés (Huang et Sagae, 2010; Andor et al., 2016).

4.2 Systèmes par transitions pour l'analyse syntaxique et l'identification d'EPs

Des modifications sur le type de systèmes présenté section 4.1 permettent de produire des sorties plus riches, et notamment coupler l'analyse syntaxique avec d'autres tâches. Un couplage possible est l'analyse en dépendances et l'étiquetage morphosyntaxique, deux tâches hautement inter-dépendantes. Bohnet et Nivre (2012) construisent un système par transitions qui réalise l'étiquetage morphosyntaxique et l'analyse syntaxique en dépendances de façon conjointe, en ajoutant la fonction d'étiquetage morphosyntaxique sur la transition SHIFT (s'inspirant de Nivre (2009)). Cette architecture permet d'améliorer la performance des deux tâches pour certaines langues morphologiquement riches.

En s'inspirant de la méthode précédente, Constant et Nivre (2016) proposent un système par transitions pour réaliser conjointement l'analyse syntaxique en dépendances et l'identification des EPs. Le système utilise un tampon, une pile pour l'analyse syntaxique comme vu à la section 4.1, et une pile supplémentaire, dite « pile lexicale » pour l'identification des unités lexicales, simples ou polylexicales. Ce système peut être considéré comme une extension du système arc-standard (voir section 4.1.2), intégrant des transitions supplémentaires pour l'identification d'EPs. Le jeu de transitions comprend en effet les trois transitions d'arc-standard, modulo le fait que Shift dépose le premier élément du tampon non seulement sur la pile syntaxique mais également sur la pile lexicale. Des transitions spécifiques à la pile lexicale permettent de grouper certains tokens (pas forcément contigus) pour identifier des EPs. Il s'agit de trois transitions supplémentaires : SYNTACTIC MERGE, LEXICAL MERGE et COMPLETE. Alors que le SYNTACTIC MERGE fusionne deux tokens sur la pile syntaxique et sur la pile lexicale, ce qui est utilisé pour les EPs syntaxiquement irrégulières, la transition LEXICAL MERGE n'opère une fusion que sur la pile lexicale, ce qui permet de dissocier irrégularité sémantique des EPs et régularité syntaxique de certaines d'entre elles. La transition COMPLETE déclare une unité lexicale comme complète et l'enregistre dans la liste des unités lexicales en cours de construction.

L'utilisation de plusieurs piles est inspirée de l'analyseur en dépendances multiplanaire de Gómez-Rodríguez et Nivre (2013) et la stratégie d'analyse syntaxique pour les EPs (hiérarchiques) est très similaire à la méthode d'analyse déterministe en constituants de Crabbé (2014). Les expérimentations menées sur deux jeux de données différents montrent que l'approche améliore l'identification des EPs de manière significative par rapport aux approches conjointes existantes et qu'elle arrive parfois à améliorer l'analyse syntaxique.

Le système par transitions que nous avons utilisé dans notre travail est largement inspiré de celui de Constant et Nivre (2016), dont nous avons repris et modifié les transitions concernant la pile lexicale.

4.3 Réseaux de neurones pour les systèmes par transitions

Les réseaux de neurones sont massivement utilisés en TAL depuis le début des années 2010. Un aspect clé de ces architectures non linéaires est la possibilité de représenter les symboles atomiques (les mots, les POS, les étiquettes de dépendances...) par des vecteurs denses. Cela permet en particulier d'intégrer des vecteurs de mots appris sur gros corpus bruts, qui capturent une similarité lexicale distributionnelle. Les mots représentés ne sont pas des symboles totalement distincts mais sont positionnés dans un espace vectoriel, et les systèmes peuvent utiliser la similarité plus ou moins grande entre mots distincts.

Un autre avantage des réseaux de neurones est que leur non linéarité permet de capturer des combinaisons de traits d'entrée, qui peuvent du coup être relativement simples, comparativement aux traits dans des architectures linéaires.

Nous présentons ci-dessous un bref historique de l'utilisation des réseaux de neurones pour l'analyse syntaxique par transitions. Nous ne pouvons pas viser l'exhaustivité étant donné le nombre de travaux, mais citons des travaux emblématiques de ce courant.

4.3.1 Réseaux de neurones à propagation avant

Afin de réduire l'effort requis pour la construction des systèmes par transitions pour l'analyse syntaxique en dépendances, Chen et Manning (2014) utilisent des traits simples et indépendants de la langue en entrée d'un réseau de neurones, à la place des traits sophistiqués et incluant des combinaisons d'attributs linguistiques utilisés dans des classifieurs linéaires (comme par exemple Zhang et Nivre (2011b)).

Le réseau neuronal utilisé est un modèle Perceptron Multicouche (MLP) (« multilayer perceptron »), composé d'une entrée, d'une couche cachée avec une fonction d'activation cubique et d'une couche de sortie softmax.

Dans ce modèle, les traits fournis en entrée sont en effet une simple concaténation de vecteurs creux (« one-hot ») pour quelques tokens spécifiques de la configuration courante (par exemple « le premier et le deuxième élément du tampon », « le dépendant le plus à gauche du premier élément de la pile », etc.), que nous appellerons les "tokens ciblés"[30]. Pour chaque token ciblé, l'entrée concatène un vecteur one-hot pour la forme fléchie, l'étiquette morphosyntaxique et parfois l'étiquette de dépendances déjà prédites pour eux.
Notez que les paramètres entre la couche d'entrée et la couche cachée correspondent à des vecteurs denses représentant les symboles atomiques (forme fléchie, étiquette morphosyntaxique, étiquette de dépendance), ce que l'on appelle « plongement » en français (« embedding » en anglais). Enfin, une couche softmax permet de fournir en sortie une distribution de probabilités sur les transitions pour une configuration donnée, le tout étant utilisé dans un système arc-standard (voir section 4.1.2).

Ce modèle entraîne une amélioration significative des performances, pour les treebanks usuels pour l'anglais et le chinois.

Weiss et al. (2015) reprennent une version légèrement améliorée du réseau de neurones de Chen et Manning (2014) (avec notamment une couche cachée supplémentaire), mais ajoutent une étape d'apprentissage par perceptron structuré. Lors de l'apprentissage, une première étape est l'entraînement d'un classifieur local pour prédire une transition à partir d'une configuration. Mais au lieu d'utiliser directement ces prédictions, les auteurs utilisent les représentations denses cachées obtenues à partir d'une configuration d'entrée. Ils entraînent ensuite un perceptron structuré avec mise à jour précoce (« early update » en anglais) introduit par Collins et Roark (2004) : pour une phrase donnée, une prédiction en faisceau est faite pas à pas (i.e. transition par transition), en conservant **n** meilleures séquences de transitions à chaque pas, d'après l'état courant des paramètres à apprendre. La mise à jour des paramètres intervient dès que l'arbre gold n'est atteignable par plus aucune des séquences conservées dans le faisceau. Lors de la phase de prédiction, une recherche en faisceau est utilisée. Ceci couplé avec l'utilisation d'auto-apprentissage

30. Plus précisément, les tokens ciblés dans le modèle de Chen et Manning (2014) sont : les trois premiers tokens de la pile, les trois premiers du tampon ; les premier et deuxième enfants les plus à gauche et à droite des deux premiers tokens de la pile ; et les enfants de ceux-ci les plus à gauche et à droite.

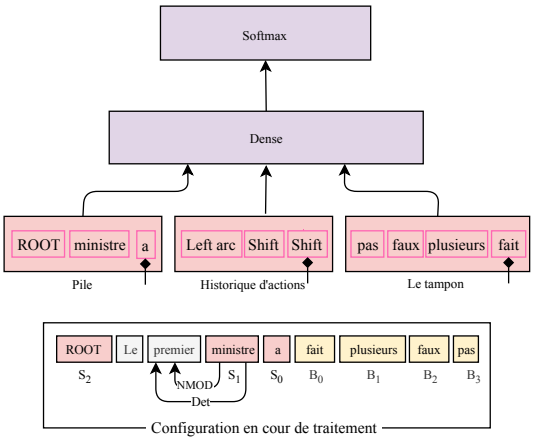

FIGURE 4.6 – **Réseau de neurones de Dyer et al. (2016) :** Une figure de Dyer et al. (2016), adaptée au français par nous, montrant l'un des états de leur système d'analyse syntaxique en dépendances pour la phrase *Le* $\boldsymbol{premier}_1$ $\boldsymbol{ministre}_1$ *a* $\boldsymbol{fait}_2$ *plusieurs* $\boldsymbol{faux}_{2,3}$ $\boldsymbol{pas}_{2,3}$. La couche dense $\mathbf{P_t}$ produit une représentation continue des trois stack-LSTM, et la transmet à une couche softmax pour donner la distribution des décisions possibles.

(« self-training ») permet d'obtenir les meilleurs résultats de l'époque sur l'anglais.

Andor et al. (2016) montrent que de simples réseaux de neurones à propagation avant sans récurrence peuvent atteindre des précisions comparables ou meilleures que les réseaux de neurones exploitant des couches récurrentes comme LSTM et GRU, à condition qu'ils soient globalement normalisés (voir figure 4.6). Alors que leur réseau est utilisé par un système par transitions et alimenté par les mêmes représentations vectorielles que celles de Chen et Manning (2014), ils effectuent une recherche en faisceau pour maintenir plusieurs hypothèses et introduisent une normalisation globale avec un objectif de type CRF. La motivation sous-jacente est d'éviter le problème dit de biais d'étiquetage (« label bias ») des modèles locaux (les modèles normalisés localement, même avec un décodage avec recherche en faisceau, ont souvent une très faible capacité à réviser les décisions antérieures (Andor et al., 2016)). Et en effet, ce modèle permet d'obtenir l'état de l'art, sur plusieurs jeux de données multilingues, sur les tâches d'étiquetage morphosyntaxique, d'analyse en dépendances et de compression de phrases. Il dépasse les modèles récurrents (section suivante) tout en étant nettement plus rapide.

4.3.2 Réseaux de neurones récurrents

Dans cette architecture que nous adapterons pour l'une des variantes d'identification des EPs via une analyse par transitions, Kiperwasser et Goldberg (2016) reprennent l'idée de cibler certains éléments de la configuration, mais pour représenter ces tokens ciblés, ils utilisent pour chacun le vecteur obtenu par une couche récurrente de type Bi-LSTM. Plus précisément, pour une phrase du corpus d'entraînement, les mots passent d'abord dans une couche de plongements lexicaux (non contextuels), puis un Bi-LSTM est appliqué, permettant d'obtenir un vecteur de chaque mot en contexte cette fois. Ensuite, pour chaque configuration gold de la phrase, l'entrée est la concaténation des vecteurs Bi-LSTM des tokens ciblés de la configuration (voir figure 4.7).

Ils expérimentent leur modèle sur deux analyseurs syntaxiques : un analyseur par transitions, avec une variante d'arc-standard appelée arc-hybrid, ainsi qu'un système d'analyse syntaxique à

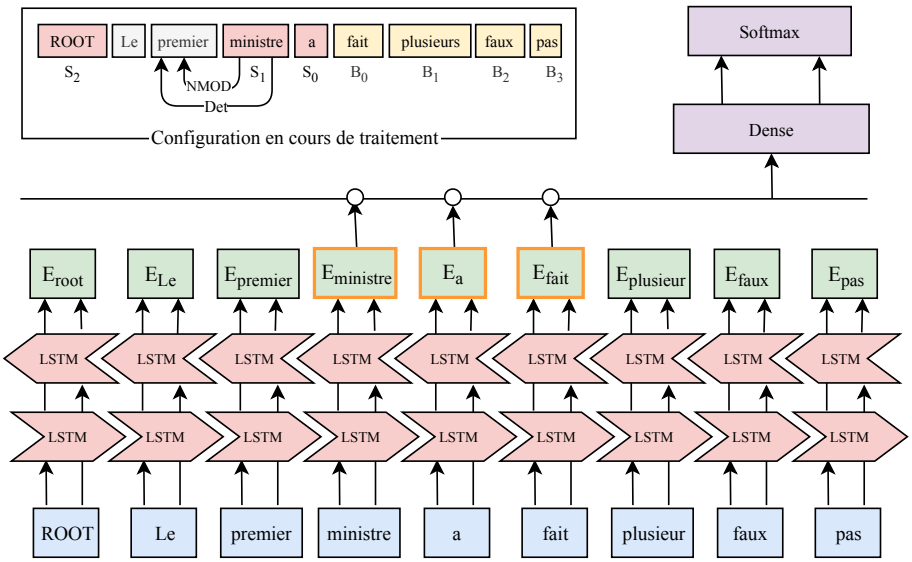

FIGURE 4.7 – **Réseau de neurones de Kiperwasser et Goldberg (2016) :** Un schéma représentant le fonctionnement de la fonction d'extraction de traits Bi-LSTM de Kiperwasser et Goldberg (2016) pour le système arc-standard. La partie supérieure illustre la configuration en cours de traitement, tandis que la structure du bas représente une instance de la propagation en avant du système. La couche dense reçoit les vecteurs d'état du module Bi-LSTM de certains token de la configuration, qui représentent les traits extraits, et la fonction softmax attribue ensuite des probabilités pour chaque transition en fonction de la représentation produite.

base de graphes[31].

Le système atteint l'état de l'art en anglais et en chinois, sans recourir à des ressources et à des techniques semi-supervisées. L'intégration des plongements lexicaux préentraînés permet d'améliorer encore la performance.

Dyer et al. (2016) dépassent l'analyseur de Chen et Manning (2014) en définissant un réseau de neurones intégrant une couche variante de LSTM, dite stack-LSTM, capable de connecter les neurones de la couche de manière dynamique en effectuant des actions d'ajout (Push) et de suppression (Pop). La dynamisation de la couche revient à l'ajout d'un « pointeur de pile ». Celui-ci détermine quel neurone du LSTM nourrit la fonction d'activation du neurone actuel. Le modèle apprend des représentations du contenu complet de la configuration, c'est-à-dire le tampon d'entrée, l'historique des actions précédentes de l'analyseur et le contenu complet de la pile de structures syntaxiques partiellement construites.

Le modèle d'analyse utilise trois stack-LSTMs : (1) une représentant le tampon l'entrée ; (2) une représentant la pile d'arbres syntaxiques partiels ; et (3) une représentant l'historique des actions d'analyse (voir figure 4.8).

Ballesteros et al. (2017) ont obtenu une amélioration en intégrant à ce modèle des plongements lexicaux à base de caractères (c'est-à-dire obtenu par combinaison de plongements des n-grammes de lettres composant le mot) pour représenter les mots de stack-LSTM, ce qui améliore la représentation des mots hors vocabulaire, en particulier pour les langues morphologiquement riches.

31. Les méthodes d'analyse en dépendances à base de graphe consistent à rechercher l'arbre le plus probable dans l'ensemble des arbres capables de représenter les relations de dépendances entre les mots d'une phrase.

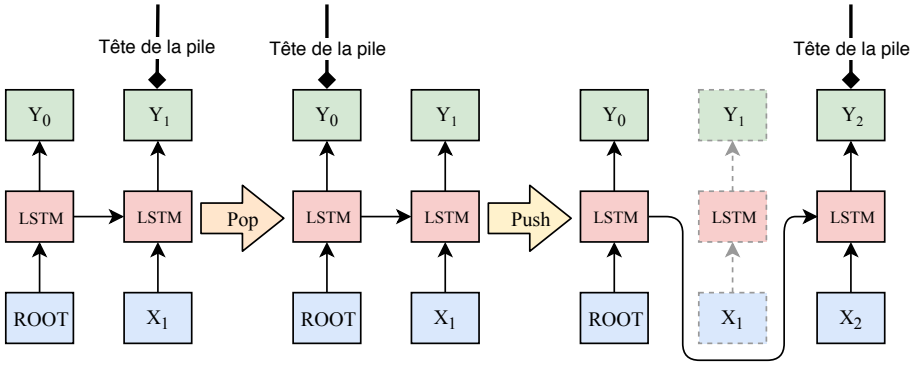

FIGURE 4.8 – **Réseau de neurones de Dyer et al. (2016) :** Une figure de Dyer et al. (2016) montrant le fonctionnement de stack-LSTM sur trois stack-LSTM. À gauche, un stack-LSTM à seul élément x_1 où l'application de l'opération *« Pop »* produit le stack-LSTM du milieu. Le *« Pop »* déplace le pointeur de la pile au vecteur y_0. L'application de *« Push »* sur le stack-LSTM du milieu connecte le token x_2 avec le vecteur d'état initial en ignorant le token x_1. Il convient de noter que les vecteurs situés dans les rangées inférieures représentent le contenu de la pile, qui représentent, à leur tour, les entrées de LSTM. Les rangées supérieures sont les sorties de LSTM, et les vecteurs du milieu sont les neurones de LSTM.

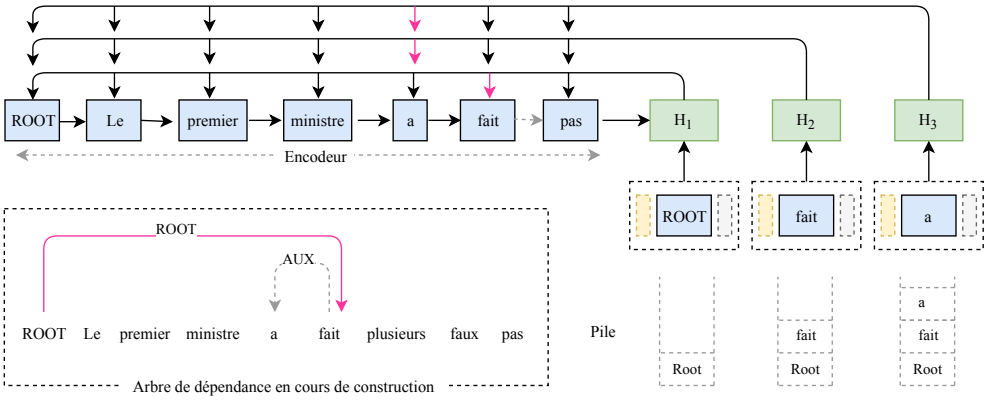

FIGURE 4.9 – **Réseau de neurones de Ma et al. (2018) :** Une figure montrant le début de l'analyse en dépendances pour la phrase *Le* ***premier***$_1$ ***ministre***$_1$ *a* ***fait***$_2$ *plusieurs* ***faux***$_{2,3}$ ***pas***$_{2,3}$. Les vecteurs dans les cases jaunes et grises indiquent respectivement le frère et le grand-parent du token en cours de traitement (tête de la pile). Source : Ma et al. (2018).

Ma et al. (2018) proposent une autre architecture récurrente, basée sur la structure encodeur-décodeur, pour l'analyse par transitions, à savoir STACK-PTR, qui atteint l'état de l'art sur de nombreux treebanks de différentes langues. L'encodeur de ce modèle lit toute la phrase à l'aide d'un module récurrent, bidirectionnel et à plusieurs couches, et le décodeur (un module RNN unidirectionnel) construit l'arbre en dépendances de haut en bas (de la racine aux feuilles). Il utilise pour cela une pile, initialisée avec un symbole spécial racine. À chaque étape, le décodeur reçoit un vecteur pour le mot h en tête de la pile (et s'ils existent, les vecteurs pour son nœud parent et son nœud frère), et utilise un mécanisme d'attention sur les vecteurs de chaque position produits par l'encodeur (voir figure 4.9). Il peut ainsi sélectionner la position recevant le plus d'attention, qui est interprétée comme étant la position d'un dépendant d pour le mot h. Le décodeur ajoute une dépendance entre h et d, et pousse d au sommet de la pile. Une astuce est utilisée pour décider quand un mot a reçu tous ses dépendants et doit être enlevé de la pile : cela est fait dès lors que le dépendant sélectionné pour h est h lui-même.

L'utilisation de l'attention sur toute la phrase permet de ne pas avoir de limite sur le contexte pris en compte, et en particulier de pouvoir utiliser le contexte droit complet.

4.4 Architectures neuronales multitâches

Nous allons dans cette section aborder les techniques de transfert et d'apprentissage multitâche, en présentant leur principe général, ainsi qu'un cas d'utilisation typique : l'apprentissage conjoint de l'analyse en dépendances et de l'étiquetage morphosyntaxique.

4.4.1 Transfert de paramètres

Le transfert est une technique d'apprentissage automatique dans laquelle, pour un réseau conçu pour une tâche **A**, on utilise comme valeurs initiales de paramètres des valeurs obtenues par apprentissage pour une tâche **B**. Cette technique ne fonctionne que si les paramètres transférés encodent des informations assez génériques. Cette technique d'optimisation est très répandue dans le domaine de TAL, puisqu'elle permet de gagner du temps d'apprentissage et d'améliorer les performances (Torrey et Shavlik, 2010). L'un des exemples les plus basiques et les plus connus en TAL de transfert de paramètres est l'utilisation de plongements lexicaux préentraînés, avec affinage ultérieur de ces plongements dans le cadre d'une autre tâche. .

4.4.2 Apprentissage multitâche

L'apprentissage multitâche consiste à partager certains paramètres entre n réseaux, entraînés chacun pour des tâches distinctes mais liées. L'intuition est que des indices utiles pour une tâche donnée peuvent s'avérer utiles pour une tâche différente mais liée. L'apprentissage multitâche est un mécanisme d'apprentissage des réseaux de neurones dont l'objectif principal est de produire une structure neuronale capable de bien généraliser pour plusieurs tâches auxiliaires (Caruana, 1997).

Le partage de paramètres peut être explicite et implicite ("hard" et "soft" en anglais (Ruder, 2017)). Le partage explicite consiste à partager une ou plusieurs couches cachées entre les tâches, tout en conservant (au moins) une couche de sortie indépendante pour chaque tâche. Dans le partage implicite de paramètres, chaque tâche a son propre modèle avec ses propres paramètres. La distance entre les paramètres des modèles est ensuite régularisée afin de trouver une convergence

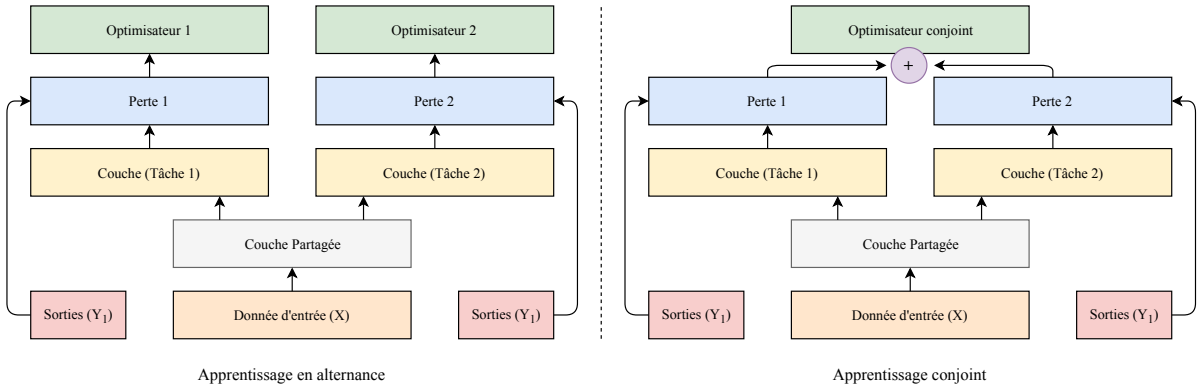

FIGURE 4.10 – **Stratégies d'apprentissage multitâche :** Un schéma des deux stratégies d'apprentissage pour les modèles multitâches. À gauche, l'apprentissage en alternance, qui consiste à calculer la perte et à optimiser le réseau pour chaque tâche séparément, en général en alternance. À droite, l'apprentissage joint, où la fonction optimisée combine les pertes des deux tâches.

(Ruder, 2017). Certaines architectures multitâches utilisent une stratégie hybride pour partager leurs paramètres.

L'apprentissage des paramètres des architectures multitâches peut s'effectuer en alternant l'optimisation entre les tâches ou bien de manière jointe. Dans l'apprentissage en alternance, la mise à jour des paramètres est faite d'après la perte des différentes tâches, mais en alternant les pertes de chaque tâche, selon un ordre donné ou stochastique. Les paramètres partagés sont mis à jour quelle que soit la tâche considérée. En revanche, dans le cas d'un apprentissage joint, une fonction objectif est définie, combinant les fonctions objectif des différentes tâches. (voir figure 4.10).

4.4.3 Apprentissage multitâche pour l'analyse en dépendances et l'étiquetage morphosyntaxique

Zhang et Weiss (2016) proposent une architecture multitâche pour l'étiquetage morphosyntaxique et l'analyse syntaxique en dépendances, partant du constat qu'une part significative des erreurs d'étiquetage morphosyntaxique n'en serait pas si le contexte syntaxique était mieux pris en compte.

L'architecture de Zhang et Nivre (2011b) est composée d'un module pour l'étiquetage morphosyntaxique et d'un autre autre pour l'analyse en dépendances. Le premier module est un simple réseau neuronal de type perceptron multicouche (MLP), qui prend en entrée des vecteurs one-hot pour le token cible et les tokens dans une fenêtre autour du token cible, ainsi que des vecteurs one-hot pour des caractéristiques de ces tokens (des suffixes et préfixes). Ces vecteurs one-hot sont d'abord transformés par une couche de plongements, puis une couche dense et enfin une couche de sortie softmax.

Le module pour l'analyse syntaxique est du même type que celui de Chen et Manning (2014) (voir section 4.3.1). L'aspect clé de la proposition de Zhang et Weiss (2016) est que ce deuxième module, au lieu de prendre pour les éléments ciblés de la configuration une concaténation de vecteurs one-hot, va concaténer pour chaque élément cible la couche dense obtenue pour lui par le module d'étiquetage morphosyntaxique.

Notez que le chapitre 5 de la seconde partie de la thèse présente une architecture multitâche,

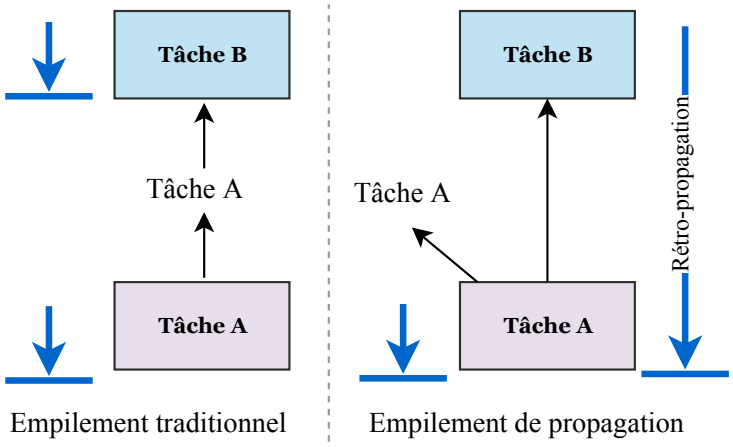

FIGURE 4.11 – **Rétropropagation en pile de Zhang et Weiss (2016) :** Illustration de la technique de « stack-propagation » (rétropropagation en pile) de Zhang et Weiss (2016) : à gauche l'empilement traditionnel où le réseau de la tâche **B** utilise des paramètres du réseau de la tâche **A**, mais où la rétropropagation s'arrête aux paramètres de la tâche **B**. À droite, pour la tâche **B**, la rétropropagation va jusqu'aux paramètres du réseau de la tâche **A**. Source : Zhang et Weiss (2016)

intégrant l'identification des EPs par transitions à cette architecture.

Les auteurs qualifient leur modèle de « stack-propagation » (rétropropagation en pile) car à l'apprentissage, la rétropropagation pour l'objectif d'analyse en dépendances va jusqu'au module d'étiquetage morphosyntaxique : l'empilement de la (rétro-)propagation permet au module d'analyse en dépendances de modifier les paramètres du module d'étiquetage morphosyntaxique en fonction du résultat de l'objectif de l'analyse en dépendances.

Taslimipoor et al. (2019) proposent l'une des premières architectures neuronales multitâches pour la prédiction des étiquettes IOB des EPs d'une part, et des arêtes et des étiquettes de dépendances syntaxiques d'autre part. Leur architecture est centrée sur la tâche d'identification des EPs et considère les deux autres tâches comme des tâches auxiliaires pouvant aider. À l'apprentissage, le modèle calcule les valeurs de perte pour les deux tâches auxiliaires et les ajoute à la perte de la tâche principale.

Le modèle, inspiré du travail de Taslimipoor et Rohanian (2018), reçoit en entrée des plongements lexicaux contextuels préentraînés et se sert de modules convolutionnels (CNN), sans pooling, comme fonction d'extraction de traits (voir 3.2.2). Le modèle comprend d'autres couches Bi-LSTM et denses (voir figure 4.13). La première couche Bi-LSTM est partagée entre les trois tâches et sert à alimenter **(i)** les couches denses des deux tâches auxiliaires, ainsi que **(ii)** une seconde couche Bi-LSTM pour la tâche principale de prédiction des étiquettes d'EPs. Cette architecture, testée sur les données anglaises des jeux de données PARSEME 1.1 seulement, produit un score plus performant que celui de l'état de l'art de l'open track de PARSEME 1.1.

Taslimipoor et al. (2019) explorent également la technique de transfert de paramètres en apprenant le même modèle, sans tâche auxiliaire, sur un grand jeu de données, le jeu de données PARSEME 1.1 de l'allemand, et en transférant les paramètres appris vers une autre langue, l'anglais, dont les données PARSEME 1.1 sont de taille modeste. Cette expérience, bien que peu élaborée en termes de recherche des hyperparamètres, se révèle prometteuse et le score produit dépasse celui de l'open track de PARSEME 1.1.

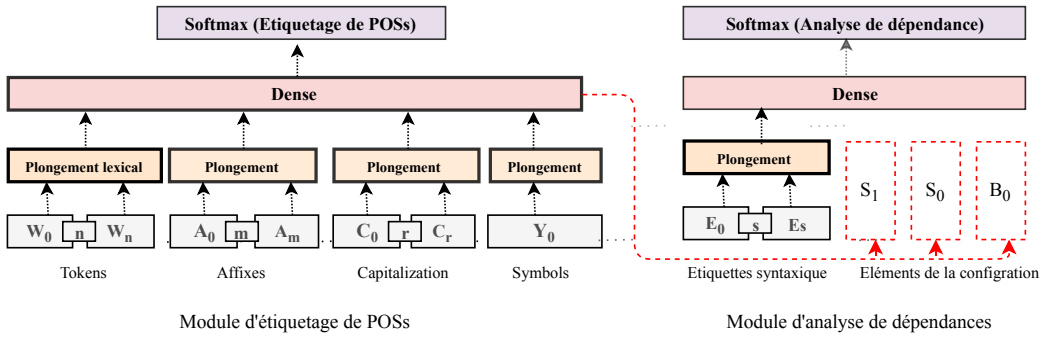

FIGURE 4.12 – **Réseau de neurones de Zhang et Weiss (2016) :** Le modèle de rétropropagation en pile (« stack-propagation ») de Zhang et Weiss (2016). À gauche, le module d'étiquetage morphosyntaxique qui reçoit en entrée le token cible et une fenêtre de tokens autour du lui, ainsi que des informations de suffixes et préfixes et de capitalisation pour une fenêtre plus réduite. « Symbols » désigne des booléens concernant la présence de caractères spéciaux dans ces tokens. À droite, le module d'analyse en dépendances utilise des plongements spécifiques à ce module $(E_0, ..., E_s)$ et pour les tokens S_1, S_0, B_0, le vecteur obtenu sur la couche dense du module de gauche.

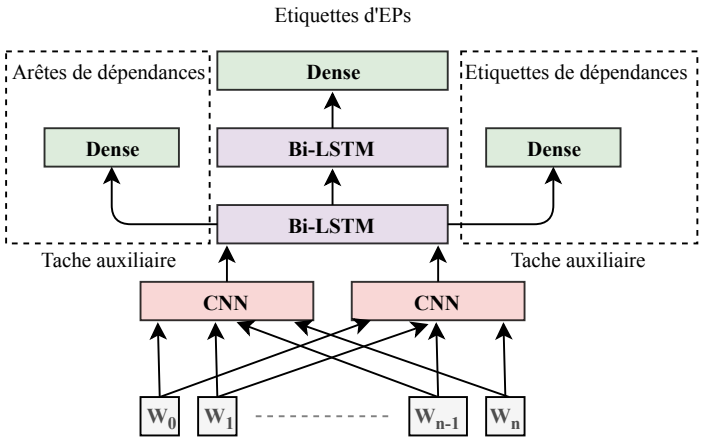

FIGURE 4.13 – **Réseau de neurones de Taslimipoor et al. (2019) :** L'architecture neuronale multitâche de Taslimipoor et al. (2019). Source : Taslimipoor et al. (2019).

4.5 Résumé

Dans ce chapitre, nous avons présenté différents travaux dont nous nous sommes inspirés pour notre système d'identification d'EPs. Nous avons commencé par définir formellement un système d'analyse dit "par transitions", abondamment utilisé en analyse syntaxique en dépendances depuis les travaux de (Yamada et Matsumoto, 2003; Nivre, 2004b), caractérisés par l'utilisation d'un tampon de tokens non encore vus, une pile d'éléments en cours de traitements, et une structure en cours de construction. Nous avons ensuite détaillé la proposition de Constant et Nivre (2016) qui ajoute à ce type de système une pile et une structure de sortie supplémentaire, de manière à réaliser conjointement l'analyse syntaxique automatique et l'identification d'EPs.

Nous sommes ensuite passé à la description d'architectures neuronales aujourd'hui couramment utilisés en TAL, en particulier les réseaux de neurones à propagation avant, et les réseaux récurrents de type GRU ou LSTM, et avons cité différents travaux les utilisant pour l'analyse en dépendances, comme (Chen et Manning, 2014) qui proposèrent pour la première fois d'utiliser un réseau de neurones pour représenter la configuration de l'analyseur au sein du classifieur devant prédire la transition à appliquer à cette configuration. Le système de Kiperwasser et Goldberg (2016) a été également détaillé dans ce chapitre, système que nous avons essayé d'adapter à l'identification d'EPs. Il utilise la représentation récurrente de tokens ciblés de la configuration courante de l'analyseur, et obtient des résultats à l'époque état de l'art en analyse en dépendance. D'autres systèmes ont été présentés, citons en particulier l'architecture de type encodeur-décodeur de (Ma et al., 2018), qui utilise un mécanisme d'attention sur les vecteurs de tous les mots de la phrase, obtenus par l'encodeur.

Enfin, nous avons présenté la technique de l'apprentissage multi-tâche, que nous avons également utilisée dans notre travail, en particulier le travail de Zhang et Weiss (2016), où les tâches considérées pour le multi-tâche sont l'étiquetage morpho-syntaxique et l'analyse syntaxique en dépendances. Le système de Taslimipoor et al. (2019) utilise l'apprentissage multi-tâche pour réaliser l'identification d'EPs, avec une architecture proche de celle du système SHOMA, et l'analyse en dépendances. Taslimipoor et al. (2019) ont expérimenté ce système sur les données anglaises de PARSEME 1.1 et montrent un gain significatif de performance pour l'identification des EPs verbales en utilisant une architecture multi-tâche (comparé à une architecture mono-tâche).

Deuxième partie

Méthode d'analyse automatique par transitions pour l'identification des EPs

Chapitre 1

Description du système

Issus de l'informatique théorique, les systèmes par transitions sont utilisés pour des problèmes de prédiction de structures. Ils décomposent ces problèmes en des séquences de prédictions locales comme le montre le chapitre 4, Partie 1, où nous avons présenté les notions et concepts principaux de ces systèmes. Ils sont couramment utilisés pour l'analyse syntaxique en dépendances, mais peuvent être utilisés pour d'autres tâches du TAL comme l'étiquetage morphosyntaxique ou la compression de phrases (Bohnet et Nivre, 2012; Andor et al., 2016).

Dans cette thèse, nous proposons un système par transitions dédié à l'identification des EPs. La structure de notre système est inspirée par le système de Constant et Nivre (2016), qui lui-même s'appuie sur le célèbre analyseur en dépendances syntaxiques proposé par (Nivre, 2004b). Constant et Nivre (2016) ont proposé un algorithme d'analyse syntaxique qui prédit de manière conjointe un arbre de dépendances syntaxiques et une forêt d'unités lexicales intégrant les EPs. Ce système intègre, en particulier, un nouveau mécanisme par transitions d'analyse lexicale permettant d'identifier les EPs.

Dans notre travail, nous nous concentrons uniquement sur l'aspect lexical, et nous proposons un système d'identification générique, indépendant des informations syntaxiques. Ainsi, nous modifions la structure du système d'origine en abandonnant la prédiction de la syntaxe, en proposant un nouvel ensemble de transitions et de nouveaux modèles de prédiction de transitions. Dans la suite de ce chapitre, nous donnons une définition formelle de notre système par transitions (section 1.1), trois ensembles de transitions pour le système (section 1.2), la puissance expressive du système pour les trois ensembles de transitions (section 1.3), la définition et le fonctionnement de l'oracle qui anime ce système (section 1.4), ainsi que l'algorithme qu'il utilise pour transformer les phrases d'un corpus annoté de référence (gold) en un jeu d'entraînement adapté à l'apprentissage des modèles (section 1.5).

1.1 Définition du système

Un système par transitions consiste à appliquer une séquence d'actions (à savoir « des transitions ») pour construire progressivement la structure de sortie attendue de manière ascendante. Chaque transition est prédite par un oracle ou un classifieur, à partir de l'état courant de l'analyseur (à savoir « la configuration »). Formellement, notre système par transitions est un quadruplet $\mathbf{S} = (\mathcal{C}, \mathcal{T}, \mathcal{F}_{\mathbf{i}}, \mathcal{C}_{\mathbf{t}})$ où :

1. $\mathbf{C}$ est un ensemble de configurations. Une configuration de notre système consiste en un triplet $\mathbf{c} = (\mathbf{S}, \mathbf{B}, \mathbf{E})$, où $\mathbf{S}$ est une pile contenant les unités en cours de traitement, $\mathbf{B}$ est un

tampon contenant les tokens d'entrée restants et **E** est un ensemble d'EPs déjà identifiées. Une unité apparaissant dans la pile **S** peut être constituée d'un seul token de la phrase d'entrée, ou bien plusieurs, auquel cas, l'unité a une structure binaire (il s'agit par construction d'une paire d'unités, elles-mêmes mono ou multitoken). Une unité de la pile peut aussi bien correspondre à une EP complète, un début d'EP en cours d'identification ou un token non membre d'une EP.

2. $\mathcal{T}$ est un ensemble des transitions, chaque transition étant une fonction $\mathbf{t} : \mathcal{C} \Rightarrow \mathcal{C}$ permettant de passer d'une configuration à une autre, suite à son application (voir section 1.2, page 80).

3. $\mathcal{F}_\mathbf{i}$ est une fonction d'initialisation qui génère la configuration initiale $\mathbf{c_s}$ de la phrase, qui dans notre système place tous les tokens dans le buffer : pour une phrase $\mathbf{s} = (\mathbf{m_1}, .., \mathbf{m_n})$ de **n** tokens, la configuration initiale est $\mathcal{F}_\mathbf{i}(\mathbf{s}) = \mathbf{c_s} = (\mathbf{S} = [\,], \mathbf{B} = [\mathbf{m_1}, ..., \mathbf{m_n}], \mathbf{E} = [\,])$;

4. $\mathcal{C}_\mathbf{t} \subset \mathcal{C}$ est l'ensemble des configurations terminales contenant toute configuration avec une pile vide et un tampon vide, i.e. de la forme $\mathbf{c_t} = (\mathbf{S} = [\,], \mathbf{B} = [\,], \mathbf{E})$.

1.2 Ensembles de transitions

Nous présentons maintenant les trois ensembles de transitions ($\mathcal{T}_\mathbf{0}$, $\mathcal{T}_\mathbf{1}$ et $\mathcal{T}_\mathbf{2}$) que nous avons mis au point et expérimentés pour notre système d'identification. Nous commençons par l'ensemble $\mathcal{T}_\mathbf{2}$ qui représente l'ensemble le plus expressif des trois. Nous décrirons ensuite les deux autres ensembles $\mathcal{T}_\mathbf{0}$ et $\mathcal{T}_\mathbf{1}$, mis au point lors d'étapes précédentes [32].

Transition	$\text{Configuration}_{avant}$		$\text{Configuration}_{apres}$	Précondition
SHIFT	$(S, i\|B, E)$	$\Rightarrow$	$(S\|i, B, E)$	$B \neq \emptyset$
REDUCE	$(S\|i, B, E)$	$\Rightarrow$	(S, B, E)	$S \neq \emptyset$
MERGE	$(S\|i, j, B, E)$	$\Rightarrow$	$(S\|(i, j), B, E)$	$\|S\| > 1$
MARK	$(S\|i, B, E)$	$\Rightarrow$	$(S\|i, B, E \cup \{i\})$	$S \neq \emptyset$ et $i \notin E$

FIGURE 1.1 – **Ensemble de transitions $\mathcal{T}_\mathbf{2}$:** Cet ensemble est utilisé par une des variantes de notre système d'identification des EPs. Pour chaque transition, on fournit la configuration avant application, après application, et les préconditions pour que la transition soit applicable.

1.2.1 Ensemble de transitions $\mathcal{T}_\mathbf{2}$

L'ensemble $\mathcal{T}_\mathbf{2}$ comprend les transitions suivantes :

1. **SHIFT** : déplace le premier élément du tampon vers le sommet de la pile ;
   — **Précondition** : le tampon n'est pas vide.
2. **REDUCE** : supprime l'élément au sommet de la pile ;

32. Nous utilisons une notation avec un | pour désigner une structure au sein de la pile ou du tampon : **S**|**i** désigne une pile constituée de **S** et d'un élément **i** en tête de pile, alors que **i**|**B** désigne un buffer constitué de l'élément **i** suivi d'une séquence (éventuellement vide) **B**.

— **Précondition** : la pile n'est pas vide.

3. **Merge** : combine les deux éléments au sommet de la pile en un seul élément. L'élément résultant correspond donc au nœud racine d'un arbre binaire. Les attributs linguistiques de cet élément résultant (tels que forme fléchie, lemme, étiquette morphosyntaxique...) sont obtenus par concaténation des attributs des deux éléments combinés, dans l'ordre où ils apparaissaient dans la pile, i.e. forcément l'ordre linéaire de la phrase. Par exemple, si les composants *avaient* et *besoin* dont les lemmes sont *avoir* et *besoin* sont combinés via **Merge**, l'unité résultante aura pour lemme *avoir_besoin*, et pour étiquette morphosyntaxique *VERB_NOUN*.

   — **Précondition** : la pile contient au moins deux éléments.

4. **Mark** : ajoute à **E** l'unité (EP) **i** du sommet de la pile.

   — **Précondition** : la pile n'est pas vide et **i** n'a pas déjà été ajoutée à **E**.

Les transitions Merge et Mark sont responsables de l'identification d'EPs. À noter que l'unité ajoutée à l'ensemble **E** par la transition Mark reste au sommet de la pile. Elle pourra ainsi entrer dans une autre EP via Merge et Mark, ou pas (via un Reduce).

1.2.2 Ensemble de transitions $\mathcal{T}_0$

Transition	Config.$_{avant}$		Config.$_{apres}$	Précondition
Shift	$(S, i\|B, E)$	$\Rightarrow$	$(S\|i, B, E)$	$B \neq \emptyset$
Reduce	$(S\|i, B, E)$	$\Rightarrow$	(S, B, E)	$S \neq \emptyset$ et i est un token
Merge	$(S\|i, j, B, E)$	$\Rightarrow$	$(S\|(i, j), B, E)$	$\|S\| > 1$
Mark_Reduce	$(S\|i, B, E)$	$\Rightarrow$	$(S, B, E \cup \{i\})$	$S \neq \emptyset$ et i est une unité multitoken
MarkTP_Reduce	$(S\|i, B, E)$	$\Rightarrow$	$(S, B, E \cup \{i\})$	$S \neq \emptyset$ et i est un seul token

Figure 1.2 – **Ensemble de transitions $\mathcal{T}_0$**, utilisé par une des variantes de notre système d'identification des EPs. Pour chaque transition, on fournit la configuration avant application, après application, et les préconditions pour que la transition soit applicable.

Dans l'ensemble $\mathcal{T}_0$ fourni à la figure 1.2, les transitions Shift et Merge sont identiques à celles de $\mathcal{T}_2$, mais on a deux différences. D'une part la transition Mark de $\mathcal{T}_2$ est divisée en deux transitions Mark_Reduce et MarkTP_Reduce[33], selon que l'unité au sommet de la pile est multitoken (cf. la précondition « **i** est une unité multitoken ») ou constituée d'un seul token (un Token Polylexical (TP)). D'autre part, ces deux transitions éliminent de la pile l'unité ajoutée à **E**, contrairement à Mark dans $\mathcal{T}_2$. Étant donné que ces deux transitions s'occupent de l'élimination des unités multitokens et des TPs de la pile, la transition Reduce se limite à la suppression, au sommet de la pile, des tokens n'appartenant pas aux EPs.

Cette différence empêche les systèmes basés sur $\mathcal{T}_0$ d'identifier les cas d'enchâssement, puisque dès qu'une EP est identifiée elle est éliminée et ne peut plus entrer dans la composition d'une autre EP.

Transition	Configuration$_{avant}$		Configuration$_{apres}$	Précondition
SHIFT	$(S, i\|B, E)$	$\Rightarrow$	$(S\|i, B, E)$	$B \neq \emptyset$
REDUCE	$(S\|i, B, E)$	$\Rightarrow$	(S, B, E)	$S \neq \emptyset$
MERGE	$(S\|i, j, B, E)$	$\Rightarrow$	$(S\|(i, j), B, E)$	$\|S\| > 1$
MERGE_MARK	$(S\|i, j, B, E)$	$\Rightarrow$	$(S\|(i, j), B, E \cup \{(i, j)\})$	$\|S\| > 1$
MARKTP	$(S\|i, B, E)$	$\Rightarrow$	$(S\|i, B, E \cup \{i\})$	$S \neq \emptyset$ i est un token

FIGURE 1.3 – **Ensemble de transitions $\mathcal{T}_1$**, utilisé par une des variantes de notre système d'identification des EPs. Pour chaque transition, on fournit la configuration avant application, après application, et les préconditions pour que la transition soit applicable.

1.2.3 Ensemble de transitions $\mathcal{T}_1$

Dans le jeu de transition $\mathcal{T}_1$ (figure 1.3), là encore SHIFT, REDUCE et MERGE sont formellement identiques à celles de $\mathcal{T}_2$. Comme pour $\mathcal{T}_0$, on sépare le traitement des EPs mono- et multitokens : les mono-tokens (les TPs) sont gérées par MARKTP, qui conserve en sommet de pile l'élément TP marqué comme EP, permettant son enchâssement dans une autre EP. Le marquage des EPs multitokens est fait avec MERGE_MARK, qui fait à la fois un merge et un marquage. Ainsi la transition MERGE est de fait réservée à un merge d'une partie d'EP, et pas d'une EP complète.

Bien qu'ayant le même pouvoir expressif que $\mathcal{T}_2$, le jeu $\mathcal{T}_1$ a plus de transitions, et ne s'est pas avéré plus efficace en pratique, c'est pourquoi nous l'avons abandonné au profit de $\mathcal{T}_2$.

1.2.4 Catégorisation : ensembles de transitions $\mathcal{T}_c$

Transition	Configuration$_{avant}$		Configuration$_{apres}$	Précondition
SHIFT	$(S, i\|B, E)$	$\Rightarrow$	$(S\|i, B, E)$	$B \neq \emptyset$
REDUCE	$(S\|i, B, E)$	$\Rightarrow$	(S, B, E)	$S \neq \emptyset$
MERGE	$(S\|i, j, B, E)$	$\Rightarrow$	$(S\|(i, j), B, E)$	$\|S\| > 1$
MARK AS c_k	$(S\|i, B, E)$	$\Rightarrow$	$(S\|i, B, E \cup \{i_{c_k}\})$	$S \neq \emptyset$ et $i \notin E$

FIGURE 1.4 – **Ensemble de transitions $\mathcal{T}_{c2}$**, variante de $\mathcal{T}_2$ incluant la catégorisation des EPs. Il y a une transition MARK AS $\mathbf{c_k}$ pour chaque catégorie d'EP $\mathbf{c_k}$. La notation $\mathbf{i_{c_k}}$ est utilisée pour une EP à laquelle le système assigne la catégorie $\mathbf{c_k}$.

.

Pour permettre à notre système d'identifier une EP tout en lui assignant une catégorie parmi un ensemble prédéfini (cf. par exemple la typologie des EPs verbales des jeux PARSEME 1.0 et PARSEME 1.1), une solution simple est de démultiplier toute transition de type MARK ou MARKTP des jeux $\mathcal{T}_{\mathbf{0,1,2}}$ en une transition par catégorie d'EP, comme illustré à la figure 1.4 qui détaille la variante avec catégorisation du jeu $\mathcal{T}_2$.

33. Dans Al Saied et al. (2017), la transition MARK_REDUCE et la transition REDUCE sont réunies sous le nom COMPLETE, alors que MARKTP_REDUCE a le nom COMPLETE_TP.

1.3 Puissance expressive

Comme vu dans Partie 1, chapitre 2, dans les données que nous visons d'utiliser, une EP peut être composée d'un seul token comme *contre-indiqué* ou de plusieurs tokens comme *prendre la relève*. Elle peut être continue ou non continue (*faire face* vs. *faire souvent face*). Elle peut être enchâssée dans une autre EP ($il_{1,2}$ $fait_{1,2}$ $l'_{1,2}$ $objet_{1,2}$ $d'une\ évaluation_2$). Deux EPs peuvent se chevaucher ($accomplir_{1,2}$ $de\ multiples\ tâches_1$ $et\ démarches_2$).

Les trois jeux de transitions $\mathcal{T}_{\mathbf{0,1,2}}$ peuvent reconnaître des EPs mono- ou multitokens, et gérer toute longueur de discontinuité. Aucun des trois jeux ne permet de reconnaître un chevauchement d'EP. En effet, prenons l'exemple de trois tokens $\mathbf{i}, \mathbf{j}, \mathbf{k}$ tels que $(\mathbf{i}, \mathbf{j})$ est une EP et $(\mathbf{i}, \mathbf{k})$ également. Pour que $(\mathbf{i}, \mathbf{j})$ soit identifiée, $\mathbf{i}$ et $\mathbf{j}$ doivent avoir fait l'objet d'une transition MERGE, mais alors $\mathbf{i}$ n'est plus utilisable seul pour un MERGE avec $\mathbf{k}$.

Pour ce qui est de l'enchâssement d'une EP dans une autre, nous avons déjà signalé que $\mathcal{T}_\mathbf{0}$ ne le permet pas du tout (car la transition MARK_REDUCE enlève de la pile l'unité marquée comme EP). Les jeux $\mathcal{T}_\mathbf{1}$ et $\mathcal{T}_\mathbf{2}$, qui ont exactement le même pouvoir expressif, peuvent en revanche traiter certains cas d'enchâssement. Nous donnons un exemple à la figure 1.5.

Plus précisément, les cas couverts sont ceux où l'on peut former un arbre projectif en attachant tous les arbres binaires représentant les EPs d'une phrase à une racine fictive, en ignorant les tokens n'appartenant à aucune EP. Une formulation alternative est la suivante : étant donné une EP composée des tokens $\mathbf{t_1}, \mathbf{t_2}, .., \mathbf{t_m}$ et un trou $\mathbf{g_1}, \mathbf{g_2}, .., \mathbf{g_n}$ apparaissant entre les composants $\mathbf{t_i}, \mathbf{t_{i+1}}$, la condition est que si des tokens $\mathbf{g_i}$ appartiennent à une autre EP, cela ne peut être qu'au sein des tokens $\mathbf{g_i}$ et pas en dehors. Ainsi, un cas non couvert se trouve par exemple dans $let_{1,2}$ the_1 cat_1 $out_{1,2}$ of_1 the_1 bag_1 *(lit : 'laisser the chat hors du sac'; 'dévoiler le pot aux roses')*, puisque l'EP *let out* a un trou contenant les tokens the_1 et cat_1, qui appartiennent à une autre EP incluant des tokens situés en dehors du trou.

1.4 Oracle O

Entraîner un système par transitions consiste à lui apprendre à prédire la transition optimale à appliquer à partir de la configuration courante. Cela nécessite un oracle qui est utilisé pour produire les données d'entraînement en transformant les phrases du corpus d'entraînement en des séquences de paires (configuration, transition optimale à appliquer). Ces données alimentent ensuite l'apprentissage supervisé d'un classifieur.

Nous avons défini un oracle statique au sens de Goldberg et Nivre (2013) (cf. section 1.4, Partie 1, chapitre 4). Il renvoie, pour une configuration donnée et un ensemble gold d'EPs, la première transition applicable en respectant l'ordre de priorité suivant : MARK, MERGE, REDUCE et SHIFT. Une transition est applicable lorsque les préconditions sont vérifiées et qu'elle est compatible avec les EPs gold. Nous présentons ci-dessous la procédure de sélection de la transition pour chaque jeu de transitions. À noter que les figures 1.6, 1.7 et 1.8 présentent des exemples pratiques du fonctionnement de l'oracle pour une phrase avec un enchâssement pour les trois jeux de transitions $\mathcal{T}_{\mathbf{0,1,2}}$.

Jeu de transitions $\mathcal{T}_\mathbf{2}$: la compatibilité avec les annotations d'EPs de référence se résume ainsi, en suivant l'ordre de priorité des transitions :

— un MARK ne peut être fait que si $\mathbf{S_0}$ est une EP gold, n'ayant pas encore été ajoutée à $\mathbf{E}$

— un MERGE entre $\mathbf{S_0}$ et $\mathbf{S_1}$ est fait si et seulement si :

	Trans.		**Configuration**
1	$\mathcal{F}_i(P)$	$\Rightarrow$	[ ], [Damit, müsste, man, sich, nun, herumschlagen], [ ]
2	Shift	$\Rightarrow$	[Damit], [müsste, man, sich, nun, herumschlagen], [ ]
3	Reduce	$\Rightarrow$	[ ], [müsste, man, sich, nun, herumschlagen], [ ]
4	Shift	$\Rightarrow$	[müsste], [man, sich, nun, herumschlagen], [ ]
5	Reduce	$\Rightarrow$	[ ], [man, sich, nun, herumschlagen], [ ]
6	Shift	$\Rightarrow$	[man], [sich, nun, herumschlagen], [ ]
7	Reduce	$\Rightarrow$	[ ], [sich, nun, herumschlagen], [ ]
8	Shift	$\Rightarrow$	[sich], [nun, herumschlagen], [ ]
9	Shift	$\Rightarrow$	[sich, nun], [herumschlagen], [ ]
10	Reduce	$\Rightarrow$	[sich], [herumschlagen], [ ]
11	Shift	$\Rightarrow$	[sich, herumschlagen], [ ], [ ]
12	Mark as VPC	$\Rightarrow$	[sich, herumschlagen$_{VPC}$], [ ], [herumschlagen$_{VPC}$]
13	Merge	$\Rightarrow$	[(sich, herumschlagen$_{VPC}$)], [ ], [herumschlagen$_{VPC}$]
14	Mark as IRV	$\Rightarrow$	[(sich, herumschlagen$_{VPC}$)$_{IRV}$], [ ], [herumschlagen$_{VPC}$, (sich, herumschlagen$_{VPC}$)$_{IRV}$]
15	Reduce	$\Rightarrow$	[ ], [ ], [herumschlagen$_{VPC}$, (sich, herumschlagen$_{VPC}$)$_{IRV}$]

FIGURE 1.5 – **Exemple - $\mathcal{T}_c$2 :** Séquence de transitions (avec le jeu $\mathcal{T}_c$**2**) pour l'analyse de la phrase allemande *Damit müsste man* ***sich***$_1$ *nun* ***herumschlagen***$_{1,2}$ (lit : 'avec-cela devoir-3sg-subjonctif quelqu'un se maintenant autour-battre' ; 'Quelqu'un devrait s'en occuper maintenant'), contenant un TP *herumschlagen* (de type VPC) lui-même inclus dans un verbe pronominal (type IRV) *sich herumschlagen*.

  - — $\mathbf{S_0}$ et $\mathbf{S_1}$ constituent une EP
  - — ou bien sont une sous-partie d'une ou plusieurs EPs, et $\mathbf{S_0}$ n'est pas une partie d'une EP comprenant des éléments encore dans le tampon. C'est cette condition qui fait qu'à la configuration 16 de la figure 1.6, l'oracle ne choisit pas un MERGE, car $\mathbf{S_0}$ =*faux* forme une EP avec *pas* qui est encore dans le tampon.
- — REDUCE s'applique si $\mathbf{S_0}$ n'est pas une partie d'EP, ou bien est une EP qui a déjà été ajoutée à **E** et n'est pas une sous-partie d'une autre EP.
- — à défaut c'est SHIFT qui doit s'appliquer.

Jeu de transitions $\mathcal{T}_0$: l'ordre de priorité et la vérification de la compatibilité avec les annotations d'EPs de référence se traduisent comme suit :

- — un MARKTP_REDUCE ne peut être fait que si $\mathbf{S_0}$ est une TP gold, n'ayant pas encore été ajoutée à **E** ;
- — un MARK_REDUCE est fait si et seulement si $\mathbf{S_0}$ est une EP ;
- — un REDUCE s'applique si $\mathbf{S_0}$ est un token qui n'appartient pas à une EP ;
- — un SHIFT s'applique si la pile est vide ou si le token du sommet de la pile n'est pas le dernier token d'une EP ;
- — un MERGE s'applique si et seulement si (1) tous les tokens d'une EP sont sur la pile ; et (2) $\mathbf{S0}$ et $\mathbf{S_1}$ constituent une EP ou une sous-partie de l'EP.

	Trans.		**Configuration**
1	$\mathcal{F}_i(P)$	$\Rightarrow$	[ ], [Le, premier, ministre, a, fait, récemment, plusieurs, faux, pas], [ ]
2	SHIFT	$\Rightarrow$	[Le], [premier, ministre, ..], [ ]
3	REDUCE	$\Rightarrow$	[ ], [premier, ministre, ..], [ ]
4	SHIFT	$\Rightarrow$	[premier], [ministre, a, ..], [ ]
5	SHIFT	$\Rightarrow$	[premier, ministre], [a, fait, ..], [ ]
6	MERGE	$\Rightarrow$	[(premier, ministre)], [a, fait, ..], [ ]
7	MARK	$\Rightarrow$	[(premier, ministre)], [a, fait, ..], [(premier, ministre)]
8	REDUCE	$\Rightarrow$	[ ], [a, fait, ..], [(premier, ministre)]
9	SHIFT	$\Rightarrow$	[a], [fait, récemment, ..], [(premier, ministre)]
10	REDUCE	$\Rightarrow$	[ ], [fait, récemment, ..], [(premier, ministre)]
11	SHIFT	$\Rightarrow$	[fait], [récemment, plusieurs, ..], [(premier, ministre)]
12	SHIFT	$\Rightarrow$	[fait, récemment], [plusieurs, faux, ..], [(premier, ministre)]
13	REDUCE	$\Rightarrow$	[fait], [plusieurs, faux, ..], [(premier, ministre)]
14	SHIFT	$\Rightarrow$	[fait, plusieurs], [faux, pas, ..], [(premier, ministre)]
15	REDUCE	$\Rightarrow$	[fait], [faux, pas, ..], [(premier, ministre)]
16	SHIFT	$\Rightarrow$	[fait, faux], [pas, .], [(premier, ministre)]
17	SHIFT	$\Rightarrow$	[fait, faux, pas], [.], [(premier, ministre)]
18	MERGE	$\Rightarrow$	[fait, (faux, pas)], [.], [(premier, ministre)]
19	MARK	$\Rightarrow$	[fait, (faux, pas)], [.], [(premier, ministre), (faux, pas)]
10	MERGE	$\Rightarrow$	[(fait, (faux, pas))], [.], [(premier, ministre), (faux, pas)]
21	MARK	$\Rightarrow$	[(fait, (faux, pas))], [.], [(premier, ministre), (faux, pas), (fait, (faux, pas))]
22	REDUCE	$\Rightarrow$	[ ], [.], [(premier, ministre), (faux, pas), (fait, (faux, pas))]
23	SHIFT	$\Rightarrow$	[.], [ ], [(premier, ministre), (faux, pas), (fait, (faux, pas))]
24	REDUCE	$\Rightarrow$	[ ], [ ], [(premier, ministre), (faux, pas), (fait, (faux, pas))]

FIGURE 1.6 – **Exemple - $\mathcal{T}_2$:** Les séquences de transitions produites par l'oracle en utilisant l'ensemble de transitions $\mathcal{T}_2$ pour la phrase $P =$ *Le* $\boldsymbol{premier}_1$ $\boldsymbol{ministre}_1$ *a* $\boldsymbol{fait}_2$ *récemment plusieurs* $\boldsymbol{faux}_{2,3}$ $\boldsymbol{pas}_{2,3}$. La phrase contient trois EPs : *premier ministre* (EP nominale) et *faire faux pas* (EP verbale : CVS) enchâssant une autre EP nominale *faux pas*.

	Trans.		**Configuration**
1	$\mathcal{F}_i(P)$	$\Rightarrow$	[ ], [Le, premier, ministre, a, fait, récemment, plusieurs, faux, pas], [ ]
	..		
4	SHIFT	$\Rightarrow$	[premier], [ministre, a, ..], [ ]
5	SHIFT	$\Rightarrow$	[premier, ministre], [a, fait, ..], [ ]
6	MERGE	$\Rightarrow$	[(premier, ministre)], [a, fait, ..], [ ]
7	MARK_REDUCE	$\Rightarrow$	[ ], [a, fait, ..], [(premier, ministre)]
	..		
10	SHIFT	$\Rightarrow$	[fait], [récemment, plusieurs, faux..], [(premier, ministre)]
	..		
15	SHIFT	$\Rightarrow$	[fait, faux], [pas, .], [(premier, ministre)]
16	SHIFT	$\Rightarrow$	[fait, faux, pas], [.], [(premier, ministre)]
17	MERGE	$\Rightarrow$	[fait, (faux, pas)], [.], [(premier, ministre)]
18	MERGE	$\Rightarrow$	[(fait, (faux, pas))], [.], [(premier, ministre)]
19	MARK_REDUCE	$\Rightarrow$	[ ], [.], [(premier, ministre), (fait, (faux, pas))]
	..		

FIGURE 1.7 – **Exemple - $\mathcal{T}_0$:** Les séquences de transitions produites par l'oracle de $\mathcal{T}_0$ pour la même phrase que la figure 1.6. « .. » correspond au traitement des tokens *le*, *a*, *récemment*, *plusieurs* et .. Chacun de ces tokens est traité, comme dans la figure 1.6, avec une séquence de SHIFT et de REDUCE.

	Trans.		**Configuration**
1	$\mathcal{F}_i(P)$	$\Rightarrow$	[ ], [Le, premier, ministre, a, fait, récemment, plusieurs, faux, pas], [ ]
	..		
4	SHIFT	$\Rightarrow$	[premier], [ministre, a, ..], [ ]
5	SHIFT	$\Rightarrow$	[premier, ministre], [a, fait, ..], [ ]
6	MERGE_MARK	$\Rightarrow$	[(premier, ministre)], [a, fait, ..], [(premier, ministre)]
7	REDUCE	$\Rightarrow$	[ ], [a, fait, ..], [(premier, ministre)]
	..		
10	SHIFT	$\Rightarrow$	[fait], [récemment, plusieurs, faux, ..], [(premier, ministre)]
	..		
15	SHIFT	$\Rightarrow$	[fait, faux], [pas, .], [(premier, ministre)]
16	SHIFT	$\Rightarrow$	[fait, faux, pas], [.], [(premier, ministre)]
17	MERGE_MARK	$\Rightarrow$	[fait, (faux, pas)], [.], [(premier, ministre), (faux, pas)]
18	MERGE_MARK	$\Rightarrow$	[(fait, (faux, pas))], [.], [(premier, ministre), (faux, pas), (fait, (faux, pas))]
19	REDUCE	$\Rightarrow$	[ ], [.], [(premier, ministre), (faux, pas), (fait, (faux, pas))]
	..		

FIGURE 1.8 – **Exemple - $\mathcal{T}_1$:** Les séquences de transitions produites par l'oracle de $\mathcal{T}_1$ pour la même phrase que la figure 1.6. « .. » correspond au traitement des tokens *le*, *a*, *récemment*, *plusieurs* et .. Chacun de ces tokens est traité, comme dans la figure 1.6, avec une séquence de SHIFT et de REDUCE.

Entrée : P : Phrase segmentée en tokens ;
E : Ensemble des EPs de référence de P ;
o : Fonction oracle

$c_0 \leftarrow F_i(P)$
$i \leftarrow 0$
$paires\_gold \leftarrow [\,]$
tant que $c_i \notin C_t$ **faire**
  $t_i \leftarrow o(c_i, E)$
  Ajouter (c_i, t_i) aux $paires\_gold$
  $c_{i+1} \leftarrow t_i(c_i)$
  $i \leftarrow i + 1$
fin
retourner $paires\_gold$

Production des données d'entraînement

Entrée : P : phrase segmentée en tokens ;
CLF : fonction de score des transitions ;

$c \leftarrow F_i(P)$;
tant que $c \notin C_t$ **faire**
  $\hat{t} \leftarrow ArgMax_{t \in L(c)} CLF(c, t)$;
  $c \leftarrow \hat{t}(c)$;
fin
retourner c

Prédiction

FIGURE 1.9 – **Oracle et prédiction :** Les algorithmes **(1)** de la production des données d'entraînement du classifieur et **(2)** de l'analyse.

Jeu de transitions $\mathcal{T}_1$: la compatibilité du jeu de transition $\mathcal{T}_1$ avec les annotations d'EPs de référence prend en compte d'autres considérations alors que l'ordre de priorité du jeu de transition $\mathcal{T}_1$, suivi ci-dessous, ne s'éloigne pas de celui de $\mathcal{T}_2$:

- un MARKTP s'applique si $\mathbf{S_0}$ est une TP gold, n'ayant pas encore été ajoutée à $\mathbf{E}$;
- un MARK_MERGE entre $\mathbf{S_0}$ et $\mathbf{S_1}$ se réalise si et seulement si $\mathbf{S_0}$ et $\mathbf{S_1}$ constituent une EP ;
- un MERGE entre $\mathbf{S_0}$ et $\mathbf{S_1}$ est fait si $\mathbf{S_0}$ et $\mathbf{S_1}$ représentent une sous-partie d'une ou plusieurs EPs, et $\mathbf{S_0}$ n'est pas une partie d'une EP comprenant des éléments encore dans le tampon ;
- un REDUCE se réalise si le sommet de la pile contient un token qui n'appartient pas à une EP ou s'il contient un arbre binaire représentant une EP, à condition que cette EP soit ajoutée à l'ensemble $\mathbf{E}$ et ne soit pas enchâssée dans une autre EP ;
- comme en $\mathcal{T}_2$, c'est le SHIFT qui doit s'appliquer à défaut.

Notez qu'aux cas d'enchâssement, l'oracle de $\mathcal{T}_0$ ne considère que les annotations d'EPs de référence de l'EP enchâssante. D'ailleurs, les oracles de $\mathcal{T}_1$ et $\mathcal{T}_2$ ignorent les annotations des EPs enchâssées non identifiables (dont l'arbre binaire ne s'arrange de manière projective avec l'arbre de l'EP enchâssante,- lors de l'attachement des deux arbres à une racine fictive).

Aux cas de chevauchement, les oracles de tous les jeux de transition ignorent les annotations d'EP de référence de l'EP la plus courte (ou la dernière au cas d'égalité de longueur).

1.5 Données d'entraînement et algorithme d'analyse

Pour transformer les phrases des jeux d'entraînement en paires de (configuration, transition), on s'appuie sur un algorithme glouton. La fonction d'initialisation $\mathcal{F}_\mathbf{i}$ génère la configuration initiale pour chaque phrase. L'oracle sélectionne ensuite la seule transition correcte en s'appuyant sur la configuration courante et les annotations de la phrase. Cette étape se répète de manière

itérative jusqu'à ce qu'une configuration terminale soit atteinte. En phase de prédiction, le classifieur occupe la place de l'oracle pour le choix de la transition à appliquer (voir Figure 1.9).

1.5.1 Longueur de la séquence des transitions

Pour l'ensemble de transitions $\mathcal{T}_\mathbf{2}$, une phrase de $\mathbf{n}$ tokens, $\mathbf{m}$ EPs (non enchâssées) et $\mathbf{p}$ cas d'enchâssement produit une séquence de $(\mathbf{2} * \mathbf{n} + \mathbf{m} + \mathbf{p})$ paires (configuration, transition), puisque chaque token n'appartenant pas à une EP exige un SHIFT et un REDUCE et chaque EP de longueur $\mathbf{r}$ exige $\mathbf{r}$ SHIFT, $\mathbf{r}-\mathbf{1}$ MERGE, un MARK et un REDUCE. Un MARK supplémentaire est requis pour chaque enchâssement d'une EP dans une autre. Ainsi pour l'analyse de la phrase de la figure 1.6 contenant 10 tokens, deux EPs non enchâssées et une EP enchâssée, produit une séquence de 23 paires (configuration, transition).

Pour l'ensemble $\mathcal{T}_\mathbf{0}$, l'oracle produit une séquence de $\mathbf{2} * \mathbf{n}$ transitions : idem chaque token n'appartenant pas à une EP exige un SHIFT et un REDUCE, chaque TP exige un SHIFT et un MARKTP et chaque EP de longueur $\mathbf{r} > \mathbf{1}$ exige $\mathbf{r}$ SHIFT, $\mathbf{r}-\mathbf{1}$ MERGE et un MARK_REDUCE (et un token ne peut appartenir qu'à une seule EP).

Enfin, pour l'ensemble $\mathcal{T}_\mathbf{1}$, l'oracle produit une séquence de $\mathbf{2} * \mathbf{n} + \mathbf{m}$ transitions pour une phrase de $\mathbf{n}$ tokens dont $\mathbf{m}$ sont des TPs : un token n'appartenant pas à une EP entraîne 2 transitions, chaque TP non enchâssé entraîne un SHIFT, un MARKTP et un REDUCE. Chaque EP non enchâssée de longueur $\mathbf{r} > \mathbf{1}$ entraîne $\mathbf{2} * \mathbf{r}$ transitions ($\mathbf{r}$ SHIFT, $\mathbf{r}-\mathbf{2}$ MERGE, un MERGE_MARK et un REDUCE). En cas d'enchâssement (d'un TP ou d'une EP multitoken) on n'a pas de transitions supplémentaires.

1.6 Résumé

Dans ce chapitre, nous avons formellement défini un système par transitions dédié à l'identification des expressions polylexicales. Ce système construit incrémentalement l'ensemble des EPs pour une phrase donnée via l'application successive d'actions (ou "transitions") choisies dans un jeu prédéfini. La prochaine transition à appliquer est déterminée soit par un oracle (en phase d'entraînement) soit par un classifieur (en phase d'analyse). Le système est caractérisé par l'utilisation d'une "configuration" constituée d'un tampon de tokens encore à traiter, une pile d'unités en cours de traitement et un ensemble d'EPs déjà identifiées, en cours de construction. Une unité apparaissant dans la pile peut être constituée d'un seul token de la phrase d'entrée, ou bien plusieurs, auquel cas, l'unité a une structure binaire (il s'agit par construction d'une paire d'unités, elles-mêmes mono ou multitoken). Une unité de la pile peut aussi bien correspondre à une EP complète, un début d'EP en cours d'identification ou un token non membre d'une EP. L'application d'une transition à la configuration courante modifie celle-ci pour donner une nouvelle configuration. Étant donnée une configuration initiale dépendant de la phrase en entrée, le système d'identification va appliquer une séquence de transitions jusqu'à obtenir une configuration finale, et ainsi identifier un ensemble d'EPs pour la phrase. En comparaison avec un système d'étiquetage par séquence, cette approche permet en particulier de bénéficier d'un contexte dynamique, dépendant des unités en cours de traitement, et pas seulement dépendant de l'ordre linéaire, et ainsi de faciliter la prise en compte des discontinuités dans les EPs.

Nous avons ensuite présenté les trois jeux de transitions que nous avons définis et utilisés dans nos expérimentations. Ces jeux combinent de différentes manières les deux types d'action centrales pour l'identification d'EPs : une "fusion", pour exprimer que deux unités font partie de la même EP, et un "marquage" pour exprimer qu'une unité est une EP complète. Les trois

jeux sont présentés dans l'ordre chronologique de conception et d'exploitation, avec une augmentation progressive de la puissance expressive. Le jeu de transitions $\mathcal{T}_0$ a été utilisé lors de notre participation à la compétition PARSEME 1.0 pour une première version de notre système par transitions. Il peut gérer l'identification des EPs mono- et multi-tokens, des EPs discontinues (i.e. à trous) et d'EPs incluses dans les trous d'autres EPs (discontinues). Par contre, il ne gère pas les enchâssements ou le chevauchement. Le jeu $\mathcal{T}_1$ est plus expressif car il permet de traiter certains types d'enchâssements. Le jeu de transitions $\mathcal{T}_2$ a la même puissance expressive que $\mathcal{T}_1$ mais est plus élégant.

Nous avons ensuite décrit précisément l'oracle statique (au sens de Goldberg et Nivre (2013)) que nous avons défini, qui permet de transformer de manière déterministe toute phrase annotée en EPs utilisée à l'entraînement en une unique séquence de couples (configuration, transition à appliquer), utilisable comme données d'apprentissage. Puis nous avons présenté l'algorithme glouton de prédiction pour l'identification des EPs. Enfin, nous montrons que le nombre de transitions à appliquer pour analyser une phrase s'exprime mathématiquement en fonction du nombre de tokens de la phrase, du nombre d'EPs à identifier, et du nombre d'enchâssements de celles-ci.

Chapitre 2

Modèle linéaire

En phase d'analyse, le système par transitions essaie d'imiter le fonctionnement de l'oracle, en remplaçant celui-ci par un classifieur multiclasse appris de manière supervisée. Dans ce chapitre, nous proposons d'utiliser un modèle linéaire pour ce classifieur, et plus précisément une machine à vecteurs de support (Support Vector Machine (SVM)). Le choix d'utiliser SVM a été effectué suite à une comparaison empirique préliminaire sur les données PARSEME 1.0 avec d'autres classifieurs linéaires (régression logistique, modèle bayésien naïf et perceptron), où SVM s'est montré plus performant. À partir d'une expérience sur les expressions verbales dans plusieurs langues, nous allons montrer dans ce chapitre que notre système utilisant un classifieur SVM peut être très performant.

Plus précisément, dans tout ce chapitre nous parlons d'expérimentations sur les données PARSEME 1.0, modifiées pour avoir des annotations morphologiques et syntaxiques prédites et pas gold (comme décrit infra section 2.4.3). Nous nous plaçons dans un scénario où nous nous permettons d'une part d'utiliser des traits morphologiques et syntaxiques (et donc supposant de se placer après analyse syntaxique) pour les langues où ces informations sont disponibles, et d'autre part d'utiliser des combinaisons d'hyperparamètres différentes d'une langue à une autre.

Pour des questions de lisibilité, nous nous concentrons ici sur un système que nous appelons SVM_1, utilisant le jeu de transitions $\mathcal{T}_1$ décrit au chapitre précédent, et un classifieur SVM [34]. Nous montrons que ce système est très performant par rapport à un système de base (« baseline ») et aux systèmes ayant participé à la compétition PARSEME 1.0.

Pour notre propre participation à la compétition PARSEME 1.0, détaillée dans (Al Saied et al., 2017), nous utilisions à l'époque un SVM avec le jeu de transitions $\mathcal{T}_0$, que nous appelons SVM_0 qui nous a permis d'obtenir les meilleurs résultats pour la majorité des langues. Nous ne le détaillons pas ici, préférant la version plus aboutie et expressive utilisant le jeu $\mathcal{T}_1$, mais ses résultats seront comparés à SVM_1 dans le chapitre. Nous avons aussi réalisé des expériences de SVM avec l'ensemble de transitions plus élégant $\mathcal{T}_2$. Ses performances seront détaillées dans le chapitre suivant à des fins de comparaison avec un système neuronal par transitions appris dans les mêmes conditions expérimentales (en particulier sur les données PARSEME 1.1 et pas PARSEME 1.0).

Dans ce chapitre, nous décrivons d'abord brièvement ce qu'est un classifieur SVM (section 2.1). La section 2.2 présente ensuite les patrons de traits que nous avons proposés pour alimenter notre modèle et qui jouent un rôle crucial dans les performances de ce dernier. Puis,

34. Cette variante est celle décrite dans le chapitre d'ouvrage que nous avons rédigé durant la première année de la thèse (Al Saied et al., 2018), publié dans le livre « Multiword expressions at length and in depth : Extended papers from the MWE 2017 workshop ».

la section 2.3 introduit la méthode de réglage des hyperparamètres de la variante SVM_1, qui contrôlent l'extraction des traits et la génération des données d'entraînement. La section 2.4 présente ensuite les performances de cette variante sur les tâches d'identification et de catégorisation de la compétition PARSEME 1.0. Dans la section 2.5, nous présentons une analyse comparative entre la variante SVM_1 et un système de base exploitant uniquement le lexique lemmatisé des EPs extrait du corpus d'apprentissage. Enfin, nous explorerons expérimentalement l'impact des différents patrons de traits sur les performances du SVM_1 sur les données françaises de PARSEME 1.0 (section 2.6).

2.1 Machine à vecteurs de support (SVM)

SVM est l'une des méthodes de classification binaire les plus connues et performantes dans le domaine de l'apprentissage automatique supervisé. Cette méthode est basée sur l'utilisation de fonctions dîtes noyaux (« kernel ») qui permet une séparation optimale des données. Elle consiste à trouver un classifieur binaire (« hyperplan ») maximisant la distance entre les deux classes pour obtenir une marge maximale entre la frontière de séparation et les exemples les plus proches. Les fonctions noyaux permettent de traiter les données non séparables linéairement, en transformant l'espace de représentation des données d'entrée en un espace de plus grande dimension, dans lequel il est probable qu'il existe une séparation linéaire. En plus de ses excellentes performances sur de nombreuses tâches de classification, cette approche se caractérise par son efficacité computationnelle en termes de gestion de mémoire. Par ailleurs, les fonctions noyaux peuvent être combinées, modifiées et ainsi adaptées à la tâche cible.

Les modèles SVM peuvent aussi être utilisés pour des tâches de classification multiclasse. Ce type de modèle est d'ailleurs l'un des plus répandus dans le domaine de l'identification des EPs (Legrand et Collobert, 2016; Salton et al., 2016; Stodden et al., 2018; Tu, 2012). Pour étendre un SVM à la classification multiclasse, les stratégies classiques consistent à transformer le problème en un ensemble de classifications binaires. La stratégie *un contre tous* consiste à entraîner un seul classifieur pour chaque classe, les exemples de cette classe étant des exemples positifs et tous les autres exemples, des négatifs. La classe gagnante est celle dont le classifieur produit la probabilité la plus élevée. Un autre exemple de stratégie est la stratégie *un contre un*. Celle-ci s'appuie sur $\mathbf{n} * (\mathbf{n} - \mathbf{1})/\mathbf{2}$ classifieurs binaires différents, $\mathbf{n}$ étant le nombre total de classes. Chaque classifieur est associé à une paire différente de classes. La classe gagnante est celle qui aura été prédite le plus grand nombre de fois par l'ensemble des classifieurs (Liu et Zheng, 2005). Par exemple, supposons que nous avons trois classes, $\mathbf{A}, \mathbf{B}$ et $\mathbf{C}$, la stratégie *un contre un* créera $\mathbf{3} * (\mathbf{3} - \mathbf{1})/\mathbf{2} = \mathbf{3}$ classifieurs binaires. Le premier discriminera $\mathbf{A}$ de $\mathbf{B}$, le second $\mathbf{A}$ de $\mathbf{C}$ et le troisième $\mathbf{B}$ de $\mathbf{C}$. Par conséquent, un exemple $\mathbf{X}$ est présenté à chaque classifieur binaire et la classe gagnante est celle qui réunit le plus de voix. Il existe d'autres stratégies plus complexes comme *Error-Correcting Output-Code* (Dietterich et Bakiri, 1994). Dans notre travail, nous avons adopté, pour le SVM, la stratégie *un contre tous* avec un noyau linéaire, suite à une comparaison empirique préliminaire des stratégies sur les données PARSEME 1.0.

$$\mathbf{C_{Hinge}}(\mathbf{y\prime}, \mathbf{y}) = \mathbf{max}(\mathbf{0}, \mathbf{1} - \mathbf{y} * \mathbf{y\prime}) \tag{2.1}$$

Par ailleurs, notre classifieur SVM utilise la fonction de coût Hinge, qui pénalise les prédictions erronées ($\mathbf{y}, \mathbf{y\prime}$ ont des signes opposés) et les prédictions correctes, mais qui n'ont pas assez de marge ($\mathbf{y} < \mathbf{1}$). Notez que $\mathbf{y}$ représente la sortie brute du classifieur, et pas le label de la classe (voir l'équation 2.1).

2.2 Patrons de traits

Les patrons de traits, connus pour avoir un impact important sur les performances des modèles linéaires, constituent un élément clé pour la construction d'un système par transitions s'appuyant sur un classifieur SVM. Cette section est dédiée à la description des patrons de traits que nous avons utilisés. Ceux-ci utilisent les éléments ciblés de la configuration courante à partir de laquelle on veut prédire la transition suivante à appliquer. Pour une meilleure lisibilité, nous formons des groupes de patrons ayant des caractéristiques similaires. Nous utilisons ci-après le symbole $\mathbf{B_i}$ pour indiquer le $i$ème élément dans le tampon. $\mathbf{S_0}$ et $\mathbf{S_1}$ représentent les deux premiers éléments de la pile. Pour chaque élément $\mathbf{X}$ de la pile ou du tampon, on note sa forme $\mathbf{X_t}$, son lemme $\mathbf{X_l}$ et son étiquette morphosyntaxique $\mathbf{X_p}$. La concaténation de deux éléments $\mathbf{X}$ et $\mathbf{Y}$ est notée $\mathbf{XY}$.

2.2.1 Patrons lexicaux et morphosyntaxiques

Les patrons de traits s'appuyant sur les attributs linguistiques classiques (forme, lemme et étiquette morphosyntaxique) des éléments en entrée jouent traditionnellement un rôle crucial pour les modèles linéaires du TAL, et en particulier pour les systèmes par transitions, cf. Zhang et Nivre (2011b) pour l'analyse en dépendances. Les EPs recouvrant, par ailleurs, des phénomènes essentiellement lexicaux, il est raisonnable de penser que ce type de patrons aura un impact essentiel pour l'identification de ces expressions. Ainsi, nous avons extrait des patrons utilisant les attributs linguistiques de base (forme, lemme et étiquette morphosyntaxique) des éléments ciblés de la pile $\mathbf{S}$ et du tampon $\mathbf{B}$ de la configuration courante. Ces patrons peuvent prendre la forme d'unigrammes, de bigrammes et de trigrammes. Par défaut, les éléments ciblés sont $\mathbf{S_0}, \mathbf{S_1}, \mathbf{B_0}, \mathbf{B_1}$ et $\mathbf{B_2}$. Les bigrammes et trigrammes utilisés sont $\mathbf{S_1S_0}, \mathbf{S_0B_0}, \mathbf{S_1B_0}, \mathbf{S_0B_1}, \mathbf{S_0B_2}$ et $\mathbf{S_1S_0B_0}$.

En tenant compte des attributs linguistiques des éléments, il existe un large ensemble de combinaisons possibles que nous fixons de la manière suivante. Les traits d'un unigramme $\mathbf{X}$ sont $\mathbf{X_t}$, $\mathbf{X_l}$ et $\mathbf{X_p}$ alors que ceux d'un bigramme $\mathbf{XY}$ sont $\mathbf{X_tY_t}$, $\mathbf{X_lY_l}$, $\mathbf{X_pY_p}$, $\mathbf{X_lY_p}$ et $\mathbf{X_pY_l}$. Les traits extraits d'un trigramme $\mathbf{XYZ}$ sont $\mathbf{X_tY_tZ_t}$, $\mathbf{X_lY_lZ_l}$, $\mathbf{X_pY_pZ_p}$, $\mathbf{X_lY_pZ_p}$, $\mathbf{X_pY_lZ_p}$, $\mathbf{X_pY_pZ_l}$, $\mathbf{X_lY_lZ_p}$, $\mathbf{X_lY_pZ_l}$ et $\mathbf{X_pY_lZ_l}$.

Pour les langues dépourvues d'annotations basiques comme les lemmes et les étiquettes morphosyntaxiques, nous essayons de compenser cette absence par des patrons s'appuyant sur les suffixes de taille 2 et 3 des éléments ciblés. Par exemple, si l'élément correspond au token *identification*, on extrait les traits correspondants $\mathbf{X_{s2}} = \mathbf{on}$ et $\mathbf{X_{s3}} = \mathbf{ion}$.

2.2.2 Patrons Syntaxiques

Outre les traits linguistiques classiques, nous explorons des traits syntaxiques, plus sophistiqués sur le plan linguistique. Tout d'abord, il est connu que la prédiction préalable de la structure syntaxique facilite l'identification des EPs (Fazly et al., 2009; Seretan, 2011; Nagy T. et Vincze, 2014). Par exemple, de nombreuses EPs séquentiellement discontinues sont syntaxiquement connectées, comme nous l'avons évoqué Partie 1, section 2.5.2 (37), citant (Moreau et al., 2018a). Par ailleurs, certaines EPs se caractérisent par leur structure syntaxique irrégulière (Candito et Constant, 2014). Ainsi des patrons de traits intégrant les liens de dépendances entre les tokens peuvent aider à repérer les EPs discontinues ou les EPs syntaxiquement irrégulières.

Nous extrayons donc des traits à partir de la configuration courante en s'appuyant sur des patrons syntaxiques (si l'information est disponible dans les données). Pour chaque token $\mathbf{B_i}$

qui apparaît dans le tampon et dépend de $\mathbf{S_0}$ avec la relation syntaxique **rs**, nous indiquons l'existence de la dépendance à l'aide des patrons :

— $Dependance\_Droite(Lemme(S_0), Lemme(B_i)) = vrai/faux$

— $Etiquette\_De\_Dependance\_Droite(Lemme(S_0), Lemme(B_i)) = rs$

Lorsque la tête syntaxique **G** de $\mathbf{S_0}$ apparaît dans le tampon, nous utilisons également les traits inverses :

— $Est\_Gouverne\_Par(Lemme(Lemme(S_0), Lemme(G)) = vrai/faux$

— $Est\_Gouverne\_Par\_l'Etiquette(Lemme(S_0), Lemme(G)) = rs$

Enfin, le patron $Relation\_Syntaxique(Lemme(S_0), Lemme(S_1)) = \pm rs$ tient compte de la relation syntaxique rs entre les deux éléments au sommet de la pile $\mathbf{S_0}$ et $\mathbf{S_1}$. La direction ($\mathbf{S_0}$ est le gouverneur ou gouverné de/par $\mathbf{S_1}$) est indiqué grâce au caractère $\pm$. À noter que, nous n'extrayons des traits syntaxiques que si les éléments considérés dans les relations syntaxiques correspondantes sont des tokens, et pas des unités multitokens.

2.2.3 Lexiques des données d'entraînement

Nous ajoutons également des patrons basés sur deux lexiques automatiquement extraits des jeux d'entraînement. Il s'agit de lexiques lemmatisés pour les langues où les lemmes sont disponibles, sinon les formes fléchies sont utilisées. Le premier lexique LEX_{TP} contient tous les TPs du jeu d'entraînement et le deuxième $LEX_{tokens(EPs)}$ contient tous les tokens apparaissant comme composant d'au moins une EP du jeu d'entraînement. Les patrons utilisant ces lexiques sont **(i)** un patron binaire qui s'active lorsque $\mathbf{S_0}$ appartient au lexique des TPs ($S_0 \in LEX_{TP} = vrai/faux$) ; et **(ii)** un patron binaire par entité ciblée $\mathbf{S_0}, \mathbf{S_1}, \mathbf{B_0}, \mathbf{B_1}$ ou $\mathbf{B_2}$, qui s'active quand l'élément en question appartient à $LEX_{tokens(EPs)}$, par exemple $S_0 \in LEX_{tokens(EPs)} = vrai/faux$.

2.2.4 Historique des transitions prédites

Nous avons également expérimenté d'autres patrons traditionnels des systèmes par transitions, tels que l'historique des transitions appliquées avant d'arriver à la configuration courante. Nous ajoutons donc des patrons de traits pour représenter la séquence des transitions précédentes de longueurs 1, 2 et 3.

2.2.5 Distance entre les éléments

La distance entre les éléments de la phrase est également connue pour faciliter l'analyse en dépendances par transitions (Zhang et Nivre, 2011b). Par conséquent, nous exploitons des patrons reposant sur la distance entre $\mathbf{S_0}$ et $\mathbf{B_0}$ et la distance entre $\mathbf{S_0}$ et $\mathbf{S_1}$, ce qui peut informer le modèle des trous possibles au sein des EPs, ainsi que de leurs tailles habituelles.

2.2.6 Longueur de la pile

L'utilisation de la taille effective de la pile en tant que patron supplémentaire peut également se révéler bénéfique puisque, par exemple, une taille importante de pile pourrait favoriser un Reduce ou un Merge, les trous au sein des EPs étant souvent de taille limitée, cf. (Savary et al., 2017) pour le cas des expressions verbales.

2.3 Réglage des hyperparamètres

Cette section aborde la méthode et les résultats du réglage des hyperparamètres de la variante SVM_1, ayant pour but de sélectionner les patrons de traits à utiliser parmi ceux que nous venons de lister.

2.3.1 Méthode de réglage

Nous avons utilisé une méthode manuelle pour le réglage des hyperparamètres de la variante SVM_1, c'est-à-dire ici pour sélectionner quels patrons de traits sont à utiliser. Le réglage se décompose en deux étapes principales : **(1)** un réglage préliminaire pour un sous-ensemble de langues « pilotes » choisies parmi les 18 langues, suivi par **(2)** un réglage pour toutes les langues de PARSEME 1.0.

Pour choisir les langues pilotes de la première étape, nous divisons les jeux de données en trois groupes en fonction du type d'annotations disponibles :

A) le bulgare, l'hébreu et le lituanien, pour lesquels il n'y a aucune annotation morphologique ou syntaxique ;

B) le tchèque, l'espagnol, le farsi, le maltais et le roumain, accompagnés par des fichiers CoNLL-U contenant uniquement les annotations morphologiques (lemmes, étiquettes morphosyntaxiques),

C) les autres langues [35], munies d'un fichier CoNLL-U entièrement annoté (annotations morphologiques et syntaxiques).

Nous sélectionnons ensuite une langue « pilote » pour chaque groupe (le bulgare, le tchèque et le français). Nous avons en outre ajouté l'allemand comme langue pilote pour couvrir le cas où la proportion de TPs et d'EPs enchâssées est significative.

Nous avons ensuite essayé de trouver les ensembles de patrons de traits les plus performants pour chacune des quatre langues pilotes. Compte tenu de l'explosion combinatoire du nombre de jeux de patrons de traits à tester, nous ne pouvions pas appliquer une recherche systématique pour les langues pilotes et a fortiori pour toutes les langues. Nous avons expérimenté de nombreuses combinaisons d'hyperparamètres sur les quatre langues pilotes, en utilisant deux modes : **(i)** un apprentissage sur une partie du jeu d'entraînement (80 %) avec évaluation sur le reste ; et **(ii)** une validation croisée à 5 plis sur tout le jeu d'entraînement. Nous avons sélectionné les ensembles de patrons donnant des bonnes performances dans les deux modes. À partir de ces ensembles de patrons, nous avons ensuite investigué quelques variations, langue par langue, sans avoir, cela dit, pu viser l'exhaustivité.

2.3.2 Résultats : ensembles de patrons de traits retenus

Le tableau 2.1 de gauche présente les ensembles de patrons de traits retenus pour chaque langue de PARSEME 1.0. Chaque ligne du tableau indique les codes des patrons de traits actifs pour la langue, alors que le tableau de droite documente les codes des patrons de traits. Le polonais (PL), par exemple, est associé aux symboles *A A' B C H L*, ce qui signifie que cette langue utilise les unigrammes lexicaux et morphosyntaxiques **(A, A')**, les bigrammes lexicaux et morphosyntaxiques à l'exception de S_0B_2 **(C)** les traits syntaxiques **(B)**, l'historique de transitions de longueur 3 **(H)** et le dictionnaire des tokens d'EPs **(L)**.

35. Ces langues sont l'allemand, le grec, le français, le hongrois, l'italien, le polonais, le portugais, le slovène, le suédois et le turc.

Lang.	Patrons de traits
BG	A A" C D F G I L
CS	A A' C F G H I J K L M
DE	A A' B C D E J L N
EL	A A' B C E J K L
ES	A A' C D F G H I J K L
FA	A A' C I J K
FR	A A' B C E I J K L
HE	A A" C E F G H K L
HU	A A' B C D F G H K L N
IT	A A' B C H J L
LT	A A" C D E F G H I J K L M
MT	A A' C F G H J L M
PL	A A' B C H L
PT	Tous les patrons sauf A"
RO	A A' C D E F G H I J K
SL	A A' B C F G H I K N
SV	Tous les patrons sauf N et A"
TR	A A' B C F G H I K

	Patron de traits
A	Unigrammes de forme fléchie
A'	Utilisation des lemmes et des POS
A"	Utilisation des suffixes
B	Utilisation de patrons syntaxiques
C	Bigrammes $S_0S_1, S_0, B_0, S_0, B_1, S_1, B_0$
D	Trigrammes $S_1S_0B_0$
E	Bigramme S_0B_2
F	Historique de transition (longueur 1)
G	Historique de transition (longueur 2)
H	Historique de transition (longueur 3)
I	Distance entre S_0 et S_1
J	Distance entre S_0 et B_0
K	Unigramme B_1
L	$LEX_{tokens(EPs)}$
M	Longueur de la pile

TABLE 2.1 – **Résultat du réglage des hyperparamètres :** À gauche : pour chaque langue de PARSEME 1.0, la colonne **Patrons de traits** répertorie les patrons de traits que nous activons à la suite du réglage manuel. À droite, les patrons de traits avec leurs codes.

On peut constater à la lecture de ce tableau que toutes les langues utilisent les unigrammes **(A)** et les bigrammes **(C)**, ce qui reflète l'importance des traits lexicaux pour l'identification. D'un autre côté, les traits syntaxiques prouvent leur efficacité, puisqu'ils sont actifs pour toutes les langues où ils sont disponibles.

Alors que les trigrammes **(D)** et le bigramme S_0B_2 **(E)** ne se révèlent efficaces que pour huit langues, les historiques de transitions **(F, G** et **H)**, les distances entre les éléments de la configuration (**I** et **J**), l'utilisation de l'unigramme B_1 et les traits de dictionnaires [36] sont actifs pour une bonne partie des langues. Par contre, la longueur de la pile ne se montre pas bénéfique pour la plupart des langues.

2.4 Expérimentations et résultats

Dans cette section, nous présentons les résultats d'identification et de catégorisation des EPs de cette variante. Avant cela, nous présentons le cadre expérimental de ces expérimentations : les métriques d'évaluation, un système de base (« baseline ») ainsi qu'une procédure responsable de la production des annotations morphologiques et syntaxiques prédites pour certaines langues, remplaçant les annotations gold.

36. Il est à noter que le trait $\mathbf{S_0} \in \mathbf{LEX_{TP}}$ ne fait pas l'objet du réglage et qu'il est systématiquement activé sur les langues dont le jeu d'entraînement comprend des TPs.

2.4.1 Métriques d'évaluation

Nous évaluons la performance des systèmes d'identification d'EPs avec des mesures classiques telles que le rappel (ci-après $\boldsymbol{R}$), la précision ($\boldsymbol{P}$) et leur moyenne harmonique , i.e. la F-mesure ($\boldsymbol{F}$), en comparant les EPs prédites par le système et les EPs dans la référence (voir Partie 1, section 2.6, page 39). Cette comparaison est calculée soit au niveau des composants des EPs (ci-après eval-cmpts) soit au niveau des EPs prises dans leur globalité (ci-après, eval-eps). Pour mesurer cette performance sur un ensemble de langues, nous suivons la métrique officielle de PARSEME, qui consiste à calculer la moyenne macro des F-mesures sur toutes les langues (ci-après $\boldsymbol{F_{avg}}$).

Par rapport à utiliser un moyennage micro, un moyennage macro (i.e. où chaque langue a le même poids) est approprié car les nombres d'EPs des corpus de développement et de test sont similaires pour la majorité des langues de PARSEME 1.0 et PARSEME 1.1. En revanche, nous avons constaté des différences notables de résultats dans la manière d'obtenir une F-mesure globale : la moyenne des scores $\mathbf{F}$ que constitue la métrique officielle $\boldsymbol{F_{avg}}$est mathématiquement et empiriquement différente de la F-mesure des précisions et rappels moyens. Nous utilisons néanmoins $\boldsymbol{F_{avg}}$ à des fins de comparaison avec les résultats des deux campagnes internationales, mais nous rapportons également la F-mesure des moyennes macros des précisions et des rappels de toutes les langues considérées (ci-après $\mathbf{F_g}$).

Nous utilisons $\boldsymbol{F_{avg}}$ et $\boldsymbol{F_g}$ tout au long de la thèse, aussi bien pour l'évaluation eval-cmpts que pour eval-eps.

2.4.2 Système de base (baseline)

Afin de pouvoir mieux interpréter le niveau de performance par nos systèmes, nous avons mis en place un système de base (« baseline ») qui s'appuie sur un lexique lemmatisé des EPs vues dans le corpus d'apprentissage pour chaque langue. Chaque entrée de ce lexique lemmatisé d'EPs correspond à l'ensemble des lemmes des composants de l'EP correspondante. Pour chaque phrase du jeu d'évaluation et chaque entrée de ce lexique, le système annote une occurrence si tous les lemmes des composants de l'EP sont présents dans la version lemmatisée de la phrase, dans n'importe quel ordre. Pour limiter un peu le bruit, la distance tolérée entre deux tokens d'une EP supposée est égale à la distance maximale entre ces deux tokens dans le jeu d'entraînement de la langue. Cette règle permet de donner une latitude assez large pour identifier les EPs discontinues et variantes syntaxiques, tout en diminuant un peu les chances d'identifier des cooccurrences accidentelles de composants. Ce système peut donc traiter les EPs discontinues, enchâssées, et en chevauchement, en plus des variantes flexionnelles et syntaxiques d'EPs.

Pour les langues où les lemmes des tokens ne sont pas fournis, on utilise les formes fléchies, sans généralisation morphologique.

2.4.3 Production d'annotations morphologiques et syntaxiques prédites

Certaines des annotations supplémentaires des données PARSEME 1.0 sont des annotations validées manuellement. Les participants à la campagne internationale avaient la possibilité d'utiliser ces données, mais il s'agit d'un scénario irréaliste de prédiction d'EPs, étant donné que l'on ne peut pas tabler sur disposer d'annotations gold pour un texte tout-venant. Pour les langues accompagnées d'annotations gold, nous avons produit des annotations morphologiques et syntaxiques prédites automatiquement, par apprentissage supervisé sur les annotations gold : les étiquettes morphosyntaxiques de quatre langues (tchèque, français, hongrois et polonais) ainsi que les annotations syntaxiques de deux langues (français et hongrois).

Cette étape a été appliquée afin de se placer dans des conditions plus réalistes et homogènes pour toutes les langues. Si lors de l'apprentissage, des traits morphologiques et syntaxiques sont utilisés, il vaut mieux se placer dans des conditions les plus proches de ce qui sera vu lors de la phase de prédiction sur du nouveau texte, et donc utiliser des annotations morphologiques et syntaxiques prédites. Pour les étiquettes morphosyntaxiques comme pour les annotations syntaxiques, nous appliquons la technique du jackknifing à 10 plis qui consiste à :

1. découper le corpus d'entraînement de la langue en 10 plis ;
2. pour chaque plis, utiliser les neuf autres parties comme corpus d'entraînement d'un modèle d'étiquetage morphosyntaxique ou d'analyse syntaxique et s'en servir pour annoter le plis ;
3. reconstituer le corpus d'entraînement complet, avec les annotations ainsi prédites ;
4. l'utiliser pour entraîner un modèle, lui-même utilisé ensuite pour prédire les annotations pour le corpus de test.

Nous avons utilisé l'étiqueteur morphosyntaxique Marmot [37] pour les étiquettes morphosyntaxiques et la chaîne d'outils d'analyse Mate [38] pour l'annotation des dépendances syntaxiques.

2.4.4 Résultats d'identification

Le tableau 2.2 présente les résultats de SVM_1 sur la tâche d'identification des EPs verbales des jeux de données PARSEME 1.0 et les compare avec les performances de plusieurs systèmes. Plus précisément, d'une part nous comparons SVM_1 avec le système de base (baseline) en mode validation croisée (à 5 plis) sur le corpus d'entraînement. D'autre part, nous comparons les performances du SVM_1 sur les corpus de test avec les systèmes participants à la compétition PARSEME 1.0, à la fois pour chaque langue, et toutes langues confondues.

Comparaison au système de base : L'évaluation croisée montre que le modèle SVM_1 est globalement nettement supérieur au système de base avec 9 points d'écart environ sur les jeux d'entraînement en validation croisée ($F_{avg} = 53,7$ versus $44,8$). Le système de base est cependant meilleur pour trois langues : l'hébreu, l'italien et le maltais. Nous constatons également que le système de base est compétitif (moins de 5 points d'écart) sur les langues sans aucune annotation supplémentaire (le bulgare, et le lituanien). Ainsi, le manque de données et le manque d'informations linguistiques généralisant au-dessus des formes fléchies (lemmes, étiquettes) semblent handicaper l'apprentissage du modèle SVM par rapport au système de base.

Comparaison aux systèmes de PARSEME 1.0 : Nous comparons maintenant les résultats de SVM_1 avec notre système participant à la *closed track* (sans données externes) de la compétition PARSEME 1.0, et avec les meilleurs résultats au sein des 6 autres participants. Cette comparaison est à la faveur de SVM_1, mais il faut garder à l'esprit que nous avons pris plus de temps pour régler SVM_1 que n'en ont eu les systèmes participants à la compétition (dont SVM_0).

Pour la comparaison toutes langues confondues, nous avons calculé la moyenne des meilleurs résultats par langue obtenus par les participants de la compétition (ce qui correspond à une performance artificiellement haute, prenant le meilleur système par langue). Deux langues (LT et HE) n'ayant pas été traitées par les participants de la compétition sauf par notre variante SVM_0, nous avons réalisé la moyenne sur les 16 autres langues.

Pour la métrique $\mathbf{F(eval - eps)}$, toutes langues confondues SVM_0 et SVM_1 ont de très bons résultats $\mathbf{F_{avg}(eval - eps)}$, avec une légère avance pour SVM_1, et un écart de 7 points par

37. `http://cistern.cis.lmu.de/marmot/`
38. `http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/matetools.en.html`

Lang.	Validation croisée		Jeu de test					
	eval-eps		eval-eps			eval-cmpts		
	baseline	SVM_1	SVM_1	SVM_0	PrsM	SVM_1	SVM_0	PrsM
BG	48.3	**53.0**	55.8	**61.3**	34.7	60.2	**66.2**	59.2
CS	60.1	**68.9**	70.9	**71.7**	64.2	**73.9**	73.7	72.9
DE	39.9	**47.6**	**45.8**	41.1	40.5	44.8	41.1	**45.5**
EL	48.0	**57.3**	**42.8**	40.1	31.9	46.3	**46.9**	43.1
ES	61.2	**66.0**	**58.9**	57.4	44.3	**60.3**	58.4	49.2
FA	67.3	**81.1**	84.3	**86.6**	80.1	84.8	**90.2**	85.4
FR	66.0	**73.8**	**60.6**	57.7	50.9	**62.6**	60.3	61.5
HE	**30.0**	26.8	29.9	**33.4**	-	30.5	**31.3**	-
HU	73.7	**83.7**	**74.8**	69.9	74.0	**72.1**	67.5	70.8
IT	**43.6**	27.4	28.2	**39.0**	23.1	39.5	**43.6**	34.9
LT	20.7	**21.5**	**34.1**	28.4	-	**31.7**	25.3	-
MT	**7.7**	7.2	6.9	**14.4**	6.4	9.4	**16.3**	8.9
PL	70.0	**73.6**	**75.1**	69.1	68.0	**75.5**	70.6	72.7
PT	65.2	**67.5**	**69.6**	67.3	58.1	**71.4**	70.9	70.2
RO	61.8	**86.0**	**86.3**	75.3	77.8	**87.0**	79.1	83.6
SL	17.3	**40.8**	42.9	**43.2**	37.1	45.7	**46.6**	45.6
SV	6.9	**24.7**	30.1	**30.4**	30.3	**34.3**	30.7	31.5
TR	19.3	**60.1**	53.8	**55.4**	51.8	53.9	**55.3**	52.9
F_{avg}	44.8	**53.7**	**52.8**	52.3	-	**54.7**	54.1	-
$F_{avg(16)}$	47.3	**57.4**	**55.4**	55.0	48.3	**57.6**	57.3	55.2

TABLE 2.2 – **Identification** Les scores **F** du système de base, de nos variantes SVM_0, SVM_1 et des systèmes de la compétition PARSEME 1.1. Pour chaque langue de PARSEME 1.0, les colonnes 2 et 3 fournissent les scores au niveau des EPs (eval-eps) en validation croisée sur le jeu d'entraînement, de la baseline et de notre variante $\mathbf{SVM_1}$. Les scores fournis représentent la moyenne des 5 itérations de l'évaluation croisée. Les six colonnes suivantes concernent les corpus de test, et comparent le score **F** au niveau des EPs (eval-eps) ou des composants (eval-cmpts) de $\mathbf{SVM_1}$ avec les scores du système $\mathbf{SVM_0}$ (le système avec lequel nous avons participé à la compétition PARSEME 1.0) et les meilleurs scores de PARSEME 1.0 excluant SVM_0 (**PrsM**). Par exemple, PrsM présente le score du deuxième système de PARSEME 1.0 lorsque SVM_0 obtient le meilleur score. La ligne **AVG** présente la moyenne des scores **F** sur toutes les langues, alors que la ligne $\mathbf{AVG_{16}}$ **présente la même moyenne sur les 16 langues où un système participant autre que** $\mathbf{SVM_0}$ **a fourni un résultat (toutes sauf l'hébreu et le lituanien)**. La couleur de la langue indique si elle est accompagnée par des annotations morphologiques et syntaxiques (noir), morphologiques seulement (bleu) ou s'il n'y que les tokens (rouge). Les langues en gras sont celles qui ont de grands corpus d'entraînement, c'est-à-dire, leur nombre de tokens est supérieur à la moyenne du nombre de tokens de toutes les langues.

rapport aux autres systèmes participants à PARSEME 1.0. Pour chaque langue, c'est soit SVM_1 soit SVM_0 qui donne le meilleur $\mathbf{F_{avg}}(\mathbf{eval-eps})$.

En termes de $\mathbf{F_{avg}}(\mathbf{eval-cmpts})$, nous constatons à peu près la même tendance : SVM_0 et SVM_1 sont globalement au-dessus, mais l'écart est nettement réduit avec les autres systèmes participants (2 points environ), et SVM_0 et SVM_1 sont battus pour une des langues (l'allemand). On constate donc une meilleur avantage compétitif de notre système par transitions avec la métrique $\mathbf{F}(\mathbf{eval-eps})$, ce qui peut peut-être s'expliquer par l'existence d'une transition spécifique pour marquer une EP complète, par rapport à des systèmes de type étiquetage IOB avec CRF. Ceux-ci peuvent favoriser des identifications d'EPs partielles, étant donné que l'historique des prédictions est limité (juste l'étiquette prédite précédente dans le cas d'un CRF linéaire), mais cela peut provoquer des problèmes pour l'identification complète d'EPs longues ou discontinues.

SVM_0 et SVM_1 : Comme vu à l'instant, les systèmes SVM_0 et SVM_1 ont des performances très proches. SVM_1 a une performance globale légèrement meilleure avec des écarts de 0,5 et 0,6 point respectivement en termes de $\mathbf{F_{avg}}(\mathbf{eval-eps})$ et de $\mathbf{F_{avg}}(\mathbf{eval-cmpts})$, mais on notera que chacun est meilleur que l'autre sur 9 langues avec la mesure $\mathbf{F}(\mathbf{eval-eps})$. Si l'on essaye de comparer ces résultats pour les différentes langues en tenant en compte desquelles sont relativement mieux fournies en EPs enchâssées (potentiellement identifiables par SVM_1), i.e. hébreu, allemand, tchèque, italien, et desquelles ont bénéficié de contraintes codées en dur, nous ne trouvons malheureusement pas de corrélations flagrantes.

Bien que très ressemblante à SVM_1, la variante SVM_0 est cependant conceptuellement moins satisfaisante que son successeur chronologique SVM_1. Tout d'abord, elle utilise l'ensemble de transitions $\mathcal{T}_0$ avec une plus faible puissance expressive que $\mathcal{T}_1$ utilisé par SVM_1 (ex. possibilité d'identification des EPs enchâssées pour SVM_1 et pas pour SVM_0). Par ailleurs, SVM_0 inclut l'application systématique de transitions si certaines conditions (codées en dur) dans la configuration courante sont satisfaites, et ne fait donc pas appel au classifieur dans ces cas-là. Par exemple, l'identification des TPs a été effectuée systématiquement lorsque le lemme de $\mathbf{S_0}$ se trouvait dans un lexique des EPs du corpus d'apprentissage. De même, lorsque les lemmes de $\mathbf{S_0}$ et $\mathbf{B_0}$ appartenaient au lexique des composants d'EPs du corpus d'apprentissage, une transition MERGE était systématiquement appliquée. Ce dernier cas a été activé pour certaines langues dont les EPs sont caractérisées par leur longueur importante et/ou par la proportion élevée d'EPs discontinues dans leurs jeux d'entraînement. Cet ensemble de langues est composé de l'allemand, de l'hébreu, du hongrois, de l'italien, du lituanien, du maltais, du portugais et du slovène.

En conclusion, ce tableau montre que notre système par transitions guidé par un classifieur SVM obtient des performances largement supérieures à un système de base simple. Ses performances constituent l'état de l'art (de l'époque) pour quasiment toutes les langues et sont très compétitives pour les quelques langues restantes.

2.4.5 Catégorisation

La compétition PARSEME 1.0 proposait également de catégoriser les EPs en plus de les identifier. Nous n'avions pas traité cette tâche avec notre système participant SVM_0, mais l'avons fait a posteriori avec SVM_1. Nous rappelons que pour faire la catégorisation avec notre architecture par transitions, il suffit de décliner toute transition réalisant un MARK en autant de catégories d'EP visées (cf. section 1.2.4). En l'occurrence pour les données PARSEME 1.0, on a quatre catégories (CVS verbes support, CVP verbes à particule, VIR verbes réfléchis intrinsèques et ID pour les idiomes verbaux (cf. Partie 1, section 2.3.1).

Le tableau 2.3 fournit les scores $\mathbf{F_{avg}}(\mathbf{eval-eps})$ obtenus par SVM_1 pour chacune de ces

	CVS	VIR	CVP	ID
%	29.5	43.6	34.8	27.5
SVM_1	**38.3**	**62.0**	**53.7**	**35.0**
PARSEME 1.0	32.9	50.4	39.3	24.9

TABLE 2.3 – **Résultats de catégorisation :** Proportions de chaque catégorie d'EPs (ligne %), et scores $\mathbf{F_{avg}}(\mathbf{eval-eps})$ du modèle SVM_1 et des meilleurs systèmes de la campagne PARSEME 1.0, parmi ceux réalisant la catégorisation en plus de l'identification. Les $\mathbf{F_{avg}}(\mathbf{eval-eps})$ par catégorie sont les moyennes des scores $\mathbf{F}(\mathbf{eval-eps})$ de 13 langues pour CVSs, de 9 langues pour VIRs, de 7 langues pour CVPs et de 11 langues pour IDs, pour lesquelles il y a au moins un système autre que SVM_1 qui donne un résultat (alors que SVM_1 fournit un résultat pour toutes les langues).

catégories, comparés aux meilleurs scores des systèmes ayant participé à la compétition PARSEME 1.0. On peut constater que les meilleures performances du modèle SVM_1 valent pour toutes les catégories, avec pour chacune un écart de plus de 6 points par rapport aux systèmes de PARSEME 1.0. Les systèmes ont relativement plus de difficultés à traiter les expressions idiomatiques (ID) et les constructions à verbe support (CVS), en comparaison avec les constructions à verbe à particule (CVP) et les verbes intrinsèquement réfléchis (VIR). Cela pourrait s'expliquer par le fait que ces deux derniers types d'expressions ont des patrons syntaxiques très contraints, alors que c'est moins le cas pour les autres. Par exemple, un VRI contient obligatoirement un pronom réfléchi et un verbe ; un CVP est constitué d'un verbe et d'une particule. Le tableau montre également que les écarts entre les scores de catégorisation de la variante SVM_1 et ceux des systèmes de PARSEME 1.0 sont plus importants que les écarts des scores d'identification. On ne peut malheureusement pas tirer de conclusions car les systèmes de PARSEME 1.0 ayant participé à la tâche de catégorisation ont généralement de moins bons résultats sur la tâche d'identification que la plupart des autres systèmes. Par exemple, le système RACAI, qui produit les meilleurs scores sur la classe CVS pour huit langues, a un score d'identification $\mathbf{F_{avg}}(\mathbf{eval-eps})$ égal à 37,8 sur les huit langues alors que SVM_1 a un score de 47,2 et les meilleurs systèmes du PARSEME 1.0 (en excluant SVM_0) ont un score de 41,1 sur les mêmes langues.

2.5 Analyse : SVM_1 et système de base

Le tableau 2.4 compare une nouvelle fois les performances en termes de $\mathbf{F}(\mathbf{eval-eps})$ de notre variante SVM_1 et de notre système de base, mais cette fois pour trois classes formelles d'EPs : les EPs continues, discontinues et les TPs. Les deux systèmes sont évalués en validation croisée à 5 plis sur les jeux d'entraînement de toutes les langues de PARSEME 1.0.

Le tableau montre que SVM_1 surpasse largement le système de base sur les EPs continues et discontinues, avec cependant un plus grand écart dans le cas continu (écart de plus de 10 points, versus un peu moins de 6 points pour les discontinues). Ceci peut s'expliquer par le fait que le système de base gère de la même manière les EPs continues et discontinues, alors que les prédictions locales de SVM_1 à partir des quelques éléments ciblés dans la configuration courante peuvent handicaper l'identification des EPs discontinues.

Les performances des deux systèmes sont très variables d'une langue à une autre, mais on peut noter que les langues où le système de base surpasse SVM_1 sont des cas où SVM_1 a une performance très faible. On notera également que le système de base est plus performant que SVM_1 sur toutes les classes (pertinentes) pour le hébreu et l'italien. Nous n'avons pas pu réaliser

Lang.	Toutes		Continues		Discontinues		TPs	
	SVM$_1$	basel.	SVM$_1$	basel.	SVM$_1$	basel.	SVM$_1$	basel.
BG	53.0	48.3	58.6	52.3	16.8	**21.2**		
CS	68.9	60.1	75.7	66.6	60.4	53.3		
DE	47.7	39.9	56.4	51.8	35.8	22.6	63.2	**63.7**
EL	57.3	48.0	64.0	52.1	47.0	40.7		
ES	66.0	61.2	73.3	67.9	46.0	41.8		
FA	81.1	67.3	85.7	72.1	53.3	46.2		
FR	73.7	66.0	82.6	72.1	57.7	53.4		
HE	26.8	**30.0**	30.2	**32.3**	10.2	**19.9**		
HU	83.7	73.7	85.4	77.4	63.0	41.2	89.0	81.2
IT	27.3	**33.7**	30.6	**37.2**	7.9	**13.4**		
LT	21.5	20.7	26.2	23.7	10.5	**14.6**		
MT	7.1	**7.7**	7.6	**8.1**	5.8	5.2		
PL	73.6	70.0	82.5	78.3	42.4	39.8		
PT	67.4	65.2	71.8	71.0	60.9	55.8	65.4	**71.9**
RO	85.9	61.8	87.4	59.1	83.2	67.9		
SV	24.7	6.9	29.5	9.7	0.0	0.0		
SL	33.2	17.3	31.7	14.0	35.3	22.0		
TR	60.1	19.2	65.2	13.7	55.0	31.0		
F_{avg}	53.3	44.3	58.0	47.7	38.4	32.8	54.5	54.2
P_{avg}	64.6	56	66.2	57	57.5	49.9	60.8	64.3
R_{avg}	47.8	44.8	54.4	50	31.8	31.9	53.1	48.4

TABLE 2.4 – **Résultats de SVM$_1$ et du système de base pour différentes classes d'EPs :** scores **F**(**eval** − **eps**) d'identification de **SVM**$_1$ et du système de base (**basel.**) sur **toutes** les EPs, les EPs **continues**, les EPs **discontinues** et les EPs composées d'un seul token (**TPs**) pour toutes les langues. Les deux systèmes sont évalués par validation croisée à 5 plis sur les jeux d'entraînements de PARSEME 1.0. Les cas où le système de base est le plus performant sont en gras.

d'analyse d'erreur spécifique à ces langues pour expliquer cette faible performance.

En termes de précision et rappel, sans surprise le système de base a une moins bonne précision sur toutes les classes d'EPs, mais on constate que le système de base n'a finalement pas un très bon rappel, malgré une technique d'identification assez permissive. En particulier le niveau de rappel pour les EPs discontinues est similaire à SVM$_1$.

Enfin, le tableau montre que le système de base est plutôt compétitif pour la classe des tokens polylexicaux (TPs), avec des résultats comparables en considérant la moyenne sur les trois langues concernées. Par contre, si l'on regarde langue par langue, on remarquera des résultats opposés pour le portugais et le hongrois : un gain de plus de 6 points pour le système de base sur le portugais et un gain d'un peu moins de 8 points pour SVM$_1$ sur le hongrois.

Ceci peut s'expliquer en partie par le fait que les tokens polylexicaux sont moins fréquents pour le portugais que pour le hongrois dans les données d'entraînement : le portugais et le hongrois comptent respectivement moins de 80 et plus de 2000 occurrences de tokens polylexicaux, alors que le corpus du portugais est plus grand que celui du hongrois en termes de tokens et d'annotations. L'apprentissage du modèle SVM sera défavorisé par le relatif manque de données sur les TPs pour le portugais, ce qui favorisera une méthode par règles telle que celle qu'utilise

le système de base.

2.6 Analyse : Impact de patrons de traits

Attribut	Uni	Bi	Tri	**F**	**#Traits (k)**
Forme	+			11.5	80
	+	+		11.6	430
	+	+	+	11.4	440
POS	+			00.0	-
	+	+		17.4	-
	+	+	+	17.2	-
Lemme	+			13.0	60
	+	+		17.9	360
	+	+	+	17.9	370
POS + forme	+			54.0	80
	+	+		65.7	440
	+	+	+	65.2	440
POS + lemme	+			52.5	60
	+	+		**73.6**	600
	+	+	+	73.5	600
Forme + lemme	+			14.5	140
	+	+		16.9	800
	+	+	+	16.8	800
Forme + lemme + POS	+			55.4	140
	+	+		73.4	1.050
	+	+	+	73.4	1.080

TABLE 2.5 – **Impact des patrons de traits lexicaux :** Un tableau de plusieurs expérimentations lancées sur les jeux de données françaises de PARSEME 1.0 (divisées en un jeu d'entraînement 80 % et un jeu d'évaluation 20 %). **Attribut** représente les attributs linguistiques que l'on utilise lors de l'extraction d'unigrammes (**Uni**), de bigrammes (**Bi**) et de trigrammes (**Tri**). Le caractère + indique si le patron de traits est activé. La colonne (**F**) représente le score $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$ du modèle sur le jeu d'évaluation et la dernière colonne # indique le nombre de traits extraits (en milliers).

Dans cette section, nous étudions l'impact des patrons de traits sur notre variante SVM_1, sur les données françaises de PARSEME 1.0. Dans un premier temps, nous évaluons l'impact des différents attributs linguistiques des éléments ciblés de la configuration courante, ainsi que les types de n-grammes considérés dans les patrons de traits. Ainsi, dans cette étude, nous ne considérons que les patrons de traits correspondant à des n-grammes d'éléments ciblés, plus précisément des uni-, bi-, ou trigrammes de différents attributs linguistiques (forme, lemme et étiquette POS) de ces éléments.

Le tableau 2.5 présente les performances de SVM_1 en faisant varier la longueur des n-grammes et les attributs linguistiques utilisés. Chaque ligne du tableau représente une combinaison de patrons de traits ainsi que son score $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$ et le nombre de traits exploités. Le tableau

Alias	**F**	#	**Patrons**
Standard	74.2	1.92	A B C E I J K L
Sans syntaxe	73.3	1.62	A C E I J K L
Sans B_i S_0B_2	74.0	1.43	A B C I J K L
Sans Distance S_0B_0	74.7	1.92	A B C E J K L
Sans Distance S_0S_1	74.2	1.92	A B C E I K L
Sans unigramme B_1	73.6	1.84	A B C E I J L
Sans lexique	74.4	1.92	A B C E I J K

	Patrons
A	Tokens, POS et lemmes
B	Dépendances syntaxiques
C	Bigrammes S_1S_0. S_0B_0. S_1B_0 and S_0B_1
E	bigramme S_0B_2
I	Distance entre S_0 et S_1
J	Distance entre S_0 et B_0
K	B_1
L	$LEX_{tokens(EPs)}$

TABLE 2.6 – **Impact des patrons de traits :** Un tableau donnant les résultats de plusieurs expérimentations lancées sur les données français de PARSEME 1.0 (divisées en jeu d'entraînement 80 % et un jeu d'évaluation 20 %) en utilisant la variante SVM_1. L'**Alias** représente le nom explicatif de la combinaison des patrons de traits exploités ; **F** représente le score $\mathbf{F(eval-eps)}$ du modèle sur le jeu d'évaluation ; # est le nombre de traits extraits (en millions) ; **Patrons** : indique les patrons de traits exploités. Ces patrons sont représentés par une séquence de lettres documentées dans le tableau de droite.

montre d'abord qu'un modèle appris sur les n-grammes composés de formes seulement, de lemmes seulement ou d'étiquettes seulement, produit des résultats très faibles, puisque l'on atteint au mieux un score de 17,9 % avec l'utilisation de bigrammes ou trigrammes de lemmes. Par contre, l'utilisation d'une forme lexicale (forme fléchie ou lemme) combinée avec celle d'une étiquette se montre décisive, avec des gains spectaculaires par rapport au cas précédent : on passe de scores sous les 20 % à des scores au-dessus de 50 %. L'utilisation des trois attributs linguistiques ensemble n'apporte guère de gain par rapport au cas précédent, tout en faisant augmenter le nombre de traits extraits. Par ailleurs, l'utilisation exclusive des unigrammes ne peut pas produire des scores compétitifs, ce qui n'est pas une surprise étant donné le caractère multitoken des EPs en français. C'est l'utilisation combinée des unigrammes et des bigrammes qui montre l'impact le plus fort, alors que l'ajout des trigrammes n'a pas d'impact positif. Enfin, le modèle qui utilise les unigrammes et les bigrammes de lemmes et d'étiquettes produit le score le plus élevé de cette étude (73,6), et est très proche du score obtenu par réglage manuel, avec utilisation de beaucoup plus de traits, en particulier des syntaxiques (73,8).

Le tableau 2.6, à son tour, explore l'impact des autres patrons de traits sur la performance de SVM_1, en décrivant les résultats d'une étude par ablation. Par exemple, la deuxième ligne du tableau représente une expérimentation dont les patrons exploités sont ceux de la grille standard de patrons (*A B C E I J K L*), à l'exception des patrons de traits syntaxiques (voir section 2.2.2, p. 93). Ce tableau montre que l'impact des patrons de traits considérés pour ablation est très léger, en ne dépassant jamais 0,9 point d'écart. Certains patrons du système standard semblent même légèrement détériorer les performances. Notons tout de même que les résultats de ce tableau valent pour le français seulement et que cette tendance devrait être vérifiée sur plus de langues.

2.7 Résumé

Ce chapitre est dédié à la mise au point et à l'utilisation d'un modèle linéaire, de type SVM, pour le classifieur prédisant la transition à appliquer à la configuration courante. Le modèle est nourri par des traits utilisant la forme des tokens, leurs lemmes, leurs étiquettes morphosyn-

taxiques et leurs relations syntaxiques avec d'autres tokens de la phrase. Des traits basés sur des lexiques d'EPs issus des jeux d'entraînement sont également utilisés, ainsi que d'autres traits formels et techniques.

Ce classifieur SVM, utilisé avec le jeu de transitions $\mathcal{T}_1$ forme la variante de notre système que nous avons appelée SVM_1. Toutes les expérimentations décritse dans ce chapitre concerne les données PARSEME 1.0, modifiées pour avoir des annotations morphologiques et syntaxiques prédites et pas gold. Contrairement aux chapitres suivants, nous nous sommes placé dans un scénario où nous nous permettons d'une part d'utiliser des traits morphologiques et syntaxiques (et donc supposant de se placer après analyse syntaxique) pour les langues où ces informations sont disponibles, et d'autre part d'utiliser des combinaisons d'hyperparamètres différentes d'une langue à une autre, avec un réglage manuel des hyperparamètres.

Les résultats produits par cette variante SVM_1 montrent qu'une méthode d'analyse par transitions est très performante pour l'identification des EPs. Tout d'abord, SVM_1 présente des résultats bien meilleurs qu'un système de base s'appuyant sur un lexique lemmatisé des EPs vues dans le corpus l'apprentissage ($F_{avg} = 53,7$ versus $44,8$). Par ailleurs, elle bat assez largement tous les systèmes ayant participé à PARSEME 1.0 en termes de scores globaux, si l'on ne tient pas compte de notre propre variante SVM_0 (variante "ancêtre" de SVM_1, utilisant le jeu de transitions $\mathcal{T}_0$ qui a participé à PARSEME 1.0). Elle est aussi la plus performante sur toutes les langues, à l'exception de l'allemand (en eval-cmpts). À noter que nos deux variantes SVM_0 et SVM_1 ont des scores globaux très comparables bien que SVM_1 soit légèrement meilleure. D'autre part, les performances de notre variante SVM_1 sur la tâche supplémentaire de catégorisation d'EPs confirment les tendances observées pour la tâche d'identification seule.

Enfin, une analyse plus fine de la performance de la variante SVM_1 sur le français, en fonction des patrons de traits utilisés, a montré que les traits lexicaux (unigrammes et bigrammes de lemmes et étiquettes morphosyntaxiques) sont primordiaux pour l'identification. Par contre, les traits syntaxiques, les traits issus des lexiques du jeu d'entraînement, ainsi que les traits formels et techniques ont un faible impact pour cette langue.

Chapitre 3

Perceptron multicouche pour l'identification des EPs

Le domaine du TAL a récemment connu une évolution spectaculaire avec le remplacement quasi systématique des modèles linéaires statistiques classiques par des modèles non linéaires s'appuyant sur des réseaux de neurones. Ce renouveau a conduit à une amélioration notable des performances des outils du TAL, notamment pour l'analyse syntaxique en dépendances par transitions (Chen et Manning, 2014; Kiperwasser et Goldberg, 2016; Andor et al., 2016; Kulmizev et al., 2019a). Les réseaux de neurones ont l'avantage d'être souvent utilisés avec une approche de bout en bout (« end-to-end ») qui évite en théorie de développer des patrons de traits fins et spécifiques à la tâche, contrairement aux modèles statistiques classiques comme SVM (cf. chapitre 2). Par contre, de tels modèles impliquent l'utilisation d'un jeu important d'hyperparamètres qu'il s'agit d'optimiser pour obtenir le modèle le plus performant possible.

Dans ce chapitre, nous proposons de remplacer le classifieur linéaire de notre système d'identification, par un simple perceptron multicouche modélisé sous la forme d'un réseau de neurones. Dans nos évaluations expérimentales, nous utilisons un système par transitions reposant sur le jeu de transitions $\mathcal{T}_2$ qui est le plus abouti. Dans toutes nos expériences, le système est comparé systématiquement avec notre variante SVM_2 reposant sur le même jeu de transitions en utilisant les mêmes conditions expérimentales (i.e. mêmes données, même méthode de réglage des hyperparamètres, etc.). Contrairement au chapitre précédent, une autre différence importante est que nous nous plaçons dans un scénario où nous disposons d'annotations morphologiques (lemmes et étiquettes morphosyntaxiques), mais pas d'annotations syntaxiques.

Ce chapitre commence par décrire la procédure que nous utilisons pour produire les vecteurs denses (ou plongements) des éléments en entrée de notre modèle neuronal (section 3.1). La section 3.2 présente en détail l'architecture de notre perceptron multicouche (ci-après MLP, pour « Multi Layer Perceptron »). La section 3.3 discute des méthodes de ré-échantillonnage que nous utilisons pour équilibrer la distribution des transitions dans les jeux de données d'entraînement et pour stabiliser l'apprentissage de notre réseau de neurones. À noter que, dans les trois sections sus-mentionnées, nous listons systématiquement les hyperparamètres utilisés.

Les autres sections du chapitre sont consacrées aux expériences réalisées et à la présentation et l'analyse des résultats. La section 3.4 présente la méthode aléatoire utilisée lors du réglage des hyperparamètres et compare les combinaisons d'hyperparamètres retenues pour les deux variantes. La section 3.5 présente les résultats d'identification obtenus pour différentes configurations expérimentales : identification des expressions verbales sur les données PARSEME 1.1, avec ou sans initialisation des plongements lexicaux par des vecteurs préentrainés, identification

de tous types d'expressions sur le français et l'anglais en utilisant respectivement les jeux de données FTB et DiMSUM. La section 3.6 décrit une analyse détaillée des résultats. Par exemple, elle fournit une analyse comparative de la performance du MLP et de SVM_2 pour différents types d'EPs : ex. vues vs. non vues, continues vs. discontinues, différentes longueurs, etc. Elle examine aussi l'impact des différentes techniques de ré-échantillonnage, de la taille des corpus d'apprentissage, et également le type de vocabulaire utilisé pour les formes lexicales. Enfin, la section 3.7 propose de combiner les deux variantes comparées tout au long de ce chapitre, en expérimentant un système d'empilement des deux variantes, dans les deux directions. Le principe est l'une des deux variantes bénéficie aussi de la prédiction de l'autre pour prédire la prochaine transition étant donné une configuration courante.

3.1 Plongements

Notre réseau de neurones reçoit en entrée des éléments ciblés de la configuration courante, pour lesquels on considère comme attribut linguistique la forme (fléchie), le lemme et l'étiquette (morphosyntaxique). Ces attributs sont représentés par des plongements (« embeddings » en anglais) grâce à des couches dédiées (ci-après, des couches de plongements). Un plongement est un vecteur (dense) de valeurs réelles, comptant souvent des dizaines ou des centaines de dimensions. Ainsi un plongement de taille **n** correspond à un point d'un espace vectoriel à **n** dimensions. Chaque type d'attribut est associé à une couche de plongements spécifique, i.e. il existe une couche pour les formes des éléments, une autre pour les lemmes, et une dernière pour les étiquettes. Les plongements les plus connus sont les plongements lexicaux associés à des formes lexicales (formes fléchies ou lemmes). La taille de chaque type de plongement sont des hyperparamètres du réseau, et dépend entre autres de la taille du vocabulaire représenté : typiquement les plongements lexicaux ont souvent des centaines de dimensions, alors que les plongements des étiquettes peuvent être limitées à quelques dizaines de dimensions.

Chacune des couches de plongements définit un vecteur dense par élément du vocabulaire (des formes, des lemmes ou des étiquettes). L'ensemble des plongements constituent des paramètres du réseau. Lors de l'apprentissage, ils sont appris en même temps que les paramètres des couches ultérieures. Ils sont initialisés soit aléatoirement soit, dans le cas de plongements lexicaux, avec des vecteurs denses préentrainés, en général de manière non supervisée à partir de très grands corpus de textes. Il existe différents algorithmes d'apprentissage non supervisé. Par exemple, les modèles *word2vec* (Mikolov et al., 2013) utilisent des réseaux de neurones qui apprennent à reconstruire le contexte linguistique de l'élément ciblé. FastText est un autre algorithme, qui représente les mots sous forme de n-grammes de caractères et additionne les vecteurs associés aux n-grammes pour obtenir la représentation complète du mot. Cet algorithme permet d'intégrer davantage de caractéristiques morphologiques à partir des formes lexicales considérées (Mikolov et al., 2013).

En pratique, dans nos réseaux, chaque élément ciblé de la configuration courante est transformé en un plongement pour son étiquette morphosyntaxique, et un plongement lexical : un hyperparamètre détermine si ce sont les formes fléchies ou les lemmes qui sont utilisés pour les plongements lexicaux. Pour **k** éléments ciblés, on obtient $\mathbf{2} * \mathbf{k}$ plongements, qui sont concaténés avant d'être passés à la couche suivante.

3.1.1 Hyperparamètres

Chaque couche de plongements est associée à un jeu d'hyperparamètres : la taille des vecteurs denses, la méthode utilisée pour les initialiser (aléatoirement ou avec des vecteurs préentrainés),

le type d'attributs sélectionnés et le vocabulaire utilisé. Notez que ces hyperparamètres sont partagés pour l'ensemble de nos architectures neuronales qui seront utilisées dans cette thèse.

Attributs : L'hyperparamètre LEMMATISATION détermine si les plongements lexicaux doivent représenter les lemmes des éléments (valeur TRUE) ou leurs formes fléchies (valeur FALSE).

Dimensions : DIMENSION DE TOKEN et DIMENSION DE POS fournissent les dimensions des plongements lexicaux (forme fléchie ou lemme) et des plongements des étiquettes morphosyntaxiques.

Initialisation : L'hyperparamètre PRÉENTRAÎNÉ détermine si les plongements lexicaux sont initialisées par des plongements préalablement appris de manière externe (valeur TRUE) ou s'ils sont initialisés aléatoirement [39] (valeur FALSE). Les plongements des étiquettes sont systématiquement initialisés aléatoirement.

Étant donné que les unités de la pile peuvent contenir plusieurs tokens, avec une structure d'arbre binaire, nous avons deux possibilités pour les représenter dans nos réseaux : (1) comme des éléments atomiques (i.e. indécomposables) ou bien (2) comme des structures arborées. Nous avons opté pour la première solution : dans nos réseaux une unité est un élément atomique dont chacun des attributs est constitué de la concaténation (dans l'ordre séquentiel) des attributs de même type des composants de l'unité (i.e. les tokens de l'unité). Par exemple, l'unité correspondant à l'expression *fait face* aura pour forme *fait_face*, pour lemme *faire_face* et pour étiquette *VERB_NOUN*. L'hyperparamètre MOYENNAGE détermine si les valeurs initiales des plongements lexicaux associés aux unités multitokens de la pile sont les moyennes (valeur TRUE) ou les sommes (valeur FALSE) des plongements lexicaux des nœuds de l'arbre.

Vocabulaire : Un hyperparamètre que nous appelons VOCAB. COMPACT détermine deux modes pour le choix du vocabulaire des éléments lexicaux utilisés dans nos modèles.

La stratégie *vocabulaire exhaustif* (valeur FALSE pour l'hyperparamètre) considère que tous les tokens apparaissant au moins 2 fois dans le corpus d'entraînement appartiennent au vocabulaire. Si le token ne se produit qu'une seule fois et n'appartient à aucune EP, une condition probabiliste ($\alpha > 0,5$) déterminera s'il sera retenu dans le vocabulaire ou non. Tous les tokens appartenant à des EPs du jeu d'entraînement sont retenus dans le vocabulaire. Pour les unités multitokens de la pile qui sont créées lors de l'apprentissage, leur concaténation est également intégrée au vocabulaire. Par exemple, la forme *pomme_de* est ajoutée au vocabulaire quand l'unité correspondante a été créée et utilisée pour identifier l'expression *pomme de terre* lors de la phase d'entraînement.

La stratégie *vocabulaire compact* (valeur TRUE) réduit drastiquement la taille du vocabulaire, puisqu'elle ne retient que les tokens qui se produisent dans au moins une EP du jeu d'entraînement. Pour les unités multitokens, le vocabulaire compact suit la même stratégie que celle du vocabulaire exhaustif. Cette stratégie de réduction agressive a été expérimentée parce qu'elle réduit la taille de la matrice de plongements lexicaux, ce qui peut faciliter l'apprentissage en réduisant le nombre de paramètres du modèle.

Notons que les entrées du vocabulaire des éléments lexicaux, qu'il soit compact ou exhaustif, sont constituées soit des formes soit des lemmes des éléments ciblés, selon la valeur de l'hyperparamètre LEMMATISATION. Par ailleurs, le vocabulaire comprend trois entrées fictives supplémen-

39. L'initialisation aléatoire que nous adoptons génère des valeurs réelles entre 0.5 et -0.5 selon une distribution uniforme.

taires afin de représenter **(1)** les éléments vides de la configuration (par exemple, quand la pile a 0 ou 1 élément, ou que le tampon est vide), **(2)** les éléments inconnus représentant les éléments dont la valeur lexicale (forme ou lemme) est considérée comme inconnue, et **(3)** les éléments numériques [40].

Le vocabulaire des étiquettes morphosyntaxiques est l'ensemble des étiquettes morphosyntaxiques des tokens se trouvant dans les phrases d'entraînement. Il inclut aussi les étiquettes obtenues par concaténation pour des unités multitokens de la pile. Il contient aussi des entrées fictives pour les éléments vides et les étiquettes inconnues.

Généralisation du vocabulaire : Notre réseau de neurones associe une forme lexicale à un plongement lexical. Or, dans notre cas, le vocabulaire des formes lexicales comprend des concaténations de formes ou lemmes pour les unités multitokens, mais uniquement celles vues à l'apprentissage. Ceci peut limiter le pouvoir de généralisation du modèle pour identifier des EPs non vues dans le corpus d'entraînement. Il se trouve que certains composants d'EPs sont très fréquemment utilisés au sein du lexique des EPs du corpus d'entraînement. Les constructions à verbe support en *faire* sont par exemple très nombreuses : *faire un choix*, *faire un pari*, *faire une promenade*, *faire un pas*, ...

À des fins de généralisation, il peut être intéressant de remplacer les formes des unités multitokens par des formes fictives correspondant à un ensemble d'unités. Par exemple, les constructions à verbe support en *faire* citées ci-dessus pourraient être associées à une même forme fictive *faire_ JOKER* où JOKER représente n'importe quel token. Avec cet hyperparamètre, nous avons donné la possibilité d'ajouter dynamiquement des formes fictives au vocabulaire. Nous décrivons ci-dessous son fonctionnement en phase d'apprentissage, puis en phase d'analyse.

En phase d'apprentissage, l'étape préliminaire est de constituer un ensemble de formes fictives à ajouter au vocabulaire. Nous utilisons pour cela une fonction **g** prenant en entrée une unité multitoken des données d'entraînement et renvoie sa forme fictive si elle existe, ou sa forme fléchie normale autrement. Elle fonctionne comme suit : la fonction **g** remplace par JOKER tout composant dont la forme n'apparaît pas comme composant d'une EP d'entraînement plus d'un certain seuil (Seuil pour joker=5) (à moins d'obtenir uniquement des JOKERs, auquel cas **g** rend la forme de départ).

Une fois ce vocabulaire augmenté de formes fictives, il est utilisé de manière différente en phase d'apprentissage et en phase d'analyse. En phase d'apprentissage, pour un exemple (configuration, transition) lorsque $\mathbf{S_0}$ ou $\mathbf{S_1}$ donnent lieu à un remplacement par JOKER par la fonction **g**, on utilise d'abord une fois l'exemple d'apprentissage avec formes inchangées, puis si une condition probabiliste ($\beta > 0,5$) se réalise, on réutilise ce même exemple mais en remplaçant les formes normales par les formes fictives.

En phase d'analyse, étant donnée une unité multitoken à traiter, si la forme classique existe dans le vocabulaire, c'est celle-ci qui est choisie avec son plongement associé, car ayant déjà été vue, cela ne sert à rien d'activer le vocabulaire dédié à la généralisation (source d'erreurs). Sinon, si la fonction **g** renvoie une forme fictive qui a été vue à l'apprentissage, celle-ci est sélectionnée ainsi que son plongement associé. Si l'on ne rentre dans aucun des deux cas précédents, l'unité est considérée comme inconnue.

40. Cette dernière entrée fictive n'est utilisée qu'avec les jeux de données comprenant des EPs non verbales, puisque les expressions numériques sont extrêmement marginales dans les jeux de données PARSEME 1.0 et PARSEME 1.1.

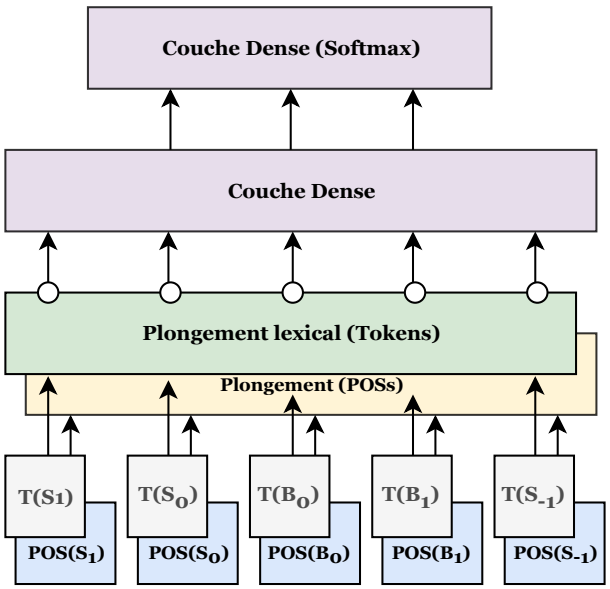

FIGURE 3.1 – **Topologie du MLP :** MLP simple pour l'identification des EPs. Les couches lexicales (couches de plongements) sont alimentées par les formes lexicales des représentations atomiques **T** et les étiquettes morphosyntaxiques **POS** des éléments ciblés $\mathbf{S_1}, .., \mathbf{S_{-1}}$ de la configuration. Cette architecture correspond au cas avec une seule couche cachée.

3.2 Perceptron multicouche (MLP)

Les réseaux de neurones entièrement connectés à propagation avant sont des modèles non linéaires qui peuvent être utilisés à la place des modèles linéaires pour la classification multiclasse ainsi que pour d'autres tâches. La non linéarité est obtenue par l'utilisation d'une fonction d'activation non linéaire en sortie de la ou les couches denses internes au réseau. La non linéarité du réseau, ainsi que la possibilité d'intégrer facilement des plongements, conduisent souvent à une amélioration significative des performances pour l'analyse syntaxique en dépendances (Chen et Manning, 2014), pour le traçage de l'état des dialogues (« dialog state tracking ») (Henderson et al., 2013), pour l'apprentissage des modèles de langue (Bengio et al., 2003), etc.

Un MLP est un réseau neuronal à propagation avant, traditionnellement vu comme composé de plusieurs « couches », chacune prenant un vecteur d'entrée, appliquant une transformation linéaire puis une fonction d'activation non linéaire.

Notre premier modèle neuronal d'identification des EPs s'appuie sur un MLP à une ou deux couches cachées. Il possède une couche de plongements pour les étiquettes des éléments ciblés (i.e. par défaut $\mathbf{S_0}$, $\mathbf{S_1}$ et $\mathbf{B_0}$, plus $\mathbf{B_1}$), et une couche de plongements pour leurs formes ou leurs lemmes. Les deux couches sont concaténées et entièrement connectées à une couche dense munie d'une fonction d'activation *ReLu*. Dans le cas d'une seule couche cachée, cette couche est connectée à la couche de sortie, utilisant la fonction d'activation *Softmax* (voir figure 3.1) qui produit une distribution de probabilités pour les classes associées aux transitions étant donné la configuration courante, et sert ensuite à prédire la transition la plus probable.

On peut détailler la fonction mathématique représentée par le réseau de la manière suivante. Notons $\mathbf{c_1}...\mathbf{c_n}$ les $\mathbf{n}$ éléments ciblés, $\mathbf{L}$ la taille du vocabulaire lexical, et $\mathbf{E}$ la taille du vocabulaire d'étiquettes, $\mathbf{n_l}$ et $\mathbf{n_e}$ la taille des plongements lexicaux et d'étiquettes, $\mathbf{P^l}$ et $\mathbf{P^e}$ les matrices des plongements lexicaux et d'étiquettes. On note $\mathbf{o^l_{c_i}}$ le vecteur de taille $\mathbf{L}$, dit « one-hot »,

comprenant des zéros sauf à une dimension correspondant au numéro de la forme de $\mathbf{c_i}$, de telle sorte que $\mathbf{p^l_{c_i}} = \mathbf{o^l_{c_i}}.\mathbf{P^l}$ donne le plongement lexical de c_i. On note de la même manière $\mathbf{o^e_{c_i}}$ de telle sorte que $\mathbf{p^e_{c_i}} = \mathbf{o^e_{c_i}}.\mathbf{P^e}$ donne le plongement d'étiquette de $\mathbf{c_i}$. Si on note $[\mathbf{x};\mathbf{y}]$ la concaténation de deux vecteurs $\mathbf{x}$ et $\mathbf{y}$, alors après passage dans les couches de plongement, on obtient par concaténation :

$$\mathbf{h_0} = [\mathbf{p^e_{c_1}}; \mathbf{p^e_{c_2}}; ...\mathbf{p^e_{c_n}}; \mathbf{p^l_{c_1}}; ...; \mathbf{p^l_{c_n}}]$$

Ce vecteur $\mathbf{h_0}$ obtenu en sortie des couches de plongement a comme taille $\mathbf{t} = \mathbf{n} \times (\mathbf{n_l} + \mathbf{n_e})$. Le passage dans la couche dense donne :

$$\mathbf{h_1} = \mathbf{ReLU}(\mathbf{h_0}.\mathbf{W^1} + \mathbf{b^1})$$

Enfin, on obtient en sortie :

$$\mathbf{y} = \mathbf{Softmax}(\mathbf{h_2}.\mathbf{W^2} + \mathbf{b^2})$$

Les paramètres du réseau sont :

— $P^l \in \mathbb{R}^{L \times n_l}, P^e \in \mathbb{R}^{E \times n_e}$

— $W^1 \in \mathbb{R}^{t \times n_d}, b^1 \in \mathbb{R}^{n_d}$

— $W^2 \in \mathbb{R}^{n_d \times n_o}, b^2 \in \mathbb{R}^{n_o}$

Cette formulation mathématique omet la technique du « dropout » en sortie de la couche cachée : une certaine proportion de dimensions du vecteur $\mathbf{h_1}$ sont sélectionnées aléatoirement et ignorées lors du passage à la couche suivante. Cette technique a montré ses preuves pour limiter le problème du sur-apprentissage. D'autre part, il y a un hyperparamètre pour déterminer si on ajoute une couche cachée supplémentaire, entre la première couche cachée et la couche de sortie.

Un MLP pour une tâche de classification multiclasse s'apprend de façon supervisée en utilisant généralement un algorithme de type gradient stochastique (SGD) ou l'une de ses variantes. Dans notre cas, nous utilisons systématiquement l'algorithme AdaGrad, qui améliore la robustesse de SGD en utilisant un taux d'apprentissage [41] adaptatif.

3.2.1 Hyperparamètres

Éléments en entrée : Par défaut, les éléments ciblés de la configuration sont $\mathbf{S_0}$, $\mathbf{S_1}$ et $\mathbf{B_0}$. On y ajoute le deuxième token du tampon (i.e. $\mathbf{B_1}$) si l'hyperparamètre UTILISER B_1 est actif, et le dernier élément ayant subi un REDUCE, noté $\mathbf{S_{-1}}$, si l'hyperparamètre UTILISER S_{-1} est actif. Dans le cas où $\mathbf{S_{-1}}$ est multitoken, on n'utilise que son dernier token. Ces deux hyperparamètres permettent d'étendre le contexte utilisé pour l'identification des EPs.

Apprentissage : L'hyperparamètre TAUX D'APPRENTISSAGE détermine la valeur initiale du taux d'apprentissage adaptatif de l'optimisateur (Adagrad). L'hyperparamètre TAILLE DE BATCH détermine le nombre d'exemples utilisés pour une itération de l'apprentissage stochastique (la perte est calculée sur un « mini-batch » d'exemples) utilisée lors de l'optimisation. Nous utilisons aussi une mesure d'arrêt anticipé (« Early stopping ») pour surveiller la perte de l'optimisateur et arrêter l'apprentissage s'il n'évolue pas. Notez que la fonction de l'entropie croisée est la fonction de perte utilisée lors de l'optimisation de ce réseau, comme pour tous les autres réseaux de

41. Le taux d'apprentissage est un hyperparamètre correspondant au facteur multiplicatif appliqué au gradient lors de la mise à jour des paramètres.

neurones proposés dans cette thèse.

Hyperparamètres divers : Nous utilisons, en plus, d'autres hyperparamètres classiques. L'hyperparamètre UTILISER UNE DEUXIÈME COUCHE CACHÉE détermine si le modèle doit comprendre une couche cachée (valeur FALSE) ou deux couches cachées (valeur TRUE). Chaque couche est associée à plusieurs hyperparamètres : NOMBRE DE NEURONES et FONCTION D'ACTIVATION définissent la dimension et la fonction d'activation de la couche. Les fonctions d'activation testées sont *tanh* et *ReLu*. DROPOUT définit la proportion de neurones qui sont ignorés en sortie d'une couche cachée.

3.3 Ré-échantillonnage

Langue	Phrases	SHIFT	REDUCE	MERGE	MARK
BG	27	49.7	46.7	0.8	2.9
FR	23	53.5	50.6	0.7	2.8
PT	18	60.5	57.0	0.7	3.4
TR	27	40.8	37.9	0.8	2.8

TABLE 3.1 – **Proportion de phrases contenant une EP verbale et distribution des transitions :** Les proportions des phrases contenant des EPs verbales (colonne **Phrases**) et des transitions dans le jeu d'entraînement pour quelques langues de PARSEME 1.1 en utilisant l'oracle de $\mathcal{T}_2$. Les valeurs correspondent à des pourcentages.

Des tests préliminaires sur les jeux de données PARSEME 1.1 pour notre perceptron multicouche ont produit une perte très instable lors de l'apprentissage et des performances très médiocres sur les données de développement. Pour identifier le ou les problèmes, nous avons commencé par étudier la distribution des phrases contenant au moins une EP verbale, ainsi que la distribution des transitions appliquées en utilisant l'oracle pour le jeu de transitions $\mathcal{T}_2$ pour quelques langues. Nous constatons que la fréquence des tokens ordinaires (ceux qui ne font pas partie d'une EP verbale) est beaucoup plus élevée que la fréquence, très basse, des EPs verbales (un ratio d'une EP verbale annotée tous les 75 tokens environ, dans les jeux d'entraînements de PARSEME 1.1). De ce fait, les transitions responsables de l'identification des EPs sont très rares et la distribution des transitions est très déséquilibrée (voir tableau 3.1). Nous avons identifié ce déséquilibre comme un problème central pour l'apprentissage de notre perceptron multicouche, et avons testé un certain nombre de stratégies de ré-échantillonage pour rééquilibrer la distribution des transitions.

Les méthodes de ré-échantillonnage visant à équilibrer la répartition des classes (les transitions dans notre cas) sont réputées être efficaces dans un tel cas (Chawla, 2009), qu'il s'agisse de sous-échantillonnage ou de sur-échantillonnage. Ces méthodes présentent cependant des inconvénients. La méthode de sous-échantillonnage aléatoire peut potentiellement supprimer certains exemples importants et le sur-échantillonnage aléatoire peut conduire à un sur-apprentissage. Chawla et al. (2002) proposent une combinaison de sur-échantillonnage de la classe minoritaire et de sous-échantillonnage de la classe majoritaire. Le sur-échantillonnage de la classe minoritaire implique notamment la création d'exemples synthétiques. He et al. (2008) utilisent une pondération pour différents exemples des classes minoritaires en fonction de leur niveau de difficulté à apprendre : davantage de données synthétiques sont générées pour les classes minoritaires les plus difficiles à

apprendre que pour les exemples de classes minoritaires les plus faciles à apprendre.

Nous avons donc utilisé plusieurs méthodes de ré-échantillonnage : une stratégie de sous-échantillonnage consistant à supprimer les phrases ne comprenant aucune EP. Cette stratégie réduit drastiquement le jeu d'entraînement : par exemple, pour les quatre langues du tableau 3.1, entre 73 % et 82 % des phrases sont exclues. Une autre méthode de sous-échantillonnage a été expérimentée au niveau des transitions cette fois-ci. Elle consiste à ignorer agressivement à l'apprentissage toutes les transitions responsables du traitement des tokens « non importants », i.e. n'appartenant à aucune EP du jeu d'entraînement. Cette méthode réduit également massivement la taille des données d'entraînement. Un corpus de $\mathbf{n}$ tokens, contenant $\mathbf{m}$ tokens importants (appartenant à $\mathbf{k}$ EPs), doit produire un jeu d'entraînement de $\mathbf{2} * \mathbf{n} + \mathbf{k}$ paires (configuration, transition) sans sous-échantillonnage pour le jeu de transitions $\mathcal{T}_\mathbf{2}$, alors que l'application du sous-échantillonnage à base de transitions réduit la taille du jeu de données à $\mathbf{2} * \mathbf{m} + \mathbf{k}$ où $\mathbf{m}$ représente moins de 3 % de $\mathbf{n}$ pour les données de PARSEME 1.1.

Nous avons également expérimenté la méthode de suréchantillonnage aléatoire qui impose une distribution uniforme à toutes les classes en multipliant aléatoirement des instances de classes minoritaires. Suite à des tests préliminaires, nous avons rapidement constaté que l'apprentissage devient stable en combinant **(1)** un sous-échantillonnage au niveau des phrases, supprimant toute phrase ne contenant pas d'EP ; **(2)** un suréchantillonnage aléatoire au niveau des transitions qui impose une distribution uniforme à toutes les transitions. Nous avons abandonné la méthode de sous-échantillonnage agressif.

Nous avons ensuite systématiquement utilisé cette méthode pour toutes les architectures neuronales que nous avons proposées. Testée a posteriori sur le SVM avec la meilleure combinaison d'hyperparamètres précédemment obtenue, elle a conduit à une baisse légère de la performance.

3.3.1 Hyperparamètres

En plus de la méthode de ré-échantillonnage hybride que nous venons de citer, utilisée dans toutes nos expériences, nous avons exploré deux méthodes complémentaires, dont l'utilisation est contrôlée par des hyperparamètres.

Le Suréchantillonnage avec seuil de fréquence vise à simuler un nombre minimum d'occurrences pour toutes les EPs. Quand cet hyperparamètre est activé, les instances d'entraînement pour les transitions responsables de l'identification des EPs sont dupliquées pour les EPs du corpus d'apprentissage jusqu'à atteindre à un seuil de fréquence $\mathbf{K}$ dans le cas où leur fréquence est inférieure à ce seuil (hyperparamètre Seuil de fréquence minimale). Étant donné que chaque EP de $\mathbf{n}$ tokens a besoin de $\mathbf{n}$ Shift, $\mathbf{n-1}$ Merge, un Mark et un Reduce, nous dupliquons toutes ces transitions $\mathbf{K-r}$ fois où $\mathbf{r}$ est la fréquence de l'EP dans le jeu d'entraînement.

L'hyperparamètre Multiplication de perte pénalise le modèle lorsqu'il ne parvient pas à prédire un Merge ou un Mark, en multipliant la perte par un Coefficient empirique. Cette mesure n'est pas à proprement parler du ré-échantillonnage, mais plutôt une alternative à celui-ci.

3.4 Réglage des hyperparamètres

Nous présentons maintenant la méthode et les résultats du réglage des hyperparamètres pour le perceptron multicouche (MLP) sur les données de développement de PARSEME 1.1 à la fois pour la « closed track » (sans données externes) et l'« open track » (avec données externes). Dans la suite de ce document, nous utilisons les notations MLP_c et MLP_o pour le

MLP en configurations closed track et open track respectivement. Dans notre cas, nous avons expérimenté l'open track en utilisant des plongements lexicaux préentrainés pour l'initialisation des plongements lexicaux au moment de l'apprentissage, au lieu d'une initialisation aléatoire. Nous avons également réglé les hyperparamètres de la variante SVM_2 avec la même méthode, mais pour la « closed track » uniquement, afin de pouvoir comparer les deux variantes dans des conditions semblables. Nous considérons donc trois cas : MLP_c et SVM_2 en closed track, et MLP_o en open track.

3.4.1 Méthode de réglage des hyperparamètres

Pour chacun des trois cas cités ci-dessus, nous avons cherché à sélectionner la même combinaison d'hyperparamètres pour toutes les langues, afin de favoriser une certaine robustesse trans-langue des hyperparamètres sélectionnés. Le nombre d'hyperparamètres est assez important, il y avait donc un enjeu à organiser au mieux les expériences de réglage. Pour réduire le temps de réglage, nous choisissons de travailler sur trois langues pilotes, issues de trois familles linguistiques différentes.

Par ailleurs, comme les différents jeux d'entraînement ont des tailles différentes, nous essayons de neutraliser cette variation en utilisant des jeux d'entraînement à taille fixe et égale à la moyenne du nombre de tokens de toutes les langues de PARSEME 1.1 (=270 milles tokens). Les trois langues les plus conformes à ces critères sont : le bulgare pour les langues slaves ; le portugais pour les langues romanes ; et le turc pour les autres familles linguistiques. Par conséquent, pour effectuer ce réglage, pour ces trois langues, nous utilisons des données d'entraînement tronquées à cette taille moyenne pour l'apprentissage, et leurs données complètes de développement pour l'évaluation des différentes combinaisons d'hyperparamètres.

Recherche d'hyperparamètres. Au début de nos expérimentations sur le perceptron multicouche sur les données PARSEME 1.1 d'expressions verbales, nous nous sommes plongés dans une étape de réglage manuel préliminaire des hyperparamètres du modèle. Cette étape manuelle s'est révélée peu efficace, à cause du grand nombre d'hyperparamètres en comparaison avec les modèles linéaires, et à cause des limitations en termes de budget computationnel et de temps d'expérimentation que permet la thèse. Nous avons alors décidé de nous orienter vers une stratégie par recherche aléatoire. L'étape préliminaire nous a cependant permis de fixer une fois pour toutes certains hyperparamètres :

— Utiliser une deuxième couche cachée : *False* ;
— Fonction d'activation : *Relu* ;
— Suréchantillonnage aléatoire (rééquilibrage des classes) : True ;
— Sous-échantillonnage (phrases avec EPs uniquement) : True ;
— Optimisateur : *AdaGrad* ;
— Arrêt anticipé de l'apprentissage (Early stop) : *True*.

Bergstra et Bengio (2012) montrent que l'on peut obtenir de meilleurs modèles, à budget computationnel constant, en utilisant une recherche aléatoire plutôt qu'une recherche par quadrillage (i.e. qui teste chaque combinaison possible), car cela permet de rechercher dans un espace d'hyperparamètres plus grand : on peut utiliser des plages de valeurs un peu plus grandes, et c'est compensé par une non exhaustivité. Nous utilisons donc la méthode de recherche aléatoire pour optimiser les hyperparamètres sur les trois langues pilotes. Plus précisément, nous lançons mille expériences sur les trois langues pilotes (le bulgare, le portugais et le turc) pour nos trois cas de

	Hyperparamètre	Plage	Dis.	CPP_c	CPP_o	CT_c	CT_o
Basique	Lemmatisation	{T, F}	U	T	T	T	T
	Utiliser B_1	{T, F}	U	T	T	T	T
	Utiliser S_{-1}	{T, F}	U	T	F	T	T
Plongements	Dimension de token	[100, 600]	L	157	300	300	300
	Dimension de POS	[15, 150]	L	147	132	35	57
	Preentraîné	{T, F}	U	F	T	T	T
	Modifiable	{T, F}	L	T	T	T	T
	Moyennage	{T, F}	L	F	T	T	T
	Vocab. compact	{T, F}	U	T	F	T	T
	Généralisation	{F, T}	L	F	F	F	F
Dense	# neurones	[25, 600]	L	85	56	75	157
	Dropout	[.1,.., .6]	U	.3	.1	.4	.4
Ré-éch.	Ré-éch. avec seuil de fréq.	{T, F}	U	F	F	F	F
	Seuil de fréquence min.	{5, .., 30}	U	-	-	-	-
	multiplication de perte	{T, F}	U	F	F	F	F
	Coefficient	[1, 40]	L	-	-	-	-
App.	Taux d'apprentissage	[.01, .2]	L	.017	.095	.03	.04
	Taille de batch	{16,.., 128}	U	128	16	48	56
F_g				61.2	57.8	64.3	63.5

TABLE 3.2 – **MLP : Réglage des hyperparamètres** Les hyperparamètres du MLP avec : leurs plages de valeurs (**Plage**) ; la distribution utilisée pour la génération des valeurs (**Dis**) ; la combinaison d'hyperparamètres la plus performante lors du réglage du closed track sur les trois langues pilotes $\mathbf{CPP}_c$; celle du réglage de l'open track sur les mêmes langues $\mathbf{CPP}_o$; la combinaison d'hyperparamètres de tendances de la closed track et celle de l'open track (respectivement $\mathbf{CT}_c$ et $\mathbf{CT}_o$). La distribution de génération des valeurs peut être uniforme (U) ou suivant la loi log-normale (L). La plage complète de la taille de batch est {16, 32, 64, 96, 128} et celle du seuil pour joker est {5, 10, 15, 20, 25, 30}. La dernière ligne indique les scores $\mathbf{F_g}$ du modèle MLP, appris sur les jeux d'entraînement et évalué sur les jeux de développement de toutes les langues de PARSEME 1.1. Pour faciliter la lisibilité du tableau, nous remplaçons certains mots par leurs premiers caractères tels que **T**rue, **F**alse, **Dis**tribution, **App**rentissage, **Ré-éch**antillonnage.

figure (MLP_c et SVM_2 en closed track, et MLP_o en open track). Pour le MLP, chaque expérience est lancée deux fois avec une graine d'initialisation aléatoire différente à chaque fois (= 0 et 1). Pour le SVM, la méthode aléatoire utilise une distribution uniforme pour sélectionner les valeurs des hyperparamètres, qui sont par ailleurs des valeurs booléennes (i.e. un patron de traits est activé ou pas). Pour le MLP, la valeur d'un hyperparamètre peut être issue d'un ensemble de valeurs discrètes en utilisant une distribution uniforme ou d'une plage de valeurs continues en utilisant une distribution log-normale.

Sélection des combinaisons d'hyperparamètres. Dans le cadre de la recherche aléatoire d'hyperparamètres sur les trois langues pilotes, nous avons utilisé deux stratégies pour sélectionner la combinaison gagnante du réglage. La première, classique, consiste simplement à sélectionner la combinaison d'hyperparamètres la plus performante (CPP), c'est-à-dire, celle qui réalise le meilleur score $\mathbf{F_{avg}}$ sur les trois langues pilotes pour les deux graines. Cependant, nous avons constaté, pour le MLP notamment, que **(1)** les valeurs de nombreux hyperparamètres varient

Hyperparamètre	Type	CPP	CT
Unigrammes à base de Lemme et de POS (S_0, S_1, B_0)	Pré	+	+
Bigrammes $S_0S_1, S_0, B_0, S_0, B_1, S_1, B_0$	Pré	+	+
S_0 dans le dictionnaire de TP	Pré	+	+
Ré-échantillonnage	Pré	-	-
Utilisation des formes fléchies dans les n-grammes	Alé	-	-
Trigramme $S_1S_0B_0$	Alé	+	-
Bigramme S_0B_2	Alé	+	-
Historique de transition (longueur 1)	Alé	+	+
Historique de transition (longueur 2)	Alé	-	+
Historique de transition (longueur 3)	Alé	+	-
Distance entre S_0 et S_1	Alé	+	+
Distance entre S_0 et B_0	Alé	+	+
Unigramme B_1	Alé	+	-
Dictionnaire des composants des EPs	Alé	+	-
Longueur de la pile	Alé	-	-
$\mathbf{F_g}$		60.7	58.5

TABLE 3.3 – **SVM_2 : Réglage des hyperparamètres** Résultat du réglage des hyperparamètres de la variante SVM_2 en utilisant un préréglage manuel puis un réglage par recherche aléatoire. Tous les hyperparamètres sauf le dernier sont binaires, i.e. dés/activation d'un ou plusieurs patrons de traits lors de l'extraction des traits. Pour chaque hyperparamètre, la colonne **Type** indique si l'hyperparamètre est soumis au réglage par recherche aléatoire **Alé** ou si sa valeur est fixe suite à l'étape de préréglage **Pré**. Les deux dernières colonnes déterminent la valeur de l'hyperparamètre dans la combinaison d'hyperparamètres la plus performante **CPP** et la combinaison d'hyperparamètres de tendances **CT**. La dernière ligne indique les scores $\mathbf{F_g}$ du modèle SVM_2, appris sur les jeux d'entraînement et évalué sur les jeux de développement de toutes les langues de PARSEME 1.1.

beaucoup parmi les combinaisons les plus performantes et **(2)** que la combinaison la plus performante ne reflète pas toujours les tendances observées dans les combinaisons concurrentes les plus performantes.

Nous avons donc décidé d'explorer la construction d'une « combinaison de tendances » (CT), où la valeur de chaque hyperparamètre correspond à la tendance observée pour cet hyperparamètre parmi les **k** meilleures combinaisons (avec $\mathbf{k} = 250$). Plus précisément, nous choisissons les valeurs en utilisant la méthode de vote majoritaire pour les hyperparamètres à plage discrète et la moyenne des valeurs des **k** premières combinaisons pour les hyperparamètres à valeur continue.

En conséquence, à l'issue du réglage, chacun des cas de figure (MLP_c et SVM_2 en closed track, et MLP_o en open track) est associé à deux combinaisons d'hyperparamètres (CPP et CT). Nous avons donc 6 configurations au total. Celles-ci sont ensuite entraînées sur les données d'entraînement complètes de PARSEME 1.1 pour toutes les langues en utilisant deux graines (0 et 1). Enfin, nous sélectionnons la meilleure des graines pour chaque langue individuellement à partir des résultats sur le corpus de développement.

3.4.2 Résultats du réglage des hyperparamètres

Nous examinons maintenant les résultats obtenus suite au réglage des hyperparamètres tel que nous venons de le présenter, donnés dans les tableaux 3.2 et 3.3 pour le MLP et SVM_2 respectivement.

Résultats du réglage des hyperparamètres du MLP : Pour le MLP, le tableau 3.2 indique les valeurs des hyperparamètres qui ont été sélectionnées pour chacune des 4 configurations (closed ou open track, sélection CPP ou CT). Le tableau ne comprend pas les hyperparamètres fixés suite aux expériences préliminaires : utilisation d'une seule couche cachés, fonction d'activation *Relu*, et ré-échantillonnage hybride (phrases avec EPs uniquement et rééquilibrage aléatoire des nombres d'exemples d'apprentissage de chaque transition).

Le tableau donne aussi le score $\mathbf{F_g}$ obtenu par la combinaison d'hyperparamètres sélectionnée pour chacune de ces configurations sur les données de développement de PARSEME 1.1. Nous pouvons observer que la combinaison de tendances (CT) bat largement la combinaison la plus performante issue de la recherche aléatoire (CPP), à la fois pour la closed track (3,1 points d'écart) et l'open track (5,7 points). Ceci montre que choisir des valeurs d'hyperparamètres indépendamment les unes des autres peut être une bonne stratégie si l'on choisit des valeurs robustes en utilisant les **k** combinaisons les plus performantes au lieu d'une seule.

Pour chaque hyperparamètre, le tableau montre la plage de valeurs possibles et les valeurs retenues pour les différentes configurations CPP_c, CPP_o, CT_c, CT_o. Nous noterons en particulier que certaines valeurs d'hyperparamètres sont sélectionnées pour toutes ces configurations :

— utiliser $\mathbf{B_1}$ (i.e. le deuxième élément du tampon) dans les vecteurs d'entrée,

— utiliser les lemmes au lieu des formes fléchies

— permettre à la rétropropagation de modifier les plongements des lexicaux lors de l'apprentissage

Nous constatons également le petit nombre de neurones nécessaire pour la couche dense. Par ailleurs, les méthodes complémentaires de ré-échantillonage se sont révélées inutiles (seule la méthode hybride de ré-échantillonage est utile).

Résultats du réglage des hyperparamètres de SVM_2 : Le tableau 3.3 présente les résultats du réglage des hyperparamètres de la variante SVM_2. Les hyperparamètres correspondent à quels patrons de traits sont activés (comme décrit au chapitre 2, section 2.2). Notons que nous avons fixé au préalable la valeur de quelques hyperparamètres à la suite d'une étape de préréglage manuel. En particulier, comme vu au chapitre 2, le ré-échantillonnage est désactivé à cause de l'impact négatif de cette technique sur les langues pilotes. Par ailleurs, nous avons fixé l'utilisation d'unigrammes et de bigrammes à base de lemmes et d'étiquettes morphosyntaxiques (cf. 2.2.1).

Les résultats montrent que contrairement au MLP, c'est ici la combinaison des hyperparamètres la plus performante (CPP) qui donne de meilleurs résultats globaux, et pas la combinaison des tendances (CT) (2,2 points d'écart de $\mathbf{F_g}$). Cette différence par rapport au MLP pourrait signifier que le nombre d'expériences pour la recherche aléatoire est suffisant pour le cas du modèle linéaire, mais pas pour le MLP, et que la stratégie par tendances est avantageuse dans le cas d'un budget computationnel limité.

Nous constatons aussi que les combinaisons CT et CPP du réglage éliminent les n-grammes utilisant des formes fléchies, ce qui réduit énormément le nombre de traits et montre l'importance et l'efficacité des lemmes pour les traits lexicaux. Par ailleurs, la longueur de la pile se révèle

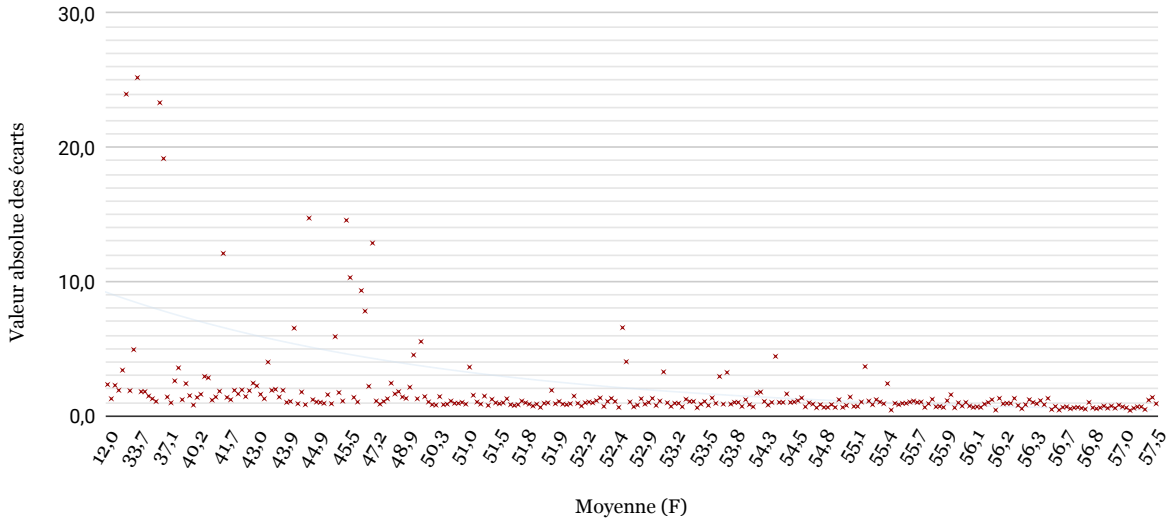

FIGURE 3.2 – **Stabilité de l'apprentissage du MLP :** Écarts de scores $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$ en fonction des scores $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$, pour 300 combinaisons d'hyperparamètres. En abscisses : scores $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$ moyens sur les 3 langues pilotes du modèle MLP, sur 10 runs obtenus avec 10 graines différentes (par ordre croissant). En ordonnées : Écart absolu moyen sur les 10 runs.

inutile. Il est intéressant de noter que la stratégie par tendances limite le nombre d'hyperparamètres activés contrairement à la stratégie de la sélection de la combinaison la plus performante obtenue lors de la recherche aléatoire.

Résultat final de la sélection des combinaisons d'hyperparamètres : Comme dernière étape des hyperparamètres, nous sélectionnons la meilleure combinaison parmi les deux possibles (CPP et CT) pour chaque cas de figure (SVM_2 et MLP_c en closed track, MLP_o en open track), ce qui donne les choix suivants :

— SVM_2 : la combinaison la plus performante obtenue par la recherche aléatoire (CPP)

— MLP_c : la combinaison des tendances (CT)

— MLP_o : la combinaison des tendances (CT)

Pour MLP_c et MLP_o, on sélection également la meilleure graine pour chaque langue. Sauf mention contraire, ce sont ces combinaisons d'hyperparamètres que nous utilisons dans la suite pour nos expériences.

3.4.3 Impact des graines sur les variations de performance

Dans notre phase de réglage des hyperparamètres pour le MLP, nous n'avons utilisé que deux graines d'initialisation, ce qui permet de limiter le temps de réglage des hyperparamètres. Avant de faire ce choix, nous avons cependant étudié les variations de performances du MLP pour une même combinaison d'hyperparamètres, en utilisant plusieurs graines.

Plus précisément, nous générons 300 combinaisons différentes d'hyperparamètres pour le modèle MLP, en utilisant les mêmes distributions que dans la section 3.4.1. Ensuite, nous lançons chaque combinaison sur les trois langues pilotes en utilisant dix différentes graines d'initialisation

(0, 1, 2, ..., 9). Les dix graines sont les mêmes pour toutes les 300 combinaisons d'hyperparamètres générées. Enfin, pour chaque expérience, nous calculons la déviation absolue moyenne entre les scores $\mathbf{F}(\mathbf{eval-eps})$ des dix graines, ainsi que la moyenne des scores $\mathbf{F}(\mathbf{eval-eps})$ des dix graines.

La figure 3.2 montre un ensemble de 300 points correspondants aux 300 combinaisons d'hyperparamètres. Chaque point associe la déviation absolue moyenne et le score moyen de la combinaison correspondante. La figure nous montre en particulier que la déviation absolue moyenne n'atteint des valeurs élevées que si le score moyen de la combinaison est mauvais. En revanche, quand la combinaison d'hyperparamètres produit une performance compétitive, la déviation absolue moyenne a tendance à être faible.

Ainsi, sur les combinaisons d'hyperparamètres les plus performantes, le changement de graines d'initialisation n'entraînera qu'un léger changement dans la performance. En conséquence, il ne nous a pas semblé utile d'avoir un nombre trop important de graines lors du réglage des hyperparamètres. Nous nous sommes limités à deux graines pour pouvoir gérer a minima les accidents, i.e. des cas où la déviation absolue moyenne serait élevée pour une combinaison d'hyperparamètres performante.

3.5 Expérimentation et résultats

Cette section décrit les expérimentations menées sur le MLP suite au réglage des hyperparamètres (section 3.4.1) et compare les résultats obtenus avec ceux de SVM_2 sur différents jeux de données définis dans la section 2.3 de la Partie 1. Nous décrivons section 3.5.1 les métriques d'évaluation que nous avons utilisées et section 3.5.2 les résultats des variantes sur les données de test de PARSEME 1.1, que nous comparons avec les résultats des meilleurs systèmes de la closed track de la compétition. La sous-section 3.5.3 montre les résultats des deux systèmes sur les mêmes données d'évaluation, en utilisant cette fois des plongements lexicaux préentraînés, et les compare avec les résultats des meilleurs systèmes de l'open track de la compétition PARSEME 1.1. Enfin, la section 3.5.4 donne leurs résultats sur d'autres jeux de données (FTB et DiMSUM), afin de tester la capacité d'adaptation des deux systèmes au changement de types d'expressions à identifier.

3.5.1 Métriques d'évaluation et test de significativité

Dans ce chapitre, les systèmes d'identification sont évalués en utilisant les mesures d'évaluation au niveau des EPs uniquement et pas au niveau des composants (cf. Partie 1, section 2.6) : le rappel $\mathbf{R_{eval-eps}}$, la précision $\mathbf{P_{eval-eps}}$ et la F-mesure $\mathbf{F_{eval-eps}}$. Pour évaluer les systèmes sur toutes les langues et en particulier sur les données de PARSEME 1.1 pour les expressions verbales, nous utilisons la métrique officielle de la compétition PARSEME 1.1 qui est la moyenne macro des F-scores $\mathbf{F_{eval-eps}}$ sur toutes les langues (ci-après $\mathbf{F_{AVG}}$), à des fins de comparaison avec les résultats de la compétition PARSEME 1.1. Comme pour le chapitre 2, nous utilisons aussi la moyenne harmonique entre le rappel moyen et la précision moyenne sur toutes les langues, qui donne des résultats plus fiables (ci-après $\mathbf{F_g}$).

Afin de vérifier la significativité statistique des différences de scores entre SVM_2 et MLP, nous appliquons la test de McNemar (Dietterich, 1998). C'est un test statistique permettant de comparer deux séries de valeurs binaires, calculées pour les mêmes éléments. On l'utilise ici pour étudier la significativité des différences de performance de deux systèmes d'identification A et B. Les éléments considérés sont les EP de référence du corpus d'évaluation. Les deux valeurs binaires considérées pour chaque EP de référence sont « est-ce que le système A a identifié

cette EP (True/False) » et idem pour le système B. Le calcul de la valeur de significativité n'utilise que les valeurs discordantes dans les deux séries : notons a le nombre d'EPs de référence uniquement repérées par le système A, et b celles uniquement par le système B. Le test calcule $K = (a-b)^2/(a+b)$, et cette statistique est ensuite comparée à la valeur seuil dans la table de la loi du Chi-deux avec un degré de liberté de 1. Si K est strictement supérieur à la valeur seuil, alors on rejette l'hypothèse nulle qui supposait que les différences observées entre les valeurs n'étaient dues qu'au hasard [42].

3.5.2 Résultats d'identification sans plongements préentraînés (MLP_c)

Dans cette section, nous évaluons notre perceptron multicouche sur les jeux de test des langues de PARSEME 1.1 en configuration closed track, et nous comparons ses résultats avec les résultats de SVM_2 et ceux des meilleurs systèmes de PARSEME 1.1 en configuration closed track. Pour notre MLP et SVM_2, nous avons utilisé la combinaison d'hyperparamètres résultante du réglage des hyperparamètres pour chacun d'eux, c'est-à-dire la combinaison des tendances (CT) pour le MLP et la combinaison la plus performante issue de la recherche aléatoire (CPP) pour le SVM_2.

Le tableau 3.4 donne la F-mesure **F**, la précision **P** et le rappel **R** de l'identification des EPs verbales sur les jeux de test de PARSEME 1.1, pour les systèmes SVM_2 et MLP ainsi que les mêmes scores pour les meilleurs systèmes de la compétition internationale PARSEME 1.1.

Les résultats montrent que le MLP bat tous les autres systèmes en configuration closed track. En particulier, MLP_c bat SVM_2 avec un écart de 2 points. Les résultats montrent que les différences entre le MLP et SVM_2 sont statistiquement significatives pour toutes les langues, sauf le basque, le hongrois, l'italien et le polonais, en utilisant le test de McNemar. Alors que la variante SVM_2 est la plus précise (81,1 % vs. 79,7 %, soit un écart de 1,4 point), MLP_c produit un meilleur rappel (51,5 % vs. 48,6 %, soit un écart de 2,9 points). Ce qui est frappant, c'est la très forte différence entre rappel et précision avec environ 30 points d'écart pour les deux variantes en faveur de la précision. Si l'on compare les résultats du MLP avec les systèmes de la compétition PARSEME 1.1, l'écart avec la moyenne des meilleurs scores par langue est de plus de 4 points. Par ailleurs, le MLP bat nettement le système vainqueur de la compétition en configuration closed track avec un écart de plus de 7 points. Il est intéressant de noter que nos deux variantes battent les systèmes de PARSEME 1.1 en produisant un score de précision beaucoup plus élevé (15 points d'écart environ), alors que le rappel est comparable. La meilleure précision de nos systèmes est donc la clé de leur succès vis-à-vis des systèmes de la compétition PARSEME 1.1.

Si l'on examine maintenant les résultats au niveau des langues, le MLP est classé premier pour 11 langues, alors que SVM_2 et les meilleurs systèmes de PARSEME 1.1 par langue arrivent en tête pour trois et cinq langues respectivement. Ces résultats tendent à valider la robustesse de notre approche trans-langue. Si l'on regarde les résultats par familles de langues, on s'aperçoit que, pour les langues romanes, SVM_2 est globalement meilleur que le MLP et notre système surpasse les meilleurs résultats de la compétition (sauf pour le roumain). Par ailleurs, nous observons des résultats surprenants sur les langues indo-iraniennes (l'hindi et le farsi). Nos deux systèmes sont moins performants que les meilleurs systèmes de la compétition pour ces langues. Il est à noter que la proportion de phrases possédant au moins une EP dans ces deux langues est beaucoup plus importante que pour la moyenne des langues : 88 % pour le farsi et 62 % pour l'hindi, vs. 39 % en moyenne. La longueur moyenne des EPs dans ces deux langues est équivalente à ce que l'on observe en moyenne sur l'ensemble des langues (longueur de 2,1).

42. Source : wikipedia, `https://fr.wikipedia.org/wiki/Test_de_McNemar`. Nous utilisons la bibliothèque Python StatsModels pour le calcul (`https://www.statsmodels.org/dev/index.html`)

Langue	SVM_2			MLP_c			PARSEME		
	F	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*
BG	63.3	85.1	50.4	**66.8**	77.1	59.0	62.5	63.6	61.5
DE	49.5	82.3	35.4	**51.5**	74.3	39.4	45.3	53.3	39.4
EL	57.8	87.1	43.3	**62.2**	85.4	48.9	49.8	61.7	41.7
EN	28.4	58.4	18.8	31.4	46.8	23.6	**32.9**	38.4	28.7
ES	39.2	39.6	38.8	**40.0**	38.8	41.2	34.0	29.5	40.0
EU*	80.7	91.4	72.2	**82.1**	90.6	75.0	75.8	84.7	68.6
FA	75.4	88.2	65.9	70.6	92.6	57.1	**77.8**	78.2	77.5
FR	**61.1**	81.8	48.8	59.0	83.5	45.6	56.2	77.2	44.2
HE	43.3	92.2	28.3	**45.2**	75.0	32.3	23.3	50.3	15.1
HI	66.8	89.6	53.2	64.9	92.3	50.0	**73.0**	78.0	68.6
HR	55.4	76.2	43.5	**59.3**	96.0	42.9	55.3	68.0	46.6
HU*	91.7	97.8	86.3	**92.4**	99.6	86.2	90.3	92.4	88.3
IT*	**55.7**	84.9	41.4	55.0	79.4	42.1	49.2	63.1	40.3
LT	38.0	77.3	25.2	**45.7**	81.5	31.8	32.2	45.8	24.8
PL*	69.4	88.1	57.3	**71.8**	92.1	58.8	67.0	77.0	59.2
PT	**68.9**	92.7	54.8	67.8	81.5	58.0	62.1	76.8	52.1
RO	80.9	91.1	72.7	83.5	77.6	90.3	**85.3**	84.5	86.1
SL	53.5	53.6	53.4	62.7	68.9	57.6	**64.3**	79.4	54.0
TR	47.5	83.2	33.2	**52.1**	81.5	38.3	45.2	44.5	46.1
AVG	59.3	81.1	48.6	**61.3**	79.7	51.5	56.9	65.6	51.7
$\mathbf{F_g}$	60.8			**62.6**			57.8		
PrsM							54.0	67.6	45.0

TABLE 3.4 – **SVM_2 et MLP : Identification** Les scores F, P, R (eval-eps) sur les jeux de test des langues de PARSEME 1.1, en utilisant les deux systèmes SVM_2 et MLP_c, appris sur les jeux d'entraînement et de développement (lorsqu'ils sont disponibles), en closed track. **PARSEME** représente les scores du système le plus performant de PARSEME 1.1 pour chaque langue en closed track. Notez que tous les systèmes de PARSEME 1.1 exploitent les jeux de développement dans l'apprentissage, à l'exception d'un système (*varIDE*) qui produit le meilleur score pour le bulgare en closed track. Notez aussi que les langues en gras sont celles qui ont de grands corpus, c'est-à-dire, leur nombre de tokens est supérieur à la moyenne du nombre de tokens sur toutes les langues. La dernière ligne du tableau (PrsM) correspond au score moyen du meilleur système de la compétition. Les langues à astérisque sont celles dont la différence entre leurs scores MLP et SVM_2 ne se montrent pas significatives, d'après le test de significativité McNemar ($p < 0.05$).

Analyse par classe formelle d'EPs

Le tableau 3.5 donne les scores du MLP et de SVM_2, ainsi que ceux des meilleurs systèmes de la closed track de la campagne PARSEME 1.1 sur plusieurs classes d'EPs verbales :

— les EPs **continues** et **discontinues**

— les EPs composées de plusieurs tokens (**Multitokens**) et les tokens polylexicaux (**TPs**)

— les EPs vues dans les corpus d'apprentissage (**Vues dans train**) et les nouvelles EPs (**Non vues dans train**). Une EP du jeu d'évaluation est considérée vue dans le corpus d'apprentissage s'il existe dans le corpus d'apprentissage au moins une EP ayant le même ensemble de lemmes, dans n'importe quel ordre et quelles que soient les discontinuités

	MLP(CT)			SVM_2 (CPP)			ParseMe		
	F	P	R	F	P	R	F	P	R
Continues	**67.0**	81.2	58.8	65.8	83.2	55.7	57.6	68.2	49.8
Discontinues	**46.7**	76.2	35.3	42.3	76.4	31.9	44.4	61.1	34.8
Multitokens	**60.4**	79.7	50.3	58.0	80.5	47.2	55.8	74.7	44.6
TPs	44.3	44.6	50.8	**53.6**	57.8	50.7	32.8	35.1	30.8
Vues (train)	**80.3**	81.3	80.9	78.7	86.2	73.3	72.9	86.5	63.0
Non vues (train)	01.3	10.8	0.70	07.2	26.0	04.8	**19.7**	14.3	31.5
Variantes	**69.5**	73.2	68.7	65.6	80.6	56.2	65.0	76.6	56.5
Identiques	89.6	87.7	93.0	**89.8**	89.4	91.1	83.7	90.4	77.9

TABLE 3.5 – **SVM_2 et MLP : analyse de performance :** Les scores $\mathbf{F_{AVG}}$, $\mathbf{P_{AVG}}$ et $\mathbf{R_{AVG}}$ en (eval-eps) de MLP(CT) et SVM_2 (CPP) ainsi que les scores des meilleurs systèmes de PARSEME 1.1 sur différentes classes formelles d'EPs (scores fournis par le script d'évaluation de la compétition internationale PARSEME 1.1).

qu'elle contient.

— au sein des EPs vues à l'apprentissage, on distingue en outre les identiques (**identique**), c'est-à-dire ayant les mêmes formes fléchies, le même ordre et les mêmes discontinuités qu'au moins une EP du corpus d'apprentissage, de celles qui sont des (**Variantes**).

Ce tableau montre que nos deux variantes sont très compétitives l'une par rapport à l'autre sur toutes les classes d'EPs. Pourtant, MLP gère mieux les EPs continues (1,2 point d'écart), discontinues (4,4 points), Multitokens (2,4 points), vues (1,6 point) et variantes (3,9 points) et dépasse SVM_2 sur la plupart des catégories grâce à un rappel élevé. En revanche, MLP se montre très faible en ce qui concerne l'identification des nouvelles EPs (1,3). SVM_2 produit un score plus élevé sur cette classe, mais le meilleur système de PARSEME 1.1 surpasse nettement nos variantes sur cette catégorie, ce qui reflète la quasi incapacité de généralisation de notre MLP ainsi que celle du SVM_2 dans le cadre d'un système par transitions.

En résumé, la variante MLP s'impose devant SVM_2 et les systèmes de PARSEME 1.1 pour la plupart des langues. SVM_2 se montre également très performant et compétitif avec MLP et bat les systèmes de PARSEME 1.1 pour la plupart des langues. D'autre part, MLP dépasse SVM_2 et les systèmes de PARSEME 1.1 sur l'identification des EPs discontinues, variantes et vues dans le corpus d'apprentissage. En revanche, MLP se montre totalement incompétent sur l'identification des EPs non vues dans le corpus d'apprentissage.

3.5.3 Résultats d'identification avec plongements lexicaux préentrainés

Afin de découvrir l'impact des plongements lexicaux préentraînés de manière non supervisée, nous remplaçons l'initialisation aléatoire des plongements lexicaux utilisés dans le MLP par l'utilisation des plongements préentrainés. Nous relançons les expériences d'identification des expressions verbales sur les données PARSEME 1.1 dans cette nouvelle configuration. Nous utilisons la combinaison d'hyperparamètres résultante de la phase de réglage des pour MLP_o (MLP en open track).

Dans nos expériences, nous utilisons les plongements lexicaux de FastText, qui sont également ceux utilisés par le meilleur système de la compétition PARSEME 1.1 en open track (Taslimipoor et Rohanian, 2018). FastText représente les mots sous forme de n-grammes de caractères et additionne les vecteurs associés aux n-grammes pour obtenir la représentation complète de chaque

Langue	MLP_c			MLP_o			PrsM		
	F	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*
BG	66.8	77.1	59.0	**67.7**	78.4	59.6	65.6	86.0	53.0
DE	**51.5**	74.3	39.4	49.9	72.8	38.0	45.5	60.9	36.4
EL	**62.2**	85.4	48.9	61.4	84.9	48.1	58.0	62.8	53.9
EN	31.4	46.8	23.6	31.9	46.9	24.2	**33.3**	33.8	32.7
ES	**40.0**	38.8	41.2	39.7	39.6	39.8	38.4	31.7	48.8
EU	**82.1**	90.6	75.0	80.2	91.8	71.2	77.0	81.8	72.8
FA	70.6	92.6	57.1	70.6	92.3	57.1	**78.4**	86.1	71.9
FR	59.0	83.5	45.6	58.8	85.4	44.8	**60.9**	71.9	52.8
HE	45.2	75.0	32.3	**47.3**	72.4	35.1	38.9	61.4	28.5
HI	64.9	92.3	50.0	64.9	91.9	50.2	**72.7**	76.8	69.0
HR	**59.3**	96.0	42.9	59.0	93.5	43.1	47.8	59.6	40.0
HU	92.4	99.6	86.2	**92.6**	99.6	86.5	85.8	90.1	82.0
IT	55.0	79.4	42.1	**56.5**	82.8	42.9	45.4	50.4	41.3
LT	**45.7**	81.5	31.8	45.3	81.3	31.4	22.9	35.7	16.8
PL	71.8	92.1	58.8	**72.2**	91.9	59.4	63.6	73.1	56.3
PT	67.8	81.5	58.0	**70.4**	90.6	57.5	68.2	71.1	65.5
RO	83.5	77.6	90.3	82.0	75.1	90.2	**87.2**	87.8	86.6
SL	**62.7**	68.9	57.6	61.2	67.1	56.2	52.3	58.6	47.2
TR	52.1	81.5	38.3	47.9	84.5	33.4	**58.7**	81.4	45.9
AVG	**61.3**	79.7	51.5	61.0	80.1	51.0	57.9	66.4	52.7
$\mathbf{F_g}$	**62.6**			62.3			58.8		
PrsM							58.1	76.2	54.3

TABLE 3.6 – **MLP : Impact des plongements préentraînés :** scores **F**, **P** et **R** en (eval-eps) pour les langues de PARSEME 1.1 sur les jeux de tests obtenus par notre **MLP** en open track et en closed track. Tous les systèmes sont ici appris sur les jeux d'entraînement et de développement (lorsqu'ils sont disponibles). Les colonnes **PrsM** représente les **scores $\mathbf{F_g}$, P, R** du système le plus performant de PARSEME 1.1 pour chaque langue. La dernière ligne PrsM indique les scores du meilleur système de la compétition.

mot, ce qui permet d'intégrer des caractéristiques morphologiques des formes lexicales considérées (voir Partie 1, section 2.8.3.0, page 46).

Le tableau 3.6 indique les scores **F**, **P** et **R** en (eval-eps) de l'identification des EPs verbales pour toutes les langues des données de test de PARSEME 1.1. Ces résultats sont comparés aux scores obtenus par MLP_c (MLP en closed track) et par les meilleurs systèmes de la compétition PARSEME 1.1. Les résultats montrent que nos deux systèmes MLP_c et MLP_o ont des performances comparables, avec MLP_c très légèrement meilleur que MLP_o en moyenne. Cela indique que, globalement, l'impact des plongements préentrainés est nul voire très légèrement négatif. Lorsque l'on examine les résultats par langue, on note des écarts faibles entre les deux variantes, sauf pour le turc où l'on observe plus de 4 points d'écart en faveur du MLP_c.

Comparé à l'état de l'art, MLP_c surpasse donc les systèmes ayant participé à la campagne internationale PARSEME 1.1. Plus récemment cependant, (Rohanian et al., 2019) ont présenté un modèle utilisant la convolution de graphes et l'auto-attention (cf. Partie 1, section 3.2.2) produisant des scores qui dépassent nettement les nôtres pour les 4 langues que ces auteurs ont ciblées (sur jeux de test de PARSEME 1.1, EN : 41.9, DE : 59.3, FR : 71.0, FA : 80.0). On peut

Hyperparamètre	Réglage	FTB		DiMSUM	
		CPP	CT	CPP	CT
Utilisation des tokens dans les n-grammes	Rdm Sch	T	T	F	F
Uni-grammes à base de Lemme et de POS (S_0, S_1, B_0)	Prelim	T	T	T	T
Uni-gramme B_1	Rdm Sch	F	T	T	T
Bi-grammes $S_0S_1, S_0, B_0, S_0, B_1, S_1, B_0$	Prelim	T	T	T	T
Bi-gramme S_0B_2	Rdm Sch	F	F	F	F
Tri-gramme $S_1S_0B_0$	Rdm Sch	T	T	T	T
Historique de transition (longueur 1)	Rdm Sch	F	F	T	T
Historique de transition (longueur 2)	Rdm Sch	F	T	F	F
Historique de transition (longueur 3)	Rdm Sch	F	F	T	F
Distance entre S_0 et S_1	Rdm Sch	T	T	F	F
Distance entre S_0 et B_0	Rdm Sch	T	F	T	T
Dictionnaire des composants des \eps	Rdm Sch	T	T	T	T
S_0 dans le dictionnaire de \mwt	Prelim	T	T	T	T
Longueur de la pile	Rdm Sch	T	T	F	F
Ré-échantillonnage	Prelim	F	F	F	F
F		81.2	80.7	46.8	46.3

TABLE 3.7 – **SVM_2 : Réglage sur FTB et DiMSUM** Les hyperparamètres des combinaisons d'hyperparamètres les plus performantes de la variante SVM_2, issues du réglage de la variante SVM_2 sur les jeux de données de FTB et de DiMSUM. La dernière ligne présente les scores F des combinaisons d'hyperparamètres sur les jeux d'évaluation utilisés dans le réglage.

cependant remarquer que ce modèle exploite les informations syntaxiques ce qui n'est pas notre cas.

3.5.4 FTB et DiMSUM

Nous testons maintenant nos variantes SVM_2 et MLP sur d'autres corpus annotés en EPs (DiMSUM et FTB) qui contiennent des EPs de toute catégorie morphosyntaxique (et pas seulement des EPs verbales). Pour rappel, FTB est un corpus arboré français incluant l'annotation d'EPs essentiellement continues de tous types (les EPs verbales y sont rares). DiMSUM est un corpus anglais (de taille modeste) contenant les annotations de tous types d'EPs incluant à la fois des EPs continues et discontinues (voir Partie 1, section 2.3.3)

Dans un premier temps, nous apprenons les modèles sur les jeux d'entraînement des deux corpus en utilisant la meilleure combinaison d'hyperparamètres résultant du réglage des hyperparamètres sur les données PARSEME 1.1 pour chacune des variantes, i.e. combinaison des tendances pour le MLP en closed track et combinaison la plus performante issue de la recherche aléatoire pour SVM_2. Nous évaluons ensuite les modèles appris sur les jeux de test. L'objectif est de tester la robustesse de notre réglage initial des hyperparamètres, lorsque nos variantes sont apprises et évaluées pour d'autres corpus et d'autres types d'expressions.

Dans un deuxième temps, nous lançons ensuite un tour de réglage des hyperparamètres sur les jeux d'entraînement et de développement de ces nouvelles données (FTB et DiMSUM), afin de comparer les résultats avec réglage sur données PARSEME et réglage spécifique. Le réglage fait ici ressemble à celui des mêmes modèles sur les données de PARSEME 1.1, puisque nous lançons

	Hyperparamètre	FTB			DiMSUM		
		CT	CPP_c	CPP_o	CT	CPP_c	CPP_o
Basique	Lemmatisation	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
	Dimension de token	300	383	300	300	470	300
	Dimension de POS	60	15	125	60	90	72
	Utiliser B_1	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
	Utiliser S_{-1}	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
Plon. lex.	Vocab compact	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE
	Généralisation	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE
	Pre-entraîné	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
	Modifiable	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
	Moyennage	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Dense	Nombre de neurones	200	174	105	200	140	580
	Dropout	0.3	0.3	0.1	0.3	0.1	0.1
Ré-éch.	Sur-éch avec seuil de fréq.	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
	Seuil pour joker	16	10	-	-	-	-
	multiplication de perte	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
	Coefficient	-	-	-	8	-	-
App.	Taux d'apprentissage	0.04	0.015	0.067	0.035	0.016	0.031
	Taille de batch	64	64	128	64	16	48
F(eval-eps)		**78.6**	78.4	78.4	49.0	51.2	**51.6**

TABLE 3.8 – **MLP : Réglage sur FTB et DiMSUM** Les hyperparamètres de (1) la combinaison d'hyperparamètres la plus performante sans plongements lexicaux préentraînés $\mathbf{CPP}_c$; (2) la combinaison d'hyperparamètres la plus performante avec plongements lexicaux préentraînés $\mathbf{CPP}_o$; (3) la combinaison d'hyperparamètres de tendances **CT** lors du réglage de la variante MLP sur les jeux de données de FTB et de DiMSUM. La dernière ligne présente les scores F(eval-eps) des combinaisons d'hyperparamètres sur les jeux d'évaluation utilisés dans le réglage.

mille expériences avec différentes combinaisons d'hyperparamètres, générées aléatoirement, sur une partie tronquée du jeu d'entraînement du FTB (270 milles tokens, comme sur les langues pilotes de PARSEME 1.1), et sur 80 % du jeu d'entraînement du DiMSUM. Les combinaisons du FTB sont évaluées sur le jeu de développement du FTB. Et nous sélectionnons la combinaison d'hyperparamètres la plus performante (CPP) et la combinaison de tendances (CT). Les combinaisons pour DiMSUM sont évaluées sur les 20 % restant du jeu d'entraînement.

On a cependant quelques différences par rapport au réglage sur les données PARSEME 1.1 : nous n'utilisons ici qu'une seule graine d'initialisation et pas 2, pour réduire le temps du réglage. En outre, nous ajoutons un hyperparamètre booléen contrôlant l'exploitation de plongements lexicaux préentrainés ou pas. Autrement dit, le réglage a été réalisé à la fois sur la closed track et l'open track.

Les tableaux 3.8 et 3.7 détaillent les valeurs des hyperparamètres résultant du réglage des variantes SVM_2 et MLP sur les jeux de données de FTB et DiMSUM. Nous observons les mêmes tendances que celles observées lors du réglage des deux variantes sur PARSEME 1.1. Par contre, DiMSUM favorise la généralisation du vocabulaire de la variante MLP et FTB a tendance à explorer le suréchantillonnage avec seuil de fréquence pour le MLP. Par ailleurs, l'exploitation

	FTB						DiMSUM					
	Avant			Après			Avant			Après		
	F	P	R	F	P	R	F	P	R	F	P	R
SVM_2	79.8	84.7	75.4	**80.7**	85.2	76.7	23.2	56.7	14.6	21.5	52.8	13.5
MLP	**78.3**	81.5	75.4	78.2	78.4	78.0	**23.6**	50.2	15.4	**39.0**	41.9	36.4
Constant-16	**81.1**						**-**					
Schneider-16	-						**54.8**					

TABLE 3.9 – **FTB et DiMSUM- Identification** Les scores **F**, **P** et **R** en (eval-eps) des variantes SVM_2 et MLP sur les jeux de données FTB et DiMSUM avant et après le réglage des hyperparamètres sur ces jeux de données. Les combinaisons d'hyperparamètres utilisées avant le réglage sont issues du réglage des deux variantes sur les données de PARSEME 1.1. Constant-16 et Schneider-16 représentent les scores **F** des modèles proposés dans Constant and Nivre (2016); Schneider et al. (2016) respectivement (développés sur les mêmes jeux de données).

de plongements lexicaux préentrainés et modifiables à l'entraînement semblent avoir un effet bénéfique pour les deux corpus, contrairement au données PARSEME 1.0. La dernière ligne de ces deux tableaux inclut le F-score des combinaisons d'hyperparamètres CPP et CT pour les deux variantes et pour les deux corpus. Pour FTB, les résultats sont conformes aux résultats observés sur les données PARSEME 1.1, à savoir : pour SVM_2, c'est la combinaison la plus performante (CPP) qui obtient les meilleures performances ; pour MLP, c'est la combinaison des tendances (CT). Concernant DiMSUM, on observe des résultats un peu différents. Pour SVM_2 et le MLP, c'est toujours la combinaison CPP qui est la meilleure. Enfin, contrairement aux données PARSEME 1.1, les écarts de performance entre les combinaisons CPP et CT est assez ténue : au maximum 1,6 points d'écart (pour DiMSUM pour le MLP) alors qu'on observait plus de 3 points d'écart entre CT et CPP pour le MLP sur les données PARSEME 1.1.

Le tableau 3.9 indique les scores **F**, **P** et **R** en (eval-eps) des deux variantes sur les jeux de test de FTB et DiMSUM en utilisant deux classes de combinaisons d'hyperparamètres : la première comprend les combinaisons issues du réglage des variantes SVM_2 et MLP sur les données de PARSEME 1.1 **(Avant)**, alors que la deuxième comprend les meilleures combinaisons issues du réglage des mêmes variantes sur les données de FTB et DiMSUM **(Après)**.

Concernant FTB, les résultats montrent une supériorité de la variante SVM_2 sur le MLP, que ce soit avant ou après réglage des hyperparamètres sur FTB. Il n'y a qu'un seul point d'écart entre la combinaison issue du réglage sur PARSEME 1.1 et celle issue du réglage sur FTB. Enfin, SVM_2 obtient un score ($\mathbf{F_{apres}}$) très proche de celui de Constant et Nivre (2016), l'écart ne dépassant pas 0,5 point. Il est à noter que le score de Constant et Nivre (2016) représente, à notre connaissance, le score état-de-l'art sur ce jeu de données. Par contre, les scores ne sont pas vraiment comparables car Constant et Nivre (2016) exploitent aussi des lexiques externes contrairement à notre système. Pour le MLP, alors que le réglage des hyperparamètres (sur FTB) permet d'améliorer le rappel produit, il conduit à une détérioration de la précision. Et, au final, le réglage des hyperparamètres des variantes MLP sur FTB ne permet pas d'améliorer les résultats. Par ailleurs, ces résultats sur FTB tendraient à montrer la robustesse de notre approche de réglage et confirment la forte corrélation entre la performance et la proportion des EPs vues dans le corpus d'apprentissage (85,7 %).

Concernant DiMSUM, les deux variantes sont très loin d'obtenir des scores compétitifs par rapport au score de Schneider et al. (2016), puisqu'il existe un écart de plus de 30 points avec

la variante SVM_2, avant et après le réglage. Le MLP avant le réglage souffre du même écart que celui de SVM_2. Le réglage des hyperparamètres permet de diminuer cet écart à 16 points environ. L'analyse des résultats de SVM_2 et MLP sur les jeux de données de PARSEME 1.1 a montré qu'il existe une forte corrélation entre le score **F** et la proportion des EPs du jeu d'évaluation, vues dans le corpus d'apprentissage. Par rapport à DiMSUM, nous constatons que cette corrélation est maintenue dans la performance obtenue par SVM_2 (avant et après le réglage) et par MLP (avant le réglage), puisque les EPs vues ne représentent que 22,6 %. Le réglage des hyperparamètres du MLP sur DiMSUM conduit à une augmentation importante du rappel (25 points), et une détérioration de la précision de 10 points. Par conséquent, la performance s'améliore significativement (+15 points environ) pour le MLP. D'ailleurs, l'analyse de la performance des MLP sur les EPs vues et non vues dans le corpus d'apprentissage montre que la performance s'améliore énormément (25 points d'écart) sur les EPs non vues pour toutes les variantes MLP.

On suppose que l'activation de l'hyperparamètre responsable de la généralisation du vocabulaire (qui représente le changement le plus saillant dans la combinaison d'hyperparamètres sélectionnée par rapport à PARSEME 1.1) pourrait être à l'origine de cette amélioration, puisqu'elle est conçue pour améliorer la généralisation sur les EPs dont l'un des tokens se produit fréquemment dans d'autres EPs (voir section 3.1.1). D'un autre côté, et bien que 93 % des EPs de ce jeu de données ne se produisent qu'une ou deux fois, certains tokens tels que les particules, les adpositions, les verbes supports,.. se reproduisent au sein de multiples EPs, ce qui permet d'enrichir le vocabulaire du modèle par un riche ensemble d'entrées fictives capables de modéliser les EPs partiellement vues.

3.6 Analyse et discussion

Dans cette section, nous analysons plus en profondeur les résultats du MLP sur les données de la compétition internationale PARSEME 1.1, toujours en comparaison avec la variante SVM_2, en configuration closed track. L'objectif est d'étudier le comportement du réseau de neurones modélisant le MLP et de s'en inspirer pour développer d'autres réseaux de neurones améliorant les performances. En particulier, nous étudions l'impact de certains facteurs sur les performances : utilisation du ré-échantillonage (sous-section 3.6.1), proportion des EPs déjà vues dans le corpus d'apprentissage (sous-section 3.6.3), la catégorie de l'EP (sous-section 3.6.4), la taille du corpus d'apprentissage (sous-section 3.6.2), le type du vocabulaire utilisé en entrée de notre réseau de neurones (sous-section 3.6.5).

Les différentes analyses de cette section sont issues des résultats des variantes SVM_2 et MLP, apprises sur les jeux d'entraînement et évalués sur les jeux de développement lorsqu'ils sont disponibles. Sinon, nous entraînons les variantes sur 80 % du jeu d'entraînement et les évaluons sur les 20 % restant.

3.6.1 Impact du ré-échantillonnage

Dans cette section, nous étudions l'impact du ré-échantillonnage lors de l'apprentissage de nos modèles MLP et SVM_2 sur les performances de ces derniers. Le tableau 3.10 rend compte des résultats d'une expérimentation étudiant l'impact des différentes techniques du ré-échantillonnage sur les jeux de développement de toutes les langues de PARSEME 1.1. Le tableau donne les scores $\mathbf{F_{AVG}}$, $\mathbf{P_{AVG}}$ et $\mathbf{R_{AVG}}$ en (eval-eps) pour MLP et SVM_2 en faisant varier les techniques de ré-échantillonnage et en les combinant. Le tableau fournit également le score de la déviation absolue moyenne ($\mathbf{DAM_{avg}}$) du MLP, puisque chaque expérimentation du MLP (et du SVM_2)

Expérimentations	MLP				SVM_2		
	F_{avg}	P_{avg}	R_{avg}	DAM_{avg}	F_{avg}	P_{avg}	R_{avg}
0. Sans ré-échan.	33.0	**87.7**	23.1	12.2	60.7	**85.5**	49.2
1. Sous-échan.	36.1	76.1	26.3	13.9	**62.2**	74.5	**55.1**
2. Sur-échan. aléa.	60.3	86.4	48.1	1.0	60.8	85.0	49.4
3. ré-échan. acec seuil de fréq.	58.3	84.3	45.6	1.8	60.7	**85.5**	49.2
4. Multiplication de perte	26.6	31.1	26.3	3.2	60.7	**85.5**	49.2
1 + 2	**62.3**	81.9	**52.0**	**0.7**	62.1	74.3	55.0
1 + 2 + 3	60.7	81.0	50.6	1.0	62.1	74.3	55.0
1 + 2 + 4	60.8	81.0	50.6	1.0	62.1	74.3	55.0

TABLE 3.10 – **MLP et SVM_2 : Ré-échantillonnage :** Les scores F_{avg}, P_{avg}, R_{avg} ainsi que le score de la déviation absolue moyenne DAM_{avg}, résultant de l'application de différentes stratégies de ré-échantillonnage sur MLP(CT) et SVM_2 (CPP), évalués sur les jeux de développement de PARSEME 1.1. Les trois dernières lignes représentent des expérimentations pour les quelles plusieurs techniques de ré-échantillonnage sont couplées. Notez que chaque expérimentation de MLP est répétée cinq fois et que nous n'indiquons pas les déviations absolues moyennes pour les expérimentations de SVM_2 parce qu'ils sont proches du zéro.

est répétée cinq fois, ce qui permet de mesurer la stabilité du modèle, la fiabilité des scores et donc l'efficacité des techniques du ré-échantillonnage. Les scores DAM_{avg} du SVM_2 ne sont pas intégrés dans le tableau vu la stabilité que montre cette variante en termes de performance pour chaque expérience ($DAM_{avg} \approx 0$). Les techniques de ré-échantillonage que nous étudions sont :

1. le sous-échantillonnage qui ne garde que les phrases contenant au moins une EP dans les données d'entraînement ;
2. le sur-échantillonage aléatoire qui consiste à imposer une distribution uniforme à toutes les transitions dans les données d'entraînement ;
3. le suréchantillonnage avec seuil de fréquence qui consiste à faire en sorte que les transitions en charge d'identifier les EPs rares soient dupliquées dans le jeu d'entraînement jusqu'à atteindre une fréquence minimale ;
4. la méthode de multiplication de la perte qui consiste à augmenter la perte pour certaines transitions clés pour l'identification en la multipliant par un coefficient lors de l'apprentissage.

Pour plus de détails sur ces techniques, se référer à la section 3.3.

Pour SVM_2, les techniques de ré-échantillonage ne modifient pas les performances, sauf pour le sous-échantillonnage pour lequel on observe un petit gain de 1,5 point en termes de F-score. Pour ce dernier cas, il faut noter que c'est surtout le rappel qui est favorisé car il permet de gagner près de 6 points sur cette mesure. On noterons une perte de précision de près de 11 points.

Pour MLP, le tableau montre que son problème principal est le faible rappel, puisqu'il existe un écart de 26 points entre le rappel du MLP et celui de SVM_2 lorsque l'on n'applique pas de techniques de ré-échantillonnage. Le ré-échantillonnage aléatoire représente une étape décisive pour la stabilisation de l'apprentissage du MLP et il lui permet de gagner 27 points en F-score (avec gain de 15 points en termes de rappel), comparé à l'expérimentation sans ré-échantillonnage. L'impact du sous-échantillonnage est très marginal : il permet juste un petit gain de 3 points en

F-score. Par contre, son couplage avec le suréchantillonnage aléatoire permet d'obtenir le meilleur score global.

Le suréchantillonnage avec seuil de fréquence (sensé améliorer la distribution des EPs rarement vues dans le corpus d'apprentissage) permet d'améliorer le rappel de manière significative, mais n'arrive pas à atteindre les scores du ré-échantillonnage aléatoire. La multiplication de la perte (sensé former une alternative aux techniques de ré-échantillonnage) a un impact négatif sur la performance, puisqu'il conduit à une détérioration dramatique de la précision ($P = 31,1$ versus $87,7$ pour le modèle sans aucune technique du ré-échantillonnage). Enfin, combiner trois techniques de ré-échantillonnage ensemble ne permet pas de battre la combinaison de techniques que nous avons constamment utilisée lors de nos expériences.

En résumé, ce tableau témoigne du rôle crucial de certaines techniques de ré-échantillonnage dans la stabilisation de l'apprentissage de la variante MLP. La combinaison du sous-échantillonnage et du suréchantillonnage aléatoire permet d'équilibrer la distribution des transitions des jeux d'entraînement, ce qui se traduit par une augmentation énorme du rappel. Pourtant, ces techniques ne conduisent pas à une amélioration très significative de la performance du SVM_2.

3.6.2 Impact de la taille du corpus d'apprentissage

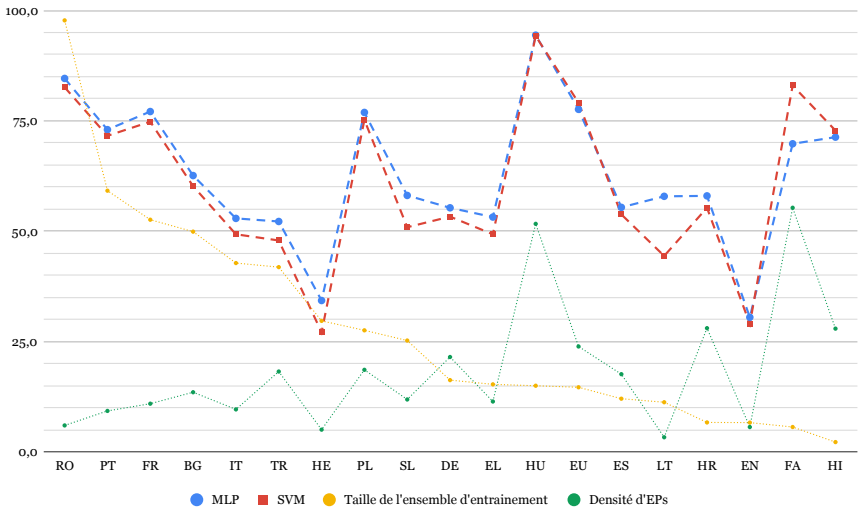

FIGURE 3.3 – **Impact de la taille du corpus d'apprentissage :** Graphique des scores **F(eval−eps)** des variantes SVM_2 (en rouge) et MLP (en bleu), les tailles du corpus d'apprentissage (en jaune) ainsi que les densités des EPs dans les corpus d'apprentissage (en vert) pour chaque langue de PARSEME 1.1 (triées d'après la taille de leurs corpus d'apprentissage). Les tailles des jeux d'apprentissage sont mises à l'échelle en divisant sur la taille approximative du plus grand corpus de PARSEME 1.1 ($\approx$ 800 milles tokens). La densité des EPs est le quotient du nombre d'occurrences d'EPs sur le nombre de tokens du corpus d'apprentissage (multiplié par dix pour la rendre plus visible).

La littérature sur l'apprentissage profond évoque la taille du corpus d'apprentissage comme l'une des conditions incontournables de la réussite de l'apprentissage pour de nombreuses tâches. Nous étudions donc l'impact de la taille du corpus d'apprentissage sur la performance de nos modèles. Le graphique 3.3 visualise les scores **F(eval − eps)** des variantes MLP et SVM_2 pour

toutes les langues, en vis-à-vis avec les tailles des jeux d'apprentissage et les densités des EPs dans ces données. La densité des EPs est la division du nombre d'occurrences d'EPs sur le nombre de tokens du corpus d'apprentissage. Les langues sont triées en fonction de la taille du corpus d'apprentissage dans l'ordre décroissant.

Le graphique ne montre aucune corrélation solide entre les scores $\mathbf{F}(\mathbf{eval}-\mathbf{eps})$ et les tailles du jeu d'entraînement. Pour les densités des EPs dans les jeux de données, la situation est relativement similaire, même si la densité des EPs dans les jeux d'entraînement semble corroborer avec les F-scores des deux variantes (surtout SVM_2) pour toutes les langues à l'exception des cinq langues ayant le corpus d'apprentissage le plus grand (roumain, portugais, français, bulgare et italien).

Nous allons voir dans la section suivante que c'est en réalité la proportion d'EPs vues à l'apprentissage qui est déterminante pour la performance des systèmes.

3.6.3 Impact de la proportion d'EPs vues à l'apprentissage

Les expériences précédentes ont montré le faible pouvoir de généralisation de nos systèmes, avec des scores très faibles, voire quasi nuls pour les expressions verbales inconnues. Les EPs vues dans le corpus d'entraînement ont donc un rôle fondamental. L'expérience suivante cherche à confirmer cela. La figure 3.4 illustre la corrélation entre performance et proportions d'EPs vues à l'apprentissage. Plus précisément, la figure comprend les proportions par langue d'EPs vues à l'apprentissage (en bleu), et parmi elles celles vues à l'identique (en jaune, cf. la définition de ces classes supra section 3.5.2.0, 122), à comparer avec les scores $\mathbf{F}(\mathbf{eval}-\mathbf{eps})$ de SVM et MLP_c.

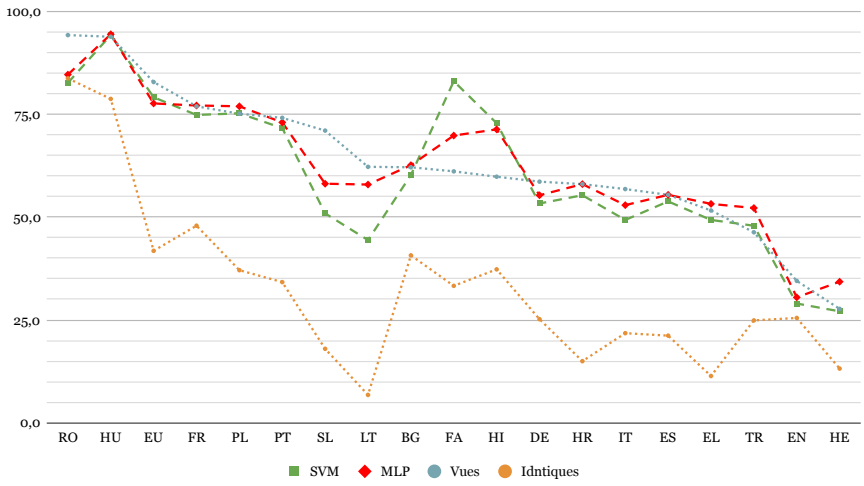

FIGURE 3.4 – **Impact de la proportion d'EPs vues à l'apprentissage :** scores $\mathbf{F}(\mathbf{eval}-\mathbf{eps})$ (eval-eps) des variantes SVM_2 et MLP (courbes verte et rouge) ; proportion des EPs des jeux d'évaluation ayant été vues à l'apprentissage (courbe bleue) ; proportion des EPs des jeux d'évaluation ayant été vues à l'identique (courbe jaune), pour chaque langue de PARSEME 1.1. Une EP vue est considérée vue à l'apprentissage s'il existe au moins une EP dans le corpus d'apprentissage ayant les mêmes lemmes, indépendamment de l'ordre des composants et des discontinuités éventuelles.

On peut constater à la lecture de cette figure qu'il existe une très forte corrélation entre les scores $\mathbf{F}(\mathbf{eval}-\mathbf{eps})$ des deux variantes d'une part et la proportion d'EPs vues à l'apprentissage

Catégorie	%	SVM_2		MLP	
		%	R_{avg}	%	R_{avg}
CVS.full	38.5	36.5	42.0	36.3	42.8
ID	24.1	18.1	35.7	19.0	35.6
VIR	16.4	22.6	67.0	22.6	71.4
CVP.full	8.3	10.5	35.6	10.9	40.6
VIA	4.8	5.7	41.2	5.9	46.6
CVS.cause	3.3	1.9	19.9	2.4	22.7
CPV	3.0	3.0	23.9	1.1	8.9
CVP.semi	1.4	1.7	46.2	1.7	46.4
R_{AVG}			38.9		39.4

TABLE 3.11 – **Impact des catégories d'EPs sur la performance d'identification :** Pour les catégories principales de la typologie PARSEME 1.1, le tableau donne la proportion de la catégorie dans les annotations gold (deuxième colonne) ; la proportion de la catégorie dans les EPs prédites pour les jeux de développement pour les deux variantes SVM_2 et MLP (3ème et 5ème colonnes) ; le rappel moyen sur toutes les langues (R_{avg} en eval-eps) de ces variantes pour chaque catégorie (4ème et 6ème colonnes). La dernière ligne donne la moyenne sur toutes les catégories. Nous avons omis quelques catégories présentes marginalement dans PARSEME 1.1, et retenu uniquement : les constructions à verbe support où le verbe support est neutre (CVS.full) ou à sens causatif CVS.cause) ; les idiomes verbaux (ID) ; les verbes intrinsiquement réfléchis (VIR) ; les constructions avec verbe à particule entièrement non compositionnelles (CVP.full) et semi-compositionnelles (CVP.semi) ; les verbes intrinsèquement adpositionnels (VIA), et les constructions à plusieurs verbes (CPV).

(modulo ordre, discontinuité, et flexion), et également avec la proportion d'EPs vues à l'identique. Un calcul du **coefficient de corrélation linéaire** (coefficient de Pearson) donne une mesure plus précise. Pour la proportion d'EPs vues, on obtient un coefficient de Pearson de $0,84$ avec la performance MLP, et de $0,81$ avec la performance SVM. Pour la proportion d'EPs identiques, la corrélation est un peu moins forte : $0,74$ avec la performance du MLP, et $0,77$ avec la performance du SVM. Ceci confirme que la proportion d'EPs vues à l'apprentissage est une caractéristique déterminante pour une bonne performance, et que le MLP est légèrement meilleur que le SVM pour gérer des variantes d'EPs vues à l'apprentissage.

3.6.4 Impact de la catégorie des EPs

Cette analyse vise à étudier l'impact des catégories linguistiques des EPs sur leur identification. Le tableau 3.11 fournit le rappel obtenu par SVM_2 et MLP pour les principales catégories de la typologie PARSEME 1.1. On constate que les deux systèmes ont un très bon rappel sur les verbes intrinsèquement réfléchis (VIR) (supérieur à 65 % vs. une moyenne de 40 % environ sur les différentes catégories). Cela corrobore les observations faites dans la section 2.4.5 sur la tâche de catégorisation par un système par transitions utilisant un modèle SVM. Cette performance peut s'expliquer par le patron syntaxique très contraint de ce type d'expression verbale formée d'un pronom réfléchi et d'un verbe. C'est de loin la meilleure catégorie, avec un écart de 21 points avec la deuxième catégorie la plus performante (CPV.SEMI). On constate aussi que malgré leur faible fréquence, les verbes intrinsèquement adpositionnels (VIA) et les constructions à verbe à particule (CVP) obtiennent de relativement bons résultats. Là encore, cela pourrait

Catégorie	Exhaustif			Compact		
	$\mathbf{F_g}$	P_{avg}	R_{avg}	$\mathbf{F_g}$	P_{avg}	R_{avg}
Toute	60.5	66.2	55.7	**64.3**	81.1	53.2
Vue	81.0	83.2	78.9	**81.9**	82.1	81.8
Non vue	**15.4**	19.6	12.7	2.1	9.9	1.2

TABLE 3.12 – **MLP- Impact du type de vocabulaire utilisé :** Les scores $\mathbf{F_g}$, $\mathbf{P_{avg}}$ et $\mathbf{R_{avg}}$ du modèle MLP sur toutes les EPs ainsi que les EPs des jeux d'évaluation vues et non vues dans le corpus d'apprentissage, en utilisant deux combinaisons d'hyperparamètres de tendances à une seule différence : le type du vocabulaire. Le nom du modèle reflète le type exploité.

aussi s'expliquer par le patron syntaxique très contraint de ces constructions.

Les résultats montrent également que le MLP est largement meilleur que SVM_2 sur les verbes intrinsèquement réfléchis (VIR), les verbes intrinsèquement adpositionnels (VIA) et les constructions à verbe à particule (non compositionnelles, CVP.full), en montrant des écarts supérieurs à 4 points. Le MLP a un rappel particulièrement bon pour les EPs dont le patron syntaxique est très contraint. À l'inverse, les constructions à plusieurs verbes (CPV) qui sont étonnement bien reconnues avec SVM_2 par rapport au MLP (13 points d'écart de rappel).

3.6.5 Impact du type de vocabulaire utilisé

Lors du réglage des hyperparamètres du MLP, nous avons utilisé deux types de vocabulaire : un vocabulaire dit exhaustif et un vocabulaire dit compact. Le vocabulaire exhaustif contient en gros toutes les formes lexicales du corpus d'entraînement à la fois pour les tokens et les unités multitokens. Le vocabulaire compact contient uniquement les formes lexicales des tokens des expressions polylexicales, ainsi que les formes lexicales des unités multitokens.

Les résultats du réglage de la variante MLP sur les données PARSEME 1.1 montrent que le vocabulaire compact est présent à la fois dans les combinaisons d'hyperparamètres les plus performantes issues de la recherche aléatoire (CPP) et les combinaisons de tendances (CT), ce qui reflète l'importance de cet hyperparamètre en termes de performance. Pourtant, la faiblesse du MLP face aux EPs non vues dans le corpus d'apprentissage, pose des questions sur l'utilisation de ce type de vocabulaire.

Le tableau 3.12 explore l'impact du type du vocabulaire en comparant deux combinaisons d'hyperparamètres pour le MLP. La seule différence est le type du vocabulaire, puisque (**MLP-Exhaustif**) utilise un vocabulaire exhaustif alors que (**MLP- Compact**) utilise un vocabulaire compact (voir section 3.1.1). À noter que MLP utilisant un vocabulaire compact utilise la combinaison d'hyperparamètres issus du réglage des hyperparamètres.

Le tableau montre très clairement que l'utilisation d'un vocabulaire compact est une raison importante de la faiblesse de la variante MLP pour gérer les EPs non vues dans le corpus d'apprentissage pour atteindre un F-score de 15,4 % (au lieu de 2,1 %). À noter que les meilleurs systèmes de la compétition PARSEME 1.1 en closed track atteignent des scores compris entre 15 et 20 % de F-score sur le corpus de test.

Pourtant, l'utilisation d'un vocabulaire exhaustif, qui améliore significativement la capacité de généralisation de la variante, conduit à une dégradation importante de la performance globale (60,5 % vs. 64,3 %, soit un écart d'un peu moins de 4 points). Cela signifie que l'utilisation du vocabulaire exhaustif fait baisser les performances sur les EPs vues dans le corpus d'apprentissage. Plus précisément, on observe une perte de trois points de rappel, face à un petit gain d'un point pour la précision avec le vocabulaire exhaustif.

	F_g	F_{avg}	P	R
$SVM2_{MLP}$	61.4	59.8	79.6	50.0
MLP_{SVM}	**63.0**	**62.0**	76.1	**53.7**
MLP	62.6	61.3	**79.7**	51.5

TABLE 3.13 – **Empilement des modèles SVM et MLP :** les scores $\mathbf{F_g}$, $\mathbf{F_{avg}}$, $\mathbf{P_{avg}}$ et $\mathbf{R_{avg}}$ (en eval-eps) des modèles d'empilement $SVM2_{MLP}$ et $\mathbf{MLP}_{SVM2}$ ainsi que ceux du modèle MLP sur les jeux test de toutes les langues de PARSEME 1.1.

3.7 Empilement des modèles

Lors de nos expériences, nous avons comparé un système par transitions reposant sur un modèle non linéaire (MLP) et un autre reposant sur un modèle linéaire (SVM_2). Nous avons pu nous rendre compte qu'ils étaient complémentaires sur certains aspects. Bien que le MLP soit globalement plus performant que SVM_2, le dernièr se montre plus performant sur les EPs non vues dans le corpus d'apprentissage et sur les tokens polylexicaux. Nous proposons donc de les combiner ensemble, en utilisant une méthode classique par empilement, dans le but d'améliorer encore les performances en tenant compte de la complémentarité des deux systèmes.

L'empilement des modèles est une méthode d'apprentissage automatique qui consiste à alimenter un modèle par les prédictions d'autres modèles auxiliaires dans l'objectif d'améliorer la performance du système (Ting et Witten, 1999). Nous appliquons la technique d'empilement sur nos deux modèles. Au début, nous transformons le modèle linéaire en un modèle auxiliaire alimentant le modèle MLP avec ses prédictions. En d'autres termes, le modèle MLP nourrit sa couche dense avec un vecteur creux supplémentaire d'entrée. Ce vecteur représente les prédictions du modèle linéaire pour la configuration en cours de traitement.
Afin d'entraîner les modèles linéaires, nous utilisons la technique du jackknifing à 5 plis, c'est-à-dire, nous entraînons cinq modèles linéaires, chacun sur 4/5 du jeu d'entraînement. Pour générer le vecteur supplémentaire d'entrée pour les configurations d'une phrase, nous devons sélectionner le modèle linéaire qui n'a pas été entraîné sur la même phrase. Au moment de l'analyse, nous utilisons un SVM, appris sur le jeu complet des données d'apprentissage, pour générer le vecteur d'entrée supplémentaire pour chaque configuration des phrases du test.

Nous avons également développé le modèle inverse, c'est-à-dire, nous utilisons les prédictions des modèles MLP comme un trait supplémentaire lors de l'extraction des traits du modèle SVM_2. Ces traits supplémentaires indiquent la prédiction du modèle MLP pour chaque configuration, dans l'objectif d'enrichir l'ensemble de traits et d'aider le modèle linéaire à produire de meilleures prédictions. Il convient de noter que la même procédure de jackknifing est utilisée pour entraîner les modèles MLPs.

3.7.1 Expérimentation et résultats d'identification

Étant donné que le temps d'apprentissage de tels modèles se multiplie, nous ne lançons pas la procédure de réglage des hyperparamètres pour ces modèles et utilisons donc les combinaisons d'hyperparamètres résultant du précédent réglage des hyperparamètres pour le MLP et SVM_2 pris isolément. Ainsi, le modèle principal du MLP_{SVM2} et le modèle MLP auxiliaire de $SVM2_{MLP}$ utilisent la combinaison de tendances (CT) du MLP seul en configuration closed track. Le modèles SVM auxiliaires du MLP_{SVM2} et le modèle principal du $SVM2_{MLP}$ utilisent la combinaison la plus performante (CPP) issue du réglage des hyperparamètres de SVM_2 seul.

Nous présentons les résultats des deux modèles, MLP_{SVM_2} et $SVM2_{MLP}$, sur les jeux de test

de PARSEME 1.1 (voir tableau 3.13). Ce tableau montre que ces variantes qui requièrent une augmentation importante du temps d'entraînement, ne montrent pas une amélioration significative de la performance de notre système sur les jeux de données de PARSEME 1.1. Tout d'abord, $SVM2_{MLP}$ connaît une légère baisse de performance par rapport à MLP (1,2 point de moins). Ensuite, MLP_{SVM2} permet au système de légèrement surpasser le score $\mathbf{F_g}$ de la variante MLP (0,4 point) en produisant un rappel plus élevé (2,2 points). Par ailleurs, MLP_{SVM2} montre que l'augmentation importante du rappel (2,2 points) par rapport à MLP est contre-balancée par une baisse importante de la précision (3,6 points). L'absence d'un réglage spécial des hyperparamètres des deux modèles d'empilement peut expliquer en partie l'incapacité de ces modèles à battre la variante MLP de manière significative.

3.8 Résumé

Dans ce chapitre, nous présentons cette fois l'utilisation d'une méthode neuronale pour la prédiction des transitions, au sein de notre analyseur par transitions pour l'identification d'EPs.

Nous utilisons cette fois le jeu de transitions plus abouti $\mathcal{T}_\mathbf{2}$, permettant d'analyser certains cas d'enchâssement. Pour la définition des traits, nous nous plaçons dans un scénario où la représentation syntaxique de la phrase n'est pas disponible. L'architecture du classifieur est un perceptron multicouche (MLP), dont l'entrée est constituée de plongements vectoriels pour les tokens et les étiquettes morphosyntaxiques de certains éléments « ciblés » de la configuration courante. Pour comparaison, nous entraînons également un classifieur SVM tel que décrit au chapitre précédent, mais avec le jeu de transitions $\mathcal{T}_2$, et des traits excluant les patrons syntaxiques.

Le chapitre décrit un travail expérimental conséquent, permettant de comparer la mise au point et les performances du système avec classifieur MLP versus classifieur SVM. Nous avons, en effet, décrit les méthodes et les résultats du réglage des hyperparamètres des deux variantes, ainsi que l'expérimentation des deux variantes sur des jeux de données divers (PARSEME 1.1, FTB et DiMSUM).

Il ressort en premier lieu que le modèle MLP ne fonctionne pas du tout sans ré-échantillonnage des données. Nous avons sélectionné expérimentalement deux types de ré-échantillonnage : l'exclusion de l'apprentissage des phrases sans EPs, et ré-échantillonnage aléatoire des exemples d'apprentissage pour équilibrer les classes (les transitions, cf. les transitions responsables de l'identification d'EPs étant très rares par rapport aux autres). On vérifie qu'avec ré-échantillonnage, la variation de performance en fonction des graines pour l'initialisation aléatoire des paramètres est relativement faible, alors que sans ré-échantillonnage, l'apprentissage est très instable. Le classifieur SVM est beaucoup plus robuste de ce point de vue, le ré-échantillonnage n'ayant pas d'impact.

Devant la combinatoire très importante des valeurs possibles d'hyperparamètres, nous avons tenté d'utiliser une méthodologie rigoureuse, d'abord (i) en sélectionnant trois langues pilote, et en tronquant leur données d'entraînement pour se placer dans des conditions "moyennes" d'apprentissage, ensuite (ii) en utilisant une recherche aléatoire de combinaison d'hyperparamètres plutôt qu'une recherche en grille. Nous avons montré qu'au lieu de sélectionner comme habituellement la combinaison la plus performante sur jeux de développement, il est plus efficace de sélectionner une combinaison de "tendances", en sélectionnant chaque valeur d'hyperparamètres dans un ensemble de k meilleures combinaisons.

En termes de performance, comparées aux systèmes de PARSEME 1.1, nos deux variantes produisent les scores état de l'art sur une bonne partie des langues de PARSEME 1.1 et des performances très compétitives sur les langues restantes. Le système MLP est globalement meilleur

que SVM ($F_g = 62,6$ versus $F_g = 60,8$), et que les résultats des participants à PARSEME 1.1 ($F_g = 57,8$), mais pas tout à fait pour toutes les langues. Nos deux variantes se montrent, par ailleurs, complémentaires : la variante MLP a un meilleur rappel et une moins bonne précision que SVM_2. Cependant, un empilement des deux variantes ne permet pas d'améliorer les résultats de manière significative.

Nous avons détaillé l'analyse de la performance des deux variantes en fonction de différentes caractéristiques linguistiques et formelles des EPs, et des différentes langues des données PARSEME 1.1. Le MLP est meilleur pour les EPs aussi bien continues que discontinues, mais pêche pour les EPs à un seul token (apparaissant dans certains jeux de PARSEME). Il n'apparaît pas de corrélation de la performance avec les catégories linguistiques des EPs verbales (constructions à verbe support, verbes intrinsèquement réfléchi, etc...), ni directement avec la taille du corpus d'apprentissage. En réalité, le facteur déterminant pour la performance est la proportion des EPs vues à l'apprentissage (à l'identique, ou avec variation morphologique ou variation d'ordre). En effet, notre système MLP s'avère très performant pour celles-ci, mais pratiquement nul ($F_{AVG} = 1,3$) pour repérer des EPs non vues, le SVM étant légèrement meilleur mais très insuffisant ($F_{AVG} = 7,2$), à comparer avec le score de $19,7$, certes bas mais pas autant, obtenu en considérant le meilleur score de la compétition PARSEME 1.1 pour chaque langue, sur les EPs inconnues. L'absence de pouvoir de généralisation représente l'enjeu principal pour nos deux variantes.

Une première expérimentation visant à une meilleure généralisation a consisté à utiliser simplement des plongements lexicaux pré-entraînés, mais cela s'est avéré malheureusement n'avoir qu'un impact marginal. Pourtant, nous avons utilisé les plongements *Fasttext*, qui incluent des plongements de sous-parties de mots, réputés avoir un bon pouvoir de généralisation. L'utilisation de plongements lexicaux contextuels, qui ont fait leurs preuves récemment en TAL, est certainement à étudier, en particulier dans le cadre de l'identification d'EPs, où l'idiosyncrasie peut se manifester par une différence de contexte entre occurrence idiosyncratique et occurrence littérale.

Nous avons également évalué la robustesse de nos deux variantes et de notre méthode de réglage, en les testant sur d'autres jeux de données, avec d'autres types d'EPs. Les résultats sont mitigés : d'une part, on obtient que la variante linéaire est moins sensible au changement de données. D'autre part, on obtient bien des résultats au niveau de l'état de l'art pour les EPs du French Treebank (EPs de tout type grammatical, presqu'exclusivement continues), mais les résultats sont décevants sur les données anglaises DiMSUM. Même si celles-ci ont un taux relativement haut d'EPs inconnues, les faibles performances obtenues exigent davantage d'analyse.

Chapitre 4

Réseaux de neurones contextuels pour l'identification des EPs

Ce chapitre propose des réseaux de neurones visant à intégrer davantage de contexte que ne le fait le MLP, qui n'utilise que quelques éléments ciblés de la configuration. Après la description des modèles, le chapitre expose les détails de leur expérimentation, exclusivement sur les jeux de données PARSEME 1.1.

La section 4.1 présente une simple extension du modèle MLP visant à intégrer davantage d'éléments ciblés de la configuration. La section 4.2 décrit trois variantes d'un MLP s'appuyant sur des couches récurrentes. La première variante (section 4.2.2) prend en compte le contexte global de la phrase en exploitant la sortie d'une couche récurrente sur toute la phrase. La deuxième variante (section 4.2.3) considère les éléments ciblés de la configuration comme une séquence modélisée par une couche récurrente [43]. La troisième (section 4.2.4) représente les unités multitokens de la pile avec des couches récurrentes. La section 4.3 présente une adaptation de l'architecture de Kiperwasser et Goldberg (2016) qui a été conçue à l'origine pour l'analyse syntaxique en dépendances. Ce modèle, appelé dorénavant modèle KG-2016, utilise un MLP classique alimenté par les éléments ciblés de la configuration, plongés dans le contexte de la phrase en entrée, grâce à une couche récurrente sur les tokens de la phrase et un mécanisme de sélection dynamique des éléments ciblés par le MLP.

Nous détaillons section 4.4 le réglage de ces différentes variantes et donnons section 4.5 les résultats de leur évaluation sur les jeux de tests de PARSEME 1.1. La section 4.6 fournit une analyse de la performance pour différentes classes d'EPs (sur les jeux de développement). Enfin, la section 4.7 présente un typage et une analyse des erreurs de ces variantes sur les mêmes jeux de développement.

Le but de ces expérimentations est d'établir si ces variantes plus contextuelles permettent une amélioration, dans le contexte d'un apprentissage avec relativement peu de données pour un apprentissage profond (quelques centaines ou milliers d'occurrences d'EPs verbales) selon les langues. Nous ne comparons que des modèles unitaires (non ensemblistes, i.e. sans empilement comme présenté section 3.7).

43. Pour cette variante, on ne peut pas parler d'extension du contexte pris en compte.

4.1 MLP-Wide

Nous développons une extension du modèle MLP, dite MLP-Wide, dont l'entrée comprend plus d'éléments que les seuls éléments ciblés[44]. MLP-Wide prend en entrée les éléments ciblés, plus pour chacun les étiquettes morphosyntaxiques des tokens du voisinage linéaire de l'élément ciblé (l'hyperparamètre FENÊTRE DE VOISINAGE indique la taille de la fenêtre à considérer autour d'un élément ciblé).

Outre la taille de la fenêtre, MLP-Wide utilise tous les hyperparamètres définis pour la variante MLP, à l'exception des hyperparamètres du ré-échantillonnage avec seuil de fréquence et la multiplication de la perte, exclus du réglage des variantes de ce chapitre.

4.2 Variantes récurrentes du réseau de neurones MLP

Les réseaux de neurones récurrents permettent de modéliser des séquences de taille arbitraire. Un réseau récurrent unidirectionnel de gauche à droite produit notamment un vecteur dense pour chaque élément de la séquence, qui « résume » cet élément et tout son contexte gauche depuis le début de la phrase.

Avant de détailler des variantes de notre MLP intégrant une couche récurrente, nous décrivons rapidement le principe d'une telle couche, ainsi que les deux types de couches récurrentes que nous avons expérimentées (LSTM et GRU).

4.2.1 Couches récurrentes LSTM et GRU

De manière abstraite, Goldberg (2016) décrit une couche récurrente comme une fonction qui prend en entrée un vecteur d'état initial $\mathbf{s_0}$ et une séquence de $\mathbf{n}$ vecteurs d'entrée $\mathbf{x_1}, .., \mathbf{x_n}$, notée plus rapidement $\mathbf{x_{1:n}}$ (en général il s'agit d'un vecteur dense non contextuel par token de la phrase) et renvoie une séquence de vecteurs d'état $\mathbf{s_{1:n}}$, ainsi qu'une séquence de vecteurs de sortie $\mathbf{y_{1:n}}$ (équation 4.1). Un vecteur de sortie $\mathbf{y_i}$ est obtenu par une fonction O appliquée au vecteur d'état correspondant $\mathbf{s_i}$ (équation 4.2). Les vecteurs d'entrée $\mathbf{x_i}$ sont présentés à la couche de manière séquentielle. L'aspect clé du réseau récurrent est que le vecteur d'état $\mathbf{s_i}$ est obtenu à partir de $\mathbf{x_i}$ mais également à partir de l'historique $\mathbf{s_{i-1}}$ (équation 4.2).

$$\mathbf{RNN}(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_{1:n}}, \mathbf{y_{1:n}} \tag{4.1}$$

$$s_i = R(s_{i-1}, x_i) \tag{4.2}$$

$$y_i = O(s_i) \tag{4.3}$$

La couche récurrente fournit un cadre pour conditionner la représentation à une position i sur l'ensemble de la séquence $x_1, .., x_i$ sans recourir à l'hypothèse de Markov (qui force une indépendance au contexte sauf pour un nombre fixe de positions contextuelles), traditionnellement utilisée pour la modélisation des séquences avant l'apprentissage profond (Goldberg, 2016).

Plusieurs fonctions R et O ont été proposées, notamment des fonctions dites « à porte » (*gated* en anglais) qui évitent lors du calcul des gradients que ceux-ci ne prennent des valeurs extrêmes, ce qui pose des problèmes computationnels et empêche l'apprentissage de bien se dérouler.

44. Nous rappelons que les éléments ciblés dans le modèle MLP sont par défaut S_0, S_1, B_0, et des hyperparamètres considèrent l'utilisation ou pas de B_1 et du dernier token ayant subi un REDUCE S_{-1}.

L'architecture LSTM (Long Short Term Memory) est l'architecture « à portes » la plus célèbre et a montré une grande efficacité concernant la modélisation des longues séquences dans de nombreuses tâches du TAL. Cette architecture divise le vecteur d'état en deux vecteurs (équation 4.4) : un pour les cellules de « mémoire » ($\mathbf{c_j}$) et une représentation cachée pour la position courante ($\mathbf{h_j}$), qui est en outre finalement la représentation de sortie (équation 4.5).

Le fonctionnement de LSTM est assez complexe, mais on a coutume d'interpréter que les fonctions portes de cette architecture décident combien de la nouvelle entrée x_j doit être passée en entrée dans la cellule de mémoire (porte **i** pour input), combien de la cellule de mémoire précédente doit être « oubliée » (porte **f** pour forget), et enfin ce qui est fourni en sortie (porte **o** pour output). Les équations suivantes, de Goldberg (2017), résument le fonctionnement de cette architecture :

$$\mathbf{s_j} = \mathbf{R_{lstm}}(\mathbf{s_{j-1}}, \mathbf{x_j}) = [\mathbf{c_j}; \mathbf{h_j}] \tag{4.4}$$

$$c_j = f \odot c_{j-1} + i \odot z$$

$$h_j = o \odot tanh(c_j)$$

$$\mathbf{i} = \sigma(x_j W^{xi} + h_{j-1} W^{hi})$$

$$\mathbf{f} = \sigma(x_j W^{xf} + h_{j-1} W^{hf})$$

$$\mathbf{o} = \sigma(x_j W^{xo} + h_j W^{ho})$$

$$\mathbf{z} = tanh(x_j W^{xz} + h_{j-1} W^{hz})$$

$$\mathbf{y_j} = \mathbf{h_j} \tag{4.5}$$

L'architecture GRU (Gated recurrent unit), représente une autre architecture récurrente à base de portes, plus simple et moins exigeante sur le plan computationnel, et qui a montré une performance compétitive avec LSTM sur plusieurs tâches du TAL (Goldberg, 2017). Nous reprenons la formalisation de Goldberg (2017) dans les équations 4.6.

$$\mathbf{y_j} = \mathbf{s_j} = \mathbf{R_{gru}}(\mathbf{s_{j-1}}, \mathbf{x_j}) = (\mathbf{1} - \mathbf{z}) \odot \mathbf{s_{j-1}} + \mathbf{z} \odot \mathbf{s'_j} \tag{4.6}$$

$$\mathbf{z} = \sigma(x_j W^{xz} + s_{j-1} W^{sz})$$

$$\mathbf{r} = \sigma(x_j W^{xr} + s_{j-1} W^{sr})$$

$$\mathbf{s\prime_j} = tanh(x_j W^{xs} + (r \odot s_{j-1}) W^{sg})$$

Une porte ($\mathbf{r}$) est utilisée pour contrôler l'accès à l'état précédent $\mathbf{s_{j-1}}$ et calculer une mise à jour $\mathbf{s'_j}$. Une interpolation au moyen d'une porte ($\mathbf{z}$) entre $\mathbf{s'_j}$ $\mathbf{s_{j-1}}$ est ensuite faite pour calculer le vecteur d'état $\mathbf{s_j}$, qui est également le vecteur de sortie $\mathbf{y_j}$.

4.2.2 Variante 1 : MLP-R-Sent

Le premier modèle que nous proposons pour découvrir l'effet des couches récurrentes est le modèle MLP-R-Sent, qui est une extension assez simple du modèle MLP : nous passons la concaténation des plongements des formes lexicales et étiquettes de tous les tokens de la phrase à une couche récurrente (unidirectionnelle de gauche à droite). Une couche dense est alimentée par la concaténation **(i)** des plongements des formes lexicales et étiquettes de tous les tokens de la phrase (comme dans le modèle MLP) et **(ii)** le dernier vecteur de sortie de la couche récurrente (celui du dernier token de la phrase).

Ce vecteur récurrent du dernier token de la phrase est une représentation intégrant d'une certaine manière la phrase complète (formellement, l'historique complet de tous les tokens de la phrase a été utilisé pour l'obtenir). Son intégration en entrée vise ainsi à fournir un contexte plus large.

Mise en œuvre et hyperparamètres

Éléments d'entrée : Comme pour le MLP classique, $\mathbf{S_0}$, $\mathbf{S_1}$ et $\mathbf{B_0}$ sont les éléments ciblés par défaut. UTILISER B_1 et UTILISER S_{-1} contrôlent l'ajout du deuxième token du tampon et le dernier token réduit, respectivement.

En pratique, techniquement, la librairie logicielle Keras que nous utilisons nécessite de fixer une taille précise pour la séquence d'entrée d'une couche récurrente. Nous utilisons donc pour cette couche récurrente la technique dite du « padding » [45] en anglais, c'est-à-dire tronquer les séquences plus longues qu'un seuil et remplir avec des éléments nuls les séquences moins longues que ce seuil. Ici nous utilisons un padding avec les 50 premiers tokens de la phrase.

Vocabulaire : La génération et la généralisation des vocabulaires (formes lexicales et étiquettes morphosyntaxiques) de cette variante est totalement identique à la procédure définie pour MLP.

Couches lexicales : Dans ce modèle, nous ne partageons pas les couches de plongements (forme lexicale et étiquette) fournis en entrée de la couche récurrente, et les couches de plongements (forme lexicale et étiquette) pour les éléments ciblés. Nous avons ainsi outre les hyperparamètres LEMMATISATION, DIMENSION (TOKEN), DIMENSION (POS) déjà vus pour le MLP, deux autres hyperparamètres DIMENSION INPUTREC (TOKEN) et DIMENSION INPUTREC (POS) pour la taille des plongements en entrée de la couche récurrente.

Couche dense : Comme en MLP, les hyperparamètres NOMBRE DE NEURONES, FONCTION D'ACTIVATION et DROPOUT déterminent les dimensions et le comportement de la couche dense.

Couches récurrentes : Le TYPE DE RÉCURRENCE de la couche récurrente peut être GRU ou LSTM. Les hyperparamètres DROPOUT et TAILLE VECTEURS DE SORTIE (un pour la récurrence sur plongements lexicaux et un pour la récurrence sur plongements d'étiquettes) définissent la dimension et le fonctionnement des deux couches récurrentes.

Pour cette variante ainsi que toutes les les autres variantes de cette section, la fonction d'activation de la couche récurrente est *ReLu*.

45. Nous plus de lisibilité nous gardons ce terme en anglais, littéralement « rembourrage », que l'on pourrait plus précisément traduire ici par tronquage-remplissage.

Apprentissage : Ce modèle utilise la même méthode d'optimisation que pour MLP (Optimisateur *Adagrad*). Le TAUX D'APPRENTISSAGE est soumis au réglage.

4.2.3 Variante 2 : MLP-R-Stack

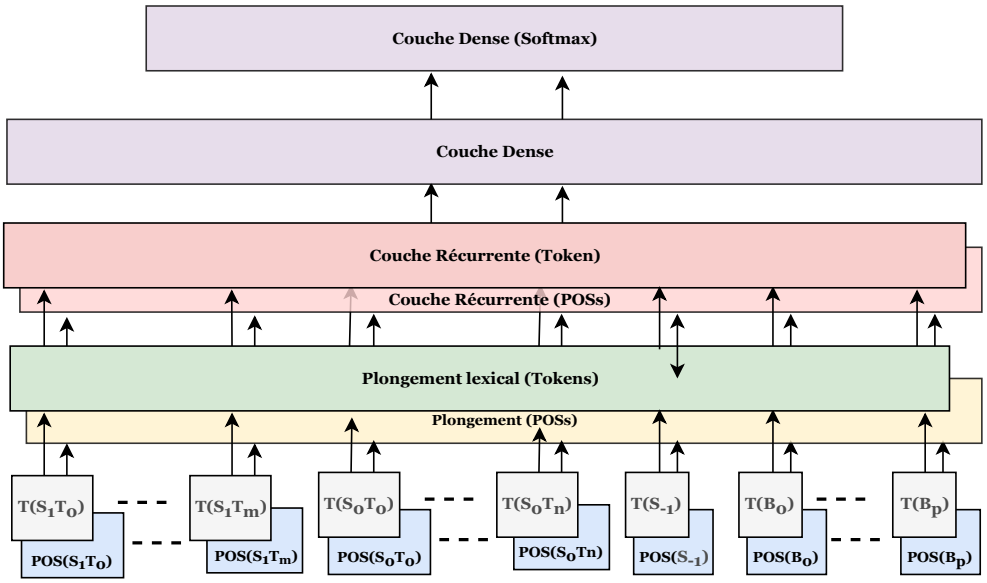

FIGURE 4.1 – **Architecture du MLP-R-Stack**

Dans ce modèle MLP-R-Stack et le suivant MLP-R-Stack2, nous travaillons sur la représentation des unités multitokens : dans les modèles précédents, la concaténation des attributs des composants des unités multitokens (comme *faire_face* ou *VERB_NOUN*) crée des éléments de vocabulaire rares, complètement séparés de la représentation de leurs composants.

Dans MLP-R-Stack, nous évitons cette concaténation et utilisons une couche récurrente pour les formes lexicales et une pour les étiquettes. Plus précisément nous créons une séquence avec les tokens des éléments ciblés (par défaut $\mathbf{S_0}, \mathbf{S_1}, \mathbf{B_0}$, plus d'autres tokens du tampon ainsi que le dernier élément réduit $\mathbf{S_{-1}}$ selon l'hyperparamétrage) et y appliquons une couche récurrente (une pour les tokens - forme ou lemme, et une pour les étiquettes). Pour garder une information de structure dans la séquence d'entrée, on utilise un nombre fixe de positions pour les unités qui peuvent être multitokens ($\mathbf{S_0}, \mathbf{S_1}, \mathbf{S_{-1}}$), plus précisément un padding de 4, 2 et 1 tokens[46] les plus à droite pour respectivement $\mathbf{S_0}$, $\mathbf{S_1}$ et $\mathbf{S_{-1}}$. Par exemple si les éléments ciblés sont réduits à $S_{-1} = m_0$, $\mathbf{S_1} = (\mathbf{m1}, (\mathbf{m_2}, \mathbf{m_3}))$, $\mathbf{S_0} = (\mathbf{m_4}, \mathbf{m_5})$ et $\mathbf{B_0} = \mathbf{m_6}$, et si on note **dummy** l'élément nul utilisé pour remplissage lors du padding, la couche récurrente s'applique à la séquence $\mathbf{S_1}$ paddé à 2, $\mathbf{S_0}$ paddé à 4, suivi de $\mathbf{B_0}$ et de S_{-2} paddé à 1, i.e. $\mathbf{m_2}, \mathbf{m_3}, \mathbf{dummy}, \mathbf{dummy}, \mathbf{m_4}, \mathbf{m_5}, \mathbf{m_6}, \mathbf{m_0}$.

L'architecture MLP-R-Stack est la même que MLP, modulo ces deux couches récurrentes, qui s'intercale entre les couches lexicales des tokens et étiquettes d'une part, et la couche dense du modèle de l'autre part (voir figure 4.1). La couche dense est alimentée soit par la concaténation de tous les vecteurs de sortie (Conc), soit uniquement par le dernier vecteur de chacune des 2 couches récurrentes (Dern), en fonction de la valeur de l'hyperparamètre VECTEURS DE SORTIE.

46. Étant donné que dans leur grande majorité les EPs sont de longueur 2 (voir par exemple la table 2.2, Partie 1, section 2.3, 25) ce padding ne paraît pas trop perdre d'information, i.e. il sera rare que des composants des unités multitokens soient tronquées

Mise en œuvre et hyperparamètres :

Éléments d'entrée : L'hyperparamètre NOMBRE DE TOKENS SUPPLÉMENTAIRES DU B contrôle le nombre des tokens du tampon à ajouter aux séquences d'entrée en plus de B_0, et UTILISER S_{-1} contrôle l'ajout le dernier token réduit.

Vocabulaire : Comme nous abandonnons la concaténation des composants des unités multi-tokens, la génération de vocabulaire est plus simple que pour le modèle MLP. Le **Vocabulaire exhaustif** (VOCAB. COMPACT = False) comprend d'une part tous les tokens ayant une fréquence supérieure à un et/ou appartenant une EP. D'autre part, pour chaque token marginal (fréquence = 1 et n'appartenant à aucune EP d'entraînement) une condition probabiliste est appliquée pour décider s'il faut l'inclure au vocabulaire ou pas. Le **Vocabulaire compact** (VOCAB. COMPACT = True) ne retient que les tokens apparaissant au moins dans une EP du jeu d'entraînement.

L'hyperparamètre LEMMATISATION détermine si les entrées du vocabulaire des tokens, compact ou exhaustif, doivent être les formes fléchies ou leurs lemmes. Chacun des deux vocabulaires comprend en outre trois entrées fictives supplémentaires afin de représenter **(1)** les positions vides de la pile ou du tampon ainsi que les positions vides des sous-séquences de S_0 et S_1 et d'un élément de tampon inexistant, **(2)** les tokens inconnus et **(3)** numériques. Le vocabulaire des étiquettes morphosyntaxiques est extrait exactement de la même manière.

Autres : Les hyperparamètres concernant les couches lexicales, la couche dense, les couches récurrentes et l'apprentissage sont les même que ceux définis pour la variante MLP-R-Sent (cf. 4.2.2.0).

4.2.4 Variante 3 : MLP-R-Stack2

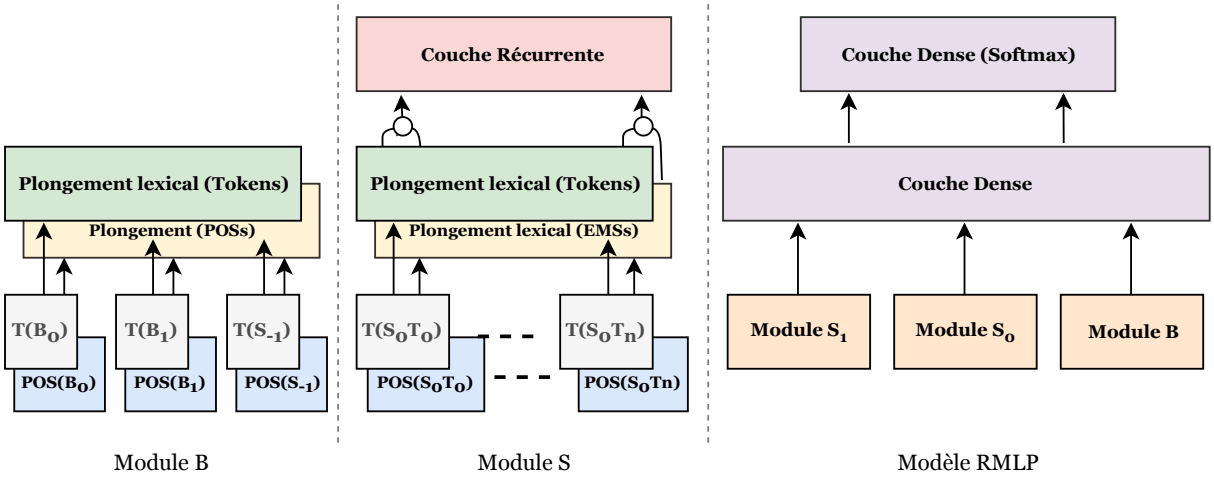

FIGURE 4.2 – **MLP-R-Stack2 :** L'architecture du modèle MLP-R-Stack2.

Comme la variante MLP-R-Stack, la variante MLP-R-Stack2 explore une représentation récurrente de $\mathbf{S_0}$ et $\mathbf{S_1}$, au lieu d'une concaténation de leurs composants. La différence avec MLP-R-Stack est qu'ici $\mathbf{S_0}$ et $\mathbf{S_1}$ ont chacun leur propre couche récurrente indépendante (comme pour MLP-R-Stack, avec un padding de 4 et 2 respectivement avant passage dans la couche récurrente, et chaque couche récurrente prend en entrée une concaténation de plongement lexical et plongement d'étiquette). Les vecteurs résultant de ces deux couches récurrentes, sensés résumer le

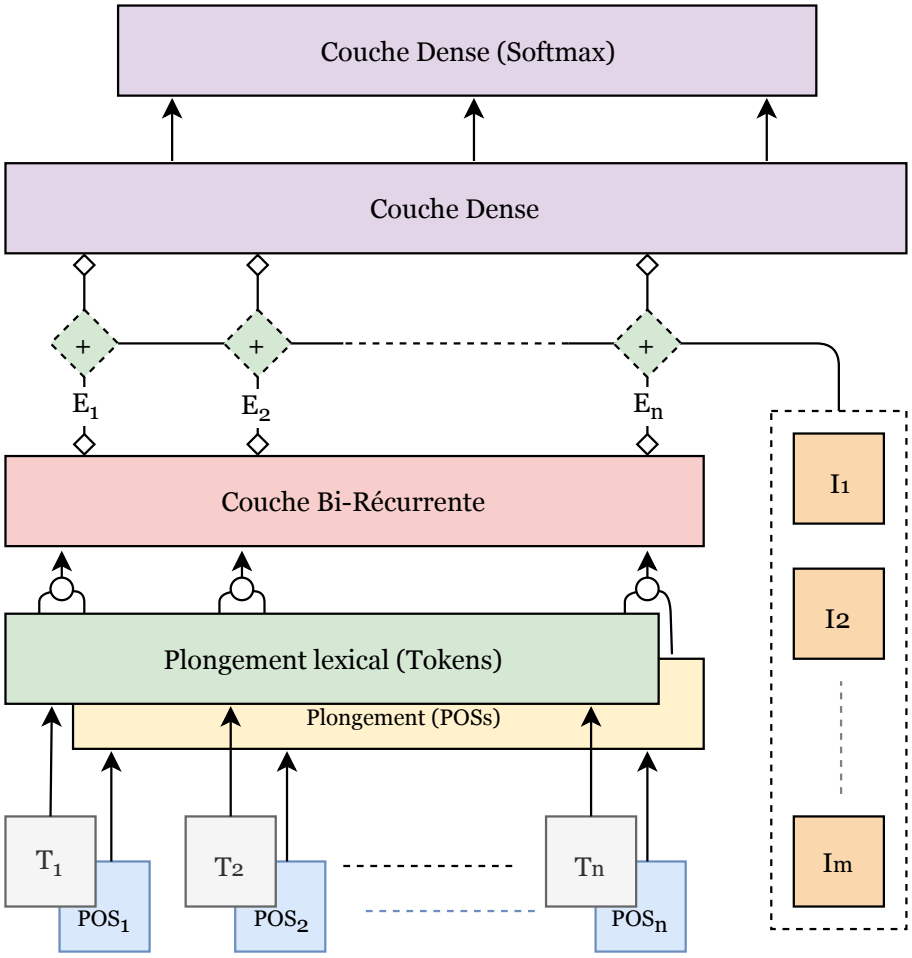

FIGURE 4.3 – **KG-2016 :** Le modèle KG-2016 reçoit en entrée les tokens (lemmes ou formes des mots) $\mathbf{T_1} : \mathbf{T_n}$ et les étiquettes $\mathbf{POS_1} : \mathbf{POS_n}$ de chaque phrase pour produire des vecteurs contextuels (via une couche bi-récurrente). Le modèle reçoit également pour chaque configuration la liste des indices des éléments ciblés $\mathbf{I_1} : \mathbf{I_m}$ ($m : 8$), ce qui permet de sélectionner la liste des vecteurs récurrents à passer à la couche dense (vecteur nul dans le cas d'un élément inexistant, par exemple si le tampon n'a qu'un élément, $\mathbf{B_1}$ n'existe pas).

contenu des éléments du haut de la pile, sont ensuite passés à une couche dense du modèle MLP, qui reçoit également directement les plongements pour les éléments ciblés du tampon (comme dans le modèle MLP, voir figure 4.2).

Cette variante partage la plupart des hyperparamètres définis pour MLP-R-Stack (on n'a cependant pas ici une couche récurrente pour les éléments lexicaux et une pour les étiquettes, on a donc un seul hyperparamètre TAILLE VECTEURS DE SORTIE).

4.3 MLP avec représentation récurrente des éléments ciblés (KG-2016)

Comme décrit dans notre état de l'art, Partie 1, section 4.3.2, page 69, Kiperwasser et Goldberg (2016) ont été les premiers à proposer d'intégrer une couche récurrente à un analyseur syntaxique à transitions avec classifieur neuronal du type de Chen et Manning (2014), obtenant ainsi un modèle plus générique (sélectionner les traits signifie choisir quels éléments de la configuration à cibler) et des scores très compétitifs. Il s'agit simplement d'appliquer, avant l'apprentissage ou la prédiction pour une phrase, un LSTM bi-directionnel [47]. Les vecteurs de sortie du LSTM constituent des représentations vectorielles contextualisées de chaque position de la phrase. On peut alors utiliser ces vecteurs pour les éléments ciblés de la configuration, au lieu de simples vecteurs denses non contextuels (comme dans notre variante MLP). Techniquement, à l'apprentissage, la mise à jour des paramètres se fait lorsque toute une phrase a été utilisée, et modifie jusqu'aux paramètres du réseau bi-LSTM.

Nous reproduisons ce modèle, mais en utilisant notre jeu de transitions $\mathcal{T}_2$, de manière à réaliser l'identification d'EPs au lieu de l'analyse syntaxique. Les éléments ciblés sont également spécifiques à notre système : nous utilisons un vecteur récurrent pour $\mathbf{B_0}, \mathbf{B_1}$, les deux premiers tokens de $\mathbf{S_1}$ et les quatre premiers tokens de $\mathbf{S_0}$ (avec toujours un vecteur nul en cas d'élément inexistant).

Hyperparamètres : À l'exception de l'hyperparamètre RNN BI-DIRECTIONNEL qui détermine si la couche récurrente doit être bidirectionnelle ou pas, et de l'hyperparamètre NOMBRE DE COUCHES qui détermine le nombre de couches récurrentes empilées, cette variante partage la plupart des hyperparamètres du MLP-R-Stack, en ce qui concerne les couches lexicales, denses et récurrentes, les vocabulaires et l'apprentissage.

4.4 Réglage des hyperparamètres

KG-2016 : Le réglage des hyperparamètres pour KG-2016 est plus difficile en pratique, car l'utilisation d'un mini-batch à l'apprentissage est beaucoup plus complexe qu'avec nos variantes vues jusqu'à présent. On ne peut pas tronquer les phrases avant l'entrée dans la couche récurrente, comme fait pour le modèle MLP-R-Sent, car on a besoin des vecteurs récurrents de potentiellement toutes les positions. Nous avons fait une implémentation avec une mise à jour des paramètres par phrase. Sans rééquilibrage des proportions des transitions dans les exemples d'apprentissage, l'apprentissage pour une langue de taille de corpus moyen prend environ 30 minutes, contre 5 par exemple pour MLP avec rééquilibrage (qui augmente pourtant drastiquement le nombre d'exemples d'apprentissage).

Pour cette raison, nous avons réduit le nombre d'expériences pour le réglage de KG-2016, en n'utilisant au départ qu'une seule langue pilote (le bulgare) et 500 expériences, entraînées sur une partie tronquée du corpus d'apprentissage (270 milles tokens) et évaluées sur le corpus de développement, avec une seule graine d'initialisation.

Le tableau 4.1, ci-dessous, rend compte de tous les hyperparamètres utilisés lors du réglage de cette variante. Les résultats très modestes obtenus, vraisemblablement à cause de l'absence de rééquilibrage des classes, nous a conduits à abandonner ce modèle pour nous concentrer sur les variantes récurrentes présentées supra.

47. Un LSTM bi-directionnel concatène les vecteurs de sortie d'un LSTM de gauche à droite et d'un LSTM de droite à gauche

Type	Hyperparamètre	Plage	Valeur
Lexicale	Lemmatisation	{True, False}	TRUE
	Dim.(Token)	[10, 100]	239
	Dim.(POS)	[5, 50]	41
	Vocabulaire	{Comp., Exhau.}	Comp.
	Plong. Modifiable	{True, False}	TRUE
Dense	Nombre de Neurones	[10, 100]	11
	Dropout	[0, .5]	.3
	Fonction d'activation	{Relu, Tanh}	Tanh
Récurrente	Type de couche	{GRU, LSTM}	LSTM
	Nombre de couches	{1, 2}	1
	Nombre de neurones	[50, 300]	90
	Bidirectionnel	{True, False}	TRUE
	Dropout	[0, .4]	.1
Appren.	Optimisateur	Adagrad*	Adagrad*
	Taux d'app.	[.01, .2]	.07
	Taille de batch	1	1

TABLE 4.1 – **Réglage des hyperparamètres du KG-2016 :** Hyperparamètres de la combinaison d'hyperparamètres la plus performante (CPP) pour KG-2016. Chaque **hyperparamètre** est accompagné par la **Plage** des valeurs possibles et sa **valeur**. Le symbole « * » réfère au fait que la valeur de l'hyperparamètre n'est pas soumise au réglage et fixée suite à un réglage préliminaire.

Réglage des autres variantes : Le réglage des variantes MLP-Wide, MLP-R-Sent, MLP-R-Stack, et MLP-R-Stack2 suit la même logique du réglage de la variante MLP, puisque nous lançons mille expériences sur chacune des mêmes langues pilotes (le bulgare, le portugais et le turc). Le modèle de chaque expérience est appris sur une partie tronquée du jeu d'entraînement (270 milles tokens) et évalué sur le jeu de développement. Comme pour le MLP, nous appliquons systématiquement un sous-échantillonnage en ignorant les phrases d'entraînement ne contenant aucune EP, et un suréchantillonnage aléatoire des transitions minoritaires, de manière à rééquilibrer les 4 transitions. Nous n'avons pas intégré au réglage ici les autres techniques de ré-échantillonnage investiguées pour le MLP (suréchantillonnage avec seuil de fréquence et multiplication de la perte) au vu de leur absence d'impact positif lors du réglage du MLP. Quant à l'initialisation, une seule graine est utilisée étant donné que ces variantes sont plus longues à entraîner, et que le rééquilibrage a clairement réglé le problème d'instabilité de performance observé initialement pour le MLP.

Concernant l'utilisation de plongements lexicaux préentraînés, nous intégrons cet hyperparamètre au réglage (contrairement à précédemment, où nous faisions un réglage en « closed track » et un en « open track », respectivement sans et avec plongements lexicaux préentraînés). Par ailleurs, nous avons fixé en amont certaines valeurs d'hyperparamètres (notées avec une astérisque dans le tableau 4.2), dans le cas où ils ressortaient de manière évidente dans des expériences préliminaires.

En ce qui concerne la sélection des combinaisons d'hyperparamètres, forts de notre expérience du réglage du MLP, nous utilisons directement les combinaisons de tendances. Pour un hyperparamètre à valeurs discrètes (respectivement continues), nous prenons la valeur la plus fréquente (respectivement moyenne) dans les 125 meilleures combinaisons.

	Hyperparamètre	Plage	MLP-Wide	MLP-R-Stack	MLP-R-Sent	MLP-R-Stack2
E. ciblés	Utiliser B_1	{T, F}	T	-	T	T
	Utiliser S_{-1}	{T, F}	T	T	T	T
	Tokens supp. du tampon	[1, 5]	-	3		-
Plongements	Lemmatisation	{T, F}	T	T	T	T
	Dim.(token)	[200, 500]	300	300	95	300
	Préentraînés	{T, F}	T	T	F*	T
	Dim.(POS)	[10, 100]	60	55	15	40
	Dim. InputRec (token)	[25, 200]	-	-	80	-
	Dim. InputRec (POS)	[5, 50]	-	-	18	-
	Fenêtre de voisinage	[1,4]	3	-	-	-
Vocab.	Vocabulaire	{Comp., Exh.}	Comp.	Exh.	Comp.	Comp.
	Généralisation	{T, F}	F	F	F	F
	Modifiable	{T, F}	T	T	T	T
	Moyennage	{T, F}	T*	-	T*	-
Dense	# Neurones	[25, 500]	215	80	115	90
	Dropout	{.1, .., .6}	.3	.3	-	.3
Couches récurrentes	Vecteurs de sortie	{Conc, Dern}	-	Dern	Dern*	Dern
	Type de récurrence	{LSTM, GRU}	-	GRU	GRU	GRU
	Taille Vec. de sortie	[50, 300]	-	-	110	75
	Taille Vec. de sortie (Rec. tokens)	[50, 300]	-	100	-	-
	Taille Vec. de sortie (Rec. POS)	[25, 100]	-	70	-	-
	Bi-directionnel	{T, F}	-	-	-	-
	Dropout	{.1, .., .5}	-	.2	0	.3
App.	Taux d'app.	[.01, .2]	.034	.044	.057	.038
	Taille du batch	{16,.. , 256}	64	160	64	256

TABLE 4.2 – **Réglage des hyperparamètres :** Combinaisons d'hyperparamètres de tendances (CT) obtenues pour les variantes MLP-Wide, MLP-R-Sent, MLP-R-Stack et MLP-R-Stack2. On donne pour chaque hyperparamètre sa **Plage** de valeurs possibles et les valeurs obtenues pour chaque variante. Le symbole « **-** » indique que l'hyperparamètre n'est pas utilisable dans la variante. Le symbole **« #** » signifie « nombre de », et **« * »** est utilisé quand la valeur de l'hyperparamètre n'est pas soumise au réglage mais fixée suite à des expérimentations préliminaires. Pour faciliter la visualisation du tableau, nous désignons True par **T**, False par **F**, Ré-échantillonnage par **Ech.**, Apprentissage par **App.**, Concaténation par **Conc**, Dernier par **Dern** et supplémentaire par **Supp.**. La plage précise pour la taille du batch est {16, 32, 48, 64, 96, 128, 256}.

Le tableau 4.2 rend compte du déroulement et des résultats du réglage des hyperparamètres des variantes MLP-Wide, MLP-R-Sent, MLP-R-Stack, et MLP-R-Stack2. Pour chaque hyperparamètre, le tableau donne la plage de valeurs possibles ainsi que les valeurs de tendances obtenues pour chaque variante de modèle.

On remarque ainsi que la lemmatisation et l'utilisation du deuxième token du tampon ($\mathbf{B_1}$) ainsi que du dernier token réduit ($\mathbf{S_{-1}}$) sont systématiquement choisies. Les plongements lexicaux préentraînés se montrent bénéfiques pour toutes les variantes qui les exploitent [48]. On constate également que la généralisation du vocabulaire n'est retenue pour aucune des variantes, et le vocabulaire compact est préféré pour toutes les variantes sauf MLP-R-Stack. Concernant la taille des plongements lexicaux et d'étiquettes, on remarque un ratio à peu près constant entre les deux (les plongements d'étiquette sont environ 1/6 des plongements lexicaux, sauf pour MLP-R-Stack2).

Le tableau montre également que toutes les variantes préfèrent l'utilisation d'une couche récurrente GRU, comprenant moins de paramètres, plus rapide à entraîner, et plus adaptée aux séquences courtes, au lieu d'une couche LSTM.

4.5 Expérimentations et résultats

KG-2016 : Le tableau 4.3 présente les scores $\mathbf{F}(\mathbf{eval-eps})$ de la variante KG-2016, ainsi que les autres variantes du chapitre, sur les jeux de test de PARSEME 1.1. Malgré nos efforts, force est de constater que la variante KG-2016 fonctionne très mal ($\mathbf{F_g}$(eval-eps) = 9.2). C'est selon nous vraisemblablement dû à l'absence de rééquilibrage des transitions à l'apprentissage. En effet, nous avons pu constater pour la variante MLP un saut énorme de performance avec l'ajout du rééquilibrage (cf. 3.6.1).

Nous rappelons que nous n'avons pas implémenté d'apprentissage en mini-batch pour KG-2016, ce qui donne des temps d'entraînement très longs. Le rééquilibrage dans ce contexte ne pouvait pas être testé, cf. il aurait multiplié par environ 100 la taille du corpus d'apprentissage, alors même que notre implémentation sans batch donne des temps de traitement pour une langue d'environ 30mn. Quelques essais de de multiplication de la perte n'ont pas amélioré les résultats, et nous avons préféré nous concentrer sur nos propres variantes.

Autres variantes : Le tableau 4.3 présente les scores $\mathbf{F}(\mathbf{eval-eps})$ de toutes les variantes, apprises sur les jeux d'entraînement et les jeux de développement (lorsqu'ils ont disponibles) et évaluées sur les jeux de test des données PARSEME 1.1, en utilisant la combinaison d'hyperparamètres de tendances (à l'exception du KG-2016 qui utilise la combinaison la plus performante (CPP)). Le tableau donne également les moyennes des scores $\mathbf{F_g}$, $\mathbf{F_{AVG}}$, $\mathbf{P_{AVG}}$ et $\mathbf{R_{AVG}}$ en (eval-eps) pour toutes les variantes. Nous ajoutons les scores des variantes SVM_2(CT) et MLP(CT) à ce tableau et aux autres tableaux de ce chapitre pour fournir un point de repère et donner une vision plus générale du comportement de notre méthode (il s'agit des scores du MLP de l'open track parce que les variantes récurrentes intègrent les plongements lexicaux préentraînés dans leur procédure de réglage).

Si on s'en tient d'abord aux résultats moyennés sur toutes les langues, on peut constater d'une part que la variante MLP-Wide (variante non récurrente, qui ajoute simplement du contexte linéaire autour des éléments ciblés) obtient la meilleure performance tous systèmes confondus, surpassant légèrement MLP. En revanche globalement, les variantes récurrentes dégradent les

48. Nous avons par erreur omis de les utiliser pour la variante MLP-R-Sent.

Langue	MLP -Wide	MLP -R-Sent	MLP -R-Stack	MLP -R-Stack2	KG-2016	MLP $_o$	SVM_2
BG	**66.4**	64.3	64.8	64.1	12.4	59.6	63.5
DE	**52.5**	48.1	45.4	36.8	6.8	51.4	47.5
EL	**64.4**	61.3	58.2	50.9	11.9	50.8	56.4
EN	32.2	31.4	23.9	26.2	10.7	**33.3**	26.9
ES	40.1	37.9	34.1	31.5	12.3	**53.7**	38.6
EU	76.8	66.7	70.0	70.7	14.6	71.1	**81.8**
FA	**77.8**	76.5	77.0	74.2	3.0	65.9	68.3
FR	59.6	57.1	54.4	44.5	3.4	**74.7**	59.2
HE	**47.3**	42.2	30.3	28.2	2.8	35.1	37.6
HI	64.0	62.6	61.9	64.3	1.5	**68.8**	64.0
HR	**56.6**	55.9	28.0	48.6	3.6	55.3	56.4
HU	93.0	91.4	91.0	89.0	9.0	**94.4**	91.1
IT	**56.8**	51.5	46.1	39.0	10.3	52.9	56.2
LT	43.6	43.6	40.1	29.0	15.2	**60.9**	36.3
PL	**73.8**	66.3	68.0	60.9	2.9	71.9	68.1
PT	66.2	52.8	63.2	55.7	11.8	**70.6**	67.7
RO	82.8	81.0	**85.1**	71.9	17.2	84.2	77.8
SL	51.9	40.3	**50.2**	45.9	9.1	58.7	**61.8**
TR	55.7	**56.0**	50.2	50.2	0.6	45.1	40.0
$\mathbf{F_g}$	**62.6**	59.0	55.6	52.7	9.2	62.4	59.3
F_{avg}	**61.1**	57.2	54.8	51.7	8.4	61.0	57.9
P_{avg}	73.4	66.3	63.9	62.6	30.2	81.4	**85.6**
R_{avg}	**54.5**	53.1	49.2	45.5	5.4	50.7	45.4

TABLE 4.3 – **PARSEME 1.1 : Résultats** les scores **F** (en eval-eps) de nos variantes pour chaque langue sur les jeux de test des données PARSEME 1.1 en utilisant la combinaison d'hyper-paramètres de tendances (à l'exception du KG-2016 qui utilise la combinaison la plus performante CPP). Les quatre dernières lignes dénotent les scores $\mathbf{F_g}$, $\mathbf{F_{AVG}}$, $\mathbf{P_{AVG}}$ et $\mathbf{R_{AVG}}$ en (eval-eps) des variantes.

performances : dans MLP-R-Sent, l'ajout du vecteur récurrent représentant toute la phrase dégrade les résultats par rapport à MLP ($\mathbf{F_g}$ =59.0 contre 62.4 pour MLP). Ces tendances masquent cependant de fortes disparités selon les langues.

En termes de rappel et précision, tout d'abord, on a globalement un rappel nettement moins bon que la précision, mais cette tendance est moins forte pour les variantes de ce chapitre par rapport à SVM_2 et MLP (pour lesquels on a un écart de plus de 30 points entre rappel et précision). On a toujours la tendance que le SVM_2 est le meilleur en précision. MLP-Wide a une performance comparable à MLP ($\mathbf{F(eval-eps)}$ =62,6 versus 62,4) mais le déséquilibre mauvais rappel / bonne précision est moins fort pour MLP-Wide que pour MLP.

MLP-Wide : La bonne performance de ce système masque de fortes disparités entre langues. MLP-Wide produit le meilleur score **F** sur huit langues, avec dans certains cas des écarts très importants (par ex. 13 points pour le grec, 12 points pour le farsi). Il produit des scores très compétitifs (à moins de 2 points du meilleur score) pour trois langues et assez compétitifs (écart de moins de 5 points) pour quatre autres langues, mais il s'avère très mauvais pour quatre langues :

le slovène (10 points d'écart), l'espagnol (14 points d'écart), le français (15 points) et le lituanien (17 points d'écart). Alors que cette baisse est prévisible pour le slovène qui ne comprend pas d'annotations morphosyntaxiques dans ses jeux de données, il est difficile de l'expliquer pour les autres langues. Pour le lituanien, la taille est également assez petite, et on peut noter la très faible densité des EPs (6 % versus une moyenne de 39 %) et par ailleurs 63 % des EPs du jeu d'entraînement sont des EPs rarement vues i.e vues moins de 5 fois (la moyenne est de 37 %). Les mauvais résultats sur le corpus de test pour le français et l'espagnol sont d'autant plus étonnants que ceux sur le corpus de développement étaient les meilleurs.

MLP-R-Sent : Cette variante produit le meilleur score pour le turc uniquement, et des scores très compétitifs (moins de deux points avec le meilleur score) pour l'anglais, le farsi et le croate, des scores assez compétitifs (moins de deux points avec le meilleur score) sur cinq autres langues, et est mauvais sur les dix langues restantes. D'après les pourcentages fournis par Pasquer et al. (2018) des occurrences idiomatiques des EPs des jeux d'entraînement de PARSEME 1.1 (par opposition aux occurrences littérales, non interprétées comme EP), les occurrences idiomatiques du turc ne dépassent pas 4 % (versus un moyen de 49 % de toutes les langues). Ces chiffres pourraient indiquer que cette variante gère mieux les cas où la lecture idiomatique est minoritaire, mais cela est contredit par les performances non compétitives pour d'autres langues avec un faible taux d'idiomaticité (le slovène 15 %, l'espagnol 32 %, l'italien 32 % et le basque 33 %).

MLP-R-Stack : MLP-R-Stack produit le meilleur score du roumain et se montre très compétitif sur le bulgare et le farsi et assez compétitif sur le hongrois, mais donne des scores très en deça pour quinze langues. Le roumain et le bulgare disposent d'un grand corpus équilibré, c'est-à-dire, un corpus ayant des caractéristiques dans la moyenne pour ce qui est de la variabilité, la discontinuité, la longueur et la densité des EPs des jeux d'entraînement.

MLP-R-Stack2 : Cette variante, qui modélise les unités multitokens avec des couches récurrentes, ne produit des scores relativement compétitifs que sur trois langues (le bulgare, le farsi, et l'hindi), et n'obtient le meilleur score pour aucune des langues. La différence d'architecture assez minime par rapport à MLP-R-Stack a un impact très négatif.

En résumé, MLP-Wide et MLP obtiennent les meilleurs scores pour quinze langues, alors que SVM_2 est très compétitif et le plus performant sur le basque et le slovène, données sans étiquettes morphosyntaxiques. MLP-R-Sent se révèle assez compétitif, au contraire des MLP-R-Stack et MLP-R-Stack2. Globalement les précisions sont plus hautes que les rappels, les variantes MLP produisent les meilleurs rappels alors que SVM_2 a la meilleure précision.

4.6 Analyse de la performance par classes d'EPs

Nous présentons dans cette section une analyse de la performance des différentes variantes, pour diverses classes formelles d'EPs, sur les jeux de développement des langues de PARSEME 1.1. Ces variantes sont ici apprises sur les jeux d'entraînement et évaluées sur les jeux de développement lorsqu'ils sont disponibles. Sinon, nous entraînons les variantes sur 80 % du jeu d'entraînement et les évaluons sur le 20 % restant. Toutes les variantes sont apprises avec les combinaisons d'hyperparamètres de tendances.

Le tableau 4.4 fournit les scores $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$ pour différentes classes d'EPs, selon leur

	%	MLP -Wide	MLP -R-Sent	MLP -R-Stack	MLP -R-Stack2	SVM_2	MLP_o
Non vue	37	12.5	12.5	17.4	**19.0**	11.6	0.5
Vue	63	**82.6**	76.3	76.8	71.5	79.6	80.5
Vue(Fréq.)	26	**80.8**	75.0	77.2	73.3	80.3	76.0
Vue(Rar.)	38	**78.8**	72.2	72.4	65.8	75.4	78.1
Identique	33	**92.0**	83.7	85.8	83.2	91.0	88.3
Variante	31	73.4	67.0	66.0	58.6	68.4	**73.8**
Continue	67	**69.2**	59.8	62.1	59.6	**69.2**	67.4
Discontinue	33	**50.8**	42.7	42.0	32.6	44.7	49.8
Enchâssée	6	11.2	9.2	10.3	7.1	**18.5**	10.0
TP(Long.1)	6	82.9	79.7	14.0	13.1	**84.4**	13.9
Multitoken	94	**63.1**	54.5	55.2	51.6	61.3	61.4
Longueur 2	78	**66.1**	57.2	58.4	56.0	64.1	64.1
Longueur 3	13	40.1	30.3	33.4	19.0	**40.2**	41.0
Longueur 4	7	**20.4**	16.7	18.1	8.5	19.9	19.4

TABLE 4.4 – **Analyse de performance pour différentes classes formelles d'EPs :** scores $\mathbf{F_{avg}(eval-eps)}$ des variantes MLP-Wide, MLP-R-Sent, MLP-R-Stack, MLP-R-Stack2, SVM_2 et MLP, apprises sur les jeux d'entraînement de PARSEME 1.1 et évaluées sur les jeux de développement si disponibles (sinon, nous entraînons les variantes sur 80 % du jeu d'entraînement et les évaluons sur le 20 % restant) en utilisant les combinaisons CT. Col 1 : classe d'EP considérée. Col 2 : proportion moyenne de cette classe d'EP dans les jeux d'evaluation gold.

fréquence dans le corpus d'entraînement (non vues versus vues – dont vues et fréquentes, vues mais rarement (< 5)), pour celles vues à l'apprentissage, selon qu'elles ont été vues à l'identique ou comme variante , selon leur caractère discontinu ou pas, et selon leur longueur.
Ce tableau permet de constater que :

— MLP-Wide se montre robuste, étant donné qu'il s'impose sur la plupart des catégories d'EPs, à l'exception notable des EPs enchâssées où SVM_2 est nettement meilleur, des EPs non vues, des EPs de longueur 1.

— MLP se montre très compétitif sur la plupart des catégories, mais totalement impuissante en ce qui concerne les EPs non vues ($\mathbf{F = 0.5}$) et non compétitive pour les EPs enchâssées ($\mathbf{F = 10.0}$). On peut remarquer que le contexte linéaire des éléments ciblés ajouté en entrée pour MLP-Wide améliore les résultats de MLP, particulièrement sur les EPs fréquentes et identiques, preuve que l'apprentissage de la partie récurrente nécessite un certain nombre d'exemples.

— SVM_2 est particulièrement bon sur les EPs enchâssées, les TPs et les EPs continues, alors qu'il a visiblement des difficultés pour les EPs discontinues.

— alors que MLP-R-Stack et MLP-R-Sent obtiennent des scores relativement médiocres pour la plupart des catégories, ils se révèlent les meilleurs sur les EPs non vues, qui sont l'énorme point faible de notre meilleur système SVM_2. Même si les performances restent modestes ($\mathbf{F(eval-eps)}$ =19.0), ceci suggère que la représentation récurrente des unités de la pile fournit un peu de pouvoir de généralisation. On peut noter que cela suggère une piste

	MLP -Wide	MLP -R-Sent	MLP -R-Stack	MLP -R-Stack2	SVM_2	MLP_o
$\mathbf{R_{avg}}$	**54.5**	53.1	49.2	45.5	45.4	50.7
Non vue	**72.5**	69.4	61.8	54.6	65.6	71.8
Vue	27.5	30.6	38.2	**45.4**	34.4	28.2
Vue(Fréq.)	5.8	7.5	**10.1**	12.7	9.2	6.0
Vue(Rare.)	21.7	23.2	28.1	**32.7**	25.2	22.3
Longueur 3	17.3	17.4	15.6	**18.6**	15.7	16.8
Longueur 2	73.2	73.4	75.3	72.2	**75.5**	74.1
MWTs	2.8	2.8	3.0	**3.1**	**3.1**	**3.1**
LVC.full	41.7	41.5	41.9	39.7	**43.3**	41.4
VID	**30.8**	29.9	28.4	29.7	29.4	30.0
IRV	10.0	10.6	**11.3**	**11.3**	9.2	9.9
VPC.full	4.7	4.7	**5.3**	**5.3**	5.2	5.2

TABLE 4.5 – **EPs non identifiées (silence) :** Ligne 2 : rappel moyen pour chaque variante. Lignes suivantes : pour chaque classe d'EPs, proportion d'EPs de cette classe parmi les EPs non identifiées (par exemple : pour MLP-Wide, 72.5 % des EPs non identifiées sont des EPs non vues). Variantes apprises sur les jeux d'entraînement de PARSEME 1.1 et évaluées sur jeux de développement si disponibles (sinon, nous entraînons les variantes sur 80 % du jeu d'entraînement et les évaluons sur le 20 % restant), en utilisant les combinaisons CT.

d'amélioration globale, en combinant les prédictions de SVM_2 pour les EPs vues, et celles de MLP-R-Stack2 pour les EPs non vues.

4.7 Analyse d'erreurs

Cette section présente une analyse des erreurs des variantes de notre méthode sur les jeux de développement des langues de PARSEME 1.1, en considérant le silence (les EPs non identifiées) et le bruit (les EPs identifiées à tort). On considère l'évaluation eval-eps, c'est-à-dire qu'une EP est considérée comme correcte si tous ces composants sont groupés et uniquement ceux-là. Comme indiqué supra lors du commentaire de la table 4.3, nos systèmes sont problématiques surtout pour ce qui est du rappel, et moins pour la précision, même si cette tendance est moins vraie pour nos variantes récurrentes que pour SVM_2 et MLP.

Le tableau 4.5 analyse le silence des différentes variantes : la première ligne montre le rappel moyen de chaque variante, et les lignes suivantes donnent pour différentes classes formelles d'EPs, la proportion d'EPs de cette classe parmi les EPs non identifiées.

On retrouve que MLP-Wide est surtout bon sur les EPs vues et a un mauvais rappel sur les EPs non vues (il a la proportion d'EPs non vues la plus forte parmi les EPs qu'il n'a pas identifiées). En comparaison, cette proportion est un peu moins grande pour les variantes intégrant une couche récurrente, par exemple les EPs non vues représentent 54.6 % des EPs non identifiées par MLP-R-Stack2). Les différences concernant les longueurs ou les catégories d'EPs ne sont pas frappantes entre les variantes.

Le tableau 4.6 analyse cette fois le bruit généré par les différentes variantes : la première ligne montre la précision moyenne de chaque variante, et les lignes suivantes donnent pour différentes classes formelles d'EPs, la proportion d'EPs de cette classe parmi les EPs identifiées à tort.

	MLP -Wide	MLP -R-Sent	MLP -R-Stack	MLP -R-Stack2	SVM_2	MLP_o
$\mathbf{P_{avg}}$	73.4	66.3	63.9	62.6	**85.6**	81.4
Non vue	45.0	60.5	75.3	**78.3**	35.2	7.5
Vue	55.0	39.5	24.7	21.7	64.8	**92.5**
Vue(Fréq.)	11.9	7.8	7.4	6.5	15.9	**26.0**
Vue(Rare.)	43.1	31.7	17.3	15.3	48.9	**66.6**
MWTs	0.6	1.6	1.6	5.1	3.2	**5.9**
Longueur 2	83.9	84.0	76.9	78.3	85.0	**85.5**
Longueur 3	8.3	8.3	**16.6**	12.1	9.9	7.3

TABLE 4.6 – **EPs identifiées à tort (bruit) :** Ligne 2 : précision moyenne pour chaque variante. Lignes suivantes : pour chaque classe d'EPs, proportion d'EPs de cette classe parmi les EPs identifiées à tort (par exemple : pour MLP-Wide, 55 % des EPs identifiées à tort sont des EPs vues à l'apprentissage). Variantes apprises sur les jeux d'entraînement de PARSEME 1.1 et évaluées sur jeux de développement si disponibles (sinon, nous entraînons les variantes sur 80 % du jeu d'entraînement et les évaluons sur le 20 % restant), en utilisant les combinaisons CT.

On retrouve que SVM_2 a la meilleure précision, suivi de MLP. On rappelle que c'est MLP-R-Stack2 la meilleure de nos variantes sur les EPs non vues à l'apprentissage, alors que SVM_2 est pratiquement nul pour celles-ci. Cela donne sans surprise ici que parmi les EPs bruyantes, SVM_2 a très peu d'EPs non vues ($7,5$ %), alors que c'est MLP-R-Stack2 qui en a le plus (78.3 % de ses EPs bruyantes sont non vues).

4.8 Résumé

Dans ce chapitre, nous avons présenté des variantes neuronales pour le classifieur prédisant les transitions au sein de notre système par transitions pour l'identification d'EPs, et avons présenté les résultats obtenus sur les données PARSEME 1.1. L'objectif est d'augmenter le pouvoir de généralisation, en intégrant davantage de contexte que ne le fait le MLP, qui n'utilise que quelques éléments ciblés de la configuration.

La première variante, appelée MLP-Wide, ajoute du contexte en intégrant en entrée du classifieur les étiquettes morphosyntaxiques des voisins linéaires des éléments ciblés de la configuration. Les expérimentations sur les données PARSEME 1.1 montrent que cette extension améliore légèrement les résultats globaux du système, mais plus nettement les résultats sur les EPs non vues à l'apprentissage (F_{avg} = 12,5 vs. 0,5).

Les autres variantes utilisent des couches récurrentes. La variante MLP-R-Sent exploite la sortie d'une couche récurrente sur toute la phrase, afin de prendre en compte le contexte global de la phrase. La variante MLP-R-Stack n'ajoute pas de contexte par rapport au MLP mais considère l'ensemble des éléments ciblés de la configuration comme une séquence modélisée par une couche récurrente, afin d'en extraire un "résumé". La variante MLP-R-Stack2 représente les unités multitokens de la pile avec des couches récurrentes représentant la séquence des tokens de ces unités (alors que MLP considère une unité multitoken comme un élément atomique du vocabulaire). Les expérimentations ont montré que l'intégration des couches récurrentes au sein du modèle MLP a permis d'améliorer la généralisation de ces variantes (F_{avg} allant de 12,5 à 19), malheureusement au prix d'une dégradation importante de la performance globale de ces variantes (entre 3 et 10 points d'écart en F_{avg}).

Dans ce chapitre, nous avons également présenté un modèle inspiré du travail de Kiperwasser et Goldberg (2016), où les éléments ciblés sont contextualisés en utilisant leur représentation issue d'une couche récurrente sur toute la phrase. Cette variante, appelée KG-2016, ne produit pas de scores compétitifs et stables. En effet, cette approche ne permet pas l'application (du moins de manière simple) de notre technique de ré-échantillonnage pour rééquilibrer la proportion des classes (i.e. transitions) à l'apprentissage. Cela pourrait constituer l'une des raisons principales de cette performance décevante.

Chapitre 5

Apprentissage multitâche pour l'identification d'EPs

L'apprentissage conjoint de plusieurs tâches (ou apprentissage multitâche) est une approche florissante de l'apprentissage profond. Cette approche est utilisée avec succès pour beaucoup de tâches de TAL (Torrey et Shavlik, 2010; Zhang et Weiss, 2016; Collobert et al., 2011).

Concernant l'identification des EPs, il existe une large littérature sur la modélisation non-neuronale jointe de cette tâche avec d'autres tâches telles que l'étiquetage morphosyntaxique ou l'analyse syntaxique en dépendances (cf. le chapitre 2 de la partie 1). Cependant, elle ne permet pas vraiment de tirer de conclusions définitives sur le sujet, même si l'on observe des cas extrêmement bénéfiques comme pour l'identification de certains types de mots grammaticaux complexes apprise de manière jointe avec l'analyse syntaxique en dépendances (Nasr et al., 2015). Concernant la modélisation multitâche neuronale de l'identification des EPs, la littérature est beaucoup plus maigre. En particulier, Taslimipoor et al. (2019) montre l'impact positif d'une architecture multitâche pour l'identification des EPs par étiquetage séquentiel, quand elle intègre l'étiquetage morphosyntaxique et l'analyse syntaxique comme tâches auxiliaires.

Dans ce chapitre, nous souhaitons évaluer l'impact que l'intégration de ces deux mêmes tâches auxiliaires dans une architecture multitâche peut apporter à notre tâche cible d'identification des EPs par un système par transitions. Pour cela, nous utilisons la méthode d'empilement de Zhang et Weiss (2016) qui a été utilisée pour combiner l'analyse syntaxique par transitions et l'étiquetage morphosyntaxique.

Nous allons donc, dans ce chapitre, décrire l'architecture de notre système multitâche et de ses composants (section 5.1). La section 5.2 aborde les algorithmes d'optimisation des réseaux de neurones proposés et les différents modes d'apprentissage du système multitâche. La section 5.3 détaille les changements du cadre expérimental qu'exige l'expérimentation de ces réseaux. Enfin, la section 5.4 rend compte de la performance de ce système, et la compare à celle d'un système de base et celle de notre meilleur système d'identification.

5.1 Architecture et composants

Notre système multitâche est inspiré du travail de Zhang et Weiss (2016) à la fois pour l'architecture du réseau de neurones et son algorithme d'optimisation. L'architecture se caractérise par le partage d'un module produisant une représentation vectorielle contextualisée pour chaque élément en entrée (ci-après **représentation contextualisée**). Cette représentation est ensuite utilisée pour nourrir trois modules distincts, chacun dédié à une tâche, i.e. étiquetage morphosyn-

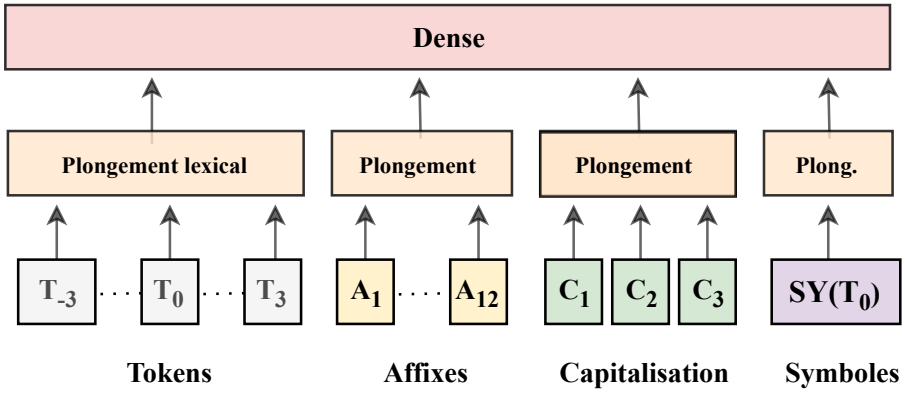

FIGURE 5.1 – **Module partagé (MP) :** Module partagé par tous les composants de la structure multitâche (dont les paramètres sont modifiés pour l'apprentissage de toutes les tâches) . $\mathbf{T_0}$ représente le token à contextualiser de la phrase alors que $\mathbf{T_{1,2,3}}$ représentent les tokens voisins à droite et $\mathbf{T_{-1,-2,-3}}$ représentent les tokens voisins à gauche. $\mathbf{A_i}$ représente les suffixes et préfixes de deux et trois caractères de T_{-1}, T_0, T_1, $\mathbf{C_i}$ représente les traits de capitalisation de T_{-1}, T_0, T_1 (si le premier caractère est majuscule ou pas et si le token est tout en majuscule ou pas) et $\mathbf{SY(T_0)}$ représente un vecteur creux comprenant les traits de symboles de $\mathbf{T_0}$ (s'il contient des chiffres, des ponctuations ou des symboles spéciaux '_', '-').

taxique, analyse syntaxique et identification des EPs. L'étiquetage morphosyntaxique s'appuie uniquement sur la représentation contextualisée du token courant pour prédire son étiquette. L'analyse syntaxique et l'identification des EPs utilisent une méthode par transitions où, durant une analyse, pour chaque configuration (i.e. état courant de l'analyse) une transition (i.e. action) est prédite par un classifieur MLP. Le principe est le même que celui décrit au chapitre 1 : certains éléments de la configuration, les « éléments ciblés » sont passés en entrée. Ici, le classifieur pour les transitions syntaxiques (respectivement le classifieur pour les transitions d'identification d'EPs) reçoit en entrée les représentations contextualisées des éléments ciblés de la configuration syntaxique (respectivement la configuration d'identification d'EPs).

Le module de représentation contextualisée d'un élément (un simple token ou une unité multi-tokens) est donc commun aux trois tâches, et ses paramètres sont partagés. Nous l'appelons ci-après **module partagé**. Chaque module dédié à une tâche a une fonction de perte propre et ses paramètres sont appris indépendamment des deux autres tâches, modulo les paramètres du module partagé. Lors de l'apprentissage d'une tâche, la rétropropagation met à jour les paramètres du module dédié à la tâche, ainsi que ceux du module partagé. Ainsi, l'apprentissage d'un module dédié à une tâche se nourrit de l'apprentissage des deux autres, via la représentation contextualisée partagée mise à jour lors de l'apprentissage des trois tâches.

Nous décrivons maintenant en détail les différents modules utilisés.

5.1.1 Module partagé de représentation contextualisée

Nous commençons par décrire le module partagé, appliqué pour « contextualiser » chacun des éléments d'entrée pour les différentes tâches, qu'il s'agisse de tokens simples ou d'unités multi-tokens pouvant apparaître dans la tâche d'identification d'EPs. Ce module est illustré par la figure 5.1.

Pour construire la représentation contextualisée d'un élément $\mathbf{T_0}$, nous utilisons une fenêtre fixe de taille 7 centrée autour de $\mathbf{T_0}$ dans la phrase à analyser. Lorsque l'élément $\mathbf{T_0}$ est un token,

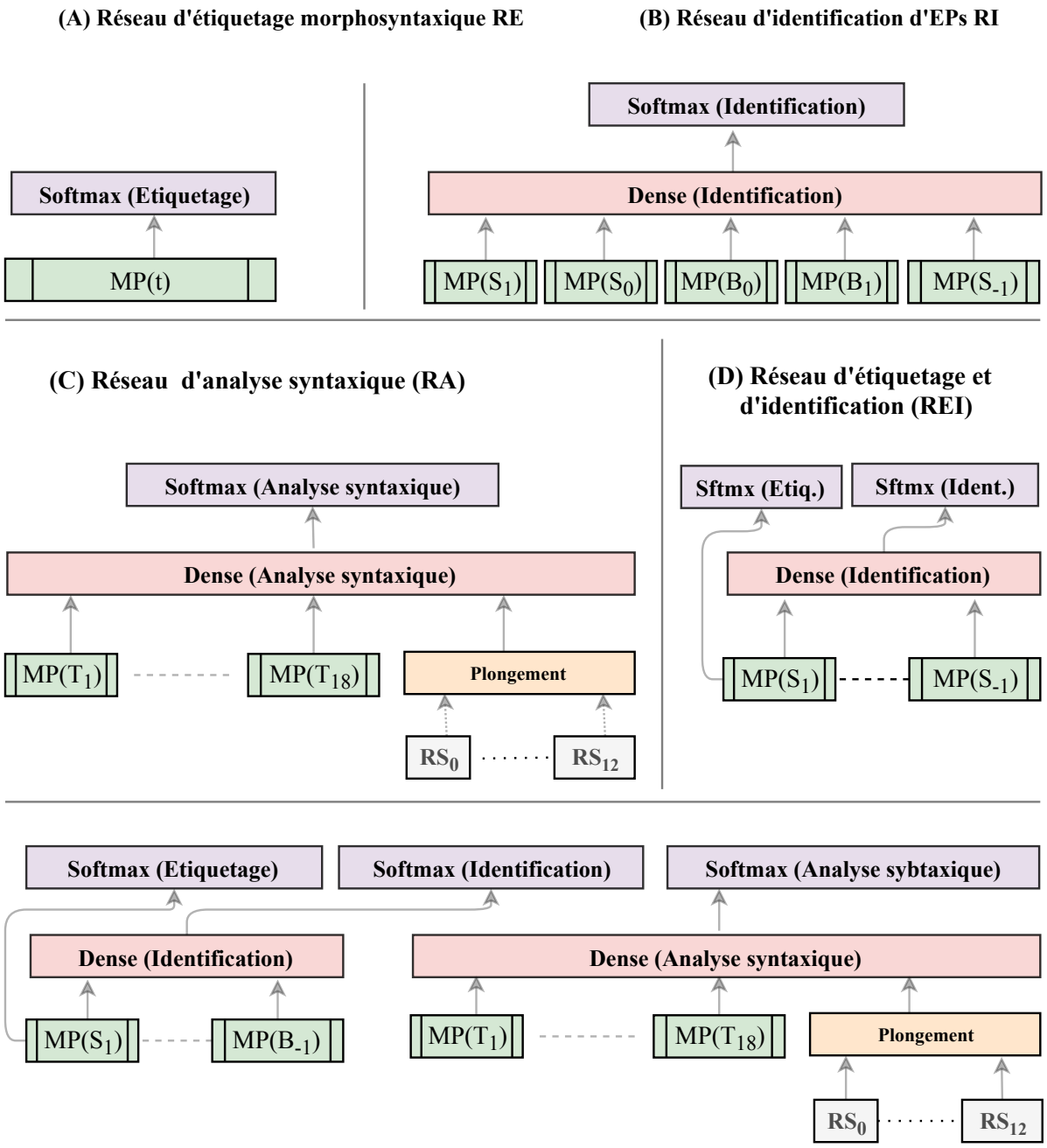

FIGURE 5.2 – **Structure multitâche et ses réseaux :** Une figure illustrant six réseaux de neurones, dont **(A)** représente le réseau d'étiquetage morphosyntaxique tout seul (**RE**) ; **(B)** représente le réseau d'identification des EPs (**RI**) ; **(C)** est le réseau d'analyse syntaxique en dépendances (**RA**). **(D)** et **(E)** représentent des structures multitâches, puisque **D** réunit l'étiquetage morphosyntaxique avec l'identification des EPs ((**REI**)) et **E** Réunit les trois tâches à la fois (**REIA**). Dans tous les blocs, les rectangles MP, répétés ici pour plus de lisibilité, correspondent en fait tous au même module partagé de contextualisation (**MP**), dont les paramètres sont systématiquement partagés. $\mathbf{S_0}, \mathbf{S_1}, \mathbf{B_0}, \mathbf{B_1}, \mathbf{S_{-1}}$ représentent les tokens ciblés des configurations d'identification, $\mathbf{T_1}, .., \mathbf{T_{18}}$ représentent les 18 tokens ciblés de la configuration syntaxique et $\mathbf{RS_0}, \mathbf{RS_{12}}$ représentent les labels des relations syntaxiques des 12 dépendances des deux premiers tokens de la pile.

son contexte sera formé de lui-même, des ses trois tokens voisins gauche dans l'ordre linéaire de la phrase ($\mathbf{T_{-1}}$, $\mathbf{T_{-2}}$, $\mathbf{T_{-3}}$), ainsi que de ses trois tokens voisins droits dans l'ordre linéaire de la phrase ($\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{T_3}$). Lorsque l'élément $\mathbf{T_0}$ est une unité multitoken, son contexte sera formé de lui-même, des trois tokens voisins gauche de son token le plus à gauche ($\mathbf{T_{-1}}$, $\mathbf{T_{-2}}$, $\mathbf{T_{-3}}$), ainsi que des trois tokens voisins droits de son token le plus à droite ($\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{T_3}$).

À partir de ce contexte, nous extrayons un ensemble de traits et donnons en entrée du module les plongements qui leur sont associés dans les couches de plongement du module. Les traits extraits se basent tous sur la forme lexicale des éléments du contexte. Pour rappel, la forme lexicale d'une unité multitoken correspond à la concaténation des formes lexicales de ses tokens dans l'ordre d'apparition dans la phrase.

Les vecteurs donnés qui alimente la couche dense sont les suivants :

1. les plongements lexicaux associés à $\mathbf{T_0}$ et aux six tokens voisins (soit 7 vecteurs) ;
2. les plongements associés aux différents suffixes et préfixes de deux et trois caractères pour $\mathbf{T_{-1}}, \mathbf{T_0}$ et $\mathbf{T_1}$ (soit 12 vecteurs) ;
3. des plongements associés aux informations sur la capitalisation des caractères de la forme lexicale de $\mathbf{T_{-1}}, \mathbf{T_0}$ et $\mathbf{T_1}$. Les informations indiquent si le premier caractère est en majuscule ou si tout le token est en majuscule ;
4. un plongement associé à un trait indiquant si la forme lexicale de l'élément courant comprend des chiffres, des ponctuations ou des symboles spéciaux ('_'. '_').

Alors que les deux premiers ensembles de vecteurs sont systématiquement utilisés dans ce composant, l'utilisation des deux derniers ensembles est optionnelle et contrôlée par les hyperparamètres Utiliser la capitalisation et Utiliser les symboles.

La concaténation de ces plongements sera donnée en entrée d'une couche dense (avec une transformation linéaire et l'application de la fonction d'activation *ReLu*) qui correspondra au final à la représentation contextualisée de l'élément courant.

Hyperparamètres : Utiliser la capitalisation et Utiliser les symboles déterminent respectivement s'il faut intégrer les traits de capitalisation et les traits des symboles dans le module partagé. D'autres hyperparamètres sont responsables de contrôler les dimensions des vecteurs utilisés dans le module tels que :

— Dimension de token pour la taille des plongements lexicaux associés aux formes lexicales des tokens ;

— Dimension des affixes pour la taille des plongements associés aux affixes ;

— Dimension de la capitalisation pour la taille des plongements associés aux informations de capitalisation ;

— Dimension de symboles pour la taille des plongements associés aux informations sur les symboles spéciaux ;

— # Neurones dense pour la taille de la couche dense du module.

Enfin, l'hyperparamètre Lemmatisation détermine si les formes lexicales utilisées sont les formes fléchies ou les lemmes.

5.1.2 Module d'étiquetage morphosyntaxique

Nous décrivons maintenant le module d'étiquetage morphosyntaxique, qui est illustré par la partie **A** de la figure 5.2. Celui-ci prend en entrée un élément **t** qui est ensuite contextualisé par

le module partagé. La représentation contextualisée produite de **t** est ensuite passée à une couche *Softmax* qui produira une distribution de probabilités pour les différentes étiquettes morphosyntaxiques possibles. En phase d'analyse, comme pour toutes les tâches du système multitâche, on utilise un algorithme glouton, sélectionnant simplement l'étiquette la plus probable à chaque position.

5.1.3 Module d'identification des EPs

Le module d'identification des EPs verbales, illustré par le bloc **B** de la figure 5.2, reçoit en entrée les éléments ciblés de la configuration en cours de traitement, tels que $\mathbf{S_1}$, $\mathbf{S_0}$, $\mathbf{B_0}$, mais aussi optionnellement $\mathbf{B_1}$ et $\mathbf{S_{-1}}$. Chacun des ces éléments est ensuite contextualisé via le module partagé. La concaténation des vecteurs contextualisés produits est ensuite passée à une couche dense dédiée à l'identification, elle-même qui alimente une couche *Softmax*, responsable de la prédiction de la prochaine transition.

Hyperparamètres : Les hyperparamètres UTILISER B_1, UTILISER S_{-1} déterminent s'il faut retenir les tokens $\mathbf{B_1}$ et $\mathbf{S_{-1}}$, # NEURONES (IDENT) détermine le nombre de neurones de la couche dense du module d'identification.

5.1.4 Module d'analyse syntaxique en dépendances par transitions

Pour l'analyse syntaxique, nous adoptons le jeu de transitions Arc-standard, que nous avons décrit dans l'état de l'art (Partie 1, section 4.1.2, 63). Les éléments ciblés de la configuration en cours de traitement sont ceux qu'utilisent Chen et Manning (2014) pour leur système d'analyse syntaxique par transitions, c'est-à-dire dix-huit tokens : les trois premiers tokens de la pile, les trois premiers tokens du tampon, les deux premiers enfants les plus à gauche/droite des deux premiers tokens de la pile, l'enfant le plus à gauche de l'enfant le plus à gauche des deux premiers tokens de la pile et son homologue de droite. Le module partagé produit pour chacun de ces dix-huit tokens une représentation dense contextualisée. Le module syntaxique prend aussi en entrée les étiquettes des relations syntaxiques déjà prédites pour douze tokens avec leurs gouverneurs (les mêmes tokens à l'exception des trois premiers de la pile et du tampon, ces derniers ne pouvant pas être dépendants de relations syntaxiques déjà prédites) . Chacune de ces douze étiquettes syntaxiques est associée à un plongement d'étiquette syntaxique.

La concaténation des dix-huit vecteurs contextualisés de tokens et des douze plongements d'étiquettes syntaxiques est ensuite passée à la couche dense du composant d'analyse syntaxique. Cette dernière couche alimente une couche *Softmax* qui permet de prédire la transition à appliquer la plus probable étant donné la configuration courante. Le bloc **C** de la figure 5.2 illustre ce module d'analyse syntaxique.

Hyperparamètres : # NEURONES (SYNT) détermine le nombre de neurones de la couche dense du composant syntaxique.

5.1.5 Systèmes multitâches

Les blocs **D** et **E** de la figure 5.2 illustrent les deux systèmes multitâches produits à partir des différents modules précédemment décrits. Le bloc **D** décrit la combinaison de l'étiquetage morphosyntaxique et de l'identification d'EPs. Il reprend le module d'identification d'EPs, à la différence près que chaque représentation contextualisée des éléments ciblés de la configuration

est branchée à la couche *Softmax* de l'étiquetage, qui permettra de prédire l'étiquette morpho-syntaxique des éléments ciblés. Enfin, le bloc **E** combine les trois tâches : il reprend le bloc **D** auquel il ajoute le bloc **C** pour l'analyse syntaxique.

Pour rappel, ce sont des architectures multitâches car les paramètres du module partagé de contextualisation sont effectivement partagés.

5.2 Algorithmes d'optimisation

Afin d'entraîner conjointement le système multitâche sur les trois tâches (ou deux tâches si l'on enlève l'analyse syntaxique), nous définissons un algorithme d'optimisation en alternance, qui utilise trois objectifs d'optimisation (un par tâche). L'algorithme réalise d'abord **n** époques sur la tâche d'étiquetage morphosyntaxique (**n** étant déterminé par l'hyperparamètre Nombre d'époque d'initialisation). Ensuite, les trois (ou deux) tâches sont apprises en alternance en fonction d'une condition probabiliste qui décide pour quelle tâche l'époque doit se dérouler. L'optimisation pour chaque époque utilise l'algorithme du gradient stochastique (mini batch). À chaque itération, la rétropropagation permet de calculer le gradient des paramètres pour la tâche considérée, ce gradient étant utilisé pour mettre à jour les paramètres, y compris ceux du module partagé.

Afin d'éviter le sur-apprentissage, nous ajoutons à l'algorithme une mesure d'arrêt de l'apprentissage (« Early stopping »), qui réserve une partie des données d'entraînement, appelées données de validation, et les exploite pour évaluer le modèle à la fin de chaque époque. Si le modèle n'obtient pas une amélioration significative de ses performances sur ces données pour une séquence de **k**(=4) époques, l'algorithme considère que l'optimisation est terminée. Comme nous nous intéressons principalement à l'identification des EPs, la procédure de Early stopping se base uniquement sur les performances d'identification.

Modes d'apprentissage : Afin de comprendre les résultats de notre système et d'évaluer l'impact de l'apprentissage conjoint sur la performance, nous implémentons des versions « unitaires », chacune ne réalisant l'apprentissage que sur une seule des trois tâches.

Hyperparamètres : La fonction de perte utilisée est l'entropie croisée pour tous les systèmes et *AdaGrad* est la technique d'optimisation choisie pour les trois objectifs. L'apprentissage est contrôlé par trois hyperparamètres Taux d'apprentissage X, qui est utilisé pour un objectif X (X correspond à une des trois tâches). La mesure d'arrêt de l'apprentissage est déterminée par un hyperparamètre Moniteur qui détermine si l'on doit observer la perte ou la certitude (« accuracy ») du modèle sur les données de validation. Un autre hyperparamètre Delta indique la proportion d'amélioration nécessaire pour ne pas réaliser l'arrêt. Autrement dit, si les écarts des valeurs du Moniteur pour les quatre dernières époques ne dépassent pas la valeur du Delta, l'apprentissage doit s'arrêter. Taille de batch détermine la taille de batch d'apprentissage de chaque objectif d'optimisation.

5.3 Cadre expérimental

Nous présentons dans cette section le cadre expérimental pour tester notre système multitâche. La sous-section 5.3.1 présente la méthode du réglage des hyperparamètres, alors que la sous-section 5.3.2 montre le mécanisme de la sélection des combinaisons d'hyperparamètres de

tendances. La sous-section 5.3.4 définit les systèmes de base avec lesquels nous comparons les résultats de nos expérimentations.

Avant de commencer, nous fixons des acronymes pour les différentes versions comparées, en utilisant **R** pour réseau, puis **E**, **I** et/ou **A** selon les tâches qu'ils réalisent : **E** pour étiquetage morphosyntaxique, **I** pour identification d'EPs, **A** pour analyse syntaxique. Ainsi **REIA** apprend les trois tâches en même temps, **REI** apprend l'étiquetage et l'identification, **RE**, **RI** et **RA** apprennent respectivement l'étiquetage, l'identification d'EPs et l'analyse syntaxique, isolément.

5.3.1 Réglage des hyperparamètres

	Configuration	Plage	RE	RI	RA	REI	REIA$_g$	REIA$_i$
Module partagé	lemmatisation	{T, F}	T	T	T	T	T	T
	Capitalisation	{T, F}	T	T	T	T	T	T
	Symbole	{T, F}	T	T	T	T	T	T
	Dim.(token)	[25, 200]	69	89	80	83	84	86
	Dim.(affixe)	[5, 25]	13	13	12	13	12	12
	Dim.(capit.)	[1, 10]	3	3	3	3	4	4
	Dim.(symbole)	[5, 15]	9	9	9	8	9	9
	#Neurones(dense)	[25, 200]	89	87	80	90	79	82
Étiq.	Taille de batch	[64, 256]	51	-	-	39	138	141
	Taux d'app.	[.01, .2]	.035	-	-	.04	.03	.029
Identification	B1	{T. F}	-	T	T	T	T	T
	Bx	{T, F}	-	T	T	T	T	T
	#Neurones(Ident)	[25, 200]	-	81	-	64	78	79
	Taux d'app.	[.01, .2]	-	.044	-	.039	.015	.015
	Taille de batch	[16, 256]	-	42	-	41	140	137
Syntaxe	Dim.(Étiq. syn.)	[5. 50]	-	-	19	-	20	19
	#Neurones (Synt)	[25, 200]	-	-	108	-	106	101
	Taux d'app.	[.01, .2]	-	-	.022	-	.012	.017
	Époques d'init.	{1, 2, 3}	-	-	-	2	1	1
	Taille de batch	[64, 256]	-	-	83	-	128	134
	Early Stop	{T, F}	-	-	T	-	1	1
	Delta	[.005, .1]	-	-	.02	-	.036	.033
	Moniteur	{Perte, Préc.}	-	-	Perte	-	1	1

TABLE 5.1 – **Réglage des hyperparamètres :** Un tableau des hyperparamètres utilisés dans les différents réseaux neuronaux du chapitre. Chaque hyperparamètre est accompagné par la plage de valeurs utilisée lors du réglage et ses valeurs dans les combinaisons CT des différents réseaux. La plage de valeur prend la forme d'un domaine continu de valeur ([ ]) ou d'un ensemble de valeurs discrètes { }. Pour rendre le tableau plus lisible, nous utilisons *T* et *F* pour *True* et *False*. La plage complète de taille de batch est {16, 32, 64, 96, 128, 256}. Il est à noter que la plage de taux d'apprentissage de l'objectif syntaxique et la plage de taille de batch sont rétrécies pour le réglage du réseau REIA à [.01, 0.05] et [64, 128, 256], dans l'objectif d'accélérer l'apprentissage et réduire le temps du réglage.

Pour le réglage des hyperparamètres des réseaux définis dans ce chapitre, nous suivons la même méthode de recherche aléatoire que celle suivie lors du réglage des hyperparamètres des variantes neuronales (voir section 4.4), avec cependant un choix différent de langues pilotes pour

le réglage du réseau dédié à l'analyse syntaxique et le réseau intégral REIA. Dans ces deux cas, le portugais et le turc ont été respectivement remplacés par l'espagnol et l'hébreu, qui respectent les critères en termes de diversité des familles linguistiques et de la taille du jeu d'entraînement. Ce changement des langues pilotes vise à utiliser des langues dont les annotations syntaxiques sont prédites et non manuellement corrigées. Par conséquent, nous écartons le portugais qui dispose d'un jeu d'entraînement dont les annotations syntaxiques sont partiellement gold. Le turc, à son tour, est écarté à cause d'une difficulté technique, puisque le système de base (*UDPipe*) utilisé pour l'analyse syntaxique en dépendances, n'arrive pas à entraîner son modèle sur les jeux de données de cette langue [49].

Impact du taux d'apprentissage : Lors de nos expériences préliminaires, nous avons constaté que la valeur du taux d'apprentissage de l'objectif syntaxique joue un rôle crucial pour les performances de l'analyseur syntaxique. La figure 5.3 montre qu'un score zéro est quasi systématique avec un taux d'apprentissage supérieur à 0.08. Du coup, pour la recherche aléatoire nous avons réduit la plage de valeurs pour cet hyperparamètre pour le réglage du module d'analyse syntaxique.

5.3.2 Sélection des combinaisons d'hyperparamètres

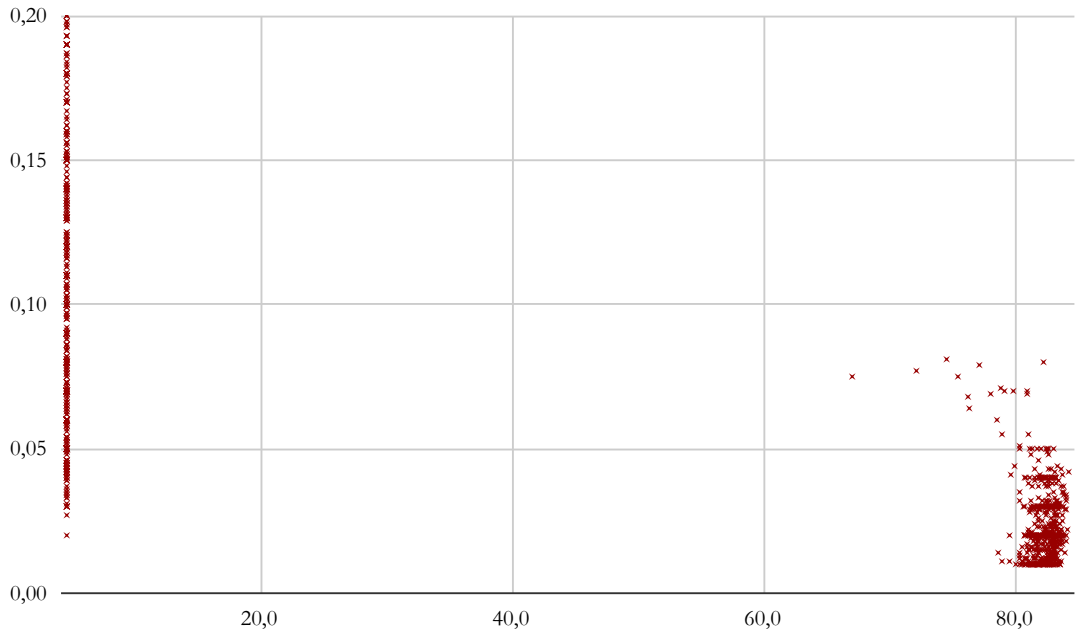

FIGURE 5.3 – **Analyse en dépendances - Impact du taux d'apprentissage** Un graphique des scores **F** des expériences du réglage des hyperparamètres du réseau syntaxique pour l'analyse syntaxique en dépendances en fonction du taux d'apprentissage. En abscisses : scores moyens $\mathbf{F_{avg}}$ du RA, évalués sur les jeux de développement des langues pilotes. En ordonnées : valeurs des taux d'apprentissage de chaque expérience.

49. L'outil UDPipe produit une erreur concernant la lisibilité des fichiers d'entraînement (« Cannot load training data from file .., Node ID X form Y has too large head : Z ! »).

Variante	Score ident.	Score synt.	Score étiq.
RI	+	-	-
RE	-	-	+
RA	-	+	-
REI	+	-	-
REIA_i	+	-	-
REIA_g	+	+	+

TABLE 5.2 – **Sélection des combinaisons - Scores :** Choix des scores pour comparer les combinaisons d'hyperparamètres en fonction des variantes. **Score ident.** correspond à la moyenne des scores $\mathbf{F}(\mathbf{eval} - \mathbf{eps})$ pour la tâche d'identification sur toutes les langues, sur les corpus de développement. **Score synt.** correspond à la moyenne des scores **UAS** et **LAS** sur toutes les langues. **Score étiq.** correspond à la moyenne des précisions d'étiquetage morphosyntaxiques sur toutes les langues. Pour la variante REIA_g, les trois scores sont sélectionnés : cela implique que le score de comparaison est la moyenne des trois scores sélectionnés.

Pour sélectionner la meilleure combinaison d'hyperparamètres de chaque variante pour l'évaluation, nous comparons le score de la combinaison d'hyperparamètres la plus performante (CPP) lors de la recherche aléatoire, avec le score de la combinaison d'hyperparamètres de tendances (CT).

Étant donné que notre tâche cible est l'identification des EPs et que les autres tâches (analyse syntaxique et étiquetage morphosyntaxique) sont considérées comme auxiliaires, le score permettant de comparer deux combinaisons d'hyperparamètres est la moyenne des F-scores (eval – eps) pour la tâche d'identification sur toutes les langues, sur les corpus de développement. Lorsque le tâche d'identification n'est pas activée comme dans les variantes RE et RA, ce sont les scores d'étiquetage (moyenne des précisions d'étiquetage sur toutes les langues) et ceux d'analyse syntaxique (moyenne des **UAS** et **LAS**[50] sur toutes les langues) respectivement qui sont pris en compte.

À des fins de comparaison, nous créons également une variante REIA_g de REIA où l'on considère les trois tâches comme des tâches cibles. C'est alors la moyenne de tous les scores d'étiquetage, d'identification et d'analyse syntaxique qui sont considérés. Le tableau 5.2 récapitule le score de comparaison des combinaisons d'hyperparamètres pour chaque variante.

Comme attendu, les combinaisons CT sont plus performantes que les combinaisons CPP pour tous les réseaux. Le tableau 5.1 fournit une liste exhaustive des hyperparamètres utilisés dans les différents réseaux de neurones. Chaque hyperparamètre du tableau est accompagné avec la plage de ses valeurs possibles ainsi que la valeur optimale (de tendances) pour chaque réseau de neurones. Le tableau permet de constater que les traits de capitalisation et de symboles sont présents dans toutes les combinaisons d'hyperparamètres de tendances sélectionnées pour tous les réseaux de neurones. De manière générale, nous constatons que les valeurs des hyperparamètres de tendances tendent à converger sur tous les réseaux de neurones.

50. **UAS** et **LAS** correspondent à deux métriques utilisées pour évaluer l'analyse syntaxique en dépendances. **UAS** (Unlabeled Attachment Score) correspond à la proportion de tokens qui ont été bien rattaché à la bonne tête. **LAS** (Labeled Attachment Score) correspond à la proportion de tokens qui ont été bien rattaché à la bonne tête avec la bonne étiquette de relation syntaxique.

5.3.3 Langues d'évaluation

Alors que nous exploitons les jeux de données PARSEME 1.1 dans nos expérimentations, nous excluons six langues (l'anglais, le lituanien, le polonais, le slovène, l'hindi et le turc) de l'évaluation pour tous les réseaux de neurones, puisque l'anglais et le polonais manquent d'homogénéité dans leurs annotations morphosyntaxiques et syntaxiques (mélangeant annotations gold et automatiques), le lituanien n'a pas d'annotations syntaxiques et le slovène n'a pas d'annotations morphosyntaxiques. Par ailleurs, nous écartons l'hindi et le turc parce que nous n'arrivons pas à entraîner notre système de base (UDPipe) sur les fichiers d'entraînement de ces langues (l'outil UDPipe produit une erreur concernant le chargement des fichiers d'entraînement). Par conséquent, l'ensemble de langues d'évaluation se limite à treize langues : le bulgare, l'allemand, le grec, l'espagnol, le basque, le farsi, l'hébreu, le croate, le hongrois, l'italien, le portugais et le roumain. Il est à noter que les jeux d'entraînement de ces langues sont constitués d'une majorité d'arbres de dépendances projectives (95 %), à l'exception du basque (EU) et du hongrois (HU), dont les phrases projectives ne représentent que 80 % de leurs jeux d'entraînement.

5.3.4 Systèmes de base

Nous utilisons l'outil UDPipe 2.0 comme système de base, avec lequel nous allons comparer nos scores d'étiquetage morphosyntaxique, ainsi que nos scores d'analyse syntaxique. Nous apprenons des modèles UDPipe 2.0 (Straka et Straková, 2017) sur les jeux d'entraînement des langues d'évaluation et les évaluons sur leurs jeux de développement pour les tâches d'étiquetage morphosyntaxique et d'analyse syntaxique. UDpipe est une boite à outils appliqués en pipeline, capable d'apprendre des modèles pour la segmentation des phrases, l'analyse lexicale, l'étiquetage morphosyntaxique, la lemmatisation et l'analyse en dépendances.

Quant à l'analyse syntaxique, l'outil utilise un système arc-standard avec un oracle dynamique. cet outil désactive l'option (*single_ root*) pour l'analyse syntaxique de certaines langues [51], pour permettre d'analyser des arbres avec plusieurs relations syntaxiques avec l'étiquette « root » désignant la racine. Il est à noter que nous obligeons l'outil (lors de l'étiquetage et de l'analyse syntaxique) à exploiter les annotations des jeux de données telles quelles (*use_ gold_ tags=1*), et que nous utilisons les valeurs standards des autres hyperparamètres de l'outil.

5.4 Résultats et analyses

Cette section porte sur l'expérimentation de la structure multitâche complète et des autres variantes.

Comme c'est notre tâche cible, nous commençons par examiner les résultats de performance sur l'identification des expressions polylexicales obtenus par les variantes RI, REI et REIA, sur les données de développement [52] des treize langues évaluées. Le tableau 5.4 montre les scores par langue et les scores globaux de ces différentes variantes.

De manière générale, ces premiers résultats sont décevants. En effet, l'inclusion des tâches d'étiquetage morphosyntaxique et d'analyse syntaxique ne semble pas bénéfique pour l'identification des EPs : la variante MLP est plus performante en F-score que tous les autres réseaux pour toutes les langues sauf pour le farsi.

51. L'allemand, l'espagnol, le basque, le hongrois et l'italien.

52. Nous ne fournissons pas de résultats de ces réseaux sur les jeux de test, vu le manque de compétitivité qu'ils montrent sur les jeux de développement.

	RI			REI			$REIA_g$			$REIA_i$			MLP		
	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R
BG	57.9	57	59	0.1	0	0	57.9	63	54	55.4	52	59	62.6	70	57
DE	48.3	47	50	47.6	53	43	40.4	40	41	42.5	51	36	55.3	76	44
EL	52.5	54	51	51.3	74	39	48.1	48	48	0.0	0	0	53.2	88	38
ES	44.8	54	38	40.9	69	29	44.2	42	47	47.1	47	48	55.4	81	42
EU	67.9	64	73	66.6	64	70	69.8	73	67	61.5	57	67	77.6	86	71
FA	80.0	84	76	79.4	87	73	66.0	56	80	76.9	74	80	69.8	94	56
FR	66.1	63	69	74.6	75	74	68.8	66	72	66.6	61	73	77.1	91	67
HE	22.9	21	25	28.8	68	18	21.4	48	14	26.4	55	17	34.3	78	22
HR	49.7	54	46	52.7	79	39	47.7	65	38	44.9	70	33	58.0	91	43
HU	92.9	95	91	94.0	99	89	83.4	79	88	87.4	85	90	94.4	100	90
IT	43.3	52	37	44.1	55	37	40.6	39	42	41.5	46	38	52.9	80	40
PT	59.4	51	70	66.4	67	66	57.4	52	64	59.3	54	66	73.0	88	62
RO	82.8	75	93	0.0	0	0	81.0	74	89	79.7	71	91	84.6	80	90
AVG	59.1	59	60	49.7	61	45	55.9	57	57	53.0	56	54	65.2	85	55
AVG_{10}	57.5	59	58	59.5	72	54	54.0	56	55	55.4	60	55	64.8	86	54

TABLE 5.3 – **Identification des EPs verbales - Résultat** Les scores **F**, **P** et **R** en (eval-eps) de l'identification des EPs verbales des variantes RI, REI, $REIA_g$, et $REIA_i$ et de notre variante MLP sur les jeux de développement de plusieurs langues de PARSEME 1.1. Il est à noter que tous les modèles sont appris sur les jeux d'entraînement des langues en utilisant les combinaisons d'hyperparamètres de tendances. Notez que les meilleurs scores sont en rouge alors que les deuxièmes scores sont en bleu. Avg_{10} représente les scores $\mathbf{F_g}$, $\mathbf{P_{avg}(eval-eps)}$ et $\mathbf{R_{avg}(eval-eps)}$ des 10 langues d'évaluation : toutes sauf le bulgare, le grec et le roumain, sur lesquels des scores zéro se sont produits.

Les résultats montrent que la variante MLP bat le réseau RI avec 6 points d'écart en termes de F-score sur l'ensemble des langues. On observe en particulier un écart supérieur à 9 points pour l'espagnol, le basque, le français, l'hébreu, l'italien et le portugais, soit majoritairement des langues romanes. Il est à noter que le réseau RI obtient le meilleur F-score pour le farsi, avec plus de 10 points d'écart sur le MLP, ce qui tendrait à montrer que les informations supplémentaires utilisées sur les tokens et sur le contexte des éléments ciblés sont importantes pour cette langue. Par contre, pour toutes les autres langues, ces informations supplémentaires ne sont pas pertinentes par rapport à MLP et pénalisent même le système. Ces résultats sont cependant à nuancer. Alors que l'ajout de ces informations a un effet désastreux sur la précision moyenne avec une chute de 16 points, elles apportent un plus significatif pour le rappel moyen avec un gain de 5 points.

Afin de tenir compte de l'impact effectif de l'inclusion de l'étiquetage morphosyntaxique et de l'analyse syntaxique en dépendances comme tâches auxiliaires, on compare maintenant les différentes variantes multitâches au réseau RI. La première observation est que certaines variantes multitâches connaissent de gros accidents pour quelques langues : les variantes REI (identification + étiquetage) et $REIA_i$ (identification + étiquetage + analyse en dépendances, avec l'identification comme objectif d'apprentissage) ont des scores nuls pour au moins une de ces trois langues : le bulgare, le grec et le roumain. À des fins de comparaison plus informative, nous donnons également les scores globaux pour toutes les langues sauf pour ces trois langues (soit dix langues au total) dans le tableau 5.4 (ligne AVG_{10}).

Lang.	RI			REI			REIA_g			REIA_i			MLP		
	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R
Vue	80.9	90	74	80.0	89	73	78.1	88	70	79.1	89	71	**81.9**	88	77
Vue(Fréq.)	**88.4**	95	83	85.9	93	80	86.3	94	80	87.0	93	82	85.4	94	78
Vue(Rar)	76.9	88	69	76.6	88	68	73.0	86	64	74.2	87	65	**79.7**	85	75
Non vue	**24.1**	22	27	20.0	25	17	18.7	15	27	19.7	16	25	0.3	20	0
Continue	63.8	62	65	67.2	74	61	62.0	61	63	63.1	64	62	**69.4**	88	57
Discon.	41.2	45	38	43.8	60	35	37.8	41	35	40.2	46	35	**47.0**	85	32
Identique	**90.1**	95	86	89.1	95	84	88.2	95	83	88.1	95	82	89.8	94	86
Variante	72.2	84	63	71.6	83	63	69.0	82	60	70.8	83	62	**74.8**	82	69
Enchâssée	21.4	26	18	13.4	23	10	14.1	15	13	14.2	20	11	**27.5**	62	18
TP	77.9	73	84	79.2	77	81	67.0	59	77	68.4	66	71	**81.4**	91	74
Multitoken	56.6	58	55	59.6	71	51	55.0	58	53	56.9	61	53	**61.6**	87	48
Long. 2	59.3	60	59	63.2	75	55	58.7	61	56	60.7	66	56	**64.2**	87	51
Long. 3	40.9	47	37	37.9	47	32	33.7	33	35	34.4	35	34	**46.7**	82	33
Long.$\geq$4	12.6	15	11	12.3	18	10	13.4	21	10	**14.9**	27	10	14.1	49	8

TABLE 5.4 – **Identification des EPs verbales - Analyse de performance** Les scores **F**, **P** et **R** en (eval-eps) des variantes RI, REI REIA_g, REIA_i et MLP sur plusieurs classe formelles des EPs des jeux de développement des 13 langues d'évaluation (à l'exception du bulgare, du grec et du roumain, sur lesquels des scores zéro se sont produits). Il est à noter que tous les modèles sont appris sur les jeux d'entraînement des langues en utilisant les combinaisons d'hyperparamètres de tendances.

Pour évaluer l'impact de l'étiquetage morphosyntaxique comme tâche auxiliaire, nous comparons les réseaux RI et REI. La performance du réseau REI dépasse celle de RI sur sept langues et se montre très compétitive avec celle de RI sur trois langues. Cependant, REI connaît deux gros accidents avec des scores nuls pour le bulgare et le turc, ce qui pénalise son score global. Si l'on ne tient compte que des dix langues pour lesquelles il n'y eu aucun accident sur l'ensemble des expériences, on s'aperçoit que la moyenne des F-scores est de 59,5 % pour REI contre 57,5 % pour RI, soit un écart de 2 points en faveur de la variante multitâche. Cette différence s'explique en grande partie par une forte augmentation de la précision moyenne en faveur de REI (72 % vs. 59 %, soit un écart de 13 points).

Si l'on rajoute maintenant la tâche d'analyse syntaxique, on s'aperçoit que les réseaux REIA_g et REIA_i produisent des scores modestes avec une baisse de 4 points en F-score moyen pour REIA_i (le meilleur des deux) par rapport à REI si l'on tient compte des dix langues sans accident uniquement. Cela s'explique par une baisse importante de la précision moyenne (60% vs. 72%, soit 12 points d'écart). À noter que le rappel augmente très légèrement de 1 point (55% vs. 54%). Si l'on examine les résultats par langue, le système REI est systématiquement meilleur que les variantes REIA, sauf pour le basque et l'espagnol. Enfin, on s'aperçoit que la variante REIA_i dépasse légèrement de 1,4 point en F-score sur l'identification, la variante REIA_g sur les 10 langues sans accident. Ce résultat est logique étant donné que REIA_i est optimisé sur la tâche d'identification uniquement, ce qui n'est pas le cas de REIA_g optimisé pour l'ensemble des tâches.

Ainsi, à partir de ces premiers résultats, l'inclusion de la tâche d'étiquetage morphosyntaxique comme tâche auxiliaire semble améliorer la tâche d'identification. Par contre, ce n'est pas le cas de la tâche d'analyse syntaxique qui détériore les performances lorsqu'elle est combinée avec

l'étiquetage morphosyntaxique et l'identification des EPs.

Le tableau 5.4 indique les scores des différents réseaux (RI, REI, REIA$_i$, REIA$_g$) pour différentes classes d'EPs, sur les jeux de développement des dix langues pour lesquels il n'y a pas

	RE	REI	REIA$_g$	REIA$_i$	UDPipe
BG	96.6	96.6	96.5	96.5	**98.8**
DE	90.7	90.9	89.3	87.7	**95.1**
EL	**95.3**	95.1	94.3	94.1	93.3
ES	93.8	93.7	91.8	91.4	**96.9**
EU	**67.9**	67.0	66.8	67.1	55.6
FA	93.0	92.5	90.1	89.7	**95.1**
FR	96.2	96.1	95.4	95.7	**96.8**
HE	78.8	78.2	75.4	75.2	**90.3**
HR	94.5	94.4	91.3	91.3	**95.9**
HU	**98.5**	98.4	97.9	97.5	**98.5**
IT	95.4	95.2	95.1	95.2	**96.6**
PT	94.6	94.6	94.6	94.8	**96.0**
RO	96.7	96.8	96.4	96.5	**98.8**
AVG	91.7	91.5	90.4	90.2	**92.9**

TABLE 5.5 – **Étiquetage morphosyntaxique : résultat** Les scores de précision de l'étiquetage morphosyntaxique des variantes RE, REI, REIA$_g$, REIA$_i$ et du système de base UDPipe sur les jeux de développement de plusieurs langues de PARSEME 1.1. Il est à noter que nos modèles sont appris sur les jeux d'entraînement des langues en utilisant les combinaisons d'hyperparamètres de tendances. Le modèle de base est appris sur les mêmes jeux d'entraînement, en utilisant les configurations standard de l'outil.

eu d'accident. Les classes d'EPs correspondent aux EPs vues et non vues dans le corpus d'apprentissage, les EPs continues et discontinues, les EPs de longueur 1, 2 et 3 , ... Les résultats montrent que le MLP est plus performant en termes d'identification que les autres réseaux sur la plupart des classes d'EPs à l'exception des EPs identiques, vues fréquemment et non vues dans le corpus d'apprentissage. Parmi tous les réseaux testés, c'est RI qui produit le meilleur score sur ces classes. Il produit un score de 24,1 % versus 0,3 % pour MLP sur les EPs non vues, ce score s'explique par un rappel moyen quasi nul pour le MLP alors qu'il atteint près 27 % pour RI. C'est donc l'utilisation d'un plus large contexte et d'informations supplémentaires sur les tokens qui a un impact important sur la performance du système sur les EPs non vues, et non pas l'inclusion de tâches auxiliaires.

Toutes ces conclusions sont cependant à prendre avec précaution. En effet, nous avons observé des performances très étonnantes (i.e. des scores nuls) pour REI sur le bulgare et le roumain, et pour REIA$_i$ sur le grec. Nous avons tenté de comprendre plus en profondeur ce qu'il s'est passé. Une piste que nous avons explorée concerne le taux d'apprentissage qui a tendance à causer des effondrements ponctuels de performance avec des valeurs élevées. Cependant, notre procédure de réglage des hyperparamètres préférant au final des combinaisons d'hyperparamètres de tendances, permet d'éviter des valeurs élevées pour le taux d'apprentissage (0,039 pour REI et 0,015 pour REIA$_i$). Cela rend donc ces effondrements plus étonnants encore. Nous avons aussi analysé l'évolution de la perte lors de l'apprentissage des modèles REI et REIA$_i$. Durant l'apprentissage, nous utilisons la méthode du `Early stopping` qui consiste à arrêter l'apprentissage

dès que la perte ne diminue pas pendant $\mathbf{k = 4}$ époques à la suite, pour gagner du temps. Cette méthode suppose que la perte ne remonte pas trop longtemps lors de l'apprentissage, ce qu'il s'est malheureusement produit dans notre cas. Il se trouve que pour l'apprentissage de REI et REIA, la valeur de perte était très élevée (et proche de la valeur initiale de la perte) à l'arrêt de l'apprentissage. Ce point mériterait de plus amples approfondissements.

	RA		REIA_g		REIA_i		UDPipe	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
BG	80.2	73.4	80.9	73.5	79.8	72.8	**89.1**	**87.0**
DE	67.7	54.5	66.2	53.5	66.1	53.1	**82.8**	**78.7**
EL	78.2	69.4	76.1	67.5	75.7	67.5	**86.3**	**82.7**
ES	70.8	64.0	73.7	66.0	75.2	67.4	**86.1**	**83.7**
EU	59.6	34.5	57.7	34.6	59.1	35.6	**78.0**	**72.1**
FA	61.3	38.7	59.3	36.4	57.4	35.8	**71.4**	**63.6**
FR	81.5	73.4	81.1	72.9	80.8	72.8	**87.0**	**84.6**
HE	56.6	29.2	55.8	28.7	55.2	28.8	**77.0**	**71.1**
HR	69.3	56.1	70.7	57.6	71.4	57.9	**83.3**	**78.2**
HU	76.2	69.4	76.3	68.5	76.2	68.9	**88.6**	**87.4**
IT	74.2	69.3	72.8	67.5	73.0	67.7	**76.7**	**73.1**
PT	82.6	71.0	80.7	69.2	77.5	66.9	**89.7**	**86.0**
RO	80.2	70.8	76.7	67.2	78.3	68.7	**85.5**	**82.5**
AVG	72.2	59.5	71.4	58.7	71.2	58.8	**83.2**	**79.3**
AVG_{11}	73.0	60.9	72.2	60.0	71.9	59.9	83.2	79.2

TABLE 5.6 – **Analyse syntaxique en dépendance - Résultat** Les scores UAS et LAS de l'analyse syntaxique en dépendance des variantes RA, REIA_g, REIA_i et du système de base UDPipe sur les jeux de développement de plusieurs langues de PARSEME 1.1. Les modèles des variantes sont appris sur les phrases projectives des jeux d'entraînement des langues en utilisant les combinaisons d'hyperparamètres de tendances. Le modèle de base est appris sur les mêmes jeux d'entraînement, en utilisant les configurations standard de l'outil. La dernière ligne (AVG_{11}) représente la moyenne pour toutes les langues dont au moins 95% des phrases d'entraînement sont projectives (i.e. toutes les langues sauf le basque et le hongrois).

Pour information, nous indiquons maintenant les résultats concernant les deux tâches auxiliaires : l'étiquetage morphosyntaxique et l'analyse syntaxique. Le tableau 5.5 présente les scores de précision d'étiquetage morphosyntaxique des réseaux RE, REI, REIA_g, REIA_i, et du système de base *UDPipe* sur les jeux de développement des langues d'évaluation de PARSEME 1.1. Ce tableau montre que les précisions moyennes des variantes RE, REI, REIA_g et REIA_i n'atteignent pas la précision moyenne du système de base. À noter cependant que tous nos réseaux dépassent le système de base sur le basque et le grec, mais perdent totalement leur compétitivité sur l'hébreu.

Par ailleurs, le réseau RE donne les scores les plus prometteurs en étant assez compétitif sur sept langues (écart de moins de 3 points). En revanche, il y a plus de 5 points d'écart entre les deux systèmes sur l'allemand. Les précisions du réseau REI sont proches de celles du réseau RE. Enfin, les deux variantes du réseau REIA obtiennent des scores comparables, mais légèrement moins compétitifs que les autres réseaux. Enfin, la dégradation des scores d'étiquetage morphosyntaxique de nos réseaux de neurones augmente avec le niveau de complexité des ré-

seaux. Autrement dit, l'ajout des composants d'identification et d'analyse syntaxique engendre une dégradation de la performance de notre composant d'étiquetage morphosyntaxique.

Le tableau 5.6 donne les scores **UAS** et **LAS** de la performance de l'analyse syntaxique en dépendances du réseau RA, des deux variantes du REIA et du système de base *UDPipe* sur les jeux de développement des langues d'évaluation de PARSEME 1.1. Notons que l'apprentissage de nos réseaux se limite aux phrases des jeux d'entraînement dont les arbres de dépendances sont projectifs. Le tableau montre que le système de base bat nettement les trois réseaux avec plus de 11 points en termes de **UAS** et de 20 points en termes de **LAS**.

Nous constatons que les scores de nos réseaux sont très proches les uns des autres, et que le réseau syntaxique (RA) seul est légèrement plus performant que la structure intégrale qui réunit le composant syntaxique avec les deux autres composants. La différence entre notre structure REIA et celle de Zhang et Weiss (2016) concerne les détails de l'implémentation de l'algorithme d'optimisation, puisque nous utilisons un optimisateur différent (Adagrad au lieu de ASGD avec momentum), ce qui peut expliquer en partie le manque de compétitivité de notre architecture.

5.5 Résumé

Dans ce chapitre nous avons présenté une architecture d'apprentissage multitâche pour réaliser conjointement les tâches d'identification des EPs verbale, d'étiquetage morphosyntaxique et d'analyse syntaxique en dépendances, et avons détaillé les résultats d'expérimentations sur les données PARSEME 1.1.

Le principe de notre architecture multitâche est d'utiliser un module partagé permettant de fournir une représentation contextualisée de tokens (ou de multitokens, quand il s'agit d'éléments de la pile). Il s'agit d'un MLP avec en entrée plongements lexicaux pour les tokens d'une fenêtre autour du token à représenter. Le module est dit partagé car ses paramètres vont être mis à jour à l'apprentissage pour les différentes sous-tâches visées.

Le module pour l'étiquetage morphosyntaxique ajoute une simple couche softmax au-dessus du module partagé représentant le token à étiqueter. Le module d'identification d'EPs utilise le module partagé pour représenter les éléments ciblés de la configuration, puis un MLP. Pour le module d'analyse syntaxique, nous avons implémenté un analyseur par transitions, de type arc-standard, et utilisé le module partagé pour contextualiser la représentation de 18 éléments ciblés de la configuration (en suivant le modèle de Chen et Manning (2014)), plus des plongements pour les étiquettes de dépendances déjà prédites.

Nous avons décrit l'algorithme d'apprentissage multitâche utilisé, qui commence par un certain nombre d'époques avec l'objectif d'étiquetage morphosyntaxique, puis alterne aléatoirement pour les trois objectifs (étiquetage morphosyntaxique, identification d'EPs, analyse syntaxique). Nous avons également rapporté les résultats d'apprentissage de chaque tâche isolément, et d'apprentissage de deux tâches (étiquetage et identification).

Les résultats sur un sous-ensemble des langues de PARSEME 1.1 sont décevants, car aucun des systèmes de ce chapitre n'améliore les résultats d'identification d'EPs, sauf pour une langue, le farsi. L'utilisation du module partagé en entrée du seul module d'identification (système RI) montre déjà une dégradation des résultats par rapport au MLP décrit au chapitre 3. Les systèmes véritablement multitâche, incluant la tâche d'étiquetage morphosyntaxique, et d'analyse syntaxique ont des résultats encore plus dégradés. Ces résultats demandent clairement plus d'analyse pour expliquer cette dégradation.

Chapitre 6

Applications et détails techniques

Dans ce bref chapitre, nous décrivons deux cas d'utilisation directe de notre méthode d'identification par transitions. Tout d'abord, notre variante SVM_2 a été utilisée dans le cadre d'une étude en neurosciences (section 6.1). Ensuite, notre méthode a été reprise et améliorée par des chercheurs ayant participé à la deuxième compétition PARSEME 1.1 (section 6.2). En fin de chapitre, nous donnons les détails techniques pour la mise en œuvre de nos systèmes et nos expérimentations.

6.1 Utilisation de notre variante SVM pour une étude en neurosciences

La variante SVM_2 a contribué à une étude en neurosciences visant à étudier la remémoration (« memory retrieval ») et la composition syntaxique en analysant les images IRMf (imagerie par résonance magnétique fonctionnelle) des participants à une séance d'audition de la version audio de « Le Petit Prince » en anglais (Bhattasali et al., 2019). Ces deux processus sont mesurés en s'appuyant sur des méthodes de la linguistique informatique. En particulier, la remémoration est mesurée via les expressions polylexicales qui ont de bonnes chances d'être stockées comme des unités, plutôt que construites compositionnellement. À noter que la composition syntaxique est mesurée via une analyse ascendante (« bottom-up ») qui trace le nombre d'opérations nécessaires pour composer une phrase dans un arbre. L'un des objectifs de l'étude était de trouver des corrélations entre les occurrences d'EPs et l'activation de certaines zones en mémoire en s'appuyant sur une analyse de régression.

À cette fin, nous avons entraîné notre variante SVM_2 sur le corpus Children's Book Test (Hill et al., 2015) pour que notre corpus d'entraînement ne soit pas trop éloigné du corpus cible (« Le Petit Prince ») en termes de vocabulaire en particulier. Étant donné que ce corpus ne contient ni annotations linguistiques basiques telles que les lemmes et les étiquettes morphosyntaxiques ni annotations des EPs, nous avons ajouté les annotations linguistiques basiques à ce corpus en utilisant les outils de lemmatisation et d'étiquetage morphosyntaxique de la librairie python NLTK. Nous avons également projeté des lexiques d'EPs anglaises (Paumier et al., 2009; Koenraad et al., 2003; White, 1998; Adam et al., 1995) sur le corpus pour annoter ces EPs, en utilisant notre système de base (cf. section 2.4.2) avec quelques contraintes supplémentaires. On considère chaque entrée lexicale comme une séquence de formes lemmatisées de tokens. Chaque entrée est mise en correspondance avec la version lemmatisée de notre corpus cible que l'on considère dont comme une séquence de lemmes. Il y a correspondance si l'on retrouve exactement tous les tokens de l'entrée dans l'ordre dans le texte, en autorisant un trou d'un token pour une EP d'au plus

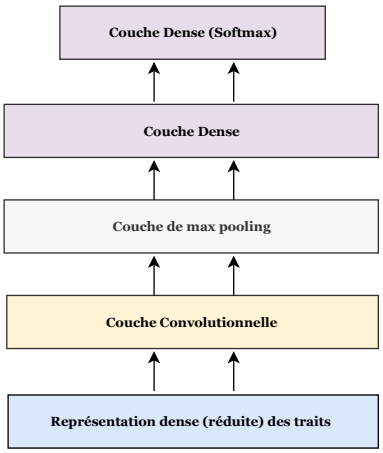

FIGURE 6.1 – **Système TRAPACC - (Stodden et al., 2018) :** Le réseau de neurones convolutionnel du système TRAPACC, (Stodden et al., 2018). Il est à noter que la couche d'entrée représente le produit de la méthode de réduction de dimensionnalité des traits extraits pour la configuration d'entrée.

deux tokens ou un trou d'au plus deux tokens pour les autres EPs. Nous avons en effet été obligé de contraindre fortement l'ordre d'apparition des tokens de l'EP et la taille des trous autorisée afin de limiter au maximum le bruit. Cette version annotée du corpus a servi ensuite de corpus d'apprentissage pour notre système SVM_2.

Une fois le modèle appris, le système SVM_2 a été appliqué au corpus « Le Petit Prince ». L'analyse de régression a permis de découvrir que la remémoration des EPs avait tendance à activer la même zone dans le cerveau qui se trouve aussi être différente de celle utilisée pour la composition syntaxique (pour plus de détails, voir Bhattasali et al. (2019)).

Nous avons également calculé pour chacune des EPs identifiées son degré de cohésion avec la mesure d'association de l'information mutuelle spécifique (« Pointwise Mutual Information ou PMI »). Cette mesure a été utilisée comme degré de cohésion de l'EP lors de l'analyse de régression, souvent interprété comme un degré d'idiomaticité de l'EP (Ramisch, 2015). L'analyse de régression a montré que les expressions avec un plus faible degré de cohésion activent aussi des noeuds bien connus pour leur implication dans la composition.

6.2 Reprise de la méthode

Nous avons participé à la première compétition PARSEME 1.0 sur l'identification des expressions verbales avec notre variante SVM_0 (Al Saied et al., 2017). Comme montré dans le chapitre 2 de cette partie, nous avons obtenu d'excellents résultats en atteignant la première position pour de nombreuses langues. Ainsi, notre système a inspiré d'autres systèmes pour l'édition suivante de la même campagne internationale.

Stodden et al. (2018) utilisent le même ensemble de transitions et le même oracle déterministe statique que nous avons utilisé dans notre variante SVM_0, pour transformer les phrases d'entrée en séquences de configurations et de transitions. Contrairement à notre système, ce système, appelé TRAPACC, utilise une technique de réduction de dimensionnalité, basée sur les projections positives aléatoires (Qasemi Zadeh et Kallmeyer, 2016) pour réduire la dimension de l'espace vectoriel produit à la suite de l'extraction des traits. TRAPACC intègre ensuite un simple modèle neuronal de plusieurs couches convolutionnelles pour classifier les représentations produites

pour les configurations, avec l'objectif d'éviter le surapprentissage et la faible généralisation du classifieur SVM de Al Saied et al. (2017).

Pour apprendre son classifieur, TRAPACC utilise tous les patrons de traits proposés par Al Saied et al. (2017) et ajoute d'autres patrons de traits, tels que la taille du tampon, la longueur des tokens, des 4-grammes et davantage de traits linguistiques tels que les étiquettes morphosyntaxiques universelles et les traits morphologiques.

Stodden et al. (2018) présentent une autre variante de leur système, TRAPACC$_S$, qui prédit les transitions de façon légèrement différente de TRAPACC. TRAPACC$_S$ réduit le module convolutionnel à une fonction d'extraction supervisée et connecte la couche dense intermédiaire du module à un modèle SVM. Stodden et al. (2018) conclut toutefois que cette variante n'a pas été aussi efficace que prévu. TRAPACC et TRAPACC$_S$ se classent troisièmes de la closed track de la compétition internationale PARSEME 1.1.

6.3 Détails techniques de la mise en œuvre

Pour terminer, nous donnons les détails techniques de la mise en oeuvre de notre système d'identification des EPs par transitions. Tout d'abord, nous utilisons le langage de programmation `python` pour la mise en œuvre de nos expérimentations, ainsi que de multiples bibliothèques logicielles pour faciliter le développement de nos modèles linéaires et de nos réseaux de neurones. Le code des expérimentations de la thèse est disponible sur le site `github` sous la licence MIT [53]. **Python** est un langage de programmation interprété, placé sous une licence libre, doté d'un typage dynamique fort, d'une gestion automatique de la mémoire et munis de nombreuses bibliothèques open source pour l'apprentissage automatique et les réseaux de neurones. Ce langage favorise la programmation impérative structurée, fonctionnelle et orientée objet et offre aux développeurs des outils de haut niveau d'abstraction et une syntaxe simple à utiliser.

Nos modèles linéaires et nos matrices creuses sont développés et gérés par des bibliothèques comme Scikit-learn, Numpy et SciPy. **Scikit-learn** est une bibliothèque open source écrite en python, développée par de nombreux contributeurs du monde académique, destinée à l'apprentissage automatique et connu pour sa complémentarité avec des bibliothèques python open-source telles que Numpy et SciPy. Cette bibliothèque comprend des modèles de régression logistique et des mises en œuvre des algorithmes de classification et des machines à vecteurs de support. SciPy et Numpy sont des bibliothèques open-source, écrites en python, destinées à traiter des matrices ou tenseurs ainsi que des fonctions mathématiques opérant sur ces structures.

Tous nos réseaux de neurones, à l'exception de KG-2016, sont développés avec Keras et Theano. Keras est une interface open source pour le développement des réseaux de neurones, écrite en Python, capable de s'exécuter sur Theano et conçue pour permettre une expérimentation fluide et rapide. **Keras** est caractérisé par son interface user-friendly, modulaire et extensible et offre des outils d'abstraction de haut niveau pour le développement des réseaux de neurones. Keras a été développé par François Chollet et est intégré dans la bibliothèque principale de TensorFlow.

Theano est une bibliothèque écrite en python qui permet de définir, d'optimiser et d'évaluer efficacement les expressions mathématiques, y compris les tenseurs. Theano est une plateforme open source, principalement développée par l'institut de Montréal des algorithmes d'apprentissage de l'Université de Montréal. Cette plateforme est connue pour sa capacité à exploiter les architectures de GPUs et pour son efficacité, sa robustesse et sa rapidité lors du calcul de différenciations symboliques (« Symbolic Differentiation »).

53. Code : `https://github.com/hazemalsaied/MWE.Identification`.

Étant donné que Keras ne permet pas de développer assez facilement des structures neuronales sophistiquées, notamment celles incluant la sélection dynamique d'éléments d'une couche récurrente, comme cela est nécessaire avec le réseau KG-2016, nous utilisons une autre bibliothèque, appelée Pytorch. **PyTorch** est une bibliothèque open source, écrite en python, et basée sur la bibliothèque Torch, développée par Facebook. Torch et son interface python permettent d'effectuer les calculs tensoriels nécessaires pour l'apprentissage profond des réseaux de neurones sur des « CPUs » ou sur des « GPUs ». PyTorch permet de manipuler les tenseurs, les structures neuronales et les algorithmes d'apprentissage, ce qui permet d'implémenter des structures sophistiquées, plus facilement qu'en Keras.

Conclusion

Conclusion

Dans cette thèse, nous avons proposé une méthode d'identification des expressions polylexicales en contexte, fondée sur une analyse automatique dite par transitions. L'analyse d'une phrase préalablement segmentée en tokens se fait par une séquence de décisions locales, appelées transitions, réalisées par un classifieur appris de manière principalement supervisée. Nous avons proposé plusieurs jeux de transitions spécifiques à l'identification d'EPs, et avons exploré plusieurs approches pour l'architecture et l'apprentissage de ce classifieur, en particulier un modèle linéaire de type machine à vecteur support, un perceptron multicouche, et des variantes neuronales utilisant des couches récurrentes sur la séquence d'entrées.

Comparée aux autres méthodes de la littérature d'identification, notre méthode s'est montrée performante, produisant les scores état de l'art pour de nombreuses langues des jeux de données PARSEME 1.0 et PARSEME 1.1, centrés sur les expressions polylexicales verbales, et des scores compétitifs sur le corpus FTB, pour le français. Elle peut traiter certains cas d'enchâssement d'EPs, et a montré, pour toutes les variantes expérimentées, une forte capacité d'identification d'EPs discontinues, et de formes constituant des variantes d'EPs vues dans le corpus d'apprentissage. Cependant, la capacité d'identifier des EPs non vues dans le corpus d'apprentissage, c.à-d. à généraliser, reste, comme pour la plupart des méthodes d'identification, très limitée.

Nous avons privilégié la mise au point de systèmes utilisant un hyperparamétrage robuste, valable pour les différentes langues des jeux de données utilisés. Étant donné le nombre de langues traitées (18 et 20 pour les données PARSEME 1.0 et PARSEME 1.1), et le nombre important d'hyperparamètres pour les différentes architectures neuronales que nous avons proposées pour le classifieur prédisant les transitions, notre thèse comporte une part expérimentale importante. Nous avons particulièrement investigué quel ré-échantillonnage des données était le plus adapté à la tâche, et mis l'accent sur la méthode de réglage des hyperparamètres. Nous avons proposé une sélection de la combinaison d'hyperparamètres basée sur les tendances observées au sein d'une recherche aléatoire dans un espace étendu de combinaisons.

Le scénario privilégié, pour la majeure partie de nos expériences, est l'identification d'EPs utilisant des informations morphologiques mais pas d'informations syntaxiques ni de lexiques externes. Nous avons cependant également investigué un apprentissage multitâche, réalisant conjointement et tirant profit de l'étiquetage morphosyntaxique, l'identification d'EPs par transitions et l'analyse syntaxique en dépendances par transitions. Ce système, s'il donne des scores peu compétitifs pour l'étiquetage et non compétitifs pour l'analyse syntaxique, donne des résultats prometteurs pour la tâche d'identification, car il obtient nos meilleurs scores sur les EPs non vues à l'apprentissage ($\mathbf{F_g}$(eval-eps) = 23.5 versus 11.6 du MLP-Wide sur 13 langues).

De ce travail nous pouvons mettre en avant les observations suivantes :

— Pour notre participation à la compétition internationale PARSEME 1.0, notre système, uniquement SVM à l'époque, a été classé premier pour la plupart des langues. Suite à l'in-

tégration d'architectures neuronales, les variantes les plus performantes sont finalement un simple perceptron multicouche (MLP), en particulier la variante où l'on ajoute en entrée les étiquettes morphosyntaxiques du contexte linéaire des éléments ciblés de la configuration (variante MLP-Wide). Cependant, le modèle SVM_2 a des scores très compétitifs également. Alors que les variantes neuronales favorisent le rappel, SVM_2 est caractérisé par une meilleure précision, et est surtout très nettement moins sensible au réglage des hyperparamètres et au déséquilibre entre classes à l'apprentissage et plus simple à régler.

— Les résultats de toutes les variantes de notre méthode ont obtenu des scores médiocres sur les EPs non vues dans le corpus d'apprentissage, comme c'est le cas également pour les systèmes état de l'art de la compétition internationale PARSEME 1.1. D'ailleurs, l'étude des résultats en fonction de la langue des données PARSEME montre clairement une corrélation très forte entre la proportion d'EPs vues à l'apprentissage et la performance des systèmes, qui explique mieux les résultats que la taille des corpus d'entraînement. Cela confirme que l'identification des EPs ne peut être améliorée qu'en intégrant des méthodes de découverte de nouvelles EPs, que ce soit en utilisant des lexiques externes et/ou en repérant des irrégularités de composition, par exploration de gros corpus, les EPs étant caractérisées par une irrégularité à différents niveaux linguistiques (Savary et al., 2019).

— La rareté des EPs (en particulier verbales) comparativement aux tokens n'entrant pas dans une EP donne une distribution déséquilibrée des transitions. Nous avons montré qu'ignorer à l'apprentissage les phrases ne contenant pas d'EPs et rééquilibrer aléatoirement la distribution des transitions dans les exemples d'apprentissage est essentiel pour assurer la stabilité et la performance de nos modèles neuronaux. Cette technique n'est pas du tout nécessaire pour la variante linéaire (SVM_2), et a un impact marginal.

— Pour le réglage des hyperparamètres d'un modèle, sélectionner la meilleure combinaison s'est avéré moins efficace que sélectionner les valeurs « tendancielles » d'hyperparamètres, i.e. majoritaires dans les k meilleures combinaisons.

— Les résultats de nos variantes MLP-R-Sent, MLP-R-Stack2, MLP-R-Stack ont montré que l'intégration « naïve » des couches récurrentes au sein d'un module MLP permet d'augmenter la généralisation de la méthode. Cependant, ce gain est contrebalancé par de moins bonnes performances sur certaines classes formelles des EPs.

— L'utilisation de plongements lexicaux non contextuels préentraînés a eu des résultats décevants pour notre modèle le plus performant (MLP), avec seulement une très légère amélioration sur certaines langues. Par contre, leur utilisation est systématiquement bénéfique pour toutes nos variantes neuronales intégrant des couches récurrentes ou bien plus de contexte pour les éléments ciblés. L'intégration de plongements lexicaux descendant au niveau de sous-parties de mots n'a pas conduit à une amélioration significative pour la variante MLP.

— Une modification simple de nos jeux de transitions permet de catégoriser les EPs en plus de les identifier, qui testée pour la variante SVM_1 n'engendre pas de perte significative de performance.

— L'architecture multitâche que nous proposons pour l'identification des EPs, l'étiquetage morphosyntaxique et l'analyse syntaxique n'a certes pas permis d'améliorer l'identification d'EPs dans son ensemble, mais l'une des variantes de cette architecture produit le meilleur score sur la classe des EPs non vues à l'apprentissage.

1 Perspectives

Comme nous l'avons vu dans la section 3.5.3, l'intégration des plongements lexicaux combinant des représentations de sous-mots ne conduit pas à une amélioration significative de la performance de la variante MLP de notre méthode. Durant notre thèse cependant sont apparus des modèles de vecteurs contextuels de mots, comme Devlin et al. (2018); Peters et al. (2018), ayant eu un fort retentissement sur de nombreuses tâches de TAL, dont l'analyse syntaxique par transitions (Kulmizev et al., 2019b). Entraînés sur de très grands corpus et capables de modéliser des caractéristiques syntaxiques et sémantiques complexes, ils sont applicables pour obtenir des représentations contextualisées d'occurrences de mots. L'intégration de telles représentations semble une piste pour améliorer en particulier la découverte d'EPs non vues à l'apprentissage, pour lesquelles nos différentes variantes ont des performances médiocres.

L'utilisation de lexiques externes est une autre piste, ainsi que l'intégration de mesures d'association, plus robustes, pouvant servir d'indices d'idiomaticité de combinaisons de tokens. Enfin, de nouvelles architectures neuronales, telles que celles intégrant le mécanisme de l'autoattention (self-attention), se sont révélées très efficaces pour des méthodes d'identification à base d'étiquetage de séquences (Taslimipoor et Rohanian, 2018; Rohanian et al., 2019). Leur utilisation au sein de notre système, formellement plus puissant, nous semble une piste prometteuse.

Par ailleurs, nous avons proposé un système robuste, traitant les différentes catégories d'EPs, de différentes langues, avec le même hyperparamétrage. Cela dit, les résultats d'identification d'EPs sont pour les différentes catégories linguistiques d'EPs. Il est ainsi envisageable de décliner différents systèmes, ciblant différents types d'EPs. En particulier, il est possible de séparer l'identification des EPs vues à l'apprentissage, pour lesquelles notre système MLP est très performant, de l'identification d'EPs non vues, pour lesquelles certaines de nos variantes sont bien meilleures.

Enfin, même si l'architecture multitâche que nous avons proposée pour l'apprentissage conjoint de l'identification d'EPs, de l'étiquetage morphosyntaxique et de l'analyse syntaxique en dépendances n'a pas permis d'améliorer la performance de notre méthode sur certaines langues de PARSEME 1.1, il reste à étudier si d'autres architectures d'apprentissage conjoint amélioreraient l'identification d'EPs, notamment si des tâches sémantiques, telles que l'étiquetage des rôles sémantiques, sont ajoutées.

Bibliographie

Abeille, A. (1988). Light verb constructions and extraction out of np in tree adjoining grammar. In *CLS. Papers from the General Session at the... Regional Meeting*, number 24-1, pages 1–16.

Abeillé, A. et Clément, L. (2003). Annotation morpho-syntaxique : Les mots simples-les mots composés, corpus le monde.

Abeillé, A., Clément, L., et Toussenel, F. (2003). Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

Adam, M., J. E., G., et Maxine, T. B. (1995). *A Dictionary of American Idioms*. Barron's Educational Series ; 3rd edition (March 1995).

Al Saied, H., Candito, M., et Constant, M. (2017). TheATILF-LLF system for parseme shared task : a transition-based verbal multiword expression tagger. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE)- 2017*, pages 127–132, Valencia, Spain. Association for Computational Linguistics.

Al Saied, H., Candito, M., et Constant, M. (2018). A transition-based verbal multiword expression analyzer. In *Multiword expressions at length and in depth : Extended papers from the MWE 2017 workshop*, volume 2, page 209. Language Science Press.

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., et Collins, M. (2016). Globally normalized transition-based neural networks. *arXiv preprint arXiv :1603.06042*.

Baldwin, T. et Kim, S. N. (2010). Multiword expressions. *Handbook of natural language processing*, 2 :267–292.

Ballesteros, M., Dyer, C., Goldberg, Y., et Smith, N. A. (2017). Greedy transition-based dependency parsing with stack lstms. *Computational Linguistics*, 43(2) :311–347.

Bengio, Y., Ducharme, R. j., Vincent, P., et Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb) :1137–1155.

Bergstra, J. et Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb) :281–305.

Berk, G., Erden, B., et Gûngôr, T. (2018). Deep-bgt at parseme shared task 2018 : Bidirectional lstm-crf model for verbal multiword expression identification. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 248–253.

Bhattasali, S., Fabre, M., Luh, W.-M., Al Saied, H., Constant, M., Pallier, C., Brennan, J. R., Spreng, R. N., et Hale, J. (2019). Localising memory retrieval and syntactic composition : an fmri study of naturalistic language comprehension. *Language, Cognition and Neuroscience*, 34(4) :491–510.

Blunsom, P. et Baldwin, T. (2006a). Multilingual deep lexical acquisition for hpsgs via supertagging. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 164–171. Association for Computational Linguistics.

Blunsom, P. et Baldwin, T. (2006b). Multilingual deep lexical acquisition for hpsgs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia. Association for Computational Linguistics.

Bohnet, B. (2010). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 89–97. Association for Computational Linguistics.

Bohnet, B. et Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., et Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv :1607.04606*.

Boukobza, R. et Rappoport, A. (2009). Multi-word expression identification using sentence surface features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 2-Volume 2*, pages 468–477. Association for Computational Linguistics.

Breidt, E., Segond, F., et Valetto, G. (1996). Formal description of multi-word lexemes with the finite-state formalism idarex. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1036–1040. Association for Computational Linguistics.

Calzolari, N., Fillmore, C. J., Grishman, R., Ide, N., Lenci, A., MacLeod, C., et Zampolli, A. (2002). Towards best practice for multiword expressions in computational lexicons. In *LREC*.

Candito, M. et Constant, M. (2014). Strategies for contiguous multiword expression analysis and dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, volume 1, pages 743–753.

Candito, M., Constant, M., Ramisch, C., Savary, A., Parmentier, Y., Pasquer, C., et Antoine, J.-Y. (2017). Annotation d'expressions polylexicales verbales en français. In *24e conférence sur le Traitement Automatique des Langues Naturelles (TALN)*.

Carpuat, M. et Diab, M. (2010). Task-based evaluation of multiword expressions : a pilot study in statistical machine translation. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 242–245.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1) :41–75.

Chanod, J.-P. et Tapanainen, P. (1996). A non-deterministic tokeniser for finite-state parsing. In *Proceedings of the Workshop on Extended finite state models of language (ECAI'96)*. Citeseer.

Chawla, N. V. (2009). Data mining for imbalanced datasets : An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer.

Chawla, N. V., Bowyer, K. W., Hall, L. O., et Kegelmeyer, W. P. (2002). Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16 :321–357.

Chen, D. et Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Collins, M. et Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 111–118, Barcelona, Spain.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., et Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug) :2493–2537.

Constant, M., Candito, M., et Seddah, D. (2013). The ligm-alpage architecture for the spmrl 2013 shared task : Multiword expression analysis and dependency parsing. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 46–52.

Constant, M., Eryiğit, G., Monti, J., Van Der Plas, L., Ramisch, C., Rosner, M., et Todirascu, A. (2017). Multiword expression processing : A survey. *Computational Linguistics*, 43(4) :837–892.

Constant, M. et Nivre, J. (2016). A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, volume 1, pages 161–171.

Constant, M. et Sigogne, A. (2011). Mwu-aware part-of-speech tagging with a crf model and lexical resources. In *Proceedings of the Workshop on Multiword Expressions : from Parsing and Generation to the Real World*, pages 49–56. Association for Computational Linguistics.

Constant, M., Sigogne, A., et Watrin, P. (2012). Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Long Papers-Volume 1*, pages 204–212. Association for Computational Linguistics.

Constant, M. et Tellier, I. (2012). Evaluating the impact of external lexical resources into a crf-based multiword segmenter and part-of-speech tagger. In *8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 646–650.

Cook, P., Fazly, A., et Stevenson, S. (2007). Pulling their weight : Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the workshop on a broader perspective on multiword expressions*, pages 41–48. Association for Computational Linguistics.

Cook, P., Fazly, A., et Stevenson, S. (2008). The vnc-tokens dataset. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 19–22.

Crabbé, B. (2014). An lr-inspired generalized lexicalized phrase structure parser. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics : Technical Papers*, pages 541–552.

Devlin, J., Chang, M.-W., Lee, K., et Toutanova, K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

Diab, M. T. et Bhutada, P. (2009). Verb noun construction mwe token supervised classification. In *Proceedings of the Workshop on Multiword Expressions : Identification, Interpretation, Disambiguation and Applications*, pages 17–22. Association for Computational Linguistics.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7) :1895–1923.

Dietterich, T. G. et Bakiri, G. (1994). Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2 :263–286.

Dyer, C., Kuncoro, A., Ballesteros, M., et Smith, N. A. (2016). Recurrent neural network grammars. *arXiv preprint arXiv :1602.07776*.

Evert, S. (2005). The statistics of word cooccurrences : word pairs and collocations.

Fazly, A., Cook, P., et Stevenson, S. (2009). Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1) :61–103.

Fellbaum, C. (1999). La représentation des verbes dans le réseau sémantique" wordnet". *Langages*, pages 27–40.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Sánchez-Martínez, F., Ramírez-Sánchez, G., et Tyers, F. M. (2011). Apertium : a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2) :127–144.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3) :268–278.

Gharbieh, W., Bhavsar, V., et Cook, P. (2017). Deep learning models for multiword expression identification. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 54–64.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57 :345–420.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1) :1–309.

Goldberg, Y. et Levy, O. (2014). word2vec explained : deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv :1402.3722*.

Goldberg, Y. et Nivre, J. (2013). Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1 :403–414.

Gómez-Rodríguez, C. et Nivre, J. (2013). Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4) :799–845.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., et Mikolov, T. (2018). Learning word vectors for 157 languages. *arXiv preprint arXiv :1802.06893*.

Gross, G. (1996). *les expressions figées*. Ophrys.

Gross, M. (1987). The use of finite automata in the lexical representation of natural language. In *LITP Spring School on Theoretical Computer Science*, pages 34–50. Springer.

Hashimoto, C. et Kawahara, D. (2008). Construction of an idiom corpus and its application to idiom identification based on wsd incorporating idiom-specific features. In *Proceedings of the conference on empirical methods in natural language processing*, pages 992–1001. Association for Computational Linguistics.

Hashimoto, C., Sato, S., et Utsuro, T. (2006). Japanese idiom recognition : Drawing a line between literal and idiomatic meanings. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 353–360. Association for Computational Linguistics.

Hashimoto, K., Xiong, C., Tsuruoka, Y., et Socher, R. (2016). A joint many-task model : Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv :1611.01587*.

He, H., Bai, Y., Garcia, E. A., et Li, S. (2008). Adasyn : Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE.

Henderson, M., Thomson, B., et Young, S. (2013). Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471.

Hill, F., Bordes, A., Chopra, S., et Weston, J. (2015). The goldilocks principle : Reading children's books with explicit memory representations. *arXiv preprint arXiv :1511.02301*.

Huang, L. et Sagae, K. (2010). Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden.

Jackendoff, R. (1997). *The architecture of the language faculty*. Number 28. MIT Press.

Katz, G. et Giesbrecht, E. (2006). Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions : Identifying and Exploiting Underlying Properties*, pages 12–19. Association for Computational Linguistics.

Keysar, B. et Bly, B. (1995). Intuitions of the transparency of idioms : Can one keep a secret by spilling the beans ? *Journal of Memory and Language*, 34(1) :89–109.

Kiperwasser, E. et Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv :1603.04351*.

Kipf, T. N. et Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv :1609.02907*.

Kirilin, A., Krauss, F., et Versley, Y. (2016). Icl-hd at semeval-2016 task 10 : Improving the detection of minimal semantic units and their meanings with an ontology and word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 937–945.

Klyueva, N., Doucet, A., et Straka, M. (2017). Neural networks for multi-word expression detection. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 60–65, Valencia, Spain. Association for Computational Linguistics.

Koenraad, K., Heather, M., Heidi, Q., Therese, A., et Kees, v. d. V. (2003). Said ldc2003t10. web download. volume 173.

Kulmizev, A., de Lhoneux, M., Gontrum, J., et Fano, E. (2019a). Deep contextualized word embeddings in transition-based and graph-based dependency parsing – a tale of two parsers revisited. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Kulmizev, A., de Lhoneux, M., Gontrum, J., Fano, E., et Nivre, J. (2019b). Deep contextualized word embeddings in transition-based and graph-based dependency parsing–a tale of two parsers revisited. *arXiv preprint arXiv :1908.07397*.

Legrand, J. et Collobert, R. (2016). Phrase representations for multiword expressions. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 67–71, Berlin, Germany. Association for Computational Linguistics.

Li, L. et Sporleder, C. (2010). Using gaussian mixture models to detect figurative language in context. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 297–300.

Li, W., Zhang, X., Niu, C., Jiang, Y., et Srihari, R. (2003). An expert lexicon approach to identifying english phrasal verbs. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 513–520. Association for Computational Linguistics.

Liu, Y. et Zheng, Y. F. (2005). One-against-all multi-class svm classification using reliability measures. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 849–854. IEEE.

Ma, X. et Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv :1603.01354*.

Ma, X., Hu, Z., Liu, J., Peng, N., Neubig, G., et Hovy, E. (2018). Stack-pointer networks for dependency parsing. *arXiv preprint arXiv :1805.01087*.

Maldonado, A., Han, L., Moreau, E., Alsulaimani, A., Chowdhury, K. D., Vogel, C., et Liu, Q. (2017). Detection of verbal multi-word expressions via conditional random fields with syntactic dependency features and semantic re-ranking. In *Proceedings of the 13th Workshop on Multiword Expressions(MWE 2017)*, pages 114–120, Valencia, Spain. Association for Computational Linguistics.

Maldonado-Guerra, A. et Emms, M. (2011). Measuring the compositionality of collocations via word co-occurrence vectors : Shared task system description. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 48–53. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., et Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*.

Mohri, M. (2005). Local grammar algorithms. *Inquiries into Words, Constraints and Contexts*, page 84.

Moreau, E., Alsulaimani, A., Maldonado, A., Han, L., Vogel, C., et Chowdhury, K. D. (2018a). Semantic reranking of crf label sequences for verbal multiword expression identification. In *Multiword expressions at length and in depth : Extended papers from the MWE 2017 workshop*, volume 2, page 177. Language Science Press.

Moreau, E., Alsulaimani, A., Maldonado, A., et Vogel, C. (2018b). Crf-seq and crf-deptree at parseme shared task 2018 : Detecting verbal mwes using sequential and dependency-based approaches. In *Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018) at the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 241–247.

Nagy T., I. et Vincze, V. (2014). Vpctagger : Detecting verb-particle constructions with syntax-based methods. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 17–25, Gothenburg, Sweden. Association for Computational Linguistics.

Nasr, A., Ramisch, C., Deulofeu, J., et Valli, A. (2015). Joint dependency parsing and multiword expression tokenization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 1116–1126.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Nivre, J. (2004a). Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing : Bringing Engineering and Cognition Together*, pages 50–57. Association for Computational Linguistics.

Nivre, J. (2004b). Incrementality in deterministic dependency parsing. In Keller, F., Clark, S., Crocker, M., et Steedman, M., editors, *Proceedings of the ACL Workshop Incremental Parsing : Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4) :513–553.

Nivre, J. (2009). Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint*

Conference on Natural Language Processing of the AFNLP : Volume 1-Volume 1, pages 351–359. Association for Computational Linguistics.

Pasquer, C., Ramisch, C., Savary, A., et Antoine, J.-Y. (2018). Varide at parseme shared task 2018 : Are variants really as alike as two peas in a pod ? In *COLING Workshop on Linguistic Annotation, Multiword Expressions and Constructions*.

Paumier, S., Nakamura, T., et Voyatzi, S. (2009). Unitex, a corpus processing system with multi-lingual linguistic resources. *eLEX2009*, 173.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., et Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Petrov, S., Das, D., et McDonald, R. (2011). A universal part-of-speech tagset. *arXiv preprint arXiv :1104.2086*.

Qasemi Zadeh, B. et Kallmeyer, L. (2016). Random positive-only projections : Ppmi-enabled incremental semantic space construction. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 189–198.

Rama, T. et al. (2014). Supertagging : Introduction, learning, and application. *arXiv preprint arXiv :1412.6264*.

Ramisch, C. (2015). Multiword expressions acquisition. *A Generic and Open Framework. Cham : Springer International Publishing*.

Ramisch, C., Cordeiro, S., Savary, A., Vincze, V., Mititelu, V., Bhatia, A., Buljan, M., Candito, M., Gantar, P., Giouli, V., et al. (2018). Edition 1.1 of the parseme shared task on automatic identification of verbal multiword expressions. In *the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 222–240.

Ramshaw, L. A. et Marcus, M. P. (1999). Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Rikters, M. et Bojar, O. (2017). Paying attention to multi-word expressions in neural machine translation. *arXiv preprint arXiv :1710.06313*.

Rohanian, O., Taslimipoor, S., Kouchaki, S., Ha, L. A., et Mitkov, R. (2019). Bridging the gap : Attending to discontinuity in identification of multiword expressions. *arXiv preprint arXiv :1902.10667*.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv :1706.05098*.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., et Flickinger, D. (2002). Multiword expressions : A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer.

Salton, G., Ross, R. J., et Kelleher, J. (2016). Idiom token classification using sentential distributed semantics.

Savary, A., Candito, M., Mititelu, V. B., Bejček, E., Cap, F., Čéplö, S., Cordeiro, S. R., Eryiğit, G., Giouli, V., van Gompel, M., HaCohen-Kerner, Y., Kovalevskaitė, J., Krek, S., Liebeskind, C., Monti, J., Escartín, C. P., van der Plas, L., QasemiZadeh, B., Ramisch, C., Sangati, F., Stoyanova, I., et Vincze, V. (2018). PARSEME multilingual corpus of verbal multiword expressions. In Markantonatou, S., Ramisch, C., Savary, A., et Vincze, V., editors, *Multiword expressions at length and in depth : Extended papers from the MWE 2017 workshop*, pages 87–147. Language Science Press., Berlin.

Savary, A., Cordeiro, S., et Ramisch, C. (2019). Without lexicons, multiword expression identification will never fly : A position statement. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 79–91.

Savary, A., Ramisch, C., Cordeiro, S., Sangati, F., Vincze, V., QasemiZadeh, B., Candito, M., Cap, F., Giouli, V., et Stoyanova, I. (2017). The parseme shared task on automatic identification of verbal multiword expressions.

Schneider, N., Danchik, E., Dyer, C., et Smith, N. A. (2014a). Discriminative lexical semantic segmentation with gaps : running the mwe gamut. *Transactions of the Association for Computational Linguistics*, 2 :193–206.

Schneider, N., Hovy, D., Johannsen, A., et Carpuat, M. (2016). Semeval-2016 task 10 : Detecting minimal semantic units and their meanings (dimsum). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 546–559.

Schneider, N., Onuffer, S., Kazour, N., Danchik, E., Mordowanec, M. T., Conrad, H., et Smith, N. A. (2014b). Comprehensive annotation of multiword expressions in a social web corpus. In *Proc. ofLREC 2014*, pages 455–461.

Schütze, H. (1998). Automatic word sense discrimination. *Computational linguistics*, 24(1) :97–123.

Seddah, D., Tsarfaty, R., Kübler, S., Candito, M., Choi, J., Farkas, R., Foster, J., Goenaga, I., Gojenola, K., Goldberg, Y., et al. (2013). Overview of the spmrl 2013 shared task : cross-framework evaluation of parsing morphologically rich languages. Association for Computational Linguistics.

Seretan, V. (2011). *Syntax-based collocation extraction*. Text, Speech and Language Technology. Springer.

Simkó, K. I., Kovács, V., et Vincze, V. (2017). Uszeged : Identifying verbal multiword expressions with pos tagging and parsing techniques. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 48–53.

Srivastava, R. K., Greff, K., et Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv :1505.00387*.

Stodden, R., QasemiZadeh, B., et Kallmeyer, L. (2018). Trapacc and trapaccs at parseme shared task 2018 : Neural transition tagging of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 268–274.

Straka, M. et Straková, J. (2017). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task : Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Taslimipoor, S. et Rohanian, O. (2018). Shoma at parseme shared task on automatic identification of vmwes : Neural multiword expression tagging with high generalisation. *arXiv preprint arXiv :1809.03056*.

Taslimipoor, S., Rohanian, O., et al. (2019). Cross-lingual transfer learning and multitask learning for capturing multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 155–161.

Ting, K. M. et Witten, I. H. (1999). Issues in stacked generalization. *Journal of artificial intelligence research*, 10 :271–289.

Torrey, L. et Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends : algorithms, methods, and techniques*, pages 242–264. IGI Global.

Tu, Y. (2012). *English complex verb constructions : identification and inference*. PhD thesis, University of Illinois at Urbana-Champaign.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., et Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vincze, V., Zsibrita, J., et Nagy T., I. (2013). Dependency parsing for identifying hungarian light verb constructions. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 207–215, Nagoya, Japan. Asian Federation of Natural Language Processing.

Waszczuk, J. (2018). Traversal at parseme shared task 2018 : Identification of verbal multiword expressions using a discriminative tree-structured model. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 275–282.

Weiss, D., Alberti, C., Collins, M., et Petrov, S. (2015). Structured training for neural network transition-based parsing. *arXiv preprint arXiv :1506.06158*.

White, J. G. (1998). *Cambridge International Dictionary of Idioms*. Cambridge University Press, Cambridge University.

Yamada, H. et Matsumoto, Y. (2003). Statistical Dependency Analysis with Support Vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, pages 195–206.

Zampieri, N., Ramisch, C., et Damnati, G. (2019). The impact of word representations on sequential neural mwe identification. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 169–175.

Zhang, Y. et Nivre, J. (2011a). Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 188–193, Portland, Oregon, USA.

Zhang, Y. et Nivre, J. (2011b). Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

Zhang, Y. et Weiss, D. (2016). Stack-propagation : Improved representation learning for syntax. *arXiv preprint arXiv :1603.06598*.