HAL
open science

Natural Language Generation : From Data Creation to Evaluation via Modelling

Anastasia Shimorina

To cite this version:

Anastasia Shimorina. Natural Language Generation : From Data Creation to Evaluation via Modelling. Computation and Language [cs.CL]. Université de Lorraine, 2021. English. NNT : 2021LORR0080 . tel-03254708

HAL Id: tel-03254708

https://hal.univ-lorraine.fr/tel-03254708

Submitted on 9 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

École doctorale IAEM Lorraine

Natural Language Generation: From Data Creation to Evaluation via Modelling

THÈSE

présentée et soutenue publiquement le 26 février 2021

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Anastasia Shimorina

Composition du jury

Rapporteurs :	Emiel Krahmer	Professor, Tilburg University, the Netherlands
	Kees van Deemter	Professor, Utrecht University, the Netherlands
Examinateur :	Dimitra Gkatzia	Associate Professor, Edinburgh Napier University, UK
Directeur de thèse :	Claire Gardent	Directrice de recherche, CNRS, LORIA, France
Co-directeur de thèse :	Yannick Parmentier	Maître de conférences, Université de Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Моим родителям

Abstract

Natural language generation is a process of generating a natural language text from some input. This input can be texts, documents, images, tables, knowledge graphs, databases, dialogue acts, meaning representations, etc. Recent methods in natural language generation, mostly based on neural modelling, have yielded significant improvements in the field. Despite this recent success, numerous issues with generation prevail, such as faithfulness to the source, developing multilingual models, few-shot generation. This thesis explores several facets of natural language generation from creating training datasets and developing models to evaluating proposed methods and model outputs.

In this thesis, we address the issue of multilinguality and propose possible strategies to semi-automatically translate corpora for data-to-text generation. We show that named entities constitute a major stumbling block in translation exemplified by the English-Russian translation pair. We proceed to handle rare entities in data-to-text modelling exploring two mechanisms: copying and delexicalisation. We demonstrate that rare entities strongly impact performance and that the impact of these two mechanisms greatly varies depending on how datasets are constructed. Getting back to multilinguality, we also develop a modular approach for shallow surface realisation in several languages. Our approach splits the surface realisation task into three submodules: word ordering, morphological inflection and contraction generation. We show, via delexicalisation, that the word ordering component mainly depends on syntactic information. Along with the modelling, we also propose a framework for error analysis, focused on word order, for the shallow surface realisation task. The framework enables to provide linguistic insights into model performance on the sentence level and identify patterns where models underperform. Finally, we also touch upon the subject of evaluation design while assessing automatic and human metrics, highlighting the difference between the sentence-level and system-level type of evaluation.

Keywords: natural language generation, data-to-text generation, surface realisation, evaluation, error analysis

Résumé

La génération en langue naturelle (*natural language generation*, NLG) est le processus qui consiste à générer du texte dans une langue naturelle à partir de données d'entrée. Ces entrées peuvent prendre la forme de textes, de documents, d'images, de tableaux, de graphes (réseaux de connaissances), de bases de données, d'actes de dialogue, ou d'autres représentations sémantiques. Les méthodes récentes en NLG, principalement basées sur des modèles neuronaux, ont apporté des améliorations significatives. Malgré ces récents progrès, de nombreux problèmes liés à la tâche de génération subsistent, tels que celui de la fidélité aux données d'entrée, du développement de modèles multilingues, ou de la génération à partir de peu d'exemples. Cette thèse explore trois aspects de la NLG : tout d'abord, la création de données d'apprentissage, puis le développement de modèles de génération, et enfin l'évaluation des méthodes proposées.

Nous abordons la question du multilinguisme et proposons des stratégies de traduction semi-automatique de corpus destinés à l'entraînement de modèles de NLG. Nous montrons que les entités nommées constituent un obstacle majeur dans la réalisation de la tâche de traduction, ici considérée de l'anglais vers le russe. Nous décrivons ensuite deux méthodes de traitement des entités rares dans les données d'apprentissages des modèles de NLG : la copie et la délexicalisation. Nous démontrons que l'effet de ces deux mécanismes varie fortement selon la manière dont les données sont construites, et que les entités rares ont un impact important sur les performances des modèles. Concernant la génération multilingue, nous développons une approche modulaire de réalisation de surface superficielle (*shallow surface realisation*, SSR) pour plusieurs langues. Notre approche consiste à diviser la tâche de SSR en trois composantes : l'ordonnancement des mots, l'inflexion morphologique et la génération de contractions. Nous montrons, via la délexicalisation, que la composante d'ordonnancement s'appuie principalement sur les informations syntaxiques. En plus de nos contributions concernant la modélisation, nous proposons un cadre d'analyse des erreurs axé sur l'ordre des mots, pour la tâche de SSR. Ce cadre permet d'obtenir un aperçu linguistique des performances des modèles au niveau de la phrase et d'identifier les cas où un modèle échoue. Enfin, nous abordons le sujet de l'évaluation de manière plus générale et comparons différentes métriques automatiques et humaines ; nous soulignons la différence entre les méthodes d'évaluation au niveau de la phrase et les méthodes d'évaluations au niveau du corpus.

Mots-clés: génération en langue naturelle, génération à partir de données, réalisation de surface, évaluation, analyse d'erreurs

Contents

Génération en langue naturelle : de la création des données à l'évaluation, en passant par la modélisation **xv**

1 Introduction **1**

2 Background **5**

2.1 Natural Language Generation 5

2.1.1 Introduction to NLG 5

2.1.2 Input Data in NLG 7

2.2 Methods in Natural Language Generation 11

2.2.1 Encoder-Decoder Framework 11

2.2.2 Attention 13

2.2.3 Copy and Coverage Mechanisms 14

2.2.4 Factored Models 16

2.2.5 Transformers, Pre-trained Language Models 16

2.2.6 Encoder-Decoder Models for NLG 18

2.3 Future Directions 19

3 Creating Training Corpora for Natural Language Generation **21**

3.1 Introduction 21

3.2 Related Work 22

3.2.1 Corpus Construction for Natural Language Generation 22

3.2.2 Automatic Post-Editing 24

3.3 WebNLG Data-to-Text Dataset 24

3.4 Creating Russian Version of WebNLG Dataset 25

3.4.1 Neural Machine Translation 25

3.4.2 Manual Post-Editing and Error Analysis 25

3.5 Automatic Post-Editing 27

3.5.1 Rule-Based Post-Editing 27

3.5.2 Automatic Post-Editing Model . . . 28
3.6 Evaluation of Rule-Based Post-Editing . . . 29
3.7 Conclusion . . . 29

4 Handling Rare Items in Natural Language Generation 31
4.1 Introduction . . . 31
4.2 Related Work . . . 32
4.3 Experiments . . . 33
4.3.1 Datasets . . . 33
4.3.2 Model Parameters . . . 38
4.3.3 Evaluation . . . 38
4.4 Results and Discussion . . . 40
4.5 Conclusion . . . 43

5 Training Models for Surface Realisation 45
5.1 Introduction . . . 45
5.2 Motivation and Related Work . . . 46
5.2.1 Motivation . . . 46
5.2.2 Related Work . . . 47
5.3 Data . . . 48
5.3.1 SR'18 . . . 48
5.3.2 SR'19 . . . 48
5.4 Model . . . 49
5.4.1 Word Ordering . . . 49
5.4.2 Morphological Realisation . . . 51
5.4.3 Contraction Generation . . . 51
5.5 Evaluation on SR'18 . . . 52
5.5.1 Word Ordering . . . 52
5.5.2 Morphological Realisation . . . 55
5.5.3 Contraction Generation . . . 56
5.5.4 Global Evaluation . . . 56
5.6 Participation in SR'19 . . . 57
5.6.1 Model Adaptation . . . 57
5.6.2 Results and Discussion . . . 58
5.7 Conclusion . . . 63

6 Evaluating Surface Realisers **65**
6.1 Introduction . . . 65
6.2 Related Work . . . 66
6.3 Framework for Error Analysis . . . 67
6.3.1 Syntactic Complexity Metrics . . . 67
6.3.2 Performance Metrics . . . 69
6.3.3 Correlation Tests . . . 70
6.3.4 Error Mining . . . 70
6.4 Data and Experimental Setting . . . 70
6.5 Error Analysis . . . 72
6.5.1 Tree-Based Syntactic Complexity . . . 72
6.5.2 Projectivity . . . 74
6.5.3 Entropy . . . 75
6.5.4 Which Syntactic Constructions Are Harder to Handle? . . . 78
6.6 Using Error Analysis for Improving Models or Datasets . . . 81
6.7 Conclusion . . . 83

7 Evaluating Natural Language Generation Systems **85**
7.1 Introduction . . . 85
7.2 Context and Motivation . . . 86
7.3 Experimental Setup . . . 87
7.3.1 Data . . . 87
7.3.2 Design . . . 87
7.3.3 Ensuring Quality . . . 88
7.3.4 Correlations . . . 88
7.4 Correlation Analysis Results . . . 88
7.5 Conclusion . . . 89

8 Conclusion **91**

Appendices **93**

A Appendix A **93**

B Appendix B **115**

Bibliography **117**

List of Figures

1 Délexicalisation et linéarisation (dans l'arbre d'analyse de la phrase de sortie, la première ligne montre les lemmes, la deuxième les formes de mots, la troisième les parties du discours et la quatrième les identifiants). Les identifiants sont assignés aux nœuds de l'arbre dans l'ordre donné par une recherche en profondeur. xviii

2.1 A game record and its corresponding document (partial view) in RotoWire. Picture from Wiseman et al. (2017). 7

2.2 A dialogue act and its corresponding text in E2E. MR: meaning representation. Picture from `http://www.macs.hw.ac.uk/InteractionLab/E2E`. 8

2.3 SR shallow track. Input for English in the CoNLL-U format and in a graphical form for the output sentence *This happened very quickly, and I wanted to make sure that I let everyone know before I left.* Picture from `http://taln.upf.edu/pages/msr2018-ws/SRST.html#examples`. 10

2.4 SR deep track. Input for English in a graphical form for the output sentence *This happened very quickly, and I wanted to make sure that I let everyone know before I left.* Picture from `http://taln.upf.edu/pages/msr2018-ws/SRST.html#examples`. 11

2.5 An RNN-based sequence-to-sequence model in the training phase. Picture modified from `http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf`. 12

2.6 A sequence-to-sequence model with attention. Picture modified from `http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf`. . 13

2.7 Pointer-generator model. Picture from See et al. (2017). 15

2.8 Factored model. Picture from Hokamp (2017). 17

2.9 Source sentence dependency tree and its factored representation after subword segmentation. B: beginning, I: inside, E: end of a word. O is used if a symbol corresponds to the full word. Picture from Sennrich and Haddow (2016). 17

5.1 Delexicalising and linearising (in the parse tree of the output sentence the first row shows the lemmas, the second–the word forms, the third–the POS tags and the fourth–the identifiers). Identifiers are assigned to the source tree nodes in the order given by depth-first search. 49

5.2 Linear regression between BLEU scores for each module and final BLEU scores. Data points are treebanks (ids are from Table 5.6). In orange: MR on gold lemma vs. final BLEU (WO + MR); in blue: WO vs. final BLEU (WO + MR). 62

6.1 A reference UD dependency tree (nodes are lemmas) and a possible SR model output. The final output is used to compute human judgments and the lemmatised output to compute BLEU and dependency edge accuracy (both are given without punctuation). 68
6.2 Spearman ρ coefficients between metrics. Ad: Adequacy z-score, Fl: Fluency z-score, DEA: dependency edge accuracy, MFS: mean flux size, MFW: mean flux weight, MA: mean arity, TD: tree depth, TL: tree length, MDD: mean dependency distance. * – non-significant coefficients at $\alpha = 0.05$ corrected with the Holm-Bonferroni method for multiple hypotheses testing. 73
6.3 Entropy of dependency relations in the treebanks used in the SR shared tasks. A cross indicates the absence of a dependency relation in a treebank. Treebanks are grouped by language families. 76
6.4 Spearman correlation coefficient heatmap between dependency edge accuracy and entropy. A cross indicates that a team did not make a submission for the treebank. Treebanks are grouped by language families. The same correlation matrix but annotated with coefficient numbers is available in the Appendix B. 77

7.1 System- and sentence-level correlation analysis. 89

B.1 Spearman ρ coefficients between dependency edge accuracy and entropy. A cross indicates that a team did not make a submission for the treebank. Treebanks are grouped by language families. 116

List of Tables

1 Exemple original de WebNLG dans la catégorie ComicsCharacter, sa traduction russe (MT), et sa traduction post-éditée (PE) ainsi que l'annotation des erreurs. Les erreurs sont colorées en bleu. *gen*—génitif; *anim*—animé; *inan*—inanimé. . . xvi

2.1 RDF triples and their corresponding text in WebNLG. 8

3.1 WebNLG original instance in the ComicsCharacter category, its Russian translation (MT), and post-edited translation (PE) along with error annotation. Errors are highlighted in blue. Links are RDF triples of the form <*English entity, sameAs, Russian entity*>. However, such links are not available for all entities in DBpedia. *gen*—genitive; *anim*—animate; *inan*—inanimate. 24
3.2 Main categories and subcategories of error classification. 26
3.3 Proportion of main error types in the manually post-edited data and Cohen's κ scores on the held-out category Athlete. 26
3.4 Corpus statistics: number of post-edited (PE) texts. Total corresponds to both PE and non-PE texts. 28
3.5 BLEU-4 scores. 29

4.1 Entry examples of the E2E (top) and WebNLG (bottom) datasets with and without delexicalisation. 34
4.2 Statistics on attribute values in E2E (above) and on RDF-triple constituents in WebNLG (below). 35
4.3 Training/development/test sets statistics in E2E and WebNLG in original (unconstrained) and constrained splits. *Instances* count is a number of (data, text) pairs; *MRs* count is a number of unique data inputs. 35
4.4 Manual annotation of text predictions for E2E and WebNLG data. Annotations are between curly braces. 39
4.5 E2E dataset (*D*: Delexicalisation, *D+C*: delexicalisation and copying, *C*: copy and coverage, *NIL*: Neither copy nor delexicalisation). The upper half of the table presents automatic evaluation results; the lower half—human evaluation results. Best scores are in bold. 40
4.6 WebNLG dataset (*D*: Delexicalisation, *D+C*: delexicalisation and copying, *C*: copy and coverage, *NIL*: Neither copy nor delexicalisation). The upper half of the table presents automatic evaluation results; the lower half—human evaluation results. Best scores are in bold. * – word repetitions present in predictions. . . . 41
4.7 Example predictions for E2E. Mistakes are in bold. 42
4.8 Example predictions for WebNLG. Mistakes are in bold. 43

5.1 Word Ordering: BLEU scores on lemmatised data. Mean and standard deviation across three random seeds. BL: Baseline. All pairwise comparisons of BL and our model showed a statistically significant difference in BLEU via the bootstrap resampling (1000 samples, $p < .05$). 52
5.2 Proportion of correct head/dependent positioning for the five selected dependency relations: *det*, *nsubj*, *obj*, *amod*, *acl*, and overall performance across all dependency relations. +1: exact match; +2: approximate match, i.e. head and dependent are in the correct order but there is a one-token difference between gold and prediction. NA: no dependency relation found in a treebank. Δ indicates the difference between our delexicalised model and the baseline. 54
5.3 Morphological Realisation Results. MR Accuracy: accuracy of the MR module. WO: BLEU scores on lemmas. WO+MR: BLEU scores on inflected tokens without contraction (S^{-c}). 55
5.4 Contraction Generation Results (BLEU scores). S^{-c}/S^{+c}: a sentence without contractions vs. a reference sentence including contractions; S^{-c}: BLEU with respect to sentences before contractions; S^{+c}: BLEU with respect to a reference sentence. The scores were computed on detokenised sequences. 56
5.5 BLEU, DIST and NIST scores on the SR'18 test data (shallow track). SR'18 is the official results of the shared task but do not include OSU scores, since they are given in the line below. We also excluded the ADAPT and NILC scores as they were obtained using data augmentation. OSU is the submission of King and White (2018). 57
5.6 The MR module applied to the gold word ordering input. Predictions and reference sentences are both tokenised. Results on the development set. 59
5.7 Accuracy of the morphological realisation component. NA: no MR component was developed. Percentage and count of lemmas with ambiguous forms found in the training data. 59
5.8 WO component performance on the development set. Predictions and references (sequences of lemmas) are both tokenised. 61
5.9 Automatic metrics on the development set (WO + MR). Predictions and reference sentences are both tokenised. 61

6.1 Metrics for Syntactic Complexity of a sentence (the values in braces indicate the corresponding value for the tree in Figure 6.1). 69
6.2 Descriptive statistics (mean and stdev apart from the first two and the last column) for the UD treebanks used in SR'18 and SR'19. S: number of submissions, count: number of sentences in a test set, MDD: mean dependency distance, MFS: mean flux size, MFW: mean flux weight, MA: mean arity, NP: percentage of non-projective sentences. For the tree-based metrics (MDD, MFS, MFW, MA), macro-average values are reported. 71
6.3 Median values for BLEU, Fluency, and Adequacy for projective/non-projective sentences for each submission. Medians for non-projective sentences which are higher than for the projective sentences are in bold. All comparisons were significant with $p < 0.001$. Human judgments were available for *ru_syntagrus* only. . . 75
6.4 Macro-average dependency edge accuracy over all submissions sorted from the lowest accuracy to the highest. Count is a number of times a relation was found in all treebanks. 79

6.5 Top-20 of the most frequent suspicious trees (dep-based) across all submissions. In case of *conj*, when tree patterns were similar, they were merged, X serving as a placeholder. Coverage: percentage of submissions where a subtree was mined as suspicious. MSS: mean suspicion score for a subtree. 80
6.6 Most frequent suspicious trees (pos-based) across all submissions. 80
6.7 Most frequent suspicious trees (dep-pos-based) across all submissions. 81

A.1 Arabic. Exact match. 94
A.2 Arabic. Approximate match. 95
A.3 Czech. Exact match. 96
A.4 Czech. Approximate match. 97
A.5 English. Exact match. 98
A.6 English. Approximate match. 99
A.7 Spanish. Exact match. 100
A.8 Spanish. Approximate match. 101
A.9 Finnish. Exact match. 102
A.10 Finnish. Approximate match. 103
A.11 French. Exact match. 104
A.12 French. Approximate match. 105
A.13 Italian. Exact match. 106
A.14 Italian. Approximate match. 107
A.15 Dutch. Exact match. 108
A.16 Dutch. Approximate match. 109
A.17 Portuguese. Exact match. 110
A.18 Portuguese. Approximate match. 111
A.19 Russian. Exact match. 112
A.20 Russian. Approximate match. 113

Génération en langue naturelle : de la création des données à l'évaluation, en passant par la modélisation

À l'heure actuelle, une part importante des données disponibles est sous forme non textuelle : tables, images, bases de données, graphes de connaissance. Un des buts de la Génération en Langue Naturelle (*natural language generation*, NLG) est de convertir ces données en texte. Ce texte doit servir l'objectif de transmettre l'information contenue dans les données au lecteur. Pour cela, il doit respecter les règles grammaticales et syntaxiques de la langue dans lequel il est écrit, et représenter fidèlement l'information contenue dans les données dont il est issu.

La production d'un texte respectant ces deux conditions est une tâche difficile. Les avancées récentes en Traitement Automatique des Langues (Natural Language Processing, NLP) ont accéléré les progrès dans le domaine de la NLG. L'émergence de nouvelles méthodes augmentant la visibilité de certaines tâches et difficultés, la modélisation de la langue offre des défis sans cesse renouvelés. Le champ du NLP a encore de nombreux défis à surmonter, puisqu'un grand nombre des capacités linguistiques des humains ne peut pas être imité par les machines.

Dans de nombreuses sciences appliquées, le cycle de recherche traditionnel consiste en une collection de données, suivie d'expériences, puis d'analyse des résultats. Dans cette thèse, nous nous penchons sur les trois phases de ce cycle en NLG : la création de données, la modélisation et l'évaluation.

De nos jours, la plupart des approches de NLG s'appuient sur des données annotées via des méthodes d'apprentissage automatique. L'annotation de ces données est un processus coûteux ; de plus, la plupart des données annotées permettant l'entraînement de modèles de NLG sont en anglais. Dans cette thèse, nous explorons comment les données annotées créées en anglais peuvent être transférées à d'autres langues et les difficultées induites par cette approche.

En ce qui concerne la modélisation, nous nous concentrons sur plusieurs aspects. Premièrement, nous étudions de quelle manière la problématique des mots peu fréquents peut être gérée dans le cadre des approches à base de réseaux de neurones. Puisque dans ces approches, les patrons sont appris de manière statistique, le traitement des mots les plus rares requiert une procédure particulière. Deuxièmement, nous nous penchons sur la tâche de réalisation de surface qui consiste en la génération de texte à partir d'une représentation linquistique et qui est une composante particulière des systèmes de NLG. Cette composante peut servir d'intermédiaire dans le processus de génération afin d'améliorer l'encodage de la sortie désirée. Nous avons développé un réalisateur de surface, grâce auquel nous montrons de quels facteurs dépend la performance de la réalisation de surface. Ce réalisateur de surface est développé dans un cadre

multilingue, ce qui permet de constater les différences de performances entre les langues.

Après qu'un système de NLG ait produit un texte, ce dernier doit être évalué. Cette évaluation est difficile, du fait de la variété des données d'entrée et du nombre infini des textes générés possibles. Pour l'analyse des modèles de réalisation de surface, nous proposons un cadre d'identification des erreurs conditionnées par certaines caractèristiques des entrées. Cette analyse est réalisée sur plusieurs systèmes multilingues. Enfin, nous nous intéressons à la méta-évaluation et évaluons le pouvoir de corrélation entre certains indicateurs et les jugements humains à propos des textes générés.

Dans ce qui suit, nous détaillons ces sujets de recherche.

Création d'un corpus d'apprentissage pour la Génération en Langue Naturelle

La génération data-to-text est une tâche clé de la NLG, qui consiste à verbaliser les informations contenues dans les données, par exemple, dans les bases de connaissance. Un corpus contenant des données mises en correspondance avec leur expression textuelles peut permettre l'apprentissage d'un modèle data-to-text. Cependant, de tels corpus sont peut courants et leur création manuelle est hautement complexe. Enfin, les quelques corpus existant ont été développés en anglais. Des méthodes permettant la création automatique de corpus data-to-text multilingues à partir de ces corpus existants seraient donc d'une grande valeur.

triplets RDF	<Asterix, creator, René Goscinny> <René Goscinny, nationality, French people>
Original	Rene Goscinny is a French national and also the creator of the comics character Asterix.
MT	Рене Госкино - французский гражданин, а также создатель комического персонажа Астерикс. Rene Goskino French national and also creator comic$_{gen}$ character$_{gen}$ Asterix$_{inan}$
PE	Рене Госинни - французский гражданин, а также создатель персонажа комиксов Астерикса. Rene Goscinny French national and also creator character$_{gen}$ comics$_{gen}$ Asterix$_{anim}$
Erreurs	entité nommée, vocabulaire, grammaire

Table 1: Exemple original de WebNLG dans la catégorie ComicsCharacter, sa traduction russe (MT), et sa traduction post-éditée (PE) ainsi que l'annotation des erreurs. Les erreurs sont colorées en bleu. *gen*—génitif; *anim*—animé; *inan*—inanimé.

Dans cette thèse nous introduisons une méthode de création d'un corpus data-to-text en russe à partir du corpus data-to-text anglophone de WebNLG (Gardent et al., 2017a). Notre méthode inclut trois étapes principales. Premièrement, nous utilisons un modèle neuronal de traduction automatique (*neural machine translation*, NMT) pour traduire en russe les textes anglophones de WebNLG. Deuxièmement, nous analysons en détail les erreurs contenues dans le texte produit par le modèle de NMT. Nous constatons que les entités nommées sont la source d'erreur la plus importante durant la traduction. La Table 1 montre un exemple tiré de WebNLG. Troisièmement, nous exploitons deux méthodes de post-édition automatique des entités nommées : l'une à base de règles de post-édition obtenues à partir de corrections manuelles, l'autre basée sur un réseau de neurones séquence-vers-séquence. Nos résultats montrent que la post-édition basée sur le modèle neuronal a des performances égales à la baseline, ce qui confirme les observations effectuées lors des tâches communes de post-édition automatique (Chatterjee et al.,

2018, 2019). D'une manière générale, nous concluons de nos expériences que l'approche à base de règles est plus robuste que sont équivalent neuronal.

Gestion des items peu fréquents en génération en langue naturelle

Les corpus de NLG décrivent souvent des informations factuelles, et par conséquent les entrées des systèmes de NLG contiennent souvent des items peux fréquents tels que des noms, lieux ou dates. Par conséquent, dans le cadre de l'apprentissage supervisé, il est difficile pour les modèles neuronaux de prédire la verbalisation de ces entrées. Typiquement les approches de génération neuronales tentent de surmonter ces problèmes par la délexicalisation (Wen et al., 2015; Dušek and Jurčíček, 2015; Trisedya et al., 2018; Chen et al., 2018) ou en utilisant un mécanisme de copie (Chen, 2018; Elder et al., 2018; Gehrmann et al., 2018). Les encodages charactère par charactère (Agarwal and Dymetman, 2017; Deriu and Cieliebak, 2018a; Jagfeld et al., 2018) et les encodages par paires d'octets ont également été utilisés (Elder, 2017; Zhang et al., 2018).

Dans cette thèse nous étudions l'impact de la copie et de la délexicalisation sur la qualité des textes générés par deux modèles sequence-to-sequence avec mécanisme d'attention : l'un utilisant un mécanisme de copie et couverture See et al. (2017), l'autre utilisant la délexicalisation. Nous évaluons leurs sorties respectives sur deux datasets, à savoir E2E (Novikova et al., 2017b) et le dataset de WebNLG (Gardent et al., 2017a). Nous comparons les deux méthodes dans deux cadres : d'une part les partitions "apprentissage/validation/test" originales données par E2E et par WebNLG, d'autre part des partitions plus contraintes visant à minimiser le nombre de redondance entre les partitions d'apprentissage, validation et test. Nous montrons que (i) les items peu fréquents impactent fortement la performance, (ii) combiner la délexicalisation et la copie donnent les améliorations les plus importantes, (iii) la copie est moins performante que la délexicalisation pour la gestion des items peu fréquents, (iv) que l'impact de ces deux mécanismes dépendent grandement de la manière dont le dataset est construit et de comment la partition "apprentissage/validation/test" est réalisée.

Entrainement des modèles de réalisation de surface

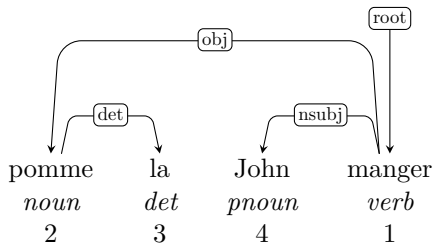
Dans les sections précédentes, nous nous sommes concentrés sur les entités nommées et avons abordé la manière de les traiter lors de la création de données d'entraînement pour la NLG et lors de la génération à partir de données riches en entités nommées. Les données d'entrée considérées étaient des triplets RDF et des actes de dialogue. Dans cette section, nous traitons d'un autre type d'entrée — les arbres de dépendance non ordonnés — et nous explorons la génération multilingue en nous concentrant sur la tâche de réalisation de surface.

La réalisation de surface (*surface realisation*, SR) est une tâche consistant à associer une représentation sémantique à une phrase. Dans le cas du data-to-text, elle s'intègre à un processus complexe visant à sélectionner, compresser et structurer les données d'entrée afin d'obtenir un texte (Reiter and Dale, 2000). Dans le cas du text-to-text, elle peut servir à reformuler une partie ou la totalité du texte d'entrée. Dans cette thèse, nous nous concentrons sur les campagnes d'évaluation de réalisation de surface peu profondes de SR'18 et SR'19, où l'entrée est un arbre de dépendance lemmatisé et non ordonné, à partir duquel une phrase bien formée doit être produite. Dans cette thèse, nous proposons une approche neuronale qui décompose la réalisation de surface en trois sous-tâches : l'ordonnancement des mots, l'inflexion morphologique et la génération de contractions (comme les clitiques en portugais ou l'élision en français). Nous fournissons une analyse détaillée de la façon dont chacun de ces phénomènes (ordre des mots,

réalisation morphologique et contraction) est traité par le modèle, et nous commentons les différences observées d'une langue à l'autre.

Entrée

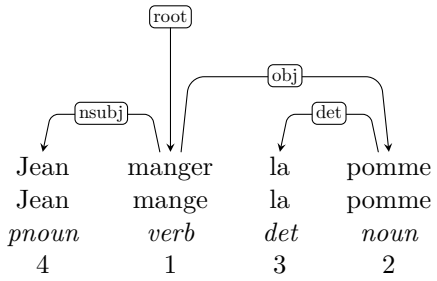
(a) Arbre source non ordonné



2:noun:obj:1 3:det:DET:2 4:pnoun:nsubj:1 1:verb:root:0

Sortie

(b) Lemmes de sortie avec l'arbre d'analyse d'or



4 1 3 2

Figure 1: Délexicalisation et linéarisation (dans l'arbre d'analyse de la phrase de sortie, la première ligne montre les lemmes, la deuxième les formes de mots, la troisième les parties du discours et la quatrième les identifiants). Les identifiants sont assignés aux nœuds de l'arbre dans l'ordre donné par une recherche en profondeur.

Dans ce qui suit, nous décrivons plus en détail notre approche de réalisation de surface. Comme nous l'avons déjà mentionné, elle comporte un module distinct pour chacun des trois composants.

Ordonnancement des mots

L'ordonnancement des mots peut être vu comme une tâche sequence-to-sequence, où un arbre d'entrée est linéarisé (voir Figure 1). Nous linéarisons les noeuds via une recherche en profondeur et délexicalisons tous les lemmes d'entrée (autrement dit, nous les remplaçons par des identifiants dans l'entrée et la sortie et les enrichissons avec des attributs). Nous entraînons ensuite, pour chaque langue, un modèle neuronal de type encodeur-décodeur factorisé, où les facteurs sont les relations de dépendance, les labels de parties du discours et les identifiants des nœuds parents (Elder and Hokamp, 2018; Alexandrescu and Kirchhoff, 2006). Lors de la relexicalisation, nous remplaçons tous les identifiants par les lemmes infléchis.

Réalisation de morphologie

Les paradigmes morphologiques sont appris à partir de paires (lemme, partie du discours+attributs) extraites des données d'entraînement (les champs `upos` et `features` du format CoNLL) en utilisant le modèle d'Aharoni and Goldberg (2017). Les lemmes sans attributs morphologiques ne sont pas utilisés. Comme les attributs ne sont pas fournis pour certains lemmes, le module de réalisation morphologique n'est pas entraîné pour les langues contenant ces lemmes. Au lieu de cela, pendant la phase d'inflexion, les lemmes sont soit copiés tels quels dans la sortie, soit un dictionnaire (construit à partir des données d'entraînement) associant chaque lemme+POS à la forme du mot est utilisé. Si la clé lemme+POS est absente du dictionnaire, le lemme est copié tel quel dans la sortie. La même règle s'applique pour tout lemme ne présentant aucune

caractéristique morphologique connue (par exemple, les URL, les mots étrangers, les chiffres, les signes de ponctuation, etc.)

Génération de contractions

La génération de contractions est implémentée pour certaines langues afin de gérer les clitiques, les contractions et l'élision. L'exemple 1 montre quelques types de contractions.

(1) Français : "Le chat dort." / "L'alouette chante." (Élision pour l'article défini *le* devant une voyelle : Le → L')

Italien : "In il mare." → "Nel mare." (Contraction de la préposition *in* et de l'article *il* : In il → Nel)

Portugais : "*Eis lo." → "Ei-lo." (Attachement du pronom clitique : Eis lo → Ei-lo)

Nous avons développé deux modules pour la génération de contractions : l'un basé sur des expressions régulières, l'autre basé sur un modèle sequence-to-sequence. Ce dernier est entraîné sur des paires de phrases avec et sans contractions. Le module à base d'expressions régulières est inspiré de la décomposition d'expression plurilexicales telles que les contractions, qui est appliquée pendant la tokenisation et le parsing.

Finallement, nous avons inclus une dernière étape de détokenisation consistant à regrouper certains tokens, certains langues exigeant des règles de détokenisation spécifiques afin de produire des phrases naturelles pour un utilisateur final.

Conclusions principales

Les résultats expérimentaux montrent qu'une délexicalisation complète améliore les performances de manière nette. Linguistiquement parlant, cela confirme l'intuition selon laquelle la correspondance entre la structure peu profonde de dépendance et l'ordre des mots peut être apprise indépendamment des mots spécifiques employés. Nous menons également une évaluation détaillée des performances de notre modèle d'ordonnancement des mots pour chaque langue de la campagne d'évaluation SR. L'évaluation des contraintes d'ordonnancement parent/enfant montre que les relations à longue distance et les contraintes d'ordres irréguliers de mots (comme par exemple la position du verbe dans les clauses principales et subordonnées en néerlandais) impacte négativement le résultat.

Le développement de cette approche multilingue nous a permis de constater la difficulté d'adapter un modèle de génération à une nouvelle langue. La composante d'ordonnancement des mots est aisément transférable d'une langue à l'autre et ne demande que peu d'effort pour être appliqué aux langues non encore traitées. En revanche, la composante de réalisation morphologique demande beaucoup d'attention et doit être réglée pour chaque langue séparément.

Evaluation des réaliseurs de surface

Dans cette partie de la thèse, nous tentons d'approfondir notre méthode d'évaluation de la réalisation de surface et de l'étendre en un cadre d'évaluation complet en considérant les sorties de tous les systèmes soumis à deux tâches de réalisation de surface multilingues.

Les mesures d'évaluation standards des modèles de NLG telles que BLEU ou METEOR échouent à indiquer quels facteurs linguistiques impactent les performances. En nous concentrant sur la réalisation de surface (c'est à dire la tâche de conversion d'un arbre de dépendance

non ordonné en une phrase bien formée), nous proposons un cadre d'analyse des erreurs qui permet d'identifier quelles caractéristiques de l'entrée affectent les résultats du modèle. Ce cadre consiste en deux composantes principales : (i) une analyse de corrélations entre de nombreuses mesures syntaxiques et des mesures de performances standards, (ii) un ensemble de techniques pour détecter automatiquement les constructions syntaxiques corrélées avec de faibles scores de performance. Nous montrons les avantages de notre cadre en effectuant une analyse des erreurs sur les résultats de 174 tests de systèmes de réalisation de surface soumis à des campagnes d'évaluation ; nous montrons que la précision des liens de dépendance est corrélée avec des mesures automatiques, fournissant une base plus interprétable pour l'évaluation ; nous suggérons des utilisations de notre modèle visant à améliorer les modèles et les données. Nous mettons ce cadre à disposition sous la forme d'un outil à destination des chercheurs souhaitant améliorer leurs modèles et datasets, ainsi que des organisateurs de campagnes d'évaluation souhaitant fournir des retours détaillés et linguistiquement interprétables sur l'état de l'art de la réalisation de surface multilingue.

1

Introduction

A lot of information in the world comes in the form of non-textual data: tables, images, databases, knowledge graphs. One of the goals of natural language generation (NLG) is to convert data to a well-formed text. This text should achieve a communicative goal: transmitting information encoded in the data to the reader. To this end, it should be fluent and accurate. A fluent text respects syntactical and grammatical rules of natural language, and an accurate text faithfully reflects the information conveyed in the data.

Producing accurate and fluent texts is a challenging task. Recent advances in natural language processing (NLP), principally based on neural networks, have given a boost to language generation tasks. However, modelling language is always a moving target. With the emergence of new methods, new tasks and challenges become more apparent, and computational linguistics as a field has a long way to go, since many human, language-related capabilities are not replicated by machines.

The traditional research cycle in many applied sciences consists in collecting some data, doing some experiments, and analysing them. In this work, we investigate the three main steps of this cycle in NLG: data creation, modelling, evaluation.

In NLG, most approaches nowadays are data-driven, i.e. learning happens on the basis of some labelled data. Performing annotation is a costly process; furthermore, most labelled data in generation has been created for English. In this thesis, we explore how annotated data created in English can be transferred to another language and what difficulties this process entails.

Regarding modelling NLG systems, we focus on several aspects. First, we investigate how non-frequent words can be handled in data-driven, neural approaches. Since in those approaches patterns are learnt statistically, one needs a special procedure for treatment of rare words. Second, we deal with surface realisation, a special component of NLG systems, which generates text from a linguistic representation. This component may serve as an intermediary in the generation process to better encode the desired output. The developed surface realiser enables to show which linguistic factors surface realisation performance depends on. Moreover, the realiser is developed in a multilingual setting, what allows to see differences in performance between languages.

Once an NLG system produces a text, it needs to be evaluated. NLG evaluation is notoriously difficult given a variety of inputs and an infinite number of possible generated texts. Given a task of surface realisation, we propose a framework for identifying errors conditioned on the input characteristics. This analysis is carried out on several multilingual systems. Finally, we also deal with meta-evaluation and assess the ability of some automatic indicators to correlate with human judgments as far as generated texts are concerned.

Thesis Outline

Chapter 2 (Background) introduces the field of Natural Language Generation. We discuss inputs to NLG systems and neural methods used in the thesis. We review recent neural-based NLG systems and point out some directions for future research.

Chapter 3 (Creating Training Corpora for Natural Language Generation) discusses how to create a data-to-text generation corpus for Russian using English data. We propose a semi-automatic approach based on machine translation and highlight that most translation errors are related to named entities. We discuss two approaches to remedy those errors: a rule-based method and an approach based on automatic post-editing.

Chapter 4 (Handling Rare Items in Natural Language Generation) focuses on handling rare input items in neural NLG systems. In such systems, the rare items are usually managed using either delexicalisation or a copy mechanism. We investigate the relative impact of these two methods on two datasets (E2E and WebNLG) and using two evaluation settings. We show that rare items strongly impact performance; that combining delexicalisation and copying yields the strongest improvement; that copying underperforms for rare and unseen items; and that the impact of these two mechanisms greatly varies depending on how the dataset is constructed and on how it is split into training, development and test sets.

Chapter 5 (Training Models for Surface Realisation) presents our model for shallow surface realisation—generation from unordered dependency trees. We propose a modular approach to surface realisation and evaluate our approach on the multiple languages covered by the SR'18 and SR'19 Surface Realisation Shared Task shallow tracks. We provide a detailed evaluation of how word order, morphological realisation and contractions are handled by the model and an analysis of the differences in word ordering performance across languages. Our experimental results show that mapping between a dependency tree and word order can be learned independently of the lexical forms.

Chapter 6 (Evaluating Surface Realisers) presents a framework for error analysis which permits identifying which features of the input affect surface realisation models' results. This framework consists of two main components: *(i)* correlation analyses between a wide range of syntactic metrics and standard performance metrics and *(ii)* a set of techniques to automatically identify syntactic constructs which often co-occur with low performance scores. We demonstrate the advantages of our framework by performing error analysis on the results of 174 system runs submitted to the Multilingual Surface Realisation shared tasks. We show that tree characteristics do not correlate with model performance, and we identify constructions where models underperform. We also suggest ways in which our framework could be used to improve models and data.

Chapter 7 (Evaluating Natural Language Generation Systems) discusses the difference between sentence-level and system-level evaluation in NLG. We validate three automatic metrics by correlating them to human judgments and show that depending on the usage of a system- or sentence-level correlation analysis, correlation results between automatic scores and human judgments are inconsistent.

Chapter 8 (Conclusion) sums up the main research conclusions of the thesis and outlines some future research directions.

List of Publications

Parts of this thesis have appeared in the following publications and reports:

- Shimorina, A., Parmentier, Y., and Gardent, C. (2021). An Error Analysis Framework for Shallow Surface Realization. *Transactions of the Association for Computational Linguistics*, 9:429–446

- Shimorina, A. and Gardent, C. (2019b). Surface realisation using full delexicalisation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3086–3096, Hong Kong, China. Association for Computational Linguistics

- Shimorina, A. and Gardent, C. (2019a). LORIA / Lorraine University at multilingual surface realisation 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 88–93, Hong Kong, China. Association for Computational Linguistics

- Shimorina, A., Khasanova, E., and Gardent, C. (2019). Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 44–49, Florence, Italy. Association for Computational Linguistics

- Shimorina, A. and Gardent, C. (2018). Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands. Association for Computational Linguistics

- Shimorina, A. (2018). Human vs automatic metrics: on the importance of correlation design. *Peer-reviewed, non-archival, presented at the Widening NLP Workshop 2018 at NAACL*, arXiv: 1805.11474

- Shimorina, A., Gardent, C., Narayan, S., and Perez-Beltrachini, L. (2018). WebNLG Challenge: Human Evaluation Results. Technical report, Loria & Inria Grand Est

2

Background

Contents

2.1 Natural Language Generation **5**
2.1.1 Introduction to NLG 5
2.1.2 Input Data in NLG 7
2.2 Methods in Natural Language Generation **11**
2.2.1 Encoder-Decoder Framework 11
2.2.2 Attention 13
2.2.3 Copy and Coverage Mechanisms 14
2.2.4 Factored Models 16
2.2.5 Transformers, Pre-trained Language Models 16
2.2.6 Encoder-Decoder Models for NLG 18
2.3 Future Directions **19**

2.1 Natural Language Generation

2.1.1 Introduction to NLG

Natural language generation (NLG) is the process of generating a natural language text from some input. This input can be texts, documents, images, tables, knowledge graphs, databases, dialogue acts, meaning representations, etc.

The purpose of NLG is to model natural language texts in their full variety. NLG is encountered in numerous applications which extend well beyond Natural Language Processing (NLP) tasks: medicine, finance, commerce, sports, journalism, and any other where there is a need to generate texts based on some data. NLG is a field on its own within NLP, however, it may be also a component of other NLP tasks whose output is language: dialogue, question answering, creative writing.

Let us consider a few examples of NLG applications:

- in health care and medicine, reports for health professionals and explanatory materials for patients are generated based on medical databases and patient records (Cawsey et al., 1997; DiMarco et al., 2007; Portet et al., 2009; Hasan and Farri, 2019);

- in finance, stock reports are generated by aggregating financial and economic indicators (Plachouras et al., 2016);
- in journalism, automated reports and news stories are generated based on machine-readable data such as polling station statistics or candidate profiles in elections (van Dalen, 2012; Leppänen et al., 2017);
- in sports, summaries of football matches are generated using game statistics (Theune et al., 2001; Chen and Mooney, 2008).

In research, NLG has been a long-standing problem. Early works in NLG deal with generation of answers from databases (McKeown, 1982) and report generation from knowledge bases (Kukich, 1983). Initial NLG systems were mainly based on rules and templates. Rule-based systems which were dominant at that time led to the construction of the standard NLG pipeline where several components are commonly identified:

1. Document Planning, which decides what information should be included.
2. Microplanning, which decides how to organise information and how to convey it.
3. Realisation, which, relying on the decisions taken in steps 1-2, converts abstract representations to a well-formed text.

More fine-grained steps of this pipeline are described in Reiter and Dale (2000), and for an overview of developments related to each of these components and their subproblems, we refer the reader to Gatt and Krahmer (2018). In early, traditional NLG systems those stages were executed in a sequence, leading to errors propagating along the pipeline. Since decisions taken upstream inevitably affect those taken downstream in the pipeline architecture, it is quite possible for early decisions to result in problems later on (e.g., because the document is planned in such a way that the microplanner lacks the resources to express relations between messages coherently). For instance, Meteer (1991) and Robin and McKeown (1996) discuss this "generation gap", which arises when a module's choices result in a dead end in a later module (e.g., an ordering of events is such that the microplanner does not know how to express temporal relations). To address that architectural issue, a number of end-to-end approaches have been developed: at first, statistical (Konstas and Lapata, 2013), and then neural-based, which constitute a dominant trend in NLG today.

Statistical methods, replacing hand-crafted rules and templates, tended to perform generation in a more unified framework, rather than strictly separating the pipeline's steps (Belz, 2008; Liang et al., 2009; Angeli et al., 2010; Kondadadi et al., 2013; Mairesse and Young, 2014). Current neural methods, more than ever, conflate all the components into a single end-to-end NLG system (Lebret et al., 2016; Wiseman et al., 2017). Neural-based NLG will be briefly reviewed below in Sections 2.2.5 and 2.2.6.

Another important strand of early NLG research has been dedicated to surface realisation from grammars. The goal consists in rendering a text given a grammar. Different grammar formalisms were used to perform surface realisation: a Probabilistic Context-Free Grammar (Belz, 2008), Head-driven Phrase Structure Grammar (Nakanishi et al., 2005a), Lexical Functional Grammar (Cahill and van Genabith, 2006), Combinatory Categorial Grammar (White et al., 2007), etc. Although many of these systems were statistical, they focused exclusively on one part of the generation process.

Figure 2.1: A game record and its corresponding document (partial view) in RotoWire. Picture from Wiseman et al. (2017).

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AS ...
Heat	11	12	103	49	47	27
Hawks	7	15	95	43	33	20

PLAYER	AS	RB	PT	FG	FGA	CITY ...
Tyler Johnson	5	2	27	8	16	Miami
Dwight Howard	4	17	23	9	11	Atlanta
Paul Millsap	2	9	21	8	12	Atlanta
Goran Dragic	4	2	21	8	17	Miami
Wayne Ellington	2	3	19	7	15	Miami
Dennis Schroder	7	4	17	8	15	Atlanta
Rodney McGruder	5	5	11	3	8	Miami
Thabo Sefolosha	5	5	10	5	11	Atlanta
Kyle Korver	5	3	9	3	9	Atlanta
...						

The Atlanta Hawks defeated the Miami Heat , 103 - 95 , at Philips Arena on Wednesday . Atlanta was in desperate need of a win and they were able to take care of a shorthanded Miami team here . Defense was key for the Hawks , as they held the Heat to 42 percent shooting and forced them to commit 16 turnovers . Atlanta also dominated in the paint , winning the rebounding battle , 47 - 34 , and outscoring them in the paint 58 - 26.The Hawks shot 49 percent from the field and assisted on 27 of their 43 made baskets . This was a near wire - to - wire win for the Hawks , as Miami held just one lead in the first five minutes . Miami (7 - 15) are as beat - up as anyone right now and it 's taking a toll on the heavily used starters . Hassan Whiteside really struggled in this game , as he amassed eight points , 12 rebounds and one blocks on 4 - of - 12 shooting ...

2.1.2 Input Data in NLG

Historically stand-alone NLG was often viewed as a data-to-text task (Reiter and Dale, 2000). With the course of time this has changed; nowadays vision-to-text and text-to-text applications—machine translation, summarisation, simplification, paraphrasing, etc.—are included into the NLG definition as well. In this thesis, we restrict the definition of NLG by excluding vision- and text-to-text generation. So our focus will be primarily on tasks including data or meaning representations as an input to NLG systems.

As discussed in Section 2.1.1, a plethora of inputs exist for NLG, depending on a task in question; this section is chiefly devoted to the inputs that are used in the thesis and some challenges related to them.

Tables and Structural Records

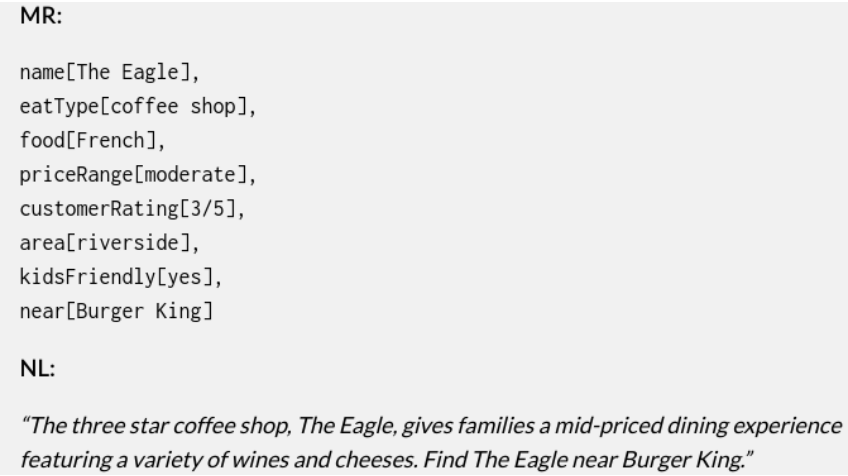
A common input to data-to-text NLG systems are tables and other structural records (Sripada et al., 2002; Chen and Mooney, 2008; Lebret et al., 2016; Parikh et al., 2020). Figure 2.1 gives an example of such input from the RotoWire dataset (Wiseman et al., 2017), which provides summaries of basketball games paired with game statistics. As illustrated by the example, table-to-text tasks quite often require a content selection process: not all the scores and entities mentioned in the tables are found in the text. Furthermore, such inputs may require reasoning about entity relationship and some numerical inference, which, in addition, makes tables an interesting target for Natural Language Understanding (NLU) tasks.

Generally, tabular data is the least specified NLG input, leaving a lot of choices for NLG systems in terms of microplanning and realisation.

RDF triples:
Arròs_negre - country - Spain
Arròs_negre - ingredient - White_rice
Text:
White rice is an ingredient of Arròs negre which is a traditional dish from Spain.

Table 2.1: RDF triples and their corresponding text in WebNLG.

Figure 2.2: A dialogue act and its corresponding text in E2E. MR: meaning representation. Picture from http://www.macs.hw.ac.uk/InteractionLab/E2E.



RDF triples

Another type of input of data-to-text systems are records from databases and knowledge bases (Elsahar et al., 2018). Here we consider an example from the WebNLG dataset (Gardent et al., 2017a), which aligns DBpedia (Lehmann et al., 2015) records to text (Table 2.1). DBpedia is a knowledge base extracted from Wikipedia, and it is constructed using technologies from Semantic Web and Linked Data. DBpedia data is stored as Resource Description Format (RDF) triples of the form subject-predicate-object where the subject is a Uniform Resource Identifier (URI), the predicate is a binary relation, and the object is either a URI or a literal value such as a string, a date or a number.

Comparing to tabular data, RDF triples can be considered as a more specified semantic representation, with predicates serving to link entities and indicating the relation between them.

In case of WebNLG, no content selection is to be modelled: all triples are found in texts. On the other hand, microplanning has a lot of operations to handle: mapping data to words (lexicalisation), using linguistic constructs to avoid repetitions (aggregation), exploiting the appropriate syntactic constructs to build sentences (realisation), and referring expression generation.

Dialogue Acts

NLG modules are used in dialogue modelling to generate utterances in conversations. Input data for utterance generation is a dialogue act expressed through a meaning representation (Mairesse et al., 2010; Wen et al., 2015; Juraska et al., 2019). Exemplified by the E2E dataset (Figure 2.2), a meaning representation consists of an unordered set of attributes/slots and their

values. Utterances in most dialogue systems also have an intent, i.e. they may serve different functions in a conversation, such as request, statement, opinion, agreement, etc. In E2E, the focus was on restaurant recommendations only, so the intent (*recommend/inform*) was omitted.

Dialogue acts give a more semantically specified representation of input than in data-to-text applications. The main challenges in generation from dialogue-based meaning representations include reproducing lexical richness and syntactic complexity and handling diverse discourse phenomena.

Semantic and Syntactic Representations

Another type of input to NLG systems stems from syntactic or semantic formalisms. Those can be regarded as the most specified representations. For NLG systems, these inputs require to perform a surface realisation step, i.e. to decide which syntactic constructs should be used to render lexical items to a well-formed text.

One of the examples of such input is the data used in the surface realisation shared tasks (Belz et al., 2011; Mille et al., 2018a). These tasks have two tracks: shallow and deep. In the shallow track, inputs are derived from syntactic trees constructed according to dependency grammar (Figure 2.3). The task of NLG systems is to reconstruct the correct word order and perform morphological inflection given lemmas and their morphological features. In the deep track, morphological and syntactical information is removed, and relations between words are now semantic, akin to predicate-argument labels (Figure 2.4).

Apart from dependency trees, other representations have been used as input for NLG: Abstract Meaning Representation (Banarescu et al., 2013), Minimal Recursion Semantics (Hajdik et al., 2019), Discourse Representation Structures (Basile and Bos, 2011), lambda calculus expressions (Lu and Ng, 2011), first-order logic (Gerdemann and Hinrichs, 1990), and various grammar formalisms (Cahill and van Genabith, 2006; White et al., 2007).

Figure 2.3: SR shallow track. Input for English in the CoNLL-U format and in a graphical form for the output sentence *This happened very quickly, and I wanted to make sure that I let everyone know before I left.* Picture from http://taln.upf.edu/pages/msr2018-ws/SRST.html#examples.

```
1       that    _       SCONJ   IN      _       5       mark    _       _
2       sure    _       ADJ     JJ      Degree=Pos      12      xcomp   _       _
3       ,       _       PUNCT   ,       _       19      punct   _       _
4       I       _       PRON    PRP     Case=Nom|Number=Sing|Person=1|PronType=Prs      19      nsubj   _       _
5       let     _       VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        2       ccomp   _       _
6       this    _       PRON    DT      Number=Sing|PronType=Dem        7       nsubj   _       _
7       happen  _       VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        0       root    _       _
8       before  _       SCONJ   IN      _       13      mark    _       _
9       know    _       VERB    VB      VerbForm=Inf    5       xcomp   _       _
10      quickly _       ADV     RB      _       7       advmod  _       _
11      and     _       CCONJ   CC      _       19      cc      _       _
12      make    _       VERB    VB      VerbForm=Inf    19      xcomp   _       _
13      leave   _       VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        5       advcl   _       _
14      I       _       PRON    PRP     Case=Nom|Number=Sing|Person=1|PronType=Prs      13      nsubj   _       _
15      .       _       PUNCT   .       _       7       punct   _       _
16      everyone        _       PRON    NN      Number=Sing     5       obj     _       _
17      to      _       PART    TO      _       12      mark    _       _
18      very    _       ADV     RB      _       10      advmod  _       _
19      want    _       VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        7       conj    _       _
20      I       _       PRON    PRP     Case=Nom|Number=Sing|Person=1|PronType=Prs      5       nsubj   _       _
```

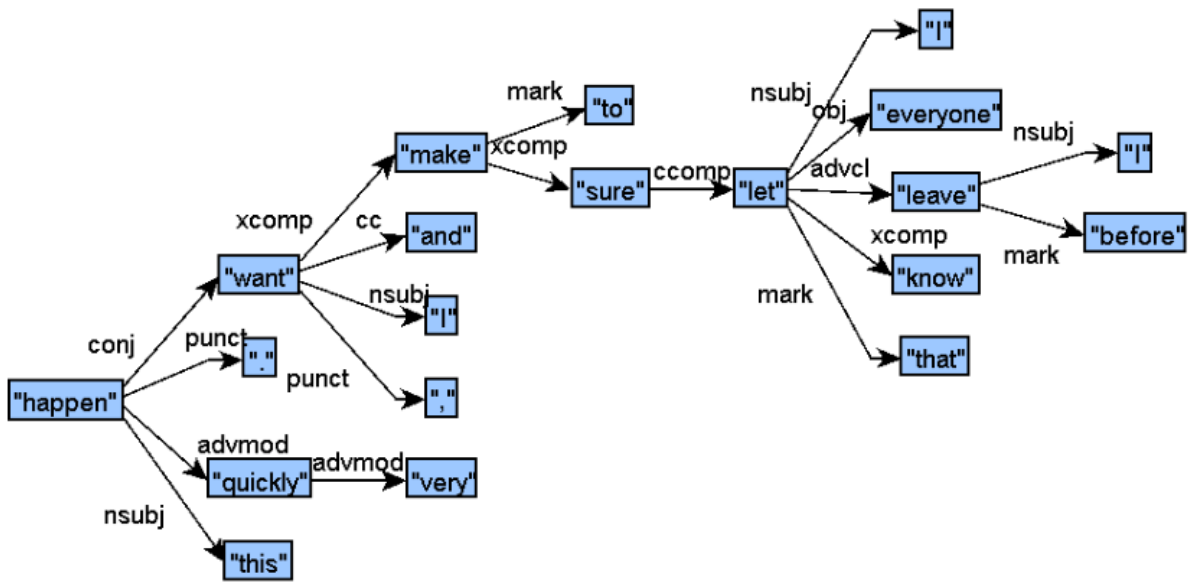
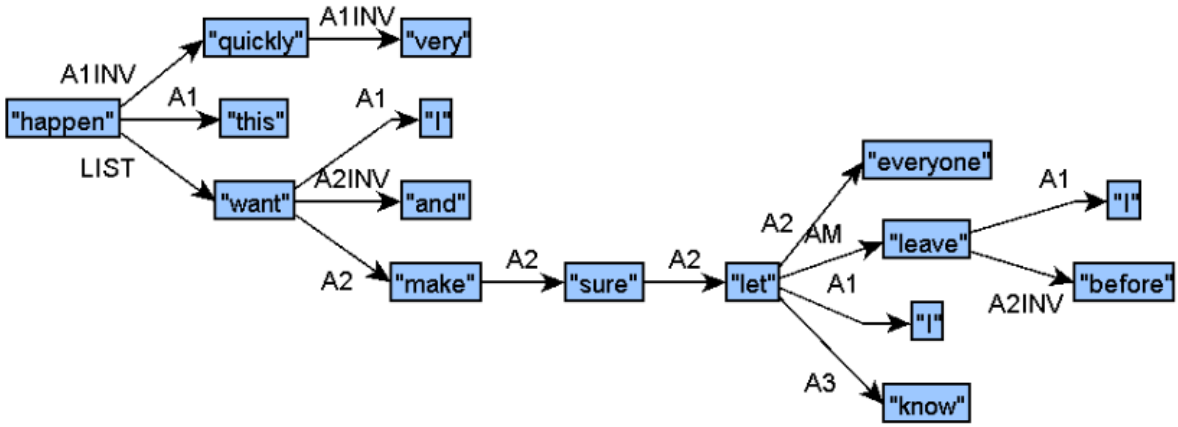


Figure 2.4: SR deep track. Input for English in a graphical form for the output sentence *This happened very quickly, and I wanted to make sure that I let everyone know before I left.* Picture from http://taln.upf.edu/pages/msr2018-ws/SRST.html#examples.



2.2 Methods in Natural Language Generation

After rule-based, grammar-based and statistical methods which were dominant in NLG, nowadays methods mostly rely on deep learning, a paradigm based on deep neural networks. In the thesis, deep learning was our primary method to model NLG systems. In Sections 2.2.1–2.2.4, we cover some concepts of neural networks, which we will use throughout the thesis. In Section 2.2.5, we review some recent developments in deep learning, and in Section 2.2.6, we provide an overview of some neural-based systems in NLG.

2.2.1 Encoder-Decoder Framework

One of the common frameworks for neural modelling is an encoder-decoder model, initially introduced for machine translation (Sutskever et al., 2014; Cho et al., 2014). In this framework, the encoder reads the input and creates its representation, the decoder uses this representation and produces the output.

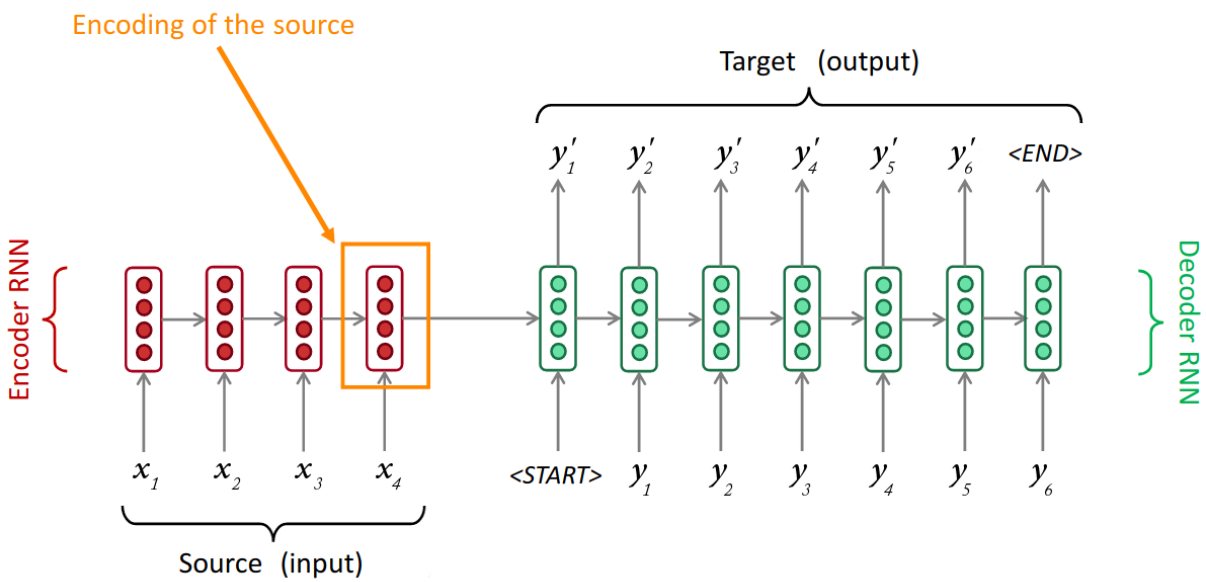
Here, we consider a type of the encoder-decoder framework, a sequence-to-sequence model based on Recurrent Neural Networks, where both encoder and decoder take a sequential form (Figure 2.5). This model is used in many NLG and NLP applications: in summarisation, a long document is encoded to produce a summary; in simplification, a complex sentence is encoded to produce a simple sentence; in data-to-text generation, a data input is encoded to produce a text describing the data, etc.

In sequence-to-sequence models, given an input sequence X: $x_1, x_2, x_3, ..., x_n$ and an output sequence Y: $y_1, y_2, y_3, ..., y_m$, the learning consists in finding a function p, which is a conditional probability $p(Y|X, \theta)$, where θ are learned parameters. Given this function, the goal is to find the most probable output sequence for the input sequence, that is

$$Y' = \underset{Y}{\operatorname{argmax}}\, p(Y|X, \theta). \tag{2.1}$$

Each sequence element is represented as a vector (*embedding*). Embeddings can be either ini-

Figure 2.5: An RNN-based sequence-to-sequence model in the training phase. Picture modified from http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf.



tialised randomly and then learned with other network parameters, or they can be pretrained. For instance, word2vec or GloVe are pretrained embeddings for words (Mikolov et al., 2013; Pennington et al., 2014). As illustrated in Figure 2.5, each input element is fed into the encoder, which is a Recurrent Neural Network (RNN), over a series of timesteps from 1 to 4. At each timestep i, the RNN takes an input vector x_i and produces a hidden state h_i:

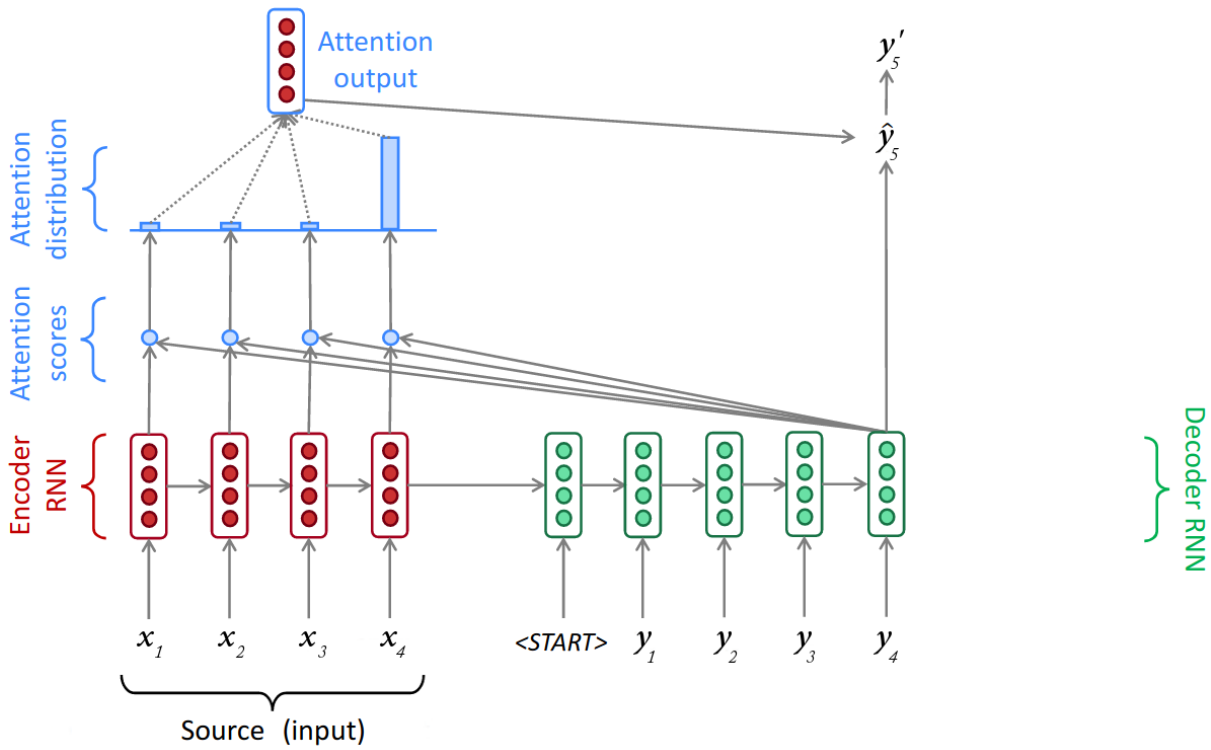
$$h_i = f(x_i, h_{i-1}, \theta), \tag{2.2}$$

where h_{i-1} is the hidden state of the previous timestep and f is an activation function. There exists a variety of activation functions ranging from logistic sigmoid and hyperbolic tangent non-linear functions to complex architectures, such as Long short-term memory (LSTMs; Hochreiter and Schmidhuber, 1997) or Gated Recurrent Units (GRUs; Cho et al., 2014). The last hidden state of the encoder is a final representation of the input sequence (in the orange frame in the figure), called a context vector.

In our example, the RNN encoder takes information from the left context only. An improved model, known as the bidirectional RNN, uses both left and right contexts and consists of two RNNs: an encoder which reads the sequence from left to right (forward pass) and a second encoder which reads the sequence from right to left (backward pass). Another possible modification of sequence-to-sequence models is to have more than one RNN layer.

The decoder in sequence-to-sequence models is also an RNN. Conditioned on the context vector, the decoding consists in generating an output sequence. The context vector is used as an initial decoder hidden state, and a special embedding representing the start of the sequence (<START>) is given to the decoder as input. A probability vector (this vector is a distribution over the output vocabulary) over the next element is outputted at each timestep i, and then the most probable element y'_i is chosen as prediction. This process is repeated with the next element

Figure 2.6: A sequence-to-sequence model with attention. Picture modified from http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf.



y_{i+1}, which is given as input at the next timestep. The method when a gold output sequence is given to the decoder during training is called *teacher forcing*. At test phase, previously predicted elements y'_i are used for feeding rather than a gold output sequence.

The training goal of sequence-to-sequence models is to predict a probability of the next element given the input and previous target elements. An objective function is used to maximise the probability that was assigned to the correct element. In sequential learning tasks, this optimisation function is often a cross-entropy loss, which then should be minimised with a gradient descent and back-propagation.

2.2.2 Attention

The context vector, a fixed-size representation of the input, is considered a main bottleneck of sequence-to-sequence models. First, it compresses all the information of the input sequence into one vector, which may be not optimal for long input sequences, and second, particular inputs of the encoder may be relevant for the decoder at each timestep.

To remedy this issue, the attention mechanism (Bahdanau et al., 2015) was introduced (Figure 2.6). Intuitively, attention creates a mapping between an output element and one or several input elements by assigning higher weights to the most important inputs. With attention, the model learns how to focus (or *to attend*) on particular parts of the source sequence during decoding, rather than using a fixed representation for the whole input (i.e. the context vector).

An attention score is computed by taking the dot product between a decoder hidden state s_t

at the timestep t and each encoder hidden state h_i. That gives attention scores for each input element:

$$e^t = \Big[s_t h_1, s_t h_2, \ldots, s_t h_n\Big]. \tag{2.3}$$

Then the softmax function is applied to all the scores to get a probability (attention) distribution:

$$\alpha^t = softmax(e^t). \tag{2.4}$$

Finally, an attention output is a weighted sum of the encoder hidden states:

$$a_t = \sum_{i=1}^{n} \alpha_i^t h_i \quad . \tag{2.5}$$

At each decoding step, the attention output is concatenated with the decoder hidden state to compute the prediction y_i' as it was done in a non-attention sequence-to-sequence model.

There are several ways to compute attention scores. Using the dot product as in the explanation; Bahdanau et al. (2015) used a multi-layer perceptron; Luong et al. (2015) proposed several methods, including their "general" alignment bilinear function, what is usually referred to as "Luong attention".

Overall, attention permits building a soft alignment between input elements and output elements; this is similar in nature to alignment between source and target words in machine translation.

2.2.3 Copy and Coverage Mechanisms

Copy Mechanism. Copy mechanism enables to copy elements from the input to the output. This mechanism is useful for NLG tasks as it permits handling out-of-vocabulary (OOV) words (not seen during training) and ensures faithfulness to the input. The latter can be illustrated by data-to-text generation tasks, where input usually contains a lot of named entities or numbers, which should be replicated in the output text.

Pointer networks introduced by Vinyals et al. (2015) use attention to model the copying strategy. They make an extension to the decoder, which predicts whether to generate an element or copy an element from the input. This decision is taken based on the attention distribution: an element with higher weights will be more probably copied to the output.

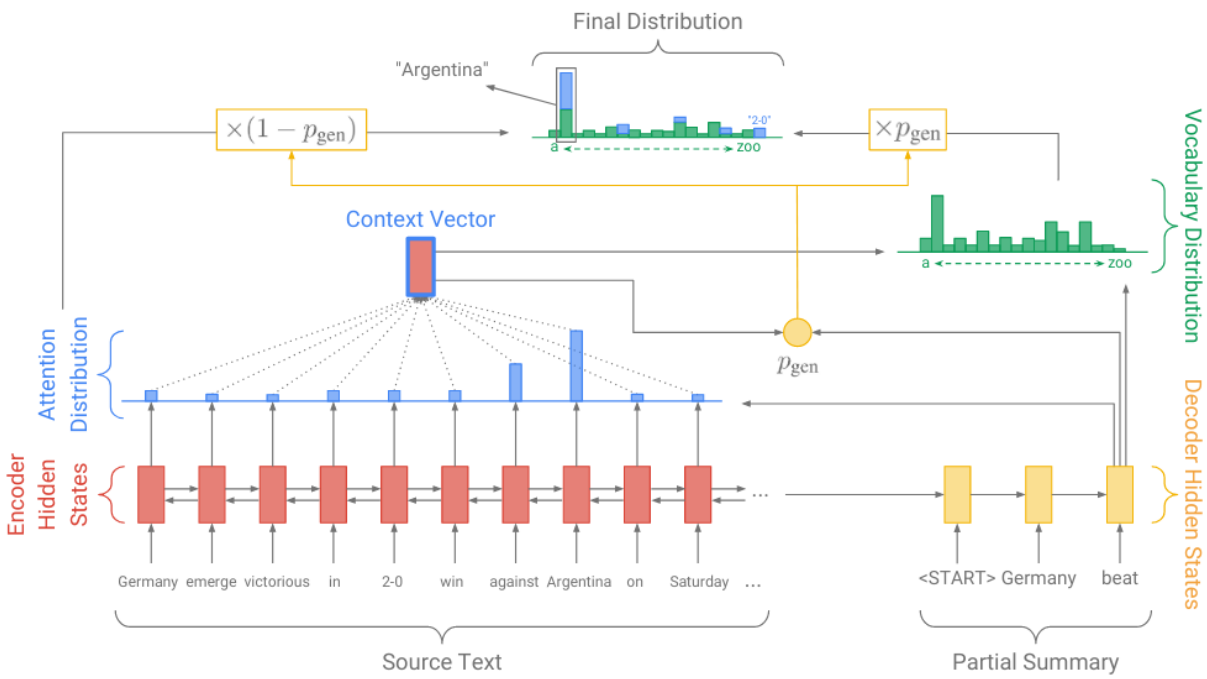
Since the introduction of pointer networks several modifications of the copy mechanism have been proposed (Gu et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016; See et al., 2017). Here we focus on the pointer-generator model of See et al. (2017) who applied it to summarisation and explain how it works.

The general schema of the model is shown in Figure 2.7. The decoder uses an extended vocabulary which consists of a predefined target vocabulary P_{vocab} which is dynamically extended at inference time with the tokens contained in the input. At each time step t during decoding, the model then decides whether to copy from the input or to generate from the target vocabulary using a probability distribution over the extended vocabulary which is computed based on a generation probability (sampling from the target vocabulary) and on the attention distribution (sampling from the input).

The more formal definition of the copy mechanism is as follows. The attention distribution α_t is calculated as in Bahdanau et al. (2015):

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + b_{attn}) \tag{2.6}$$

Figure 2.7: Pointer-generator model. Picture from See et al. (2017).



$$\alpha^t = softmax(e^t) \tag{2.7}$$

with v, W_h, W_s and b_{attn} parameters to be learned, s_t is the decoder state and h_i is a variable ranging over the encoder hidden states. P_{vocab} is a probability distribution over all elements in the target vocabulary.

$p_{gen} \in [0, 1]$ is a generation probability, i.e. the probability whether to generate or to copy. p_{gen} is then defined as:

$$p_{gen} = \sigma(W_h^T h_t + W_s^T s_t + W_x^T x_t + b_{ptr}) \tag{2.8}$$

where W_h, W_s, W_x, b_{ptr} are parameters to be learned, x_t is the decoder input and h_t is the context vector produced by the attention mechanism as the weighted sum $\sum_1^N \alpha_i^t h_i$ of the encoder states (with N the number of encoder states).

Finally, the probability distribution over the extended vocabulary, from which to predict word w, is defined as:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_i^t \quad . \tag{2.9}$$

So if word w is OOV, then $P_{vocab}(w)$ is equal to zero, hence generation of w depends on its attention score α_t^w.

Coverage Mechanism. Coverage mechanism (Tu et al., 2016) serves to keep track which elements of the input have been already attended. Initially, coverage was introduced in machine translation to indicate whether a source word was translated or not. The mechanism stimulates

to pay less attention to already translated words and to pay more attention to untranslated words. In NLG, coverage can be helpful to avoid repetitions in the output.

As defined by See et al. (2017), coverage vector c^t is the sum of attention distributions over all previous decoder timesteps:

$$c^t = \sum_{t=0}^{t-1} \alpha^t \tag{2.10}$$

Afterwards, the coverage vector is used as an additional input to the attention mechanism of Bahdanau et al. (2015):

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{attn}), \tag{2.11}$$

where w_c is a parameter to be learned.

A special coverage loss term is also added to the main loss function; it penalises focusing on elements that have already been attended.

In sum, the coverage mechanism facilitates the work of the attention mechanism. It discourages attention to focus repeatedly on the same elements, and therefore repetitions can be more likely avoided in generated output.

2.2.4 Factored Models

So far we considered sequence-to-sequence modelling using input elements only, i.e. without taking into account any other characterics they may have. To this end, Alexandrescu and Kirchhoff (2006) introduced factored representations for neural language models. Each input element, namely words in their application, is modelled, along with a usual source word embedding, with some features, or "factors". For example, for words, such factors can be part-of-speech (POS) tags, lemmas, dependency relations, etc. In factored models, a separate embedding is learned for each factor, and then all the embeddings are concatenated to form an input to the encoder (Figure 2.8). Sennrich and Haddow (2016) applied factored models to neural machine translation. In their system, based on subword representations, each source element is modelled with a subword, subword tag, POS, and dependency relation (Figure 2.9).

The formal definition of a factored representation is as follows. Let N be the number of factors f that an input element x_j has. Given a factor f, E_f is a factor embedding matrix and x_{jf} is a one-hot vector for the f-th input factor. The one-hot vector indicates which value the factor has: for instance, given POS tags as a factor, NOUN can be a possible value encoded in the one-hot vector. Then the input factored representation e_j is defined as:

$$e_j = \mathop{\Big\|}_{f=1}^{N} E_f x_{jf}, \tag{2.12}$$

where $\Big\|$ is the vector concatenation. In other words, we concatenate separate embedding vectors for each factor. This representation is passed to the encoder, and training proceeds in the usual fashion.

Beyond machine translation, factored models were used in other applications, such as automatic post-editing (Hokamp, 2017) and surface realisation (Elder and Hokamp, 2018).

2.2.5 Transformers, Pre-trained Language Models

This section discusses some recent developments in NLP: transformers and pre-training.

Figure 2.8: Factored model. Picture from Hokamp (2017).

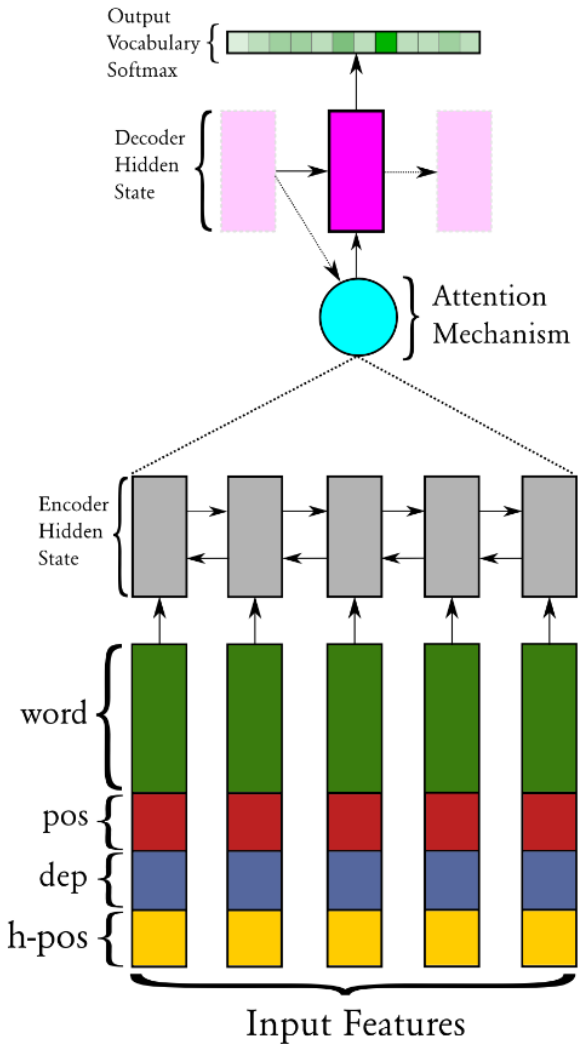
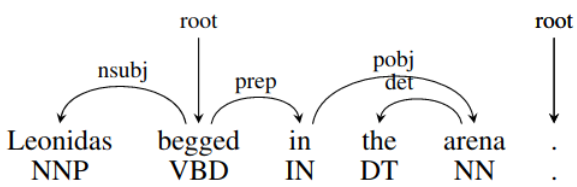


Figure 2.9: Source sentence dependency tree and its factored representation after subword segmentation. B: beginning, I: inside, E: end of a word. O is used if a symbol corresponds to the full word. Picture from Sennrich and Haddow (2016).



words	Le:	oni:	das	beg:	ged	in	the	arena	.
lemmas	Leonidas	Leonidas	Leonidas	beg	beg	in	the	arena	.
subword tags	B	I	E	B	E	O	O	O	O
POS	NNP	NNP	NNP	VBD	VBD	IN	DT	NN	.
dep	nsubj	nsubj	nsubj	root	root	prep	det	pobj	root

In Section 2.2.1, the underlying structure of the encoder-decoder model was RNNs. Vaswani et al. (2017) proposed an alternative architecture solely based on attention: a transformer model. In transformers, encoding is no longer sequential, relationship between elements is modelled using *self-attention*, an attention that is calculated from a sequence element over the sequence itself. In self-attention, all the elements of the encoder "look" at each other, and their representations get updated taking context into account. During decoding, the same self-attention mechanism is applied; additionally, information about encoder states is used at each decoder step. To account for element positions, the transformer model uses positional embeddings along with standard input embeddings. Transformer has become a standard model for sequence-to-sequence tasks, replacing an RNN-based encoder-decoder model.

The latest developments in NLG follow the main trends in NLP in general: contextual language modelling, huge pre-trained models are starting to take the field in generation as well. Large pre-trained language models, trained on open-domain corpora, are used as generators which have an "innate grasp" of a language. They know which sequences are fluent and more probable to appear in texts; in other words, they provide knowledge how to construct grammatical and fluent sentences. The standard practice to use those pre-trained models is to fine-tune them on a given task. Chen et al. (2020b) show how GPT-2 (Radford et al., 2019) can be used for a few-shot data-to-text NLG on the WikiBio dataset. To handle the vocabulary limitation, they trained byte-pair encoding (BPE) and subword vocabulary. Their model, using only several hundreds of training instances, learns how to select and copy information from tables via the field-gating encoder of Liu et al. (2018) (content selection) and uses the pre-trained language model to produce final sentences (realisation). Mager et al. (2020) also show how to fine-tune a large pre-trained language model (GPT-2), this time, for AMR-to-text generation. In their experiments, the generative model is fine-tuned on the joint distribution of text and a sequential AMR representation. Additionally, a simple rescoring technique is introduced: the best output is chosen from top-n generated texts through AMR parsing and comparing them to the gold AMR graph.

The trend of using pre-trained models can be also confirmed by looking at the results of the WebNLG+ 2020 Challenge (Castro Ferreira et al., 2020) where most of the teams used pre-training in some parts of their models. To give a flavour of the tendency, out of 15 participants, two teams developed template-based approaches, one team used a Transformer-based encoder-decoder, and 12 teams included different pre-trained models in their NLG systems: GPT-2, T5 (Raffel et al., 2020), BART (Lewis et al., 2020), and mBART (Liu et al., 2020).

2.2.6 Encoder-Decoder Models for NLG

In Sections 2.2.1–2.2.4 we described basic concepts of some neural techniques. Numerous approaches in generation have been built on these techniques. They are mainly focused on possible improvements that can be made to the encoder or decoder components, and they aim at answering such questions as *how to encode the input better?*, *how to keep track what has been verbalised during decoding?*, etc. In this section, we briefly review some works on generation from structured data, which use the neural encoder-decoder framework.

One of the first structure-aware encoder-decoder models was proposed by Wen et al. (2015). They introduce a special mechanism into an LSTM cell to track what parts of a dialogue act have been verbalised so far. Their proposal is similar in nature to the coverage mechanism discussed in Section 2.2.3. Liu et al. (2018) propose to add a field gate to LSTM units in their decoder to account for field-value records for table-to-text generation. Their field-gated LSTM performs content selection, and surface realisation is done by the generator that uses dual attention for

the field and word level. Vougiouklis et al. (2018) introduce a triple encoder with attention over RDF triples, which allows to see to which part of a triple the attention should be given at each decoding step. In a similar vein, graph-based GTR-LSTM triple encoder of Trisedya et al. (2018) considers both intra-triple and inter-triple relationships. Puduppully et al. (2019) create entity-specific representations to keep track of entities during decoding. Wiseman et al. (2018) use a hidden semi-markov model in their decoder to learn template-like structures jointly with learning to generating text.

Another type of structure-aware models are graph-to-sequence architectures where an encoder takes a form of a graph. These models can take advantage of the inherent graph structure of the data (AMRs are graphs, dependency relations form a tree) and encode them accordingly. Graph-based approaches were also proven to be a powerful tool to encode NLG data (Marcheggiani and Perez-Beltrachini, 2018; Song et al., 2018b; Beck et al., 2018; Ribeiro et al., 2019). Dual encoding of NLG data has also been proposed (Zhao et al., 2020). In their model, the input is encoded using both graph- and sequence-based encoders.

The aforementioned studies are an example of end-to-end learning where a single model is used to do the generation task bypassing the intermediate components of the traditional NLG pipeline. While end-to-end approaches are attractive by their simplicity, some neural-based approaches model each step of the NLG pipeline separately (Dušek and Jurčíček, 2016; Moryossef et al., 2019; Castro Ferreira et al., 2019).

In this section, we discussed approaches based on supervised learning with a classical prediction task (Section 2.2.1). However, other machine learning paradigms have been applied to NLG, such as reinforcement learning (Dethlefs and Cuayáhuitl, 2015; Zhang and Lapata, 2017; Perez-Beltrachini and Lapata, 2018; Narayan et al., 2018) and imitation learning (Lampouras and Vlachos, 2016); the use of Natural Language Understanding modules for NLG (Qader et al., 2019; Nie et al., 2019) or the use of autoencoders for sequence-to-sequence models (Chisholm et al., 2017), where two models are jointly trained: a forward-only model that generates text from facts and a backward-only model that predicts facts from text. We do not consider those approaches here, since they lie outside the scope of this thesis.

2.3 Future Directions

Over recent years great advancements have been made in the field of NLG, mainly due to the development of powerful machine learning methods. Despite the witnessed progress, current approaches exhibit some limitations, and a lot of areas of NLG still remain under-explored. In this section, we outline some restrictions of current practices and indicate ways for future scientific endeavours in NLG.

Currently, the majority of NLG models are developed for English, which has relatively poor morphology and mostly rigid word order. Methods developed for English may not work for other languages (Dušek and Jurčíček, 2019). This state of affairs calls for developing data in languages other than English and for building multilingual models. Furthermore, models, and hence scientific hypotheses, are usually tested on one or, less often, several NLG benchmarks, so it is not clear if the results and conclusions based on such models will generalise to other benchmarks. This, in turn, calls for testing models not only cross-lingually but on several types of data. Overall, of great interest for researchers could be generalisation and adaptability in the broadest sense: across tasks, datasets, domains, and language phenomena.

On the other hand, since dataset production is a costly process, another interesting direction of research could be training with few or zero examples (few- or zero-shot generation) and

exploring transfer learning approaches.

One of the main goals of NLG applications is to state facts encoded in data, as a result of that, affirmative clauses are more frequent to see in data verbalisations. However, stating facts can also be conveyed by other means, such as using negation or comparisons (Chen et al., 2020a). A possible direction for future research is therefore generating diverse outputs in terms of lexicon and syntax, and more generally, writing style and genre (Ficler and Goldberg, 2017; Deriu and Cieliebak, 2018b).

Finally, evaluation remains one of the main concerns in NLG methodology (Gkatzia and Mahamood, 2015; van der Lee et al., 2019; Howcroft et al., 2020). Given the variety of inputs to NLG systems and an infinite number of possible outputs, it is hard to develop a common evaluation schema across NLG tasks and to measure output correctness. This appeals to extensively discuss evaluation strategies for NLG tasks.

In this thesis, we seek to address some of the limitations by looking at how to create non-English datasets (Chapter 3), how to treat rare items in NLG (Chapter 4), how to build multilingual models (Chapter 5), and how to create task-specific evaluation (Chapter 6).

3

Creating Training Corpora for Natural Language Generation

Contents

3.1 Introduction **21**
3.2 Related Work **22**
3.2.1 Corpus Construction for Natural Language Generation 22
3.2.2 Automatic Post-Editing 24
3.3 WebNLG Data-to-Text Dataset **24**
3.4 Creating Russian Version of WebNLG Dataset **25**
3.4.1 Neural Machine Translation 25
3.4.2 Manual Post-Editing and Error Analysis 25
3.5 Automatic Post-Editing **27**
3.5.1 Rule-Based Post-Editing 27
3.5.2 Automatic Post-Editing Model 28
3.6 Evaluation of Rule-Based Post-Editing **29**
3.7 Conclusion **29**

3.1 Introduction

The majority of methods in NLP relies on supervised machine learning, which, in turn, requires a set of training examples. Neural NLG is not an exception. Developing datasets for NLG tasks has become crucial for working on those tasks. Nowadays having data to learn models, which may eventually help to resolve research questions, is a major bottleneck. Furthermore, research in NLP, and in NLG in particular, is mostly English-centric, where models and corpora are developed for the English language at the first place. This is driven by several factors: an ability to compare to other datasets/methods, English is the language that is understood by most readers, a global need for English as a language of international communication, etc. Given the current need for datasets in general and the need for non-English datasets in particular, in this chapter, we focus on developing a data-to-text dataset for Russian.

Data-to-text generation is a key task in Natural Language Generation which focuses on transforming data into text and permits verbalising the data contained in data- or knowledge

bases. However, creating the training data necessary to learn a data-to-text generation model is a major bottleneck as *(i)* naturally occurring parallel data-to-text data does not commonly exist and *(ii)* manually creating such data is highly complex. Moreover, the few parallel corpora that exist for data-to-text generation have been developed mainly for English. Methods that support the automatic creation of multilingual data-to-text corpora from these existing datasets would therefore be highly valuable.

In this chapter, we introduce a semi-automatic method for deriving a parallel data-to-text corpus for Russian from the data-to-text WebNLG corpus whose texts are in English. Our method includes three main steps. First, we use neural machine translation (NMT) model to translate WebNLG English texts into Russian (Section 3.4.1). Second, we perform a detailed error analysis on the output of the NMT model (Section 3.4.2). Third, we exploit two techniques for automatically post-editing the automatic translations (Section 3.5). As 53% of the translation errors bear on named entities, we focus on these in the present chapter and leave other error types for further research.

This chapter is based on the following publication:

Shimorina, A., Khasanova, E., and Gardent, C. (2019). Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 44–49, Florence, Italy. Association for Computational Linguistics.

I would like to gratefully acknowledge the participation of Elena Khasanova in this project without whom Russian WebNLG would have never come into the world. Elena was responsible for developing error classification, and she was involved in data annotation. The new corpus, error classification and scripts for reproducing our experiments are available at `https://gitlab.com/shimorina/bsnlp-2019`.

3.2 Related Work

3.2.1 Corpus Construction for Natural Language Generation

Various ways have been proposed to create NLG datasets. Most of them rely on automatic crawling or crowdsourcing with a few exceptions when professionally written texts are used. All techniques have their advantages and disadvantages: crawled texts allow building large-scale open-domain datasets but they are often noisy; crowdsourcing, in turn, enables a more precise mapping between meaning representations and texts but texts often suffer from being collected in an artificial setting; finally, professionally written texts are the closest to real-life applications, however they are usually limited in size.

Some examples of the datasets where texts were produced by experts are SumTime (Sripada et al., 2002) where professional weather forecasters made weather reports and RotoWire (Wiseman et al., 2017) where professionals wrote basketball game summaries targeted at fantasy basketball fans. While texts in those datasets are long and natural, they are domain-specific, and dataset size is rather small (not exceeding 5K training instances).

Automatic extraction from the web allows to compile large-scale datasets. Lebret et al. (2016) used the first sentence of a Wikipedia article and a corresponding infobox to construct WikiBio, the data-to-text dataset which deals with biographies. Similarly, Qader et al. (2018) aligned Wikipedia abstracts and articles in the Company domain with infoboxes represented as attribute-value pairs. The main challenge with the crawled datasets such as WikiBio is that mapping between data instances and text can be noisy, and the reference text does not always contain all the information present in the input (Perez-Beltrachini and Gardent, 2017;

Qader et al., 2018) or the reference contains more information than declared in the input. Other addressed this problem by applying filtering: Wang et al. (2018) aligned Wikipedia articles and tables and applied some automatic filtering, collecting data in the Person and Animal domains. Another way to create data-to-text datasets with less loose alignment is to start from text and then derive a corresponding meaning representation. In T-REx (Elsahar et al., 2018), Wikipedia abstracts were undergone several processing steps: entity and predicate extraction and linking to knowledge base URI, date and time extraction, coreference resolution, and triple alignment. These steps allowed to finally align the abstract with RDF triples from DBpedia. In a similar manner, YelpNLG (Oraby et al., 2019) includes restaurant review descriptions mined from the Yelp review website. The descriptions, which were annotated with NLP tools, are linked to automatically extracted attribute-value pairs and sentiment information.

Crowdsourced datasets are usually smaller in size (around 20–100K instances) and are often domain-specific. WebNLG (Gardent et al., 2017a) maps RDF triples to short texts providing entity descriptions; ViGGO (Juraska et al., 2019) consists of dialogue acts in the domain of video games; E2E (Novikova et al., 2017b) maps meaning representations to text in the restaurant domain. The common way to produce such datasets is to show some structured data to workers and ask to produce an utterance describing such data. Numerous ways are used in crowdsourcing to engage people in this process in order to have more natural data. Novikova et al. (2017b) proposed to use pictorial stimuli, and Balakrishnan et al. (2019) used a conversational setup to collect weather descriptions.

There are also examples where crawling and crowdsourcing come together. For creating ToTTo, Parikh et al. (2020) automatically extracted tables with descriptions from Wikipedia and later asked people both to highlight cells with necessary information matching the description and to edit the sentence.

Additionally, several datasets have been proposed recently to include reasoning and inference over data. Crowd-based CommonGen (Lin et al., 2020), targeted at commonsense reasoning, provides descriptions of concept sets, where people were asked to reason about a possible combination of concepts. In crowdsourced LogicNLG (Chen et al., 2020a), crowdworkers were asked to make logical inferences from tables.

Despite an abundance of the proposed techniques for collecting datasets and methods used, all the above-mentioned datasets are focused on English. Most non-English corpora for generation got translated or adapted using existing practices and datasets for English. Nema et al. (2018) introduced French and German versions of WikiBio following the original procedure of the corpus creation. Some portion of RotoWire was translated into German by professional translators (Hayashi et al., 2019). When creating Czech restaurant data, Dušek and Jurčíček (2019) also made use of professionals to translate Wen et al. (2015)'s restaurant corpus. Castro Ferreira et al. (2018b) used a machine translation system to create a silver-standard version of WebNLG for German.

Multilingual surface realisation data (Mille et al., 2018a, 2019) was derived from Universal Dependency treebanks (Nivre et al., 2017) by using trees as input. Based on the original trees, an additional deep syntax input was also created by converting dependency relations into argument-predicate relations and by removing function words (Mille et al., 2018b). The Abstract Meaning Representation (AMR) formalism (Banarescu et al., 2013), initially developed for English corpora, was also applied to other languages: Xue et al. (2014) experimented with translated Chinese and Czech AMRs, Sobrevilla Cabezudo and Pardo (2019) translated and adapted AMR annotations for Brazilian Portuguese. In addition, development of AMR parsers permitted creating multilingual AMR-to-text datasets (Vanderwende et al., 2015; Damonte and Cohen, 2018).

3.2.2 Automatic Post-Editing

The work presented in this chapter is also related to the field of automatic post-editing (APE) of machine translation outputs. The task of APE consists in automatically correcting "black-box" MT output by learning from human corrections. Several WMT APE shared tasks were held focusing on English-German, German-English, English-Russian, and English-Spanish language pairs[1].

Recent neural approaches to APE include, *inter alia*, multi-source training with original sentences and MT outputs (Junczys-Dowmunt and Grundkiewicz, 2018), encoding corrections by a sequence of post-edit operations (Libovický et al., 2016), as well as standard encoder-decoder architectures (Pal et al., 2016). Submissions which participate in the APE shared tasks extensively use large synthetic corpora (Negri et al., 2018). Despite that fact, a *do-nothing* baseline when MT outputs are kept unchanged is hard to beat according to the results of the APE shared task in 2018 for English-German (Chatterjee et al., 2018) and in 2019 for English-Russian (Chatterjee et al., 2019).

3.3 WebNLG Data-to-Text Dataset

RDF triples	<Asterix, creator, René Goscinny> <René Goscinny, nationality, French people>
Original	Rene Goscinny is a French national and also the creator of the comics character Asterix.
MT	Рене Госкино - французский гражданин, а также создатель комического персонажа Астерикс. Rene Goskino French national and also creator comic$_{gen}$ character$_{gen}$ Asterix$_{inan}$
PE	Рене Госинни - французский гражданин, а также создатель персонажа комиксов Астерикса. Rene Goscinny French national and also creator character$_{gen}$ comics$_{gen}$ Asterix$_{anim}$
Errors	named entity, vocabulary, grammar
Links	<René Goscinny, sameAs, Рене Госинни> <French people, sameAs, Французы>

Table 3.1: WebNLG original instance in the ComicsCharacter category, its Russian translation (MT), and post-edited translation (PE) along with error annotation. Errors are highlighted in blue. Links are RDF triples of the form <*English entity, sameAs, Russian entity*>. However, such links are not available for all entities in DBpedia. *gen*—genitive; *anim*—animate; *inan*—inanimate.

Presented in Section 2.1.2, the WebNLG data-to-text corpus (Gardent et al., 2017a)[2] aligns knowledge graphs with textual descriptions verbalising the content of those graphs. The knowledge graphs are extracted from DBpedia (Lehmann et al., 2015) and consist of sets of (one to seven) RDF triples of the form <*subject, predicate, object*>. Textual descriptions are in English, and due to the nature of the knowledge graphs, they have an abundance of named entities. The first two lines of Table 3.1 show an example of a WebNLG instance.

WebNLG provides textual descriptions for entities in fifteen DBpedia categories (Airport, Artist, Astronaut, Athlete, Building, CelestialBody, City, ComicsCharacter, Food, MeanOfTransportation, Monument, Politician, SportsTeam, University, WrittenWork). The corpus possesses a hierarchical structure: if a set consisting of more than one triple is verbalised, then

[1] WMT: workshop/conference on machine translation. See `http://www.statmt.org/wmt19/ape-task.html` for the shared task in 2019.

[2] We used the version 2.0: `https://gitlab.com/shimorina/webnlg-dataset`.

verbalisations of every single triple are to be found in the corpus. Given the example in Table 3.1, the pairs {<*Asterix, creator, René Goscinny*>: *René Goscinny created Asterix*} and {<*René Goscinny, nationality, French people*>: *René Goscinny is French*} are also present in the WebNLG data. That structure allows propagating post-edits made in texts describing one triple to those verbalising triple sets of larger sizes.

For the ease of the post-editing translation effort, we enriched the corpus with triples representing links between entities in English and Russian by virtue of the predicate *sameAs* present in the DBpedia knowledge graph between different languages. Since that predicate does not exist for all entities, the proportion of unique subjects and objects of RDF triples that got linked was 44.42%.

3.4 Creating Russian Version of WebNLG Dataset

3.4.1 Neural Machine Translation

Following Castro Ferreira et al. (2018b), who created a silver-standard German version of WebNLG, we translated the WebNLG English texts into Russian using the English-Russian NMT system developed by the University of Edinburgh for the WMT17 translation shared task (Sennrich et al., 2017)[3]. This system ranks first for the English-Russian News translation task both in automatic metrics[4] and human assessment (Bojar et al., 2017). It is learned using Nematus, an encoder-decoder with attention, based on subword units (byte pair encoding). Since the Edinburgh model was trained on sentence-to-sentence data, we split WebNLG texts into sentences using the WebSplit sentence annotation (Narayan et al., 2017), input each sentence to the NMT system, and then concatenated translations to reconstruct the target texts.

3.4.2 Manual Post-Editing and Error Analysis

To determine the most common translation errors, we start by manually annotating error types in sentences verbalising one triple.

Error Classification. The manual post-editing was done by two experts, native Russian speakers, on a part of the corpus for the categories Astronaut, ComicsCharacter, Monument, University for texts verbalising one triple only. Out of 1,076 machine translation outputs analysed, 856 texts (80%) were post-edited. The experts also classified errors that they identified in a translated text.

To define an error classification, we drew inspiration from various error typologies that were developed in the MT community and applied to different languages. See, for instance, Popović (2018) who provides an overview of different approaches to error classification. We also got some ideas from studies focused on errors made by language learners and non-experienced translators in the Russian-English and English-Russian translation directions (Kunilovskaya, 2013; Rakhilina et al., 2016; Komalova, 2017). That allowed us to extend the classification with some phenomena typical for Russian. Lastly, the classification was augmented with the notorious errors of the NMT systems: word repetitions, deletions, insertions (partly due to the subword-based nature of the applied NMT), untranslated common words, etc. Main error classes identified for the final classification are shown in Table 3.2. Named entities were treated as a

[3] http://data.statmt.org/wmt17_systems/. Specifically, we use their ensemble model consisting of four left-to-right models.

[4] http://matrix.statmt.org/matrix/systems_list/1875

Category	Subcategory
Grammar	Case marking
	Copula
	Verbal aspect
	Preposition
	Possessive
	Part-of-speech
	Agreement
	Voice, intentionality
Vocabulary	Ambiguity
	Collocation
	Incorrect translation
Structure	Word Order
	Deletion
	Insertion
Named entity	
Punctuation	

Table 3.2: Main categories and subcategories of error classification.

separate category to highlight problems while applying the NMT system on WebNLG. If a text contained more than one mistake in a particular category, then each mistake was tagged as an error. If a spotted mistake concerned an NE, annotators were allowed to add other categories to specify the error.

Category	Proportion	Agreement
Grammar	17%	0.44
Vocabulary	14%	0.52
Structure	11%	0.32
Named entity	53%	0.67
Punctuation	4%	0.0

Table 3.3: Proportion of main error types in the manually post-edited data and Cohen's κ scores on the held-out category Athlete.

Error Analysis. Table 3.3 shows the error type distribution in the post-edited texts. In total, 1,722 errors were found. Named entities is the largest source of errors with 53% (917) of all corrections. Grammatical and lexical mistakes constitute 17% and 14% of the identified errors respectively, while "Structure" (11%) ranks fourth. In fact, the majority of structural mistakes were spotted in named entities. For example, *the Baku Turkish Martyrs' Memorial* was translated as «Мемориал» «Мемориал» в Баку ('Memorial Memorial in Baku') with the following errors identified: named entity, deletion, deletion, insertion.

The most common errors found in NE translations are:

- copying verbatim English entities into Russian translations (person names, locations);

- wrong transliteration, whereas a standard transliteration exists in Russian. E.g., *Lancashire* translated as Ланкассир ('Lancassir') instead of Ланкашир;
- misinterpretation of a named entity as a common noun. E.g., *Dane Whitman* translated as датчанин Уитмен ('inhabitant of Denmark Whitman') instead of Дейн Уитмен.

It should be noted that since the Edinburgh NMT system used subword units, there were also errors with copying named entities, e.g., *Visvesvaraya Technological University* became *Visvesvaraya Technical University*. Similarly, in the example from Table 3.1, the surname *Goscinny* was misinterpreted as the acronym *Goskino* meaning 'State Committee for Cinematography'.

Inter-annotator Agreement. Erroneous words in translations can be attributed to several possible error types. To evaluate consistency between annotators and the appropriateness of the developed error classification, we calculated inter-annotator agreement (Cohen, 1960) on the 86 texts from the DBpedia category Athlete, to which annotators were not exposed before. Table 3.3 shows the kappa scores. The highest score (0.67) was reached for "Named entity", which corresponds to the substantial agreement. The main source of disagreement for named entities was a decision to perform transliteration or not, e.g., sport club names as *Tennessee Titans* can be kept 'as is' in Russian text or can be put into Cyrillic. For other categories, agreements range from moderate to fair; as for "Punctuation", the agreement is zero due to the data sparseness in this category (there were two errors only identified by one annotator). In most cases, moderate and fair agreement scores were related to the fact that annotators identified the same error but attributed different tags for it (e.g., grammar and vocabulary were quite often used interchangeably).

Overall, results show *(i)* consistency in correcting named entities, as well as *(ii)* the importance to perform more annotator training and/or establish clearer guidelines, especially for the "Structure" category.

3.5 Automatic Post-Editing

Above we described how we got Russian WebNLG by translation and what the main sources of errors are. In this section, we explore how MT outputs can be post-edited automatically, with named entities being a central point of our discussion. To improve the automatic translations, we experiment with two methods: a rule-based method based on the errors found during manual annotation (Section 3.5.1) and a neural approach (Section 3.5.2).

3.5.1 Rule-Based Post-Editing

We witnessed in the previous section that the correction of named entities was dominant in the post-editing process. So, in this section, we will focus on how to propagate changes made in texts describing one triple to texts verbalising more than one triple with the attention to the named entities only.

Based on the manual corrections applied to the 1-triple data (WebNLG instances where the input graph consists of a single triple), we extract post-edit rules by building upon the operations used to compute the edit distance (Levenshtein, 1966). For example, given the neural translation (2a) and the manually edited correction (2b), the sequence of edit operations applied to compute the Levenshtein edit distance is (2c), i.e. replace 'Альба' by 'Алба-Юлия', delete 'Юлия', keep '–', keep 'город', keep 'в', keep 'Румынии'.

(2) a. 'Альба Юлия – город в Румынии'
b. 'Алба-Юлия – город в Румынии'
c. SUB DEL KEEP KEEP KEEP KEEP
d. 'Alba Julia is a city in Romania'

Based on these edit sequences, we extracted sequences of substitution, deletion, and insertion rules along with the corresponding tokens (e.g., Альба Юлия → Алба-Юлия). We then checked these rules manually and excluded false positives: out of 856 initially extracted rules, we kept 568 rules. Lastly, we applied the validated rules to the automatic translations.

That method enabled us to increase the amount of post-edited data: after that procedure the total number of post-edited translations sums up to 4,188 (see Table 3.4).

	1 triple	2-7 triples	All triples
PE	856	3,332	4,188
Total	1,076	4,109	5,185

Table 3.4: Corpus statistics: number of post-edited (PE) texts. Total corresponds to both PE and non-PE texts.

3.5.2 Automatic Post-Editing Model

To see to which extent corrections can be learned automatically, we built a corpus of (MT, RPE) pairs where MT is an automatic translation and RPE is its correction using the rule-based system described in the preceding section and trained an APE model on it.

The baseline system is a *do-nothing* baseline where MT outputs are left unmodified. In our case, that baseline gives 82.4 BLEU between MT and RPE on the test set, which sets quite high standards for learning a new APE model.

The train/dev/test partition was 80/10/10. We used the OpenNMT-tf framework (Klein et al., 2017)[5] to train a bidirectional encoder-decoder model with attention (Luong et al., 2015). A single-layer LSTM (Hochreiter and Schmidhuber, 1997) is used for both encoder and decoder. We trained using full vocabulary and the maximal length in the source and target; all the hyperparameters were tuned on the development set. The APE model was trained with a mini-batch size of 32, a word embedding size of 512, and a hidden unit size of 512. It was optimised with Adam with a starting learning rate of 0.0005. We used early stopping based on BLEU on the development set, as a result of that, the model was trained for 23 epochs. Decoding was done using beam search with a beam size of 5. As an evaluation metric, we used BLEU-4 (Papineni et al., 2002) calculated between our model predictions and RPE. BLEU and statistical significance were calculated on tokenised texts using COMPARE-MT tool (Neubig et al., 2019), which, in turn, uses the NLTK implementation of BLEU. Results are shown in Table 3.5.

The APE model performance reached parity with the baseline on dev and test data. The difference between scores was not statistically significant via the bootstrap resampling (1000 samples, $p < 0.05$). On the training data, the model yielded 94 BLEU, which indicates a possible overfitting. We conjecture that this is due to a small amount of training data.

Our results are in line with the findings of the WMT'18 APE shared task that correcting NMT-based translations is a challenging task: gains were only up to 0.8 BLEU points in the NMT track (Chatterjee et al., 2018) for English-German. Similarly, for the English-Russian pair,

[5]version 1.22.0, https://github.com/OpenNMT/OpenNMT-tf

System	Train	Dev	Test
Baseline	81.11	81.25	82.85
Our APE model	94.45	83.00	83.65

Table 3.5: BLEU-4 scores.

the WMT'19 APE challenge reported that none of the systems was able to beat the baseline (Chatterjee et al., 2019).

3.6 Evaluation of Rule-Based Post-Editing

Evaluation was carried out only on the rule-based method output because it is more robust than the neural approach, and because the APE model did not yield better results.

We analysed a sample of total 66 lexicalisations in 4 categories: Astronaut, University, Monument (2-7 triples) and ComicsCharacter (2-5 triples). Around two thirds of analysed named entities were replaced correctly. Below we analyse common sources of errors for the erroneous NEs.

The most frequent case is unrecognised named entities. In 62% of the cases the replacement was not performed, which includes 28% of kept Latin transcriptions, 27% of kept Cyrillic translations, and 7% of acronyms. For the majority of these NEs, the original translations include unaccounted elements (not covered by the extracted rules) such as missing or wrongly inserted prepositions or punctuation marks.

Another common error is lack of grammatical adaptation of the NE. Wrong case marking occurred in 23% of all NEs (see example 3), and gender and number agreement make about 6.5%. The less frequent but important error categories are spelling errors, such as missing capitalisation, insertions of quotation marks, and gender or number agreement with anaphors, especially in texts verbalising 5-7 triples.

(3) En: 'The dean of Accademia di Architettura'
MT: 'Декан Accademia di Projecttura'
RPE: 'Декан Академия$_{nomn}$ архитектуры'
Correct: 'Декан Академии$_{gen}$ архитектуры'

To conclude, many errors are caused by irregularities in the translations (which, in turn, are often caused by misspelled input) and can be eliminated by introducing more variation to the replacement algorithm. Grammatical adaptation of NEs, however, requires more careful further investigation.

3.7 Conclusion

In this chapter, we proposed an approach for semi-automatically creating a data-to-text corpus for Russian that can be used to learn a data-to-text natural language generation model. An error analysis of the output of an English-to-Russian neural machine translation system showed that 80% of the automatically translated sentences contained an error and that 53% of all translation errors bear on named entities. We therefore focused on named entities and tested two post-editing techniques for correcting wrongly translated NEs. Specifically, we provided a rule-based method which permits correcting the errors and trained a neural post-editing model.

One of the possible directions for further research is to extend the approach to other error types and to investigate whether the neural model can be improved to help generalise post-editing to errors not captured by the rule-based method. It is also interesting to see if error annotation can help learning an automatic post-editing model—error types can be integrated to neural models in a multi-source or multi-task fashion.

Another possible direction for future research will be to identify named entities before the translation phase, perform translation on the texts stripped of named entities (see the WebNLG delexicalised version of Castro Ferreira et al. (2018b)), and then insert named entities, which were translated and verified separately. Of course, declension handling will be an additional issue in that approach. Using cross-lingual embeddings may also be useful for translation of named entities; even if they mostly capture frequent proper nouns, rather than rare, one may think of a symbolic ingestion of proper noun mapping between languages, for example, extracted from a knowledge base. Overall, handling personal names in machine translation for Slavic languages should be studied more systematically, as also found in the English-Serbian case by Lohar et al. (2019).

4

Handling Rare Items in Natural Language Generation

Contents

4.1 Introduction **31**
4.2 Related Work **32**
4.3 Experiments **33**
4.3.1 Datasets 33
4.3.2 Model Parameters 38
4.3.3 Evaluation 38
4.4 Results and Discussion **40**
4.5 Conclusion **43**

4.1 Introduction

In chapter 3 we have seen that named entities, or rare items, are problematic for NMT and are the main source of errors in translation of data-to-text corpora. This chapter is also devoted to named entities; this time we will see how to treat named entities but now from the modelling perspective in natural language generation models.

NLG corpora often describe some factual information (cf. Section 3.2.1), thus the input to NLG systems often contains rare items, i.e. low frequency items such as names, locations and dates. This makes it difficult for neural models to predict their verbalisation. To address these issues, neural approaches to generation in English typically resort either to delexicalisation (Wen et al., 2015; Dušek and Jurčíček, 2015; Trisedya et al., 2018; Chen et al., 2018) or to a copy mechanism (Chen, 2018; Elder et al., 2018; Gehrmann et al., 2018). Character-based encodings (Agarwal and Dymetman, 2017; Deriu and Cieliebak, 2018a; Jagfeld et al., 2018) and byte pair encodings have also been used (Elder, 2017; Zhang et al., 2018).

When using delexicalisation, the data is preprocessed to replace rare items with placeholders, and the generated text is post-processed to replace these placeholders with appropriate values based on a mapping between placeholders and initial values built during preprocessing. While this method is often used, it has several drawbacks. It requires additional pre- and post-processing steps. These processing steps depend on the target NLG application. The matching

procedure needed to correctly match a rare input item (e.g., Barack Obama) with the corresponding part in the output text (e.g., the former President of the United States) may be quite complex which may result in incorrect or incomplete delexicalisations. In contrast, the copy mechanisms standardly used in neural approaches to summarisation (See et al., 2017; Gu et al., 2016; Cheng and Lapata, 2016), paraphrasing (Cao et al., 2017), answer generation (He et al., 2017) and generation (Gehrmann et al., 2018; Chen et al., 2018) is a generic technique which is easy to integrate in the encoder-decoder framework and can be used independently of the particular domain and application.

In this chapter, we investigate the impact of copying and delexicalisation on the quality of generated texts using two sequence-to-sequence models with attention: one using the copy and coverage mechanism of See et al. (2017), the other using delexicalisation. We evaluate their respective output on two datasets, namely the E2E (Novikova et al., 2017b) and the WebNLG (Gardent et al., 2017a) datasets. We also compare the two methods in two different settings: the original train/dev/test partition produced by the E2E and by the WebNLG challenge *vs.* a more constrained train/dev/test split which aims to further minimise the amount of redundancy between train, dev and test data. This latter experimental setting is inspired by a recent paper by Aharoni and Goldberg (2018), which shows that the train/dev/test split may have a strong impact on how much the model learns to generalise and how much it memorises.

This chapter is based on the following publication:

Shimorina, A. and Gardent, C. (2018). Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands. Association for Computational Linguistics

All the data, scripts and evaluation results reported in this study can be found at `https://gitlab.com/shimorina/inlg-2018`.

4.2 Related Work

Delexicalisation remains one of the most popular techniques for handling rare named entities. We analysed the submissions participating in the E2E and WebNLG challenges, which used a neural approach. Among them, six teams applied delexicalisation (Chen et al., 2018; Davoodi et al., 2018; Juraska et al., 2018; Puzikov and Gurevych, 2018b; Trisedya et al., 2018; van der Lee et al., 2017), three resorted to the copy mechanism (Chen, 2018; Elder et al., 2018; Gehrmann et al., 2018), two developed character-based systems (Agarwal and Dymetman, 2017; Deriu and Cieliebak, 2018a), and another two made use of byte pair encodings (Elder, 2017; Zhang et al., 2018).

A copy mechanism allows to detect a word in an input sequence and to copy it to an output sequence (see Section 2.2.3). The copy mechanism is widely used in text production approaches where it is relevant for handling rare input but also, for instance in text summarisation, for copying input into the output. See et al. (2017), Gu et al. (2016), Cheng and Lapata (2016) introduced pointer networks (Vinyals et al., 2015) extended with a copy mechanism for text summarisation. Similarly, Cao et al. (2017) use a copy mechanism to generate paraphrases, and He et al. (2017) to generate answers. The copy mechanism is often paired with coverage, which aims at overcoming a common problem of repeated or omitted content in neural network outputs. It was used for instance in NMT (Tu et al., 2016) and summarisation (See et al., 2017).

Finally, some approaches apply neither copying nor delexicalisation. In particular, Nayak et al. (2017), working in the restaurant domain for dialogue systems, investigated ways of including slot values directly into the input representation of sequence-to-sequence models.

The most related research to ours in terms of methodology is Jagfeld et al. (2018)[6]. They compared word-based and character-based models on WebNLG and E2E datasets and concluded that both methods yielded comparable results, with character-based methods producing more diverse outputs. Other NLG-centered studies compared rule-based, statistical and neural models (van der Lee et al., 2018) and end-to-end vs. pipeline architectures (Castro Ferreira et al., 2019).

Our study suggests the following.

- Rare items strongly impact the performance of generation.
- Combining delexicalisation and copying yields the strongest improvements.
- Copying underperforms for items not, or rarely, seen in the training data.
- The content (e.g., distribution and number of named entities) and the partitioning (constraints on the test set) of the data strongly affect the impact of both copying and delexicalisation.

It should also be noted that copying and delexicalisation are methods that work for languages with no or few inflection changes of nouns; that is why this chapter is focused on English NLG corpora and research studies done on English. We point out as well that referring expression generation also lies beyond the scope of the current study, however it would be particularly interesting to see how named entities can be differently realised, rather than simply copied verbatim, in generation systems.

4.3 Experiments

4.3.1 Datasets

Two recently released corpora for generation served as experimental datasets for our study: the E2E (Novikova et al., 2017b) and the WebNLG (Gardent et al., 2017a) datasets.

In the E2E dataset, as introduced in Section 2.1.2, the input to generation is a dialogue act consisting of three to eight slot-value pairs describing a restaurant, while the output is a restaurant recommendation verbalising this input. Table 4.1 shows an example with an input consisting of six slot-value pairs. On average, each input is associated with 8.1 references, and there are 5.43 slot-value pairs per input. The number of possible values for each slot ranges from two (binary slots) to 34 (restaurant name). Tables 4.2 and 4.3 summarise the statistics of the E2E dataset.

In WebNLG, as introduced in Section 2.1.2, the aim is to verbalise a set of RDF triples describing entities of different categories. An RDF triple is of the form *(subject, predicate, object)* where *subject* and *object* denotes entities or values and *predicate* denotes a binary relation holding between *subject* and *object*. The inputs consist of sets of (one to seven) triples and the entities belong to fifteen distinct DBpedia categories[7]. Table 4.1 shows an example with an input consisting of five RDF triples, and Tables 4.2 and 4.3 summarise the data statistics.

Both dataset releases gave rise to a shared task in NLG in 2017[8]. Note though that for WebNLG, the present study relies on the version 2.0[9], which is larger than the one used for the

[6] Ironically, it was presented at the same INLG conference as our study.

[7] http://wiki.dbpedia.org/dbpedia-dataset-version-2015-10

[8] http://www.macs.hw.ac.uk/InteractionLab/E2E/, https://webnlg-challenge.loria.fr/challenge_2017/

[9] https://gitlab.com/shimorina/webnlg-dataset

MR / RDF triples	Reference
Original: name[The Cricketers], eatType[coffee shop], food[Chinese], customer rating[average], familyFriendly[no], near[The Portland Arms]	**Original**: The Cricketers is a coffee shop that also has Chinese food, located near The Portland Arms. It is not family friendly, and has an average customer rating.
Delexicalised: name[NAME], eatType[coffee shop], food[Chinese], customer rating[average], familyFriendly[no], near[NEAR]	**Delexicalised**: NAME is a coffee shop that also has Chinese food, located near NEAR. It is not family friendly, and has an average customer rating.
Original: (Bakewell pudding, region, Derbyshire Dales), (Bakewell pudding, dishVariation, Bakewell tart), (Bakewell pudding, servingTemperature, Warm or cold), (Bakewell pudding, course, Dessert), (Bakewell pudding, mainIngredients, Ground almond, jam, butter, eggs)	**Original**: Bakewell pudding, also called bakewell tart, originates from the Derbyshire Dales. Classified as a dessert which can be served warm from the oven or cold, its main ingredients are ground almond, jam, butter and eggs.
Delexicalised: (FOOD, region, REGION), (FOOD, dishVariation, DISHVARIATION), (FOOD, servingTemperature, Warm or cold), (FOOD, course, DESSERT), (FOOD, mainIngredients, MAININGREDIENTS)	**Delexicalised**: FOOD, also called DISHVARIATION, originates from the REGION. Classified as a COURSE which can be served warm from the oven or cold, its main ingredients are MAININGREDIENTS.

Table 4.1: Entry examples of the E2E (top) and WebNLG (bottom) datasets with and without delexicalisation.

Attribute	Value Range	Example
area	2	city centre, riverside
customer rating	6	3 out of 5, low, high
eatType	3	restaurant, coffee shop, pub
familyFriendly	2	no, yes
food	7	English, Chinese, Fast food
name	34	The Plough, Alimentum, Zizzi
near	19	Café Sicilia, Crowne Plaza Hotel
priceRange	6	more than £30, cheap, moderate

	Count	Example
Predicates	373	dateOfBirth, genre
Subjects	732	Buzz Aldrin
Objects	2,916	1932-03-15, jazz

Table 4.2: Statistics on attribute values in E2E (above) and on RDF-triple constituents in WebNLG (below).

	E2E Dataset			
	Unconstrained		*Constrained*	
	Instances	MRs	Instances	MRs
train	40,868	4,862	40,826	4,877
dev	4,521	547	3,946	547
test	4,577	630	5,194	615

	WebNLG Dataset			
	Unconstrained		*Constrained*	
	Instances	MRs	Instances	MRs
train	34,352	12,876	34,536	12,895
dev	4,316	1,619	4,217	1,594
test	4,224	1,600	4,148	1,606

Table 4.3: Training/development/test sets statistics in E2E and WebNLG in original (unconstrained) and constrained splits. *Instances* count is a number of (data, text) pairs; *MRs* count is a number of unique data inputs.

WebNLG Challenge 2017. Even if both datasets were created for different purposes (verbalising dialogue acts vs. verbalising knowledge bases), their inputs can be viewed as a semantic representation, so in the following we sometimes refer to both inputs as meaning representation (MR) for convenience.

Delexicalising Datasets

We derive delexicalised datasets from the original E2E and WebNLG datasets as follows.

For each dataset, we replicated the delexicalisation procedure which was applied to the baseline systems developed for the E2E (Novikova et al., 2017b) and for the WebNLG challenge (Gardent et al., 2017b) respectively. Delexicalisation for both datasets is based on exact string matching and does not deal with pronouns and other referring expressions.

As shown in Table 4.1, both input data and output text were delexicalised. In E2E, only the *name* and *near* slots were delexicalised (because contrary to the other slots, they have a large number of distinct values). In WebNLG, delexicalisation was done on the subjects and objects of RDF triples.

Delexicalisation was flawless in E2E as slot values for *name* and *near* attributes were always found in corresponding texts. WebNLG data poses additional challenges as the subject and object values in the input do not necessarily match the corresponding text fragment in the output. As a result, not all subjects and objects were delexicalised: around 25% of subjects and objects were not replaced with placeholders in texts. For instance, the object *warm or cold* from the WebNLG example in Table 4.1 was not delexicalised, because it was realised differently, namely *warm from the oven or cold*. Another limitation of delexicalisation is that it does not cover referring expressions, and it may produce incorrect delexicalisations. Given a set of two triples (The Statue of Liberty, location, New York City), (New York City, isPartOf, New York), the corresponding reference text *The Statue of Liberty is located in New York City, New York* may be wrongly delexicalised as *The Statue of Liberty is located in ISPARTOF City, ISPARTOF.* instead of the correct *The Statue of Liberty is located in LOCATION, ISPARTOF.*

In the delexicalised E2E corpus, placeholders constitute 5.7% of all tokens in target texts, while they reach 15.7% in the WebNLG data.

The Copy Mechanism

The copy mechanism is widely used in text production approaches where it is relevant for handling rare input but also, for instance, in text summarisation, for copying input into the output. Thus, Cao et al. (2017) uses a copy mechanism to generate paraphrases, Gu et al. (2016), Cheng and Lapata (2016) for text summarisation and He et al. (2017) for answer generation.

Here we make use of the copy mechanism as defined in Section 2.2.3.

Constraining Datasets

The NLP community is well aware of importance of challenging adversarial scenarios while testing developed models (Jia and Liang, 2017; Naik et al., 2018, *inter alia*). One of the basic requirements for train/dev/test split is to ensure that there is no overlap in terms of *input* between the training, the development and the test set. As Aharoni and Goldberg (2018) recently showed however, this may result in a setup where certain *input fragments* (in that case, subject and object entities present in the input set of RDF triples) are present so often in the training set that models built on this standard split, overfit and memorise rather than learn. Thus, in the split-and-rephrase application they studied, Aharoni and Goldberg (2018) observed

that, given some input containing the entity e and some set of facts $T(e)$ about this entity, the model will regularly output a text which mentions e but is unrelated to the set of facts $T(e)$. That is, instead of learning to generate text from data, the model learns to associate a text with an entity.

To better assess the impact of delexicalisation and copying on the output of generation models, we therefore consider two ways of partitioning the corpus into train, dev and test: the traditional way (*Unconstrained*) where the overlapping constraint applies to entire inputs (i.e., sets of RDF triples in WebNLG and dialogue moves in E2E) and a more challenging split (*Constrained*) where the no-overlap constraint applies to input fragments (i.e., RDF triples in WebNLG and slot-values in E2E). Table 4.3 shows the statistics for both splits for each dataset.

Unconstrained. The unconstrained split is the original split provided by dataset creators.

The E2E dataset was split into training, validation and test sets following a 76.5/8.5/15 ratio. It was ensured that the input were distinct for all three sets and that a similar distribution of input and reference text lengths was kept. We found 1,430 identical (MR, text) pairs in the original E2E data. They were deleted for the subsequent experiments.

In WebNLG, the original split follows an 80/10/10 ratio. As with the E2E dataset, there is a null intersection in terms of input between train, dev and test. In addition, sets of triples of different sizes and sets of triples of different categories were proportionally distributed between training, dev and test sets.

Constrained. We consider a second partitioning where we aim to minimise the overlap between train, dev and test in terms of input fragments.

As shown in Table 4.2, in the E2E dataset, most of the slots have under eight possible values. As these few values appear with a large number of distinct slot-value combinations (49,966 input-text instances), they are unlikely to trigger fact memorisation. We therefore focus on those slots which have a higher number of values and restrict the test set using restaurant names, a slot with 34 possible values. Four restaurant names were selected to occur only in the test data and to support a distribution of inputs types and text length similar to that of the original train/dev/test (see Table 4.3).

Nonetheless, it is worth noting that the E2E dataset was constructed in such a way that a specific restaurant name may have mutually exclusive values in different inputs, such as *low customer rating* and *high customer rating*. This might result in weak association between restaurant names and specific inputs and therefore, in little risk of memorising facts related to a specific restaurant name. As we shall see in Section 4.4, this intuition is confirmed by the results which show little differences, for the E2E data, in terms of both automatic and human-based metrics between the *Constrained* and the *Unconstrained* setting. Note also that since the E2E *Constrained* split is defined with respect to a slot value (restaurant names) which is delexicalised, the constrained *vs.* unconstrained split distinction loses its impact in the delexicalised setting.

For the WebNLG dataset, the constraint on the train/dev/test partition is in terms of triples. In addition to the exclusion from the test set all inputs (set of RDF triples) which occur in either the dev or the train set, we require that no RDF triple occurs in two of these sets. Let $t = (s, p, o)$ be an RDF-triple, with p a predicate and s, o subject and object RDF resources. In the constrained dataset, if, t is in the test set, then t may not be in either the dev or the training set but variants such as (s', p, o), (s, p, o'), (s', p, o') or (s, p', o) with $s \neq s'$, $p \neq p'$ and $o \neq o'$ may. In this way, models can be trained which must learn to verbalise predicates independently of their arguments. Again, care was taken to keep the distribution in terms of input length

similar to that of the original split (see Table 4.3).

Rare Items

By rare items we mean n-grams in input MRs, such as slot values in E2E for *name* and *near* and subjects and objects in WebNLG. These n-grams occur sparsely in the training data (unconstrained case) or they are not seen there at all (constrained case). The main motivation behind identifying and handling them separately is to learn verbalisations independently of possible entities. The majority of rare items in the considered datasets are proper names: names of restaurants in E2E and subjects of triples in WebNLG. However, since some objects in WebNLG can be dates, measurements, and string literals (such as *warm or cold* in Table 4.1), we call them rare items here.

4.3.2 Model Parameters

We trained two types of models: a standard sequence-to-sequence model and the same model augmented with a copy and coverage mechanism (denoted as C in the tables). For the standard sequence-to-sequence model, we made use of an LSTM encoder-decoder model with attention (Luong et al., 2015) from the OpenNMT-py toolkit[10], a PyTorch port of OpenNMT (Klein et al., 2017). The default parameters of OpenNMT-py were used for training and decoding. The encoder and decoder both have two layers. Models were trained for 13 epochs, with a mini-batch size of 64, a dropout rate of 0.3, and a word embedding size of 500. They were optimised with stochastic gradient descent with a starting learning rate of 1.0. We built the vocabulary using all the tokens in the training set.

As the size of the vocabulary is relatively small for both datasets, data was not lowercased, nor was it truncated (the maximal sequence length was used in the source and target).

Special options available in OpenNMT-py were used to augment the standard model with the copy and the coverage mechanisms. The OpenNMT-py implementation of training additional copy and coverage attention layers follows See et al. (2017).

4.3.3 Evaluation

Automatic Evaluation. Systems were evaluated using four automatic corpus-based metrics: BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Denkowski and Lavie, 2014), ROUGE$_L$ (Lin, 2004). We made use of the scripts used for the E2E Challenge evaluation[11]. The first three metrics were originally developed for machine translation, the last one for summarisation. Roughly speaking, BLEU calculates the n-gram precision; NIST is based on BLEU, but adds more weight to rarer n-grams; METEOR computes the harmonic mean of precision and recall, featuring also stem and synonymy matching; ROUGE$_L$ calculates recall for common longest subsequences in a reference and candidate text. Given our task—handling rare items (or mostly named entities in the corpora in question)—we also applied the slot-error rate (SER) to evaluate outputs which seems to be more suitable for evaluating the presence of named entities. SER was calculated by exact matching slot values in the candidate texts,

$$SER = \frac{S + D + I}{N},$$

[10] https://github.com/OpenNMT/OpenNMT-py
[11] https://github.com/tuetschek/e2e-metrics

where S is a number of substitutions, D is a number of deletions, I is a number of insertions, and N is a total number of slots in the reference. The resulting SER is an average of SER for each prediction. While computing SER for the dialogue slot-based E2E corpus is straightforward (the binary slot *familyFriendly* was excluded), it results in some noise for WebNLG where subjects and objects are numerous (3,648 vs. 79 values in E2E) and where they were rephrased in references (cf. also Section 4.3.1).

Manual Annotation. To allow comparisons between constrained and unconstrained settings, we intersected inputs of constrained and unconstrained test sets and gathered corresponding predictions from them for all the models. The intersection between the two test sets has 40 inputs in the E2E corpus and 153 in WebNLG. For E2E, we manually evaluated all 40 predictions available for each system (constrained and unconstrained); for WebNLG, we chose 44 predictions ensuring the presence of different sizes and categories. By manually assessing outputs for the same inputs for all the systems, contrasts between constrained and unconstrained settings are better highlighted.

Manual inspection of outputs revealed that most of generated predictions did not encounter issues with grammar or fluency. For this reason, we chose to focus on semantic adequacy of predicted texts. The evaluation was done by one human judge. After the evaluation was finished, the human annotator confirmed that, except for one system (see Section 4.4), all system outputs demonstrated fluent and grammatical English sentences.

Once presented with an input and a corresponding prediction text, a human judge was asked to evaluate semantic information present in the prediction. A minimal unit of analysis was a slot-value pair in E2E and an RDF triple element (subject, object, or predicate) in WebNLG. For each semantic unit, the judge indicated if it was rendered correctly (*right*) or incorrectly (*wrong*) in the text. If the unit was missing, it was noted as *missed*; new semantic content, not present in the source input, was labelled as *added*. Then, the number of each type of annotations was calculated for each input and converted to percentage with respect to the number of slot-value pairs (E2E) or number of triple constituents (WebNLG). Given the E2E example in Table 4.4, statistics about the example is the following: right: 2, wrong: 1, added: 1, missed: 1 (*near[Burger King]* was omitted). Total number of slots being 4, the performance in the percentage is then right: 50%, wrong: 25%, added: 25%, missed: 25%.

E2E MR prediction annotation	name[Cocum], eatType[pub], customer rating[high], near[Burger King] Cotto is a family-friendly pub with a high customer rating. Cotto {*wrong*}, family-friendly {*added*}, pub {*right*}, high customer rating {*right*}, near Burger King {*missed*}
WebNLG triple prediction annotation	(A Wizard of Mars, author, Diane Duane) A Wizard of Mars was written in the United States in 1995. A Wizard of Mars {*right*}, was written {*right*}, in the United States {*wrong*}, in 1995 {*added*}

Table 4.4: Manual annotation of text predictions for E2E and WebNLG data. Annotations are between curly braces.

WebNLG example annotations were done taking into account the three parts of a triple. If a predicate was not translated correctly, we considered that a model missed out that information. While a subject or object was not rendered correctly, they were annotated as wrong. All the semantic information beyond the size of initial set of triples was evaluated as added. The

WebNLG example in Table 4.4 received the following scores, the total number of constituents being three: right: 2 (66%), wrong: 1 (33%), missed: 0 (0%), added: 1 (33%). If semantic information was repeated, it was rated as added.

The human evaluation analysis presented above is modest due to the lack of resources. To justify it, we argue that our focus is solely on semantic adequacy which is a more objective parameter in evaluations than, say, fluency or grammaticality. Furthermore, human scores showed strong correlations with most of automatic metrics. For example, *right* exhibits statistically significant correlations of 0.9, 0.55, 0.89, 0.85, −0.87 with BLEU, NIST, METEOR, ROUGE$_L$, SER respectively (Spearman's ρ; $p < 0.05$). *Wrong* has −0.91, −0.71, −0.88, −0.96, 0.78 correlation coefficients respectively.

With no intent to question the documented unreliability of automatic metrics in NLG, we attribute such high correlations to the design of our configurations which cover some extreme cases where models are supposed to show a drastic drop in performance.

4.4 Results and Discussion

We compared the output of the sequence-to-sequence model with attention described in Section 4.3.2 on two datasets (WebNLG and E2E) and considering eight different configurations depending on how rare words are handled (without delexicalisation, with delexicalisation, with a copy-and-coverage mechanism and with both delexicalisation and a copy-and-coverage mechanism) and on how the train/dev/test partition is constructed (unconstrained *vs.* constrained).

As pointed out in Section 4.3.3, automatic scores are reported using the whole test sets whereas human evaluation is based on shared MR instances between the non-constrained and constrained test sets (40 instances for E2E and 44 for WebNLG).

The results are summarised in Table 4.5 (E2E) and 4.6 (WebNLG). Some example predictions are shown in Tables 4.7 and 4.8.

	Unconstrained				*Constrained*			
	NIL	*C*	*D*	*D+C*	*NIL*	*C*	*D*	*D+C*
BLEU	0.56	**0.68**	0.67	**0.68**	0.52	0.57	**0.72**	**0.72**
NIST	7.54	8.67	8.60	**8.74**	7.12	7.67	**8.93**	8.90
METEOR	0.38	**0.46**	0.45	**0.46**	0.39	0.41	**0.47**	**0.47**
ROUGE$_L$	0.62	**0.71**	0.70	0.70	0.58	0.62	**0.74**	**0.74**
SER	26.07%	7.25%	**4.08%**	4.56%	29.6%	17.09%	5.18%	**4.2%**
right	81.72%	95.7%	96.24%	**96.77%**	72.04%	82.8%	**95.7%**	94.09%
wrong	16.13%	0%	0%	0%	27.96%	16.13%	0%	0%
missed	2.15%	4.3%	3.76%	3.23%	0%	1.08%	4.3%	5.91%
added	6.45%	0%	5.38%	2.15%	0%	4.84%	2.15%	0%

Table 4.5: E2E dataset (*D*: Delexicalisation, *D+C*: delexicalisation and copying, *C*: copy and coverage, *NIL*: Neither copy nor delexicalisation). The upper half of the table presents automatic evaluation results; the lower half—human evaluation results. Best scores are in bold.

	Unconstrained				*Constrained*			
	NIL	*C*	*D*	*D+C*	*NIL*	*C*	*D*	*D+C*
BLEU	0.54	**0.61**	0.56	0.56	0.09	0.34	0.44	**0.48**
NIST	9.70	**10.90**	10.19	10.11	2.37	6.81	7.37	**8.09**
METEOR	0.37	**0.42**	0.39	0.39	0.10	0.29	0.33	**0.36**
ROUGE$_L$	0.64	**0.71**	0.67	0.68	0.26	0.54	0.61	**0.65**
SER	43.66%	34.76%	34.93%	**31.83%**	92.5%	66.91%	50.48%	**45.45%**
right	69.26%	83.33%	83.70%	**87.04%**	10%	41.11%*	70.00%	**76.67%**
wrong	9.63%	5.56%	9.26%	7.78%	49.26%	32.59%	17.78%	15.93%
missed	21.11%	11.11%	7.04%	5.19%	40.74%	26.30%	12.22%	7.41%
added	0.37%	0%	0%	0%	1.11%	1.11%	0%	0%

Table 4.6: WebNLG dataset (*D*: Delexicalisation, *D+C*: delexicalisation and copying, *C*: copy and coverage, *NIL*: Neither copy nor delexicalisation). The upper half of the table presents automatic evaluation results; the lower half—human evaluation results. Best scores are in bold. * – word repetitions present in predictions.

Delexicalisation and Copying *vs.* Standard Encoding-Decoding. A first observation is that, when neither delexicalisation nor copying is used, there is a strong drop in semantic adequacy. In the worst case, the SER increases by 25.4 for the E2E dataset (constrained setting, NIL *vs.* D+C) and by 47.05 points (constrained setting, NIL *vs.* D+C) in the WebNLG dataset. Similarly, the proportion of correcly predicted items (right) decreases by up to 23.66 points for the E2E dataset (constrained setting, NIL *vs.* D) and 60 points for the WebNLG dataset (constrained setting, NIL *vs.* D).

A similar, though weaker, trend can be observed for the other automatic metrics (e.g., ΔBLEU E2E, NIL *vs.* D+C, unconstrained: −0.12 points).

Delexicalisation, Copying or Both. The results show two trends. First, combining copying and delexicalisation yields the best results across the board. Second, while in the unconstrained setting, there is not much difference in terms of results between copying and delexicalisation, in the constrained setting, copying yields lower results (ΔBLEU E2E: −0.15, ΔBLEU WebNLG: −0.10, Δright E2E: −12.9%, Δright WebNLG: −28.89%, ΔSER E2E: +11.91, ΔSER WebNLG: +16.43; constrained setting, C *vs.* D). This suggests that copying only partially captures rare items. Looking at the outputs, copying seems to work better when the item to be copied has been seen in the training data. When an entity was not seen, the network often chooses to generate a frequent entity seen in the source, rather than copying. For instance, for the E2E data, restaurant names (which had not been seen in the training data) were not copied over in the constrained setting. In most cases, the input restaurant name was replaced by a restaurant name that is frequent in the training data. For example, given the MR *name[Cocum], eatType[coffee shop], near[The Rice Boat]*, the text *Near The Rice Boat there is a coffee shop called Fitzbillies* was generated, where *Fitzbillies*, a frequently occurring restaurant name in the training data (2,371 instances), was generated instead of the input restaurant name *Cocum*.

Constrained *vs.* Unconstrained Setting. There is a clear difference in terms of relative performance in the constrained *vs.* the unconstrained setting between the two datasets.

For the E2E dataset, since the constrained dataset is defined with respect to slot values (`name` and `near`) which are delexicalised, the constrained setting is in fact similar to the unconstrained setting. And indeed the scores are similar (e.g., unconstrained *vs.* constrained, D, E2E: ΔBLEU: −0.05, ΔSER: −1.1 and Δright: +0.54%). When using copying however, the results are lower in the constrained setting again suggesting that copying underperforms for items that have not been seen at training time (e.g., unconstrained *vs.* constrained, C, E2E: ΔBLEU: 0.11, ΔSER: −9.84 and Δright: 12.9%).

For the WebNLG data, the difference between constrained and unconstrained setting is much stronger for both delexicalisation and copying. For instance, for copying the BLEU score in the unconstrained setting is 0.61 *vs.* 0.34 in the constrained setting. Semantic adequacy also drops noticeably (unconstrained: 83%, constrained: 41%). This is in line with Aharoni and Goldberg (2018)'s observation that in the unconstrained setting, the model learns to memorise association between facts and entities and thereby fails to generate text that adequately captures the meaning of the input data. The low scores for the copying mechanism also confirm the observation made above that copying underperforms for rare data fragments.

This difference between datasets is further discussed in the next paragraph.

MR	name[Wildwood], eatType[pub], food[Italian], area[city centre], familyFriendly[no], near[Raja Indian Cuisine]
reference	Wildwood is an Italian pub situated in the city centre, near the Raja Indian Cuisine. It is not family-friendly.
unconstr., NIL	Wildwood is an Italian pub located in the city centre near **the Ranch**. It is not family-friendly.
unconstr., C	Wildwood is an Italian pub located in the city centre near Raja Indian Cuisine. It is not family-friendly.
unconstr., D	Wildwood is an Italian pub located in the city centre near Raja Indian Cuisine. It is not family-friendly.
unconstr., D+C	Wildwood is an Italian pub located near the Raja Indian Cuisine in the city centre. It is not family-friendly.
constr., NIL	In the city centre near Raja Indian Cuisine there is an Italian pub called the **Aromi**. It is not family-friendly.
constr., C	There is a pub near Raja Indian Cuisine in the city centre called **Aromi**. It serves Italian food and is not family-friendly.
constr., D	Wildwood is an Italian pub near Raja Indian Cuisine in the city centre. It is not family-friendly.
constr., D+C	Wildwood is an Italian pub located near the Raja Indian Cuisine in the city centre. It is not family-friendly.

Table 4.7: Example predictions for E2E. Mistakes are in bold.

Semantic Adequacy. As mentioned above, the manual and automatic evaluation metrics we used to assess semantic adequacy strongly correlate. They both show that semantic adequacy is much lower for the WebNLG data (higher SER, higher proportion of wrong and missed items). This is not surprising, since the WebNLG dataset contains a much higher number of distinct values (3,648 against 79 in the E2E dataset) and exhibits a greater mismatch between input and

MR	(108 St Georges Terrace, floorCount, 50), (108 St Georges Terrace, completionDate, 1988)
reference	108 St Georges Terrace has a floor count of 50 and was completed in 1988.
unconstr., NIL	108 St Georges Terrace **cost 120 million Australian dollars**.
unconstr., C	108 St Georges Terrace was completed in 1988 and has 50 floors.
unconstr., D	108 St Georges Terrace has 50 floors and was completed in 1988.
unconstr., D+C	108 St Georges Terrace has 50 floors and was completed in 1988.
constr., NIL	**The coach of the Democratic Party in the United States is the Conservative Party (UK)**.
constr., C	108 Georges Terrace **completionDate were created by 108 Georges**.
constr., D	108 St Georges Terrace has 50 floors and was completed in 1988.
constr., D+C	108 St Georges Terrace has 50 floors and was completed in 1988.

Table 4.8: Example predictions for WebNLG. Mistakes are in bold.

output value names[12]. That is, the delta shows that the efficiency of copying and delexicalisation varies depending on the variety and content of the dataset.

The two datasets also differ with respect to the proportion of added slots which is higher for the E2E dataset and suggests an overfitting effect due to a skewed distribution in favour of inputs containing more than 3 attributes. Thus, the human evaluation shows that the majority of cases with added slots are cases where the input consists of three slots (the minimal number of attributes in E2E). The overgeneration can be explained by the restricted number of three-slot inputs in the E2E dataset (only 2.5% MRs out of the whole corpus). That claim is also supported by predictions produced by adversarial examples. While inputting dialogue moves consisting of 2 slots (the non-existent number of attributes in E2E), all eight E2E models tend to overgenerate by predicting texts with 3 or 4 slot-value pairs.

Fluency. As mentioned in Section 4.3.3, while annotating the data for semantic adequacy, we found that almost all systems outputs were well-formed English sentences. The only exception was the WebNLG model with copy mechanism where stutterings were spotted in half of the examined instances. Despite those repetitions, it was always possible to detect the subject-predicate-object structure (e.g., *1001 kelvins is an escape velocity of 1001 kelvins*; *Asterix was created by R. Goscinny and was created by R. Goscinny*), so the annotation was not hampered.

4.5 Conclusion

We investigated the impact of copying and delexicalisation on two datasets and using two different ways of splitting the data into train, dev and test. The results show some regularities and highlight some interesting differences.

Overall, results indicate that delexicalisation outperforms copying. Furthermore, they show that copying underperforms on unseen items. Delexicalisation is a somewhat ad hoc process: it should be devised for a particular task in question, and sometimes it works only partially, e.g., in case of data where entities can be replaced with synonyms in text. So an interesting direction for future research would be to devise copying methods that are more accurate and that can better handle rare data items.

[12]In the E2E dataset, the value name in the input is usually realised by the same string in the corresponding text while in WebNLG, they often differ, e.g., USA/the United States of America (see Section 4.3.1).

Another direction for future research would be to further investigate how the content and train/dev/test split of a dataset impact learning. Our results suggest two ways in which these may induce overfitting.

In the WebNLG dataset, strong associations between entities and facts seem to result in generation models that memorise facts with entities rather than generate a text that adequately verbalises the input. This is highlighted in the manual evaluation by the high number of wrong and missed data items observed both in the constrained and in the unconstrained setting.

In the E2E dataset, on the other hand, we saw that added facts are frequent and manual evaluation suggests that this is due to an overfitting effect whereby, because most inputs consist of more than three slot-value pairs, the models tend to overgenerate by predicting texts that verbalise four or more slot-value pairs.

In both cases, the copy-and-coverage mechanism does not suffice to ensure correct output and the results further decrease in the constrained setting. It would therefore be interesting to see to what extent better methods can be devised both for creating datasets and for devising train/dev/test splits that adequately test the ability of models to generalise.

Another perspective for future work is to investigate the capability of subword- and character-based representations to handle rare input tokens in generation (Jagfeld et al., 2018). As pointed out in Section 4.2, copying and delexicalisation are restricted for languages with no or few inflection processes, so those representations would be particularly interesting for morphologically rich languages.

This chapter also leaves out how to generate referring expressions for proper names, which were the majority of the rare items considered. Generation of referring expressions for proper nouns has been studied in the literature (Siddharthan et al., 2011; van Deemter, 2016; Castro Ferreira et al., 2018a; Cao and Cheung, 2019), and it is worth exploring the ability of those proposed methods to refer to entities not seen in the training data.

5

Training Models for Surface Realisation

Contents

5.1 Introduction **45**
5.2 Motivation and Related Work **46**
5.2.1 Motivation 46
5.2.2 Related Work 47
5.3 Data **48**
5.3.1 SR'18 48
5.3.2 SR'19 48
5.4 Model **49**
5.4.1 Word Ordering 49
5.4.2 Morphological Realisation 51
5.4.3 Contraction Generation 51
5.5 Evaluation on SR'18 . . . . . **52**
5.5.1 Word Ordering 52
5.5.2 Morphological Realisation 55
5.5.3 Contraction Generation 56
5.5.4 Global Evaluation 56
5.6 Participation in SR'19 . . . . . **57**
5.6.1 Model Adaptation 57
5.6.2 Results and Discussion 58
5.7 Conclusion **63**

5.1 Introduction

In the previous chapters 3 and 4, we focused on named entities and discussed how to handle them when creating training data for NLG or when generating from data rich in named entities. The input we considered was RDF triples and dialogue acts. In this chapter, we deal with another type of input—unordered dependency trees—and explore multilingual generation, focusing on the surface realisation task. In Chapter 4, the limitation of the discussed NLG systems consisted

in treating only English, a language with relatively poor morphology, and applying methods that were specific to this language. In this chapter, we aim to show how to process a variety of languages, paying attention to their morphology.

Surface realisation (SR) maps a meaning representation to a sentence. In data-to-text generation, it is part of a complex process aiming to select, compress and structure the input data into a text (Reiter and Dale, 2000). In text-to-text generation, it can be used as a mean to rephrase part or all of the input content. For instance, Liao et al. (2018) used surface realisation to generate a summary based on the meaning representations of multiple input documents and Takase et al. (2016) to improve neural headline generation.

By providing parallel data of sentences and their meaning representation, the Surface Realisation shared tasks (Belz et al., 2011; Mille et al., 2018a, 2019) allow for a detailed evaluation and comparison of surface realisation models. Moreover, as recent SR tasks provide training and test data for multiple languages, it also allows for an analysis of how well these models handle languages with different morphological and topological properties.

As mentioned in Section 2.1.2, the SR shared tasks include two tracks: a shallow track where the input is an unordered, lemmatised dependency tree and a deep track where function words are removed and syntactic relations are replaced with semantic ones. In this chapter, we focus on the shallow track of the SR'18 and SR'19 shared tasks, and we propose a neural approach which decomposes surface realisation into three subtasks: word ordering, morphological inflection and contraction generation (e.g., clitic attachment in Portuguese or elision in French). We provide a detailed analysis of how each of these phenomena (word order, morphological realisation and contraction) is handled by the model, and we discuss the differences between languages.

This chapter is based on the following publications:

- Shimorina, A. and Gardent, C. (2019b). Surface realisation using full delexicalisation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3086–3096, Hong Kong, China. Association for Computational Linguistics

- Shimorina, A. and Gardent, C. (2019a). LORIA / Lorraine University at multilingual surface realisation 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 88–93, Hong Kong, China. Association for Computational Linguistics

The first publication introduced the model and tested it on the SR'18 data. The second publication is our system description for the SR'19 shared task: we reused our model with minor modifications and reported the results on the SR'19 data.

For reproducibility, all our experiments including data and scripts are available at `https://gitlab.com/shimorina/emnlp-2019` for SR'18 experiments and at `https://gitlab.com/shimorina/msr-2019` for SR'19 experiments.

5.2 Motivation and Related Work

5.2.1 Motivation

Surface realisation is a natural language generation task which consists in converting a linguistic representation into a well-formed sentence. SR is a key module in pipeline generation models where it is usually the last item in a pipeline of modules designed to convert the input (knowledge graph, tabular data, numerical data) into a text. While end-to-end generation models have

been proposed which do away with such pipeline architecture and therefore with SR, pipeline generation models (Dušek and Jurčíček, 2016; Castro Ferreira et al., 2019; Elder et al., 2019; Moryossef et al., 2019) have been shown to perform on a par with these end-to-end models while providing increased controllability and interpretability (each step of the pipeline provides explicit intermediate representations which can be examined and evaluated).

As illustrated in, e.g., (Dušek and Jurčíček, 2016; Elder et al., 2019; Li, 2015), SR also has potential applications in tasks such as summarisation and dialogue response generation. In such approaches, shallow dependency trees are viewed as intermediate structures used to mediate between input and output, and SR permits regenerating a summary or a dialogue turn from these intermediate structures.

Finally, multilingual SR is an important task in its own right in that it permits a detailed evaluation of how neural models handle the varying word order and morphology of the different natural languages. While neural language models are powerful at producing high quality text, the results of the multilingual SR tasks (Mille et al., 2018a, 2019) clearly show that the generation, from shallow dependency trees, of morphologically and syntactically correct sentences in multiple languages remains an open problem.

5.2.2 Related Work

Early approaches for surface realisation adopted grammar-based (Callaway, 2003; White et al., 2007) and statistical methods (Marciniak and Strube, 2004), or a mixture of symbolic and probabilistic techniques (Langkilde-Geary, 2002; Nakanishi et al., 2005b). Statistical methods included both pipeline (Bohnet et al., 2010) and joint (Song et al., 2014; Puduppully et al., 2017) architecture for word ordering and morphological generation.

Multilingual SR'18 and SR'19 were preceded by the SR'11 surface realisation task for the English language only (Belz et al., 2011). The submitted systems in 2011 had grammar-based and statistical nature, mostly relying on pipelined architecture (Guo et al., 2011; Rajkumar et al., 2011; Gervás, 2011; Stent, 2011; Bohnet et al., 2011). Recently, Marcheggiani and Perez-Beltrachini (2018) proposed a neural end-to-end approach based on graph convolutional encoders for the SR'11 deep track.

The SR'18 shallow track received submissions from eight teams with seven of them dividing the task into two subtasks: word ordering and inflection. Only Elder and Hokamp (2018) developed a joint approach, however, they participated only in the English track.

For word ordering, five teams chose an approach based on neural networks, two used a classifier, and one team resorted to a language model. As for the inflection subtask, five teams applied neural techniques, two used lexicon-based approaches, and one used an SMT system (Basile and Mazzei, 2018; Castro Ferreira et al., 2018c; Elder and Hokamp, 2018; King and White, 2018; Madsack et al., 2018; Puzikov and Gurevych, 2018a; Singh et al., 2018; Sobrevilla Cabezudo and Pardo, 2018). Overall, neural components were dominant across all the participants. However, official scores of the teams that went neural greatly differ. Furthermore, two teams (Elder and Hokamp, 2018; Sobrevilla Cabezudo and Pardo, 2018) applied data augmentation, which makes their results not strictly comparable to others.

One of the interesting findings of the shared task is reported by Elder and Hokamp (2018) who showed that applying standard neural encoder-decoder models to jointly learn word ordering and inflection is highly challenging; their sequence-to-sequence baseline without data augmentation got 43.11 BLEU points on English (a relatively moderate result).

While SR'18 shared task results showed quite a low performance of systems with respect to human-written texts, SR'19 shallow track has almost closed this gap, where the best systems

showed readability scores on a par with references for English. However, for other languages and the deep track, there is still room for improvement. As in SR'18, most participating teams split surface realisation into two subtasks: most of their components were neural with two teams employing symbolic ones for word ordering and one team—for morphological realisation (Mille et al., 2019). The best performing system in SR'19 (Yu et al., 2019a) used a Tree-LSTM (Zhou et al., 2016) to represent a tree structure where each node was encoded using its lemma, morphological features, and dependency label. Their linearisation, however neural, is based on Bohnet et al. (2010), and their neural inflection module predicts a sequence of edit operations. They also developed a separate module for contraction generation, which is a character-based sequence-to-sequence model.

Our model differs from previous work in several ways. First, it performs word ordering on fully delexicalised data. Delexicalisation has been used previously but mostly to handle rare words, e.g. named entities. Here we argue that surface realisation and, in particular, word ordering works better when delexicalising all input tokens. This captures the intuition that word ordering is mainly determined by the syntactic structure of the input. Second, we provide a detailed evaluation of how our model handles the three subtasks underlying surface realisation. While all SR participants provided descriptions of their models, not all of them performed an in-depth analysis of model performance. It is also not clear how each of those modules affects the global performance when merged in the full pipeline. In contrast, we propose a detailed incremental evaluation of each component of the full pipeline and show how each component impacts the final scores. Third, we introduce a linguistic analysis, based on the dependency relations, of the word ordering component, allowing for deeper error analysis of the developed systems. Furthermore, our model explicitly integrates a module for contraction handling, as done also by Basile and Mazzei (2018); Yu et al. (2019a). We also address all the languages proposed by the shared tasks and outline the importance of handling contractions.

Our work appeared concurrently with models of Puzikov et al. (2019); Yu et al. (2019b) on SR'18 data, that is why we do not make comparison with their models.

5.3 Data

5.3.1 SR'18

The SR'18 data (shallow track) is derived from the ten Universal Dependencies (UD) v2.0 treebanks (Nivre et al., 2017) and consists of (T, S) pairs where S is a sentence, and T is the UD dependency tree of S after word order information has been removed and tokens have been lemmatised. The languages are those shown in Table 5.1 and the size of the datasets (training, dev and test) varies between 7,586 (Arabic) and 85,377 (Czech) instances with most languages having around 12K training instances (tree/sentence pairs). For more details about the data see Mille et al. (2018a).

5.3.2 SR'19

SR'19 dataset includes 20 UD treebanks in the shallow track; for some languages several treebanks were available (see Table 5.6 for covered languages and Mille et al. (2019) for a detailed description). Like in SR'18, the size of the training set greatly differs depending on a language: from 803 trees for FRENCH_PARTUT to 48,814 for RUSSIAN_SYNTAGRUS. During the test phase, additional out-of-domain treebanks and automatically parsed data were used. Another new feature of SR'19 data was the introduction of relative order for some words, such as components

of named entities, multiple coordinations and punctuation signs.

For both shared tasks, we used only the data provided by the organisers. If several corpora were available for a language, they were merged to have one training and development dataset (that was the case for SR'19). We used original UD files for creating target files for training the word ordering component, i.e. we extracted a sequence of tokens (the field `token` in the CoNLL format) instead of using a reference sentence.

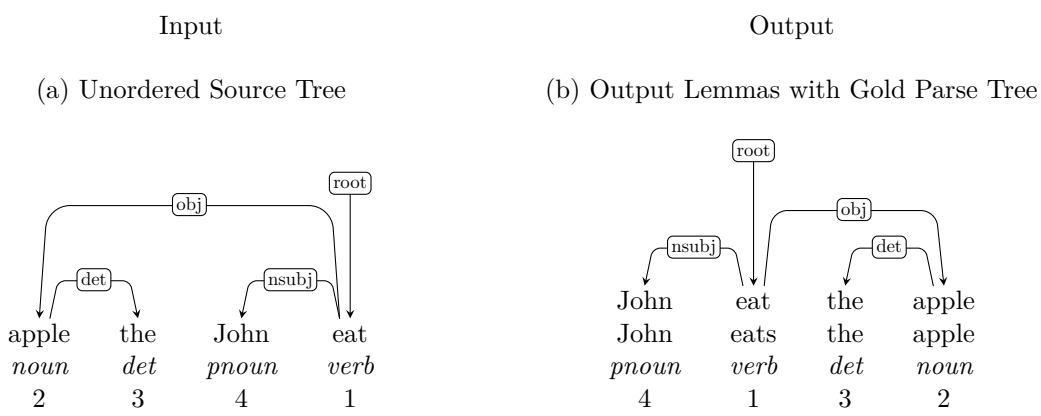
5.4 Model

As illustrated by Example 4, surface realisation from SR shallow meaning representations can be viewed as consisting of three main steps: word ordering, morphological inflection and contraction generation. For instance, given an unordered dependency tree whose nodes are labelled with lemmas and morphological features (4a)[13], the lemmas must be assigned the appropriate order (4b), they must be inflected (4c) and contractions may take place (4d).

(4) a. the find **be not** meaning of life it about
b. it be not about find the meaning of life
c. It **is n't** about finding the meaning of life
d. It **isn't** about finding the meaning of life

We propose a neural architecture which explicitly integrates these three subtasks as three separate modules into a pipeline: word ordering (WO) is applied first, then morphological realisation (WO+MR) and finally, contractions (WO+MR+C) are handled.

5.4.1 Word Ordering



2:noun:obj:1 3:det:DET:2 4:pnoun:nsubj:1 1:verb:root:0 4 1 3 2

Figure 5.1: Delexicalising and linearising (in the parse tree of the output sentence the first row shows the lemmas, the second–the word forms, the third–the POS tags and the fourth–the identifiers). Identifiers are assigned to the source tree nodes in the order given by depth-first search.

For word ordering, we combine a factored sequence-to-sequence model with an "extreme delexicalisation" step which replaces matching source and target tokens with an identifier.

[13]Features and tree structures have been omitted.

Delexicalisation. Delexicalisation has frequently been used in neural NLG to help handle unknown or rare items (Wen et al., 2015; Dušek and Jurčíček, 2015; Chen et al., 2018). Rare items are replaced by placeholders both in the input and in the output; models are trained on the delexicalised data; and a post-processing step ensures that the generated text is relexicalised using the placeholders' original value. In these approaches, delexicalisation is restricted to rare items (named entities). In contrast, we apply delexicalisation to all input lemmas. Abstracting away from specific lemmas reduces data sparsity, allows for the generation of rare or unknown words and last but not least, it captures the linguistic intuition that word ordering mainly depends on syntactic information (e.g., in English, the subject generally precedes the verb).

To create the delexicalised data, we need to identify matching input and output elements and to replace them with the same identifier. We also store a mapping (id, L, F) specifying which identifier id refers to which (L, F) pair, where L is a lemma and F is its set of morpho-syntactic features.

We identify matching input and output elements by comparing the unordered input tree provided by the SR task with the parse tree of the output sentence provided by the UD treebanks (see Figure 5.1). Source and target nodes which share the same path to the root are then mapped to the same identifier. For instance, in Figure 5.1, the lemma "apple" has the same path to the root (obj:eat:root) in both the input and the output tree. Hence the same identifier is assigned to the nodes. More generally, after linearisation through depth-first, left-to-right traversal of the input tree, each training instance captures the mapping between lemmas in the input tree and the same lemmas in the output sequence. For instance, given the example shown in Figure 5.1, delexicalisation will yield the training instance:

Input: tkn2 tkn3 tkn4 tkn1
Output: tkn4 tkn1 tkn3 tkn2

where tkn_i is the factored representation (see below) of each delexicalised input node.

Factored Sequence-to-Sequence Model. Following Elder and Hokamp (2018), we use a factored model (Alexandrescu and Kirchhoff, 2006; Sennrich and Haddow, 2016) as a means of enriching the node representations input to the neural model (see Section 2.2.4). Each delexicalised tree node is modelled by a sequence of features. Separate embeddings are learned for each feature type and the feature embeddings of each input node are concatenated to create its dense representation. As exemplified in Figure 5.1, we model each input placeholder as a concatenation of four features: the node identifier, its POS tag, its dependency relation to the parent node and its parent identifier (the parent identifier of a root node is represented as 0).

Sequence-to-sequence model. We use the OpenNMT-py framework (Klein et al., 2017)[14] to train factored sequence-to-sequence models with attention (Luong et al., 2015) and the copy and coverage mechanisms described in See et al. (2017). A single-layer LSTM is used for both encoder and decoder. We train using full vocabulary and the maximal length in the source and target for both baseline and the proposed model. Models were trained for 20 epochs, with a mini-batch size of 64, a word embedding size of 300, and a hidden unit size of 450. They were optimised with stochastic gradient descent (SGD) with a starting learning rate of 1.0. A learning rate is halved when perplexity does not decrease on the development set. Preliminary experiments showed that the lowest perplexity was reached on average at epoch 17, so this model was kept for decoding. Decoding is done using beam search with a beam size of 5. For each

[14] commit e61589d, https://github.com/OpenNMT/OpenNMT-py

language, we train three models with different random seeds, and report the average performance and standard deviation.

The model is trained on delexicalised data. At test time, the token/identifier mapping is used to relexicalise the model output.

5.4.2 Morphological Realisation

The morphological realisation (MR) module consists in producing inflected word forms based on lemmas coupled with morphological features. For that module, we used a model recently proposed by Aharoni and Goldberg (2017), which achieves state-of-the-art results on several morphological inflection datasets: the CELEX dataset (Baayen et al., 1993; Dreyer et al., 2008), the Wiktionary dataset (Durrett and DeNero, 2013) and the SIGMORPHON2016 dataset (Cotterell et al., 2016). Their model is based on a neural encoder-decoder architecture with hard monotonic attention and performs out-of-context morphologic realisation: given a lemma and a set of morpho-syntactic features, it produces a corresponding word form. As the model of Aharoni and Goldberg (2017) was trained on other tagsets, we adapted and trained it on (lemma + POS + morphological features, form) pairs extracted from the SR training data. We trained the model for 20 epochs with the default parameters provided in the implementation[15].

In our pipeline architecture, morphological realisation is applied to the output of our word ordering model using the (id, L, F) mapping mentioned above. For each delexicalised token produced by the word ordering component, we retrieve the corresponding lemma and morpho-syntactic features (L, F) and apply our MR model to it so as to produce the corresponding word form.

While associating a lemma and its features to a corresponding form, the MR module operates without taking context into account, so it cannot perform some finer grained operations, such as contraction, elision, and clitic attachment. We address that issue in the following section.

5.4.3 Contraction Generation

Contraction handling is the last step of our surface realisation pipeline. Example 5 shows some types of contractions.

(5) French: "Le chat dort." / "L'alouette chante." (Elision for the definite article *le* before a vowel: Le → L')

Italian: "*In il mare." → "Nel mare." (Contraction of the preposition *in* and the article *il*: In il → Nel)

Portuguese: "*Eis lo." → "Ei-lo." (Clitic pronoun attachment: Eis lo → Ei-lo)

We developed two modules for the contraction generation: one based on regular expressions (C_{reg}) and another based on a sequence-to-sequence model (C_{s2s}).

The sequence-to-sequence model is trained on pairs of sentences without and with contractions. The sentence with contraction (S^{+c}) is the final sentence, i.e., the reference sentence in the SR data. The sentence without contraction (S^{-c}) is the corresponding sequence of word forms extracted from the UD CoNLL data.

The regular expression module is inspired by the decomposition of multi-word expressions, such as contractions, which is applied during the tokenisation step in parsing (Martins et al., 2009). We reversed the regular expressions given in the TurboParser[16] for the surface realisation

[15] https://github.com/roeeaharoni/morphological-reinflection/blob/master/src/hard_attention.py
[16] https://github.com/andre-martins/TurboParser/tree/master/python/tokenizer

task, and also added our own to tackle, for example, elision in French. C_{s2s} and C_{reg} modules were created for three languages: French, Italian, and Portuguese[17].

5.5 Evaluation on SR'18

In this section, we evaluate our model on SR'18 data. We did not participate in the task itself because we started our model development after the task had finished. In the next section, we will discuss our system participation in the SR'19 shared task.

We evaluate each component of our approach separately. We start by providing a detailed evaluation of how the model handles word ordering (Section 5.5.1). We then go on to analyse the respective contributions of morphological realisation (Section 5.5.2) and contraction generation (Section 5.5.3). Finally, we discuss the performance of the overall surface realisation model (Section 5.5.4). Throughout the evaluation, we used the SR'18 evaluation protocol and scripts[18] to compute automatic metrics: BLEU, DIST (see Section 4.3.3), and NIST (inverse normalised character-based string-edit distance).

5.5.1 Word Ordering

BLEU scores

We evaluate our word ordering component by computing the BLEU-4 score (Papineni et al., 2002) between the sequence of lemmas it produces and the lemmatized reference sentence extracted from the UD files. The baseline is the same model without delexicalisation. As Table 5.1 shows, there is a marked, statistically significant, difference between the baseline and our approach which indicates that delexicalisation does improve word ordering.

	ar	cs	en	es	fi	fr	it	nl	pt	ru
BL	29.6±1.39	48±0.7	53.57±0.15	46.5±0.78	27.2±0.4	46.4±0.5	49.07±0.9	36.6±0.7	44.3±0.26	58.1±0.46
WO	34.9±0.2	57.97±0.06	59.1±0.36	52.33±0.31	43.1±0.53	50.0±0.0	53.17±0.5	47.03±0.59	51.77±0.32	64.73±0.23
Δ	+5.3	+9.97	+5.53	+5.83	+15.9	+3.6	+4.1	+10.43	+7.47	+6.63

Table 5.1: Word Ordering: BLEU scores on lemmatised data. Mean and standard deviation across three random seeds. BL: Baseline. All pairwise comparisons of BL and our model showed a statistically significant difference in BLEU via the bootstrap resampling (1000 samples, $p < .05$).

Word Ordering Constraints

We also investigate the degree to which our results conform with the word ordering constraints of the various languages focusing on the following dependency relations: DET (determiner), NSUBJ (nominal subject), OBJ (object), AMOD (adjectival modifier) and ACL (nominal clausal modifier). We chose them because they are more frequent and present in almost all treebanks (other dependency relation results can be found in Appendix A). For each of these dependency relations, we compare the relative ordering of the corresponding (head, dependent) pairs in the reference data and in our system predictions.

[17] Although contractions are also present in Spanish, we did not develop a module for it, since the UD Spanish AnCora treebank does not split them on the token level in contrast to other UD treebanks.

[18] http://taln.upf.edu/pages/msr2018-ws/SRST.html#evaluation

To determine whether the dependent should precede or follow its head, we use the gold standard dependency tree of the UD treebanks. Since for the system predictions we do not have a parse tree, we additionally record the distance between head and dependent (in the reference data) and we compare it with the distance between the same two items in the system output. For instance, for the DET relation, given the gold sentence (6a) and the generated sentence (6b), we extract (6c) from the UD parse tree and (6d) from the predicted sentence where each triple is of the form either (dep, head, distance) or (head, dep, distance) and distance is the distance between head and dependent.

(6) a. GOLD: The yogi tried the advanced asana
b. PRED: The yogi tried the asana advanced
c. G-triples: (the$_{dep}$, yogi$_{head}$, 1), (the$_{dep}$, asana$_{head}$, 2)
d. P-triples: (the, yogi, 1), (the, asana, 1)
Exact match: 1; Approximate match: 2

We then compute exact matches (the order and the distance to the head is exactly the same) and approximate matches (the order is preserved but the distance differs by 1 token[19]). We call this exact match metric the *dependency edge accuracy* and reuse it in chapter 6. Table 5.2 shows the results and compares them with a non-delexicalised approach.

Global Score. The **all deprels** column summarises the scores for all dependency relations present in the treebanks (not just DET, NSUBJ, OBJ, AMOD and ACL). For the exact match, most languages score above average (from 0.51 to 0.71). That is the relative word order and the position of the dependent with respect to the head is correctly predicted in more than half of the cases. Approximate match yields higher scores with most languages scoring between 0.65 and 0.80 suggesting that a higher proportion of correct relative orderings is achieved (modulo mispositioning and false positives).

Long Range Dependencies. It is noticeable that for all languages, accuracy drops for the ACL relation. We conjecture that two factors makes it difficult for the model to make the correct prediction: heterogeneity and long range dependencies. As the ACL relation captures different types of clausal modifiers (finite and non-finite), it is harder for the model to learn the corresponding patterns. As the modifier is a clause, the distance between head (the nominal being modified) and dependent (the verb of the clause modifier) can be long which again is likely to impede learning.

Irregular Order. For cases where the head/dependent order is irregular, the scores are lower. For instance, in Dutch the object may occur either before (46.9% of the cases in the test data) or after the verb depending on whether it occurs in a subordinate or a main clause. Relatedly, the OBJ exact match score is the lowest (0.38) for this language. Similarly, in Romance languages where the adjective (AMOD relation) can either be pre- (head-final construction, HF) or post-posed (head-initial construction, HI), exact match scores are lower for this relation than for the others. For instance, the Portuguese test data contains 71% HF and 29% HI occurrences of the AMOD relation and correspondingly, the scores for that relation are much lower than for the DET, NSUBJ and OBJ relations for that language. A similar pattern can be observed for Spanish, French and Italian.

[19] We could of course consider further approximates matches differing by, say 2, 3 or 5 tokens. But we refrain from this as this would increase the number of false positives.

	det		*nsubj*		*obj*		*amod*		*acl*		***all deprels***	
	+1	+2	+1	+2	+1	+2	+1	+2	+1	+2	+1	+2
ar	0.36	0.37	0.45	0.52	0.35	0.48	0.52	0.59	0.32	0.41	0.38	0.47
Δ	+0.036	+0.047	-0.036	-0.067	-0.069	-0.113	-0.088	-0.106	-0.063	-0.061	-0.044	-0.071
cs	0.86	0.9	0.49	0.63	0.51	0.64	0.83	0.87	0.47	0.62	0.63	0.74
Δ	-0.077	-0.071	-0.041	-0.054	-0.048	-0.053	-0.134	-0.126	-0.071	-0.073	-0.068	-0.074
en	0.76	0.85	0.71	0.85	0.76	0.84	0.71	0.76	0.53	0.65	0.63	0.74
Δ	-0.10	-0.086	-0.002	-0.053	-0.074	-0.064	-0.15	-0.138	-0.058	-0.104	-0.06	-0.08
es	0.73	0.83	0.55	0.71	0.54	0.70	0.43	0.49	0.39	0.58	0.55	0.68
Δ	-0.02	-0.074	-0.012	-0.073	+0.004	-0.057	+0.059	+0.036	-0.056	-0.12	-0.038	-0.088
fi	0.71	0.81	0.64	0.75	0.46	0.60	0.70	0.76	0.50	0.61	0.51	0.65
Δ	-0.241	-0.244	-0.194	-0.189	-0.118	-0.137	-0.303	-0.296	-0.261	-0.312	-0.154	-0.16
fr	0.76	0.86	0.60	0.78	0.60	0.75	0.46	0.51	0.51	0.68	0.58	0.71
Δ	-0.016	-0.235	+0.014	-0.037	+0.031	-0.029	+0.117	+0.104	-0.1	0.17	-0.038	-0.093
it	0.73	0.82	0.59	0.70	0.58	0.73	0.40	0.46	0.52	0.65	0.56	0.69
Δ	-0.022	-0.058	-0.021	-0.07	-0.044	-0.101	+0.058	+0.017	-0.054	-0.072	-0.04	-0.088
nl	0.71	0.79	0.46	0.56	0.38	0.49	0.74	0.77	0.41	0.53	0.49	0.60
Δ	-0.121	-0.115	-0.068	-0.07	-0.068	-0.087	-0.278	-0.29	-0.22	-0.277	-0.118	-0.13
pt	0.74	0.80	0.56	0.72	0.57	0.73	0.42	0.44	0.48	0.63	0.54	0.67
Δ	-0.032	-0.051	-0.072	-0.114	-0.041	-0.094	+0.034	+0.043	-0.122	-0.204	-0.068	-0.103
ru	NA	NA	0.65	0.79	0.65	0.74	0.79	0.85	0.45	0.67	0.71	0.80
Δ	—	—	-0.033	-0.057	-0.037	-0.033	-0.095	-0.1	+0.022	-0.027	-0.05	-0.059

Table 5.2: Proportion of correct head/dependent positioning for the five selected dependency relations: *det*, *nsubj*, *obj*, *amod*, *acl*, and overall performance across all dependency relations. +1: exact match; +2: approximate match, i.e. head and dependent are in the correct order but there is a one-token difference between gold and prediction. NA: no dependency relation found in a treebank. Δ indicates the difference between our delexicalised model and the baseline.

More detailed statistics, including other relations and performance with respect to head-directionality, can be found in Appendix A.

Non-delexicalised Baseline. We also compare our delexicalised model with the non-delexicalised baseline: Δ in Table 5.2 shows the difference in performance between the two models.

Overall, the scores favour the delexicalised approach (negative *delta* in the **all deprels** column for all languages) supporting the results given by the automatic metric. However, for some dependency relations, the lexicalised baseline shows usefulness of word information, for example, while predicting AMOD relations for Romance languages (positive *delta* for French, Italian, Spanish, and Portuguese). Indeed, preposed adjectives in those languages constitute a limited lexical group.

Comparison across Languages. Overall scores vary between languages but given the differences in annotation consistency in UD, number of dependency relations (the number of distinct dependency relations present in the treebank ranges between 29 (ar, es) and 44 (en)), and frequency counts for each dependency relations, it is difficult to conclude anything from these differences.

5.5.2 Morphological Realisation

Table 5.3 shows the results for the WO+MR model.

	ar	cs	en	es	fi	fr	it	nl	pt	ru
MR Accuracy	91.05	98.89	98.3	99.07	92.66	96.77	96.85	87.6	98.80	98.23
WO	34.9±0.2	57.97±0.06	59.1±0.36	52.33±0.31	43.1±0.53	50.0±0.0	53.17±0.5	47.03±0.59	51.77±0.32	64.73±0.23
WO+MR (S^{-c})	28.6±0.26	56.1±0.1	54.3±0.3	51.4±0.3	38.63±0.59	44.97±0.25	47.3±0.6	42.3±0.53	50.7±0.26	60.93±0.23
Δ	-6.3	-1.87	-4.8	-0.93	-4.47	-5.03	-5.87	-4.73	-1.07	-3.8

Table 5.3: Morphological Realisation Results. MR Accuracy: accuracy of the MR module. WO: BLEU scores on lemmas. WO+MR: BLEU scores on inflected tokens without contraction (S^{-c}).

The top line (MR Accuracy) indicates the accuracy of the MR model on the SR'18 test data which is computed by comparing its output with gold word forms. As the table shows, the accuracy is very high overall ranging from 87.6 to 99.07, with 9 of the 10 languages having an accuracy above 90. This confirms the high accuracy of the model when performing morphological inflection out of context.

The third line (WO+MR (S^{-c})) shows the BLEU scores for our WO+MR model, i.e., when the MR model is applied to the output of the WO model. Here we use an oracle setting which ignores contractions. That is, we compare the WO+MR output not with the final sentence but with the sentence before contraction applies (the ability to handle contractions is investigated in the next section).

As the table shows, the delta in BLEU scores between the model with (WO+MR) and without (WO) morphological realisation mirrors the accuracy of the morphological realisation model: as the accuracy of the morphological inflection model decreases, the delta increases. For instance, for Arabic, the MR accuracy is among the lowest (91.05) and, correspondingly, the decrease in BLEU score when going from word ordering to word ordering with morphological realisation is the largest (-6.3).

5.5.3 Contraction Generation

	ar	cs	en	es	fi	fr	it	nl	pt	ru
S^{-c}/S^{+c}	47.1	96.1	90.9	98.4	98.5	70.8	65.1	99.5	66.3	96.9
WO+MR (S^{-c})	28.6±0.26	56.1±0.1	54.3±0.3	51.4±0.3	38.63±0.59	44.97±0.25	47.3±0.6	42.3±0.53	50.7±0.26	60.93±0.23
WO+MR (S^{+c})	15.8±0.1	54.83±0.21	53.3±0.53	51.03±0.31	38.37±0.55	36.23±0.42	31.5±0.4	42.27±0.55	34.0±0.62	60.43±0.15
Δ	-12.8	-1.27	-1.0	-0.37	-0.26	-8.74	-15.8	-0.03	-16.7	-0.5
WO+MR+C_{reg} (S^{+c})						41.8±0.26	40.0±0.66		46.13±0.29	
WO+MR+C_{s2s} (S^{+c})						40.33±0.36	39.93±0.17		44.57±0.64	

Table 5.4: Contraction Generation Results (BLEU scores). S^{-c}/S^{+c}: a sentence without contractions vs. a reference sentence including contractions; S^{-c}: BLEU with respect to sentences before contractions; S^{+c}: BLEU with respect to a reference sentence. The scores were computed on detokenised sequences.

To assess the degree to which contractions are used, we compute BLEU-4 between the gold sequence of word forms from UD treebanks and the reference sentence (Table 5.4, Line S^{-c}/S^{+c}). As the table shows, this BLEU score is very low for some languages (Arabic, French, Italian, Portuguese) indicating a high level of contractions.

These differences are reflected in the results of our WO+MR model: the higher the level of contractions, the stronger the delta between the BLEU score on the reference sentence without contractions (WO+MR, S^{-c}) and the reference sentence with contractions (WO+MR, S^{+c}).

This shows the limits of out-of-context morphological realisation. While the model is good at producing a word form given its lemma and a set of morpho-syntactic features, the lack of contextual information means that contractions cannot be handled.

Adding a contraction module permits improving results for those languages where contraction is frequent (Table 5.4, Lines WO+MR+C_{reg}, WO+MR+C_{s2s}). Gains range from +5 points for French to +12 for Portuguese when comparing to WO+MR. We achieved better results with contraction module based on regular expressions (C_{reg}), rather than a neural module (C_{s2s}). In a relatively simple task, such as contraction generation, rule-based methods are more reliable, and, overall, are preferable due to their robustness and easy repair comparing to neural models, which may, for instance, hallucinate incorrect content.

5.5.4 Global Evaluation

Finally, we compare our approach with the best results obtained by the SR'18 participants and with OSU's results (King and White, 2018) using BLEU-4, DIST and NIST scores. OSU results are treated separately, since some of their scores were published after the shared task had ended. Table 5.5 shows the results. They are mixed. Our model yields the best results for Czech (BLEU: +1.63), Finnish (BLEU: +0.87), Dutch (BLEU: +9.99) and Russian (BLEU: +2.53). However it underperforms on Arabic (BLEU: -9.8), English (BLEU: -13), Spanish (BLEU: -14.27), French (BLEU: -10.23), Italian (BLEU: -4.46) and Portuguese (BLEU: -1.47). Based on the evaluation of each of our modules, these results can be explained as follows.

The languages for which our model outperforms the state of the art are languages for which the WO model performs best, the accuracy of the morphological realiser is high and the level of contractions is low. For those languages, improving the accuracy of the word ordering model would further improve results.

For four of the languages where the model underperforms (namely, Arabic, French, Portuguese and Italian), the level of contraction is high. This indicates that improvements can be

	ar	cs	en	es	fi	fr	it	nl	pt	ru
BLEU										
SR'18	16.2	25.05	55.29	49.47	23.26	**52.03**	**44.46**	32.28	30.82	34.34
OSU	**25.6**	53.2	**66.30**	**65.30**	37.5	38.2	42.1	25.5	**47.6**	57.9
Ours	15.8±0.1	**54.83±0.21**	53.3±0.53	51.03±0.31	**38.37±0.55**	41.8±0.26	40.0±0.66	**42.27±0.55**	46.13±0.29	**60.43±0.15**
DIST										
SR'18	44.37	36.48	79.29	51.73	41.21	55.54	58.61	57.81	60.7	34.56
OSU	46.7	58.1	70.2	61.5	58.7	53.7	59.7	57.8	66.0	59.9
Ours	27.63±0.06	63.53±0.15	62.77±0.15	61.33±0.15	51.83±0.55	55.23±0.72	53.73±0.21	54.13±0.15	57.0±0.4	71.23±0.12
NIST										
SR'18	7.15	10.74	10.86	11.12	9.36	9.85	9.11	8.64	7.55	13.06
OSU	7.15	13.5	12.0	12.7	9.56	8.00	8.70	7.33	9.13	14.2
Ours	6.04±0.02	13.6±0.05	10.95±0.09	11.63±0.02	10.41±0.03	8.93±0.08	8.99±0.01	9.51±0.03	9.44±0.04	13.96±0.03

Table 5.5: BLEU, DIST and NIST scores on the SR'18 test data (shallow track). SR'18 is the official results of the shared task but do not include OSU scores, since they are given in the line below. We also excluded the ADAPT and NILC scores as they were obtained using data augmentation. OSU is the submission of King and White (2018).

gained by improving the handling of contractions, e.g., by learning a joint model that would take into account both morphological inflection and contraction.

5.6 Participation in SR'19

5.6.1 Model Adaptation

Data in SR'19 has different languages and some additional features comparing to SR'18, so below we describe some system amendments that were done.

Word ordering is modelled as a sequence-to-sequence task, where an input tree is linearised. Linearisation differs from our previous approach in that it was augmented with information about the relative order of some elements, a feature that was introduced for the 2019 edition of the shared task. So nodes were linearised using the depth-first search, and then elements with the relative order feature were reordered to match the added information.

Morphological paradigms were learned from pairs of (lemma, POS+features) extracted from the training data (the `upos` and `features` fields from CoNLL) using Aharoni and Goldberg (2017)'s model. Lemmas with no morphological features were not used for training. Since features are not provided for Chinese, Japanese, and Korean treebanks, the morphological realisation module was not trained for those languages. Instead, during the inflection phase:

- for Chinese, analytic language, lemmas were copied verbatim to the output;
- for Korean, agglutinative language, morphemes in a lemma were glued together, and then the lemma was copied;
- for Japanese, synthetic language, a dictionary of the form (lemma+POS: wordform) was constructed from the training data and looked up.

If a key 'lemma+POS' was not present in the dictionary, the lemma was copied to the output verbatim. The same rule applies for any other lemma with no morphological features in any treebank (e.g., URLs, foreign words, numbers, punctuation signs, etc.)[20].

[20] We deleted features for foreign words in ru_gsd_ud for it to be consistent with ru_syntagrus_ud.

In the following, we will refer to the MR component as including the contraction generation module (CG) as well.

Eventually, one may also include detokenisation, a task of glueing tokens together, in this last step, as each language requires specific detokenisation rules to produce a final well-formed sentence, which can be shown to an end user. We used the `sacremoses`[21] library to perform detokenisation. Besides, it was also used to tokenise reference sentences; we need that for the automatic scoring.

5.6.2 Results and Discussion

We evaluate each module separately. For WO, we compared a generated sequence of lemmas with a gold sequence of lemmas extracted from UD. For MR, we calculated wordform prediction accuracy, and also applied MR to a gold sequence of lemmas instead of predicted sequence of lemmas. Finally, we performed the overall evaluation, where our system predictions were compared to reference sentences[22].

WO Evaluation

Table 5.8 shows the results of WO. BLEU scores vary from 30 to 66 depending on the language and corpus ($mean = 56.98, median = 60.01$).

We surmised that low scores for Arabic, Chinese, Indonesian are due to small sizes of training corpora (6K, 4K, 4.5K training instances, respectively), which are not enough for neural systems. Other languages' scores show a smaller variation, ranging from 51 to 66; we conjecture that the variations between languages are due to different syntactic phenomena occurring in each language and the variations between corpora are due to different annotation guidelines.

MR+CG Evaluation

The inflection module was initially measured by accuracy of producing a correct word form given a lemma and its POS together with morphological features (see Table 5.7, second column). The average accuracy is 96.14 across 8 languages, which corresponds to the state-of-the-art results in inflection tasks (Cotterell et al., 2016).

We also calculated a number of lemmas, which can have different word forms, given the same set of POS and morphological features (Table 5.7, third column). For example, the lemma *people* with `pos=NOUN, Number=Plur` as features have two word forms in the training data: *people* and *peoples*. Those ambiguous forms may stem from different sources: language variation (as in the example above) including spelling, non-standard forms and typos; annotation mistakes; underspecified morphological features. The example of the latter is an adjective in Russian, which can have different forms in the accusative case depending on animacy of the noun it modifies (animacy in that case is an underspecified feature).

To measure the effect on scores, when converting a sequence of lemmas into a sentence, we applied MR+CG to gold sequences of lemmas (they have the same word order as the reference). Results are shown in Table 5.6. In general, high accuracies of MR alone (word level, Table 5.7) do not guarantee good performance while evaluating on the sentence level. That type of evaluation enabled us to have more insight into the data used. Some of our findings are listed below.

[21] https://github.com/alvations/sacremoses

[22] After the official submission to the shared task, we fixed a bug in MR for Japanese and Korean. In this chapter we are reporting improved results.

Id	Corpus	BLEU	DIST	NIST
ar	ar_padt-ud	40.07	47.23	8.25
en1	en_ewt-ud	80.88	80.22	12.90
en2	en_gum-ud	90.73	98.74	12.80
en3	en_lines-ud	86.78	97.03	12.74
en4	en_partut-ud	86.31	96.46	10.23
es1	es_ancora-ud	93.82	98.47	14.88
es2	es_gsd-ud	89.31	99.23	13.93
fr1	fr_gsd-ud	90.53	98.12	14.04
fr2	fr_partut-ud	87.07	96.61	9.82
fr3	fr_sequoia-ud	91.00	96.38	12.38
hi	hi_hdtb-ud	91.88	96.89	13.67
id	id_gsd-ud	94.46	98.91	12.90
ja	ja_gsd-ud	77.85	99.70	11.40
ko1	ko_gsd-ud	60.38	94.38	9.45
ko2	ko_kaist-ud	97.13	99.67	13.44
pt1	pt_bosque-ud	94.09	99.10	12.68
pt2	pt_gsd-ud	57.04	91.12	10.54
ru1	ru_gsd-ud	86.87	96.89	12.18
ru2	ru_syntagrus-ud	91.25	98.15	15.53
zh	zh_gsd-ud	99.16	99.81	13.33

Table 5.6: The MR module applied to the gold word ordering input. Predictions and reference sentences are both tokenised. Results on the development set.

lang	Acc.	Amb. %	Amb. count
ar	90.87	7.29	1,815
en	96.35	0.84	226
es	98.85	0.85	418
fr	98.40	1.48	430
hi	89.95	6.46	569
id	98.52	0.55	47
ja	NA	3.62	800
ko	NA	0.86	945
pt	98.95	0.85	233
ru	97.25	0.72	933
zh	NA	0	0

Table 5.7: Accuracy of the morphological realisation component. NA: no MR component was developed. Percentage and count of lemmas with ambiguous forms found in the training data.

- English: a discrepancy in performance across datasets. The sources of the en_ewt-ud corpus are blogs, social networks, reviews, emails, where the use of contractions (*isn't*, *ain't*, etc) is dominant comparing to formal style. Since the contraction generation was not applied for English, scores for this particular dataset are lower than for others.
- Arabic. We conjecture low scores for the high variability of forms (see Table 5.7) and contractions (we did not develop a module for handling contractions in Arabic). For instance, some diacritics are optional (e.g., hamza with alif), so a word form can be written with or without them, being a valid word form in both cases.
- Japanese. MR module was not developed for Japanese, so a look-up dictionary based on training data was not sufficient to handle the morphology. The high number of ambiguous forms also impacted the scores, as in the case of Arabic.
- Portuguese. The pt_gsd-ud corpus is not annotated with morphological features, hence 57.04 score in BLEU compared to 94.09 in pt_bosque-ud.
- Korean. We do not read Korean, so we were not able to explain the difference between the two Korean corpora (97.13 vs. 60.38 BLEU). Some annotation disparity may well be the explanation.

Surface Realisation Evaluation

The performance of the overall surface realisation model is shown in Table 5.9. Automatic scores show a drop compared to the WO component performance (Table 5.8), which is consistent with the errors of the MR+CG module, described above.

Figure 5.2 aggregates the BLEU scores, shown in Tables 5.6, 5.8, 5.9. For each corpus, BLEU for each module (X-axis) is mapped to the final BLEU score (Y-axis). The scatterplots show a strong, positive association between the two variables: Pearson's $r = 0.83$ and $r = 0.86$ for WO and MR on gold lemmas respectively. The plots render the impact of both modules more visible: in the bottom left corner are languages with low scores, and in the top there are languages with higher scores. Going from the X-axis to the Y-axis allows to see the drop in scores from a single module to the final pipeline. For instance, the Japanese treebank (ja, blue point) loses around 10 points (from 56 to 46) when going from the WO evaluation to the final evaluation. On the es1 corpus, the model shows good results on MR (orange point, 93 on the abscissa) and the MR part does not almost influence the total performance: BLEU drops from 59 to 57 (blue point).

Finally, we briefly summarise our results in the SR'19 shared task based on the organisers' report (see Mille et al. (2019) for detailed results). Out of 11 teams participating in the shallow track, only four submitted for all treebanks (including our submission), which makes the comparison not based on the equal terms. A system of Yu et al. (2019a), one of those teams covering all languages, outperformed all other approaches by a large margin in automatic evaluation. Our submission performed on a par with the other two systems that developed for all the languages. As for human evaluation (carried out for several languages), we ranked fourth and second for the Meaning similarity criterion for English and Russian respectively, and sixth for Spanish and Chinese. For Readability, our submission for Chinese showed better results being on the third place, while other languages kept their ratings. Low human scores for Chinese and Spanish in our case correspond to what we have seen in the automatic evaluation on the development set outlined above. During test time, we also ran our system on out-of-domain and machine-generated data. For all languages concerned, automatic scores remained stable, which demonstrates the portability of our approach.

Corpus	BLEU	DIST	NIST
ar_padt-ud	30.45	54.72	8.86
en_ewt-ud	66.71	84.18	12.57
en_gum-ud	62.92	80.61	11.53
en_lines-ud	61.89	75.76	11.67
en_partut-ud	62.38	75.87	9.82
es_ancora-ud	59.43	75.03	12.69
es_gsd-ud	61.83	74.94	12.80
fr_gsd-ud	60.58	78.66	12.74
fr_partut-ud	61.24	82.37	9.35
fr_sequoia-ud	55.22	74.18	10.86
hi_hdtb-ud	63.07	59.87	11.74
id_gsd-ud	46.09	76.07	9.99
ja_gsd-ud	56.53	62.41	10.33
ko_gsd-ud	53.73	53.01	11.69
ko_kaist-ud	66.43	63.19	12.85
pt_bosque-ud	52.88	81.98	11.13
pt_gsd-ud	51.01	72.59	11.82
ru_gsd-ud	59.11	62.30	11.72
ru_syntagrus-ud	62.37	67.88	14.03
zh_gsd-ud	45.67	56.01	10.23

Table 5.8: WO component performance on the development set. Predictions and references (sequences of lemmas) are both tokenised.

Corpus	BLEU	DIST	NIST
ar_padt-ud	18.06	43.86	6.49
en_ewt-ud	54.45	65.45	11.32
en_gum-ud	58.17	79.68	11.16
en_lines-ud	52.53	73.48	10.79
en_partut-ud	54.79	73.98	9.10
es_ancora-ud	56.99	73.78	12.53
es_gsd-ud	59.63	74.07	12.32
fr_gsd-ud	51.94	70.43	11.63
fr_partut-ud	51.72	74.74	8.47
fr_sequoia-ud	49.08	70.27	10.13
hi_hdtb-ud	58.48	61.13	11.42
id_gsd-ud	45.28	75.50	9.88
ja_gsd-ud	46.30	62.31	9.37
ko_gsd-ud	32.21	49.43	8.73
ko_kaist-ud	64.58	61.21	12.69
pt_bosque-ud	59.35	83.84	11.20
pt_gsd-ud	35.44	69.47	9.17
ru_gsd-ud	52.90	60.59	11.13
ru_syntagrus-ud	57.97	66.87	13.70
zh_gsd-ud	45.48	55.91	10.21

Table 5.9: Automatic metrics on the development set (WO + MR). Predictions and reference sentences are both tokenised.

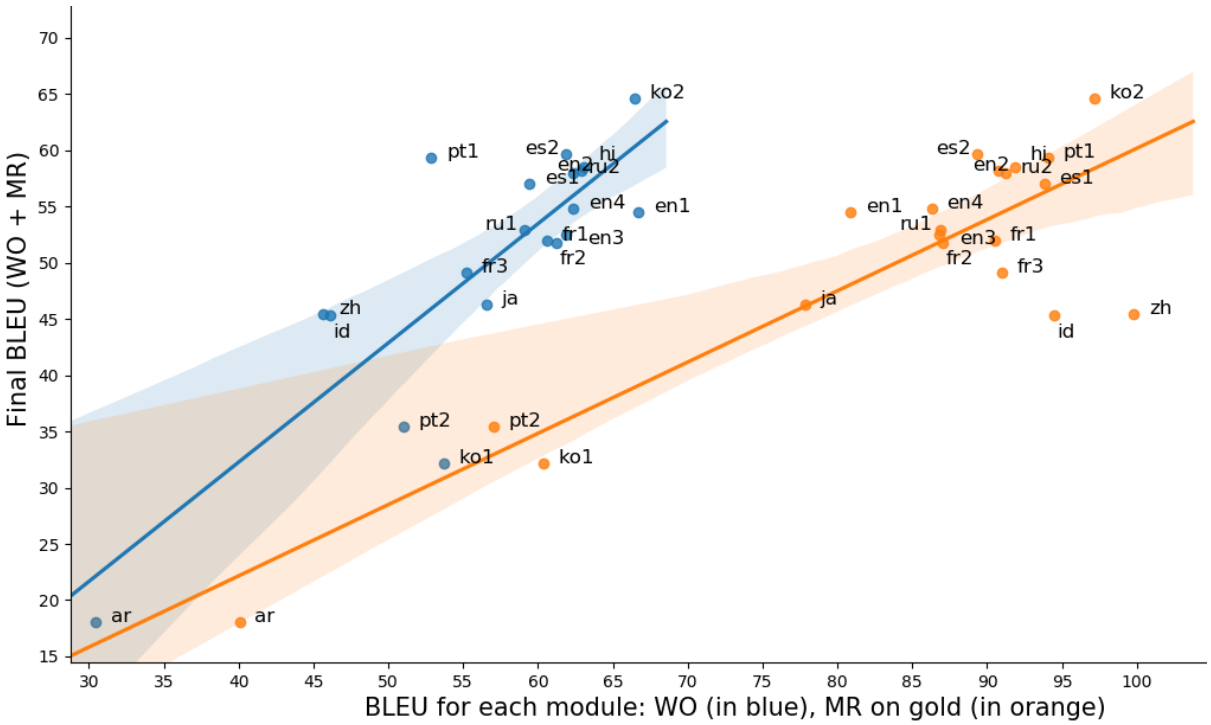


Figure 5.2: Linear regression between BLEU scores for each module and final BLEU scores. Data points are treebanks (ids are from Table 5.6). In orange: MR on gold lemma vs. final BLEU (WO + MR); in blue: WO vs. final BLEU (WO + MR).

5.7 Conclusion

While surface realisation is a key component of the NLG pipeline, most work in this domain has focused on the development of language specific models. By providing multilingual training and test set, the SR shared tasks opens up the possibility to investigate how language specific properties such as word order and morphological variation impact performance.

In this chapter, we presented a modular approach to surface realisation and applied it to all the languages of the SR shallow track in 2018 and 2019.

For word ordering, we proposed a simple approach where the data is delexicalised, the input tree is linearised using depth-first search and the mapping between input tree and output lemma sequence is learned using a factored sequence-to-sequence model. Experimental results show that full delexicalisation markedly improves performance. Linguistically, this confirms the intuition that the mapping between shallow dependency structure and word order can be learned independently of the specific words involved.

We further carried out a detailed evaluation of how our word ordering model performs on the ten languages of the SR'18 shallow track. While differences in annotation consistency, number of dependency relations and frequency counts for each dependency relations in each dataset make it difficult to conclude anything from the differences in overall scores between languages, the evaluation of head/dependent word ordering constraints highlighted the fact that long-distance relations, such as ACL, and irregular word ordering constraints (e.g., the position of the verb in Dutch main and subordinate clauses) negatively impact results.

For morphological realisation and contractions, we showed that applying morphological realisation out of context, as done by most of the SR participating systems, yields poor results for those languages (Portuguese, French, Arabic, Italian) where contractions are frequent. We explored two ways of handling contractions (a neural sequence-to-sequence model and a rule-based model) and showed that adding contraction handling strongly improves performance (from +5.57 to 12.13 increase in BLEU score for the rule-based model depending on the language). More generally, our work on contractions points to the need for SR models to better take into account the fine-grained structure of words.

By participating in SR'19, we were able to see how much effort is needed to adapt our approach to new data. The WO component is easily transferrable between languages, and it does not require much effort for applying it to unseen languages. In contrast, the MR component requires a lot of attention, and needs to be tuned for each language separately. That is mainly due to the different approaches for language annotation across UD treebanks, and, what is more unexpected, across UD treebanks for the same language, not to speak of the contraction generation and detokenisation, which are different for each language, and which should also be implemented separately.

We also would like to highlight the importance of modular evaluation. If a system design allows it, system outputs may be tested against a sequence of lemmas, not only a reference sentence, thanks to the UD annotations. We encourage future researchers not to neglect this type of evaluation to gain deeper insight into their system and data.

Given that most current models for SR split the task into two or more steps, for future work it would be interesting to explore the development of a joint model that simultaneously handles morphological realisation and word ordering while using finer grained word representations, such as fastText embeddings (Bojanowski et al., 2017) or byte pair encoding (BPE; Gage, 1994; Sennrich et al., 2016).

Speaking of the components separately, we believe that MR (including contraction generation and possibly detokenisation) would benefit from including context information, i.e. doing

inflection and necessary character transformations on a whole sentence, rather than word by word. As for word ordering, it remains a tough problem for sequence-to-sequence architectures, and it is worth exploring direct ways of encoding tree structure, as in tree- or graph-based approaches (Marcheggiani and Perez-Beltrachini, 2018; Song et al., 2018b; Beck et al., 2018; Ribeiro et al., 2019; Yu et al., 2019a). Another perspective for future development is to learn several models for word ordering: one delexicalised and another lexicalised to profit from both types of information and then use a combination of the models.

6

Evaluating Surface Realisers

Contents

6.1 Introduction **65**
6.2 Related Work **66**
6.3 Framework for Error Analysis **67**
6.3.1 Syntactic Complexity Metrics 67
6.3.2 Performance Metrics 69
6.3.3 Correlation Tests 70
6.3.4 Error Mining 70
6.4 Data and Experimental Setting **70**
6.5 Error Analysis **72**
6.5.1 Tree-Based Syntactic Complexity 72
6.5.2 Projectivity 74
6.5.3 Entropy 75
6.5.4 Which Syntactic Constructions Are Harder to Handle? 78
6.6 Using Error Analysis for Improving Models or Datasets **81**
6.7 Conclusion **83**

6.1 Introduction

In Chapter 5, we introduced surface realisation evaluation by means of dependency relations, without which we would have had to *realise* that we had barely scratched the *surface*. In this chapter, we aim to deepen our evaluation approach and extend it to a full evaluation framework by considering all system outputs submitted to two multilingual surface realisation tasks.

As discussed in Sections 5.1 and 5.2, surface realisation (SR) is a natural language generation task which consists in converting a linguistic representation into a well-formed sentence. As the use of multiple input formats made the comparison and evaluation of existing surface realisers difficult, Belz et al. (2011); Mille et al. (2018a, 2019) organised the SR shared tasks which provide two standardised input formats for surface realisers: deep and shallow dependency trees.

While the SR tasks provide a common benchmark on which to evaluate and compare SR systems, the evaluation protocol they use (automatic metrics and human evaluation) does not support a detailed error analysis. Metrics (BLEU, DIST, NIST, METEOR, TER) and human

assessments are reported on the system level, and so do not provide a detailed feedback for each participant. Neither do they give information about which syntactic phenomena impact performance.

In this chapter, we discuss a framework for error analysis which allows for an interpretable, linguistically informed analysis of SR results. While shallow surface realisation involves both determining word order (linearisation) and inflecting lemmas (morphological realisation), since inflection error detection is already covered in morphological shared tasks (Cotterell et al., 2017; Gorman et al., 2019), we focus on error analysis for word order.

Error analysis is key to understanding the performance of NLP models and to improve them. While doing error analysis, two principles are crucial: (i) analysing a large set of instances, rather than a sample (elsewise error analysis may lead to biased conclusions about the model performance); (ii) classifying errors by groups to account for a model general weaknesses and strengths. For SR models, a natural basis for error grouping is characteristics of dependency trees. Motivated by extensive linguistic studies which deal with syntactic dependencies and their relation to cognitive language processing (Liu, 2008; Futrell et al., 2015; Kahane et al., 2017), we investigate word ordering performance in SR models given various tree-based metrics.

The proposed error analysis framework consists of two main components: (i) correlation analyses between an extensive range of tree-based metrics designed to measure syntactic complexity and standard performance metrics and (ii) techniques (dependency edge accuracy, error mining) which permit automatically identifying which syntactic constructions often co-occur with low performance scores.

We apply our framework to the outputs of 174 systems submitted to the SR shared tasks and demonstrate how it can be used to highlight some global results about the state of the art.

We also indicate various ways in which our error analysis framework could be used to improve a model or a dataset thereby arguing for approaches to model and dataset improvement that are more linguistically guided.

This chapter is based on the following publication:

Shimorina, A., Parmentier, Y., and Gardent, C. (2021). An Error Analysis Framework for Shallow Surface Realization. *Transactions of the Association for Computational Linguistics*, 9:429–446

Our framework is available[23] in the form of a toolkit which can be used both by campaign organisers to provide a detailed feedback on the state of the art for surface realisation and by researchers to better analyse, interpret and improve their models.

6.2 Related Work

There has been a long tradition in NLP exploring syntactic and semantic evaluation measures based on linguistic structures (Liu and Gildea, 2005; Mehay and Brew, 2007; Giménez and Màrquez, 2009; Tratz and Hovy, 2009; Lo et al., 2012). In particular, dependency-based automatic metrics have been developed for summarisation (Hovy et al., 2005; Katragadda, 2009; Owczarzak, 2009) and machine translation (Owczarzak et al., 2007; Yu et al., 2014). Relations between metrics were also studied: Dang and Owczarzak (2008) found that automatic metrics perform on a par with the dependency-based metric of Hovy et al. (2005) while evaluating summaries. The closest research to ours, which focused on evaluating how dependency-based metrics correlate with human ratings, is Cahill (2009) who showed that syntactic-based metrics

[23] https://gitlab.com/shimorina/tacl-2021

perform equally well as compared to automatic metrics in terms of their correlation with human judgements for a German surface realiser.

Researchers, working on SR and word ordering, have been resorting to different metrics to report their models' performance. Zhang et al. (2012); Zhang (2013); Zhang and Clark (2015); Puduppully et al. (2016); Song et al. (2018a) used BLEU; Schmaltz et al. (2016) parsed their outputs and calculated the UAS parsing metric; Filippova and Strube (2009) used Kendall correlation together with edit-distance to account for English word order. Similarly, Dyer (2019) used Spearman correlation between produced and gold word order for a dozen of languages. White and Rajkumar (2012), in their CCG-based realisation, calculated average dependency lengths between grammar-generated sentences and gold standard. Gardent and Narayan (2012); Narayan and Gardent (2012) proposed an error mining algorithm for generation grammars to identify the most likely sources of failures, when generating from dependency trees. Their algorithm mines suspicious subtrees in a dependency tree, which are likely to cause errors. King and White (2018) drew attention to their model performance for non-projective sentences. Puzikov et al. (2019) assessed their binary classifier for word ordering using the accuracy of predicting the position of a dependent with respect to its head, and a sibling. Yu et al. (2019a) showed that, for their system, error rates correlate with word order freedom, and reported linearisation error rates for some frequent dependency types. In a similar manner, in chapter 5 we looked at our system performance in terms of dependency relations, which shed light on the differences between the non-delexicalised and delexicalised models.

In sum, multiple metrics and tools have been developed by individual researchers to evaluate and interpret their model results: dependency-based metrics, correlation between these metrics and human ratings, performance on projective vs. non-projective input, linearisation error rate, etc. At a more global level, however, automatic metrics and human evaluation continue to be massively used.

Here, we gather a set of linguistically informed, interpretable metrics and tools within a unified framework, apply this framework to the results of two evaluation campaigns (174 participant submissions) and generally argue for a more interpretable evaluation approach for surface realisers.

6.3 Framework for Error Analysis

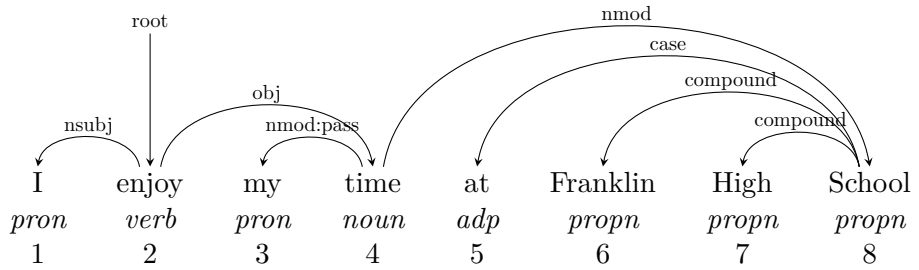
Our error analysis framework gathers a set of performance metrics together with a wide range of tree-based metrics designed to measure the syntactic complexity of the sentence to be generated. We apply correlation tests between these two types of metrics and mine a model output to automatically identify the syntactic constructs which often co-occur with low performance scores.

6.3.1 Syntactic Complexity Metrics

To measure syntactic complexity, we use several metrics commonly used for dependency trees (**tree depth and length, mean dependency distance**) as well as the ratio, in a test set, of sentences with **non-projective structures**.

We also consider the entropy of the dependency relations and a set of metrics based on "flux" recently proposed by Kahane et al. (2017).

Flux. The flux is defined for each inter-word position (e.g., 5-6 in Figure 6.1). Given the inter-word position (i, j), the flux of (i, j) is the set of edges (d, k, l) such that d is a dependency



System output (lemmatised): *I enjoy my time at High Franklin School*
BLEU= 0.65; dep edge accuracy = 0.71

System output (final): *I enjoyed my time at High Franklin School*
Fluency= 1; Adequacy = 0.7

Figure 6.1: A reference UD dependency tree (nodes are lemmas) and a possible SR model output. The final output is used to compute human judgments and the lemmatised output to compute BLEU and dependency edge accuracy (both are given without punctuation).

relation, $k \leq i$ and $j \leq l$. For example, in Figure 6.1 the flux for the inter-word position between the nodes 5 and 6 is {*(nmod, 4, 8), (case, 5, 8)*} and {*(nmod, 4, 8), (case, 5, 8), (compound, 6, 8), (compound, 7, 8)*} for the position between the nodes 7 and 8.

The **flux size** is its cardinality, i.e. the number of edges it contains: 2 for 5-6 and 4 for 7-8.

The **flux weight** is the size of the largest disjoint subset of edges in the flux (Kahane et al., 2017, p. 74). A set of edges is disjoint if the edges it contains do not share any node. For instance, in the inter-word position 5-6, *nmod* and *case* share a common node 8, so the flux weight is 1 (i.e., it was impossible to find two disjoint edges). The idea behind the flux-based metrics was to try accounting for cognitive complexity of syntactic structures, in the same fashion as in Miller (1956) who showed a processing limitation of syntactic constituents in a spoken language.

For each reference dependency tree, we calculate the metrics listed in Table 6.1. These can then be averaged over different dimensions (e.g., all runs, all runs of a given participant, runs on a given corpus, language etc.). Table 6.2 shows the statistics obtained for each corpus used in the SR shared tasks. We refer the reader to the Universal Dependencies project[24] to know more about differences between specific treebanks.

Dependency Relation Entropy. Entropy has been used in typological studies to quantify word order freedom across languages (Liu, 2010; Futrell et al., 2015; Gulordava and Merlo, 2016). It gives an estimate of how regular or irregular a dependency relation is with respect to word order. A relation d with high entropy indicates that d-dependents sometimes occur to the left and sometimes to the right of their head, i.e. their order is not fixed.

The entropy H of a dependency relation d is calculated as

$$H(d) = -p(L) \times \log_2(p(L)) - p(R) \times \log_2(p(R))$$

where $p(L)$ is the probability for a dependent to be on the left from the head, and $p(R)$ is the probability for a dependent to be on the right from the head. For instance, if the dependency

[24] https://universaldependencies.org/

Syntactic Complexity	Explanation
tree depth	the depth of the deepest node {3}
tree length	number of nodes {8}
mean dependency distance	average distance between a head and a dependent. For a dependency linking two adjacent nodes, the distance is equal to one (e.g., *nsubj* in Figure 6.1). $\{(1+2+1+4+1+2+3)/7 = 2\}$
mean flux size	average flux size of each inter-word position $\{(1+1+2+1+2+3+4)/7 = 2\}$
mean flux weight	average flux weight of each inter-word position $\{(1+1+1+1+1+1+1)/7 = 1\}$
mean arity	average number of direct dependents of a node $\{(0+2+0+2+0+0+0+3)/8 = 0.875\}$
projectivity	True if the sentence has a projective parse tree (there are no crossing dependency edges and/or projection lines). {*True*}

Table 6.1: Metrics for Syntactic Complexity of a sentence (the values in braces indicate the corresponding value for the tree in Figure 6.1).

relation *amod* is found to be head-final 20 times in a treebank, and head-initial 80 times, its entropy is equal to 0.72. Entropy ranges from 0 to 1: values close to zero indicate low word order freedom; values close to one mark high variation in head directionality.

6.3.2 Performance Metrics

Performance is assessed using sentence-level BLEU-4, dependency edge accuracy (DEA) and human evaluation scores.

Dependency edge accuracy (DEA). Introduced in Section 5.5.1, DEA measures how many edges from a reference tree can be found in a system output, given the gold lemmas and dependency distance as markers. An edge is represented as a triple (head lemma, dependent lemma, distance), e.g., (I, enjoy, -1) or (time, school, +4) in Figure 6.1[25]. In the output, the same triples can be found based on the lemmas, the direction (after or before the head), and the dependency distance. In our example, two out of the seven dependency relations cannot be found in the output: (school, high, -1) and (school, franklin, -2). Thus, dependency edge accuracy is 0.71 (5/7).

Human evaluation scores. The framework includes sentence-level z-scores for Adequacy and Fluency[26] reported in the SR'18 and SR'19 shared tasks. The z-scores were calculated on the set of all raw scores by the given annotator using each annotator's mean and standard deviation. Note that those were available for a sample of test instances for some languages only and were calculated using the final system outputs, rather than the lemmatised ones.

[25]We report signed values for dependency distance, rather than absolute ones, to account for the dependent position—after or before the head.

[26]In the original papers called *Meaning Similarity* and *Readibility* respectively (Mille et al., 2018a, 2019).

6.3.3 Correlation Tests

The majority of our metrics are numerical, which allows us to measure dependence between them using correlation. One of the metrics—projectivity—is nominal, so we apply a non-parametric test to measure whether two independent samples ("projective sentences" and "non-projective sentences") have the same distribution of scores.

6.3.4 Error Mining

Tree error mining of Narayan and Gardent (2012) was initially developped to explain errors in grammar-based generators. The algorithm takes as input two groups of dependency trees: those whose derivation was covered (P for Pass) and those whose derivation was not covered (F for Fail) by the generator. Based on these two groups, the algorithm computes a suspicion score S for each subtree f in the input data as follows:

$$S(f) = \frac{1}{2}\left(\frac{c(f|F)}{c(f)}\ln c(f) + \frac{c(\neg f|P)}{c(f)}\ln c(\neg f)\right)$$

$c(f)$ is the number of sentences containing a subtree f, $c(\neg f)$ is the number of sentences where f is not present, $c(f|F)$ is the number of sentences containing f for which generation failed, and $c(\neg f|P)$ is the number of sentences not containing f for which generation succeeded. Intuitively, a high suspicion score indicates a subtree (a syntactic construct) in the input data which often co-occurs with failure and seldom with success.

To imitate those two groups of successful and unsuccessful generation, we adapted a threshold based on BLEU. All the instances in a model output are divided into two parts: the first quartile (25% of instances)[27] with a low sentence-level BLEU was considered as failure, the rest—as success. Error mining can then be used to automatically identify subtrees of the input tree that often co-occur with failure and rarely with success. Moreover, mining can be applied to trees decorated with any combination of lemmas, dependency relations and/or POS tags.

6.4 Data and Experimental Setting

We apply our error analysis methods to 174 system outputs (runs) submitted to the shallow track of SR'18 and SR'19 shared tasks (Mille et al., 2018a, 2019). For each generated sentence in the submissions, we compute the metrics described in the preceding Section as follows.

Computing Syntactic Complexity Metrics. Tree-based metrics, dependency relation entropy and projectivity are computed on the gold parse trees from Universal Dependencies v2.0 and v2.3 (Nivre et al., 2017). Following common practice in dependency linguistics computational studies, punctuation marks were stripped from the reference trees (based on *punct* dependency relation). If a node to be removed had children, these were assigned to the parent of the node.

Computing Performance Metrics. We compute sentence-level BLEU-4 with the smoothing method 2 from Chen and Cherry (2014), implemented in NLTK. We do not include other automatic n-gram-based metrics used in the SR shared tasks because they usually correlate with each other.

[27] It is our empirical choice. Any other threshold can also be chosen.

		S	count	depth	length	MDD	MFS	MFW	MA	NP
SR'18	ar	3	676	7.37±3.29	38.5±30.38	2.61±0.93	2.61±0.93	1.44±0.26	0.94±0.08	1.48
	cs	2	9,876	3.95±1.99	14.49±9.43	2.12±0.74	2.12±0.74	1.19±0.29	0.86±0.18	9.91
	es	6	1,719	5.21±2.2	26.88±15.7	2.47±0.66	2.47±0.66	1.33±0.25	0.93±0.09	2.39
	en	8	2,061	2.71±1.88	10.57±9.55	1.86±0.95	1.86±0.95	1.02±0.42	0.75±0.3	1.65
	fi	3	1,525	3.48±1.81	11.42±7.22	2.02±0.62	2.02±0.62	1.16±0.23	0.86±0.12	5.57
	fr	5	416	4.33±1.75	21.21±12.57	2.44±0.59	2.44±0.59	1.28±0.25	0.93±0.07	2.16
	it	4	480	4.38±2.23	19.14±14.07	2.19±0.61	2.19±0.61	1.23±0.23	0.91±0.06	2.29
	nl	4	685	3.74±1.86	15.03±9.11	2.48±1.05	2.48±1.05	1.21±0.39	0.85±0.22	20.15
	pt	4	476	4.32±2.12	18.58±12.11	2.25±0.63	2.25±0.63	1.23±0.27	0.9±0.13	4.20
	ru	2	6,366	4.1±1.96	14.65±9.14	2.12±0.66	2.12±0.66	1.23±0.27	0.88±0.13	8.37
SR'19	ar_padt	4	680	7.38±3.28	38.54±30.34	2.6±0.93	2.6±0.93	1.45±0.26	0.94±0.08	1.76
	en_ewt	5	2,077	2.72±1.88	10.6±9.62	1.87±0.95	1.87±0.95	1.02±0.42	0.75±0.3	1.54
	en_gum	11	778	3.69±1.91	15.0±10.63	2.14±0.75	2.14±0.75	1.16±0.31	0.85±0.2	3.08
	en_lines	11	914	3.55±1.6	14.97±9.56	2.27±0.62	2.27±0.62	1.2±0.23	0.89±0.11	4.60
	en_partut	11	153	4.52±2.01	20.06±9.77	2.48±0.51	2.48±0.51	1.26±0.21	0.93±0.05	0.65
	es_ancora	6	1,721	5.2±2.2	26.87±15.7	2.47±0.66	2.47±0.66	1.33±0.25	0.93±0.09	2.38
	es_gsd	6	426	5.06±2.25	25.18±16.43	2.41±0.57	2.41±0.57	1.31±0.23	0.94±0.05	4.69
	fr_gsd	7	416	4.41±1.78	21.22±12.58	2.41±0.58	2.41±0.58	1.28±0.25	0.93±0.07	1.20
	fr_partut	7	110	4.85±1.82	21.84±10.01	2.44±0.46	2.44±0.46	1.29±0.21	0.94±0.03	0.91
	fr_sequoia	7	456	4.01±2.21	19.66±15.61	2.13±0.84	2.13±0.84	1.16±0.37	0.84±0.25	0.88
	hi_hdtb	5	1,684	4.19±1.48	19.6±8.99	2.96±0.82	2.96±0.82	1.48±0.23	0.94±0.03	8.91
	id_gsd	5	557	4.57±1.85	18.02±12.39	2.04±0.54	2.04±0.54	1.22±0.2	0.92±0.07	0.72
	ja_gsd	6	551	4.36±1.97	20.25±13.35	2.43±0.66	2.43±0.66	1.4±0.32	0.92±0.09	0.00
	ko_gsd	5	989	3.59±1.78	10.29±6.77	2.21±0.79	2.21±0.79	1.33±0.36	0.86±0.1	9.20
	ko_kaist	4	2,287	3.86±1.54	11.0±4.56	2.27±0.67	2.27±0.67	1.44±0.32	0.89±0.07	19.15
	pt_bosque	5	477	4.32±2.11	18.57±12.09	2.25±0.63	2.25±0.63	1.23±0.27	0.9±0.13	4.40
	pt_gsd	5	1,204	4.85±1.87	22.74±12.2	2.39±0.55	2.39±0.55	1.31±0.23	0.94±0.05	1.66
	ru_gsd	5	601	4.11±1.69	15.83±10.24	2.12±0.69	2.12±0.69	1.24±0.21	0.91±0.06	4.49
	ru_syntagrus	4	6,491	4.08±1.94	14.78±9.24	2.13±0.65	2.13±0.65	1.23±0.26	0.88±0.13	6.49
	zh_gsd	7	500	4.22±1.08	20.64±10.17	2.98±0.84	2.98±0.84	1.46±0.27	0.94±0.03	0.40

Table 6.2: Descriptive statistics (mean and stdev apart from the first two and the last column) for the UD treebanks used in SR'18 and SR'19. S: number of submissions, count: number of sentences in a test set, MDD: mean dependency distance, MFS: mean flux size, MFW: mean flux weight, MA: mean arity, NP: percentage of non-projective sentences. For the tree-based metrics (MDD, MFS, MFW, MA), macro-average values are reported.

To compute dependency edge accuracy, we process systems' outputs to allow for comparison with the lemmatised dependency tree of the reference sentence. Systems' outputs were tokenised and lemmatised; contractions were also split to match lemmas in the UD treebanks. Finally, to be consistent with punctuation-less references, punctuation was also removed from systems' outputs. The preprocessing was done with the stanfordnlp library (Qi et al., 2018).

For human judgments, we collect those provided by the shared tasks for a sample of test data and for some languages (*en, es, fr* for SR'18 and *es_ancora, en_ewt, ru_syntagrus, zh_gsd* for SR'19). Table 6.2 shows how many submissions each language received.

Computing Correlation. For all numerical variables, we assess the relationship between rankings of two variables using Spearman's ρ correlation. When calculating correlation coefficients, missing values were ignored (that was the case for human evaluations). Correlations were calculated separately for each submission (one system run for one corpus). Since up to 45 comparisons can be made for one submission, we controlled for the multiple testing problem using the Holm-Bonferroni method while doing a significance test. We also calculated means and medians of the correlations for each corpus (all submissions mixed), for each team (a team has multiple submissions), and average correlations through all the 174 submissions.

For projectivity (nominal variable) we use a Mann–Whitney U test to determine whether there is a difference in performance between projective and non-projective sentences. We ran three tests where performance was defined in terms of BLEU, Fluency, and Adequacy. As for some corpora, the count of non-projective sentences in their test set is low (e.g., 1.56% in *en_ewt*), we ran the test on the corpora that have more than 5% of non-projective sentences, that is *cs* (10%), *fi* (6%), *nl* (20%), *ru* (8%) for SR'18, and *hi_hdtb* (9%), *ko_gsd* (9%), *ko_kaist* (19%), *ru_syntagrus* (6%) for SR'19. For the calculation of the Mann–Whitney U test, we used scipy-1.4.1. Similar to the correlation analysis, the test was calculated separately for each submission and for each corpus.

Mining the Input Trees. The error mining algorithm was run for each submission separately and with three different settings: (*i*) dependency relations (dep); *(ii)* POS tags (pos); *(iii)* dependency relations and POS tags (pos-dep).

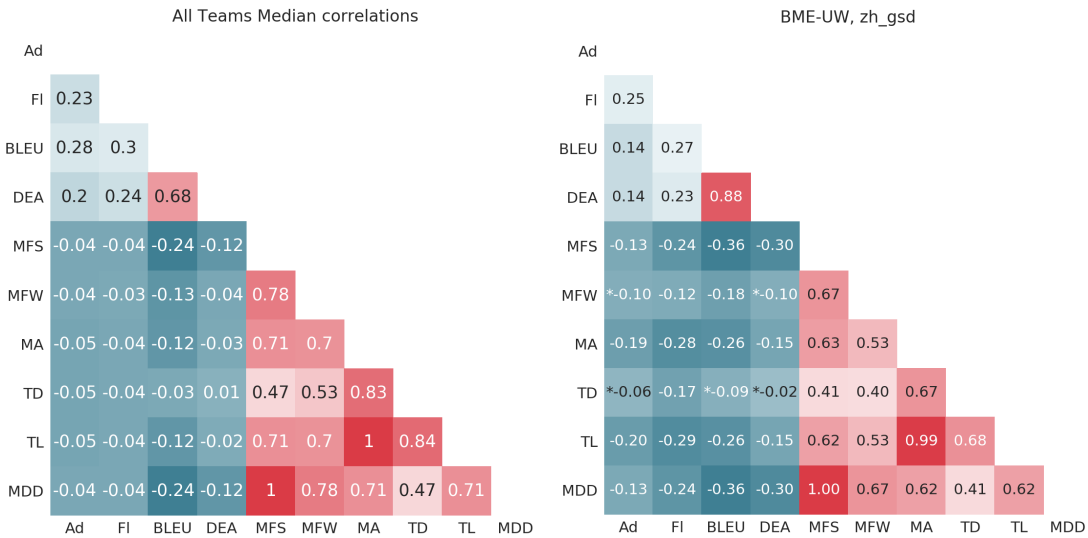
6.5 Error Analysis

We analyse results focusing successively on: tree-based syntactic complexity (are sentences with more complex syntactic trees harder to generate?), projectivity (how much does non-projectivity impact results?), entropy (how much do word order variations affect performance?), dependency edge accuracy and error mining (which syntactic constructions lead to decreased scores?).

6.5.1 Tree-Based Syntactic Complexity

We examine correlation tests results for all metrics on the system level (all submissions together) and for a single model, the BME-UW system (Kovács et al., 2019) on a single corpus/language (*zh_gsd*, Chinese). Figure 6.2a shows median Spearman ρ coefficients across all the 174 submissions, and Figure 6.2b shows the coefficients for the BME-UW system on the *zh_gsd* corpus.

We investigate both correlations between syntactic complexity and performance metrics and within each category. Similar observations can be made for both settings.



(a) Median Spearman ρ coefficients between metrics (all submissions considered).

(b) Spearman ρ coefficients between metrics for BME-UW for zh_gsd.

Figure 6.2: Spearman ρ coefficients between metrics. Ad: Adequacy z-score, Fl: Fluency z-score, DEA: dependency edge accuracy, MFS: mean flux size, MFW: mean flux weight, MA: mean arity, TD: tree depth, TL: tree length, MDD: mean dependency distance. * – non-significant coefficients at $\alpha = 0.05$ corrected with the Holm-Bonferroni method for multiple hypotheses testing.

Correlation between Performance Metrics. As often remarked in the NLG context (Stent et al., 2005; Novikova et al., 2017a; Reiter, 2018), BLEU shows a weak correlation with Fluency and Adequacy on the sentence level. Similarly, dependency edge accuracy shows weak correlations with human judgments ($\rho_{ad} = 0.2$ and $\rho_{fl} = 0.24$ for the median; $\rho_{ad} = 0.14$ and $\rho_{fl} = 0.23$ for BME-UW). Bear in mind though that using human assessments for word ordering evaluation has one downside because the assessments were collected for final sentences, and were not specifically created for word ordering evaluation. A more detailed human evaluation focused on word ordering might be needed to confirm the findings including human judgments.

In contrast, BLEU shows a strong correlation with dependency edge accuracy (median: $\rho = 0.68$; BME-UW: $\rho = 0.88$). Contrary to BLEU however, DEA has a direct linguistic interpretation (it indicates which dependency relations are harder to handle) and can be exploited to analyse and improve a model. We therefore advocate for a more informative evaluation that incorporates DEA in addition to the standard metrics. We believe this will lead to more easily interpretable results and possibly the development of better, linguistically informed SR models.

Correlation between Syntactic Complexity Metrics. Unsurprisingly, tree-based metrics have positive correlations between each other (the redish area on the right) ranging from weak to strong. Due to calculation technique overlap, some of them can show strong correlation (e.g., mean dependency distance and mean flux size).

Correlation between Syntactic Complexity and Performance Metrics. Tree-based metrics do not correlate with human assessments (ρ fluctuates around zero for median and from -0.06 to -0.29 for BME-UW).

In general no correlation between tree-based metrics and system performance was found globally, i.e. for all models and all testsets. We can use the framework to analyse results on specific corpora or languages however. For instance, zooming in on the *fr* corpus, we can observe a weak negative correlation at the system level (correlation with the median) between tree-based metrics (e.g., $\rho = -0.38$ for mean arity and tree length) and DEA. Thus, on this corpus, performance decreases as syntactic complexity (as measured by DEA) increases. Similarly, for *ar, cs, fi, it, nl*, tree-based metrics show some negative correlation with BLEU[28] whereby ρ median values between dependency metrics and BLEU for those corpora vary from -0.21 to -0.38 for *ar*, from -0.43 to -0.57 for *cs*, from -0.2 to -0.46 for *fi*, from -0.17 to -0.34 for *it*, and from -0.29 to -0.42 for *nl*.

Such increase in correlations were observed mainly for corpora, for which performance was not high across submissions (see Mille et al. (2018a)). We hypothesize that BLEU correlates more with the tree-based metrics if system performance is bad.

Significance testing. Overall, across submissions, coefficients were found non-significant only when they were close to zero (see Figure 6.2b).

6.5.2 Projectivity

Table 6.3 shows performance results with respect to the projectivity parameter.

Zooming in on the *ru_syntagrus* corpus and two models, one which can produce non-projective trees, BME-UW (Kovács et al., 2019), and one which cannot, the IMS system (Yu et al., 2019a), we observe two opposite trends.

For the BME-UW model, the median values for Fluency and Adequacy are higher for non-projective sentences. Fluency medians (proj/non-proj) are 0.15/0.19 (Mann–Whitney $U = 4109131.0$, $n_1 = 6070$, $n_2 = 421$, $p < 0.001$ two-tailed); Adequacy medians (proj/non-proj) are 0.31/0.48 ($U = 2564235.0$, $n_1 = 6070$, $n_2 = 421$, $p < 0.001$). In other words, while the model can handle non-projective structures, a key drawback revealed by our error analysis, is that for sentences with projective structures (which incidentally, are much more frequent in the data), the model output is in fact judged less fluent and less adequate by human annotators than for non-projective sentences.

Conversely, for the IMS system, median values for Fluency is higher for projective sentences (0.42 vs. 0.18 for non-projective sentences), and the distributions in the two groups differed significantly ($U = 4038434.0$, $p < 0.001$ two-tailed). For Adequacy, the median value for projective sentences (0.58) is also significantly higher than that for non-projective sentences (0.37, $U = 2583463.0$, $p < 0.001$ two-tailed). This in turn confirms the need for models that can handle non-projective structures.

Another interesting point highlighted by the results on the *ru_syntagrus* corpus in Table 6.3 is that similar BLEU scores for projective and non-projective structures do not necessarily mean similar human evaluation scores.

In terms of BLEU only, i.e. taking all other corpora with no human evaluations, and modulo the caveat just made about the relation between BLEU and human evaluation, we find that non-projective median values were always lower than projectives ones, and distributions showed significant differences, throughout all the 25 comparisons made. This underlines the need for models that can handle both projective and non-projective structures.

[28] Unfortunately no human evaluations were available for those corpora.

team	corpus	BLEU Proj/Non-Proj	Fl_z	Ad_z	Sample sizes
AX	cs	0.25/0.19	-/-	-/-	8897/979
BinLin	cs	0.49/0.38	-/-	-/-	8897/979
AX	fi	0.25/0.2	-/-	-/-	1440/85
BinLin	fi	0.44/0.33	-/-	-/-	1440/85
OSU	fi	0.47/0.38	-/-	-/-	1440/85
AX	nl	0.28/0.2	-/-	-/-	547/138
BinLin	nl	0.39/0.3	-/-	-/-	547/138
OSU	nl	0.38/0.28	-/-	-/-	547/138
Tilburg	nl	0.43/0.36	-/-	-/-	547/138
AX	ru	0.27/0.22	-/-	-/-	5833/533
BinLin	ru	0.44/0.36	-/-	-/-	5833/533
BME-UW	hi_hdtb	0.66/0.6	-/-	-/-	1534/150
DepDist	hi_hdtb	0.66/0.62	-/-	-/-	1534/150
IMS	hi_hdtb	0.82/0.73	-/-	-/-	1534/150
LORIA	hi_hdtb	0.29/0.22	-/-	-/-	1534/150
Tilburg	hi_hdtb	0.68/0.64	-/-	-/-	1534/150
BME-UW	ko_gsd	0.54/0.38	-/-	-/-	898/91
DepDist	ko_gsd	0.51/0.37	-/-	-/-	898/91
IMS	ko_gsd	0.84/0.56	-/-	-/-	898/91
LORIA	ko_gsd	0.43/0.4	-/-	-/-	898/91
Tilburg	ko_gsd	0.08/0.06	-/-	-/-	898/91
BME-UW	ko_kaist	0.51/0.39	-/-	-/-	1849/438
IMS	ko_kaist	0.82/0.6	-/-	-/-	1849/438
LORIA	ko_kaist	0.43/0.37	-/-	-/-	1849/438
Tilburg	ko_kaist	0.14/0.11	-/-	-/-	1849/438
BME-UW	ru_syntagrus	**0.58/0.59**	**0.15/0.19**	**0.31/0.48**	6070/421
IMS	ru_syntagrus	**0.76/0.77**	0.42/0.18	0.58/0.37	6070/421
LORIA	ru_syntagrus	**0.61/0.62**	0.33/0.3	**0.39/0.55**	6070/421
Tilburg	ru_syntagrus	**0.46/0.47**	-0.2/-0.37	-0.01/-0.2	6070/421

Table 6.3: Median values for BLEU, Fluency, and Adequacy for projective/non-projective sentences for each submission. Medians for non-projective sentences which are higher than for the projective sentences are in bold. All comparisons were significant with $p < 0.001$. Human judgments were available for *ru_syntagrus* only.

6.5.3 Entropy

Correlation between dependency relation entropy and dependency edge accuracy permits identifying which model, language or corpus is particularly affected by word order freedom. The meaning of dependency relations can be consulted on this page: `https://universaldependencies.org/u/dep/index.html`; originally the relations were described in de Marneffe et al. (2014).

Figure 6.3 illustrates which entropy dependency relations have in the considered treebanks. For example, the *advcl* relation has a high entropy for Finnish, Indonesian, and for the languages from the Indo-European family (redish squares), and a moderate to low entropy for other languages.

Given a DEA and an entropy for each dependency relation, we computed correlation scores

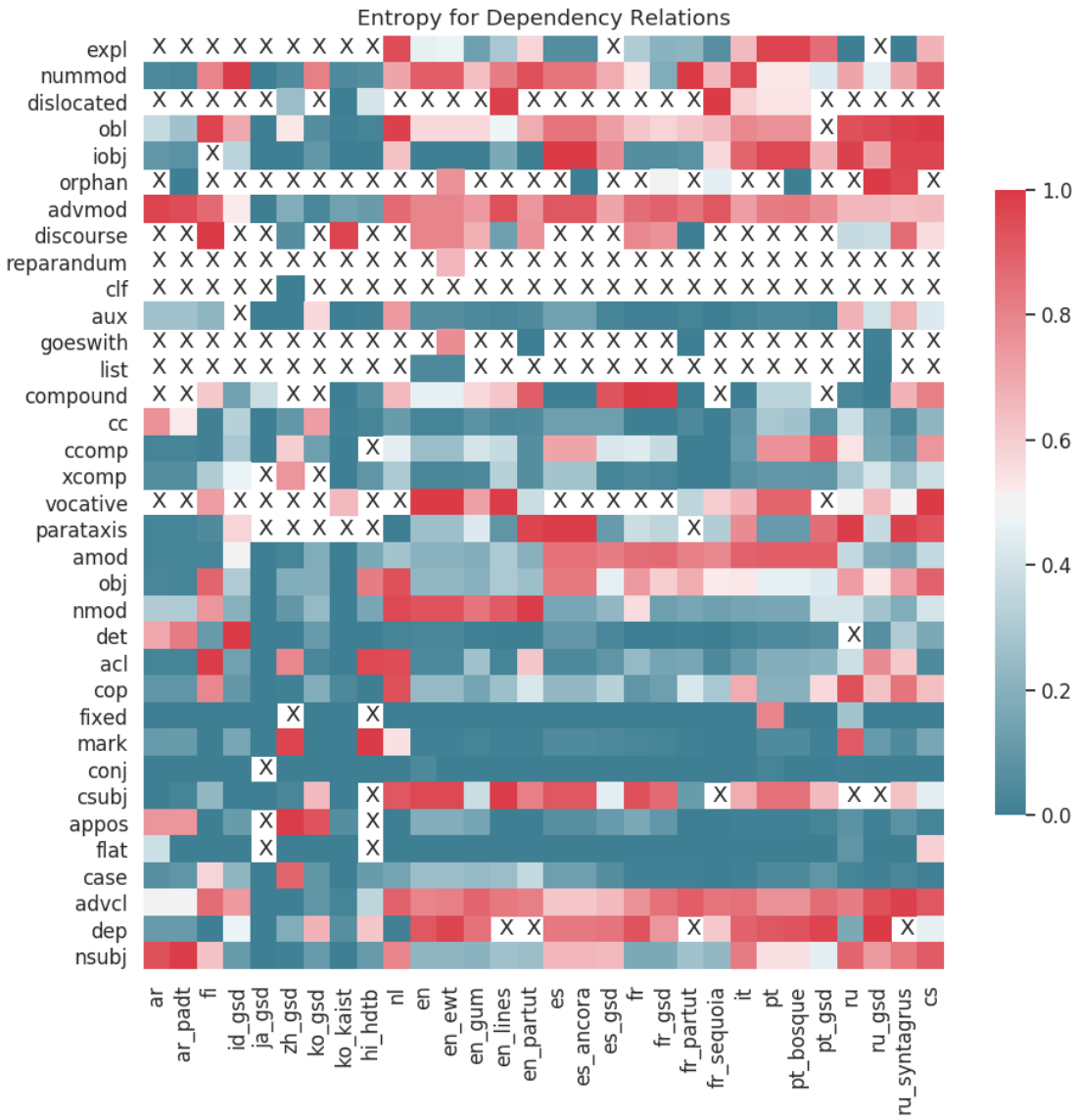


Figure 6.3: Entropy of dependency relations in the treebanks used in the SR shared tasks. A cross indicates the absence of a dependency relation in a treebank. Treebanks are grouped by language families.

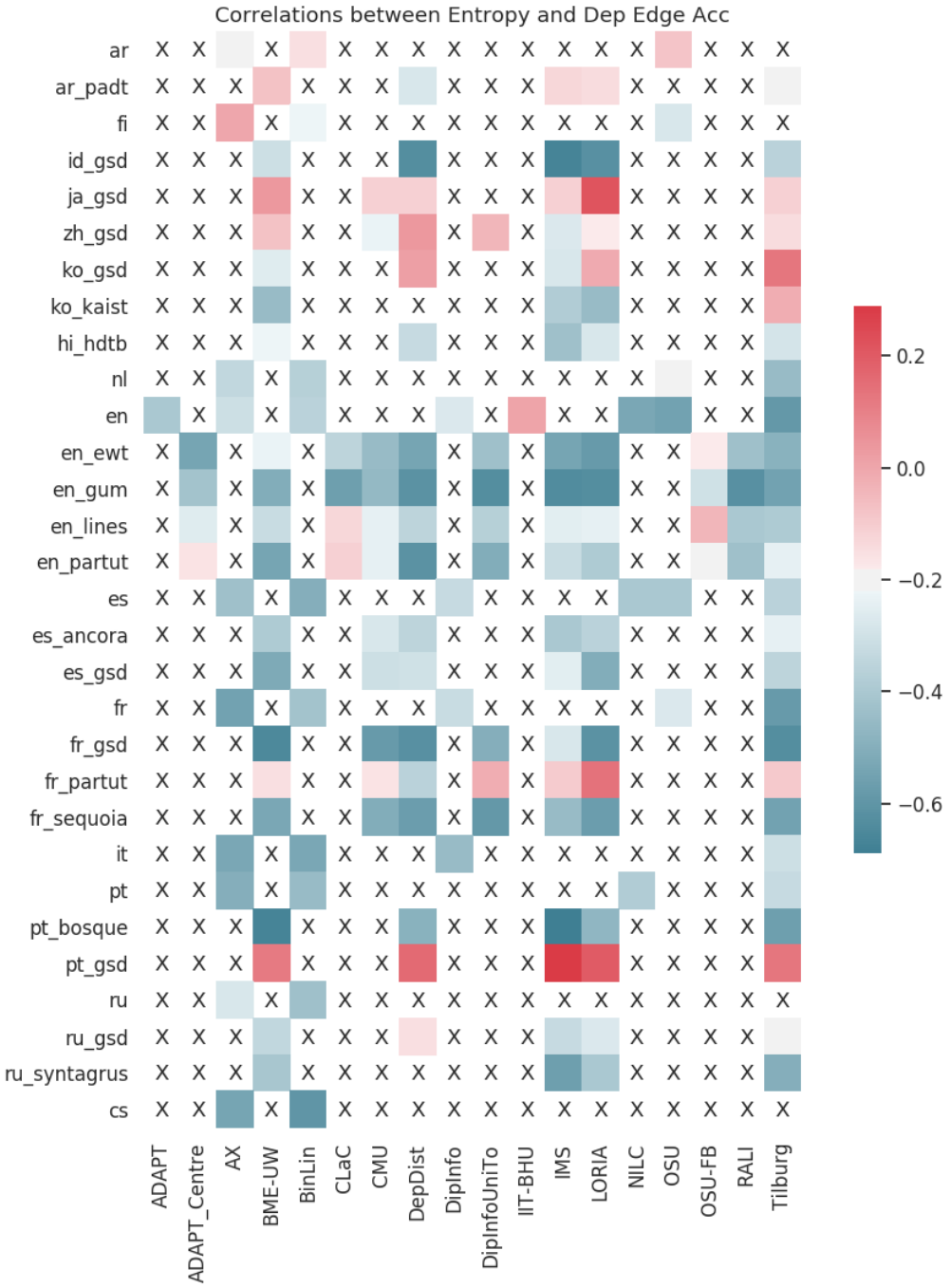


Figure 6.4: Spearman correlation coefficient heatmap between dependency edge accuracy and entropy. A cross indicates that a team did not make a submission for the treebank. Treebanks are grouped by language families. The same correlation matrix but annotated with coefficient numbers is available in the Appendix B.

between them for all systems and all corpora (Figure 6.4). Negative scores indicate the following relation between two variables: the higher the entropy, the lower the accuracy.

For instance, for the *id_gsd* corpus, three teams have a Spearman's ρ in the range from -0.62 to -0.67, indicating that their models have a tendency to underperform for dependency relations with high entropy. Conversely, two other teams showed weak correlation ($\rho = -0.31$ and $\rho = -0.36$) for the same *id_gsd* corpus.

The impact of entropy also varies depending on the language, the corpus and more generally, the entropy of the data. For instance, for Japanese (*ja_gsd* corpus), dependency relations have low entropy (the mean entropy averaged on all relations is 0.02), and so we observe no correlation between entropy and performance. Conversely, for Czech (the treebank with the highest mean entropy, $H = 0.52$), two teams show non-trivial negative correlations ($\rho = -0.54$ and $\rho = -0.6$) between entropy and dependency edge accuracy.

6.5.4 Which Syntactic Constructions Are Harder to Handle?

Dependency Edge Accuracy (DEA). For a given dependency relation, dependency edge accuracy assesses how well a model succeeds in realising that relation. To identify which syntactic constructs are problematic for surface realisation models, we therefore compute dependency edge accuracy per relation, averaging over all submissions. Table 6.4 shows the results.

Unsurprisingly, relations with low counts (first five relations in the table) have low accuracy. Because they are rare (in fact they are often absent from most corpora), SR models struggle to realise these.

Other relations with low accuracy are either relations with free word order (i.e., *advcl, discourse, obl, advmod*) or whose semantics is vague (*dep*—unspecified dependency). Clearly, in case of the latter, systems cannot make a good prediction; as for the former, the low DEA score may be an artefact of the fact that it is computed with respect to a single reference. As the construct may occur in different positions in a sentence, several equally correct sentences may match the input but only one will not be penalised by the comparison with the reference. This underlines once again the need for an evaluation setup with multiple references.

Relations with the highest accuracy are those for function words (*case*—case-marking elements, *det*—determiners, *clf*—classifiers), fixed multiword expressions (*fixed*), and nominal dependents (*amod*, *nmod*, *nummod*). Those dependencies on average have higher stability with respect to their head in terms of distance, more often demonstrate a fixed word order, and do not exhibit a certain degree of probable shifting as the relations described above. Due to those factors, their realisation performance is higher.

Interestingly, when computing DEA per dependency relation and per corpus, we found similar DEA scores for all corpora. That is, dependency relations have consistently low/high DEA score across all corpora therefore indicating that improvement on a given relation will improve performance on all corpora/languages.

Finally, we note that, at the model level, dependency edge accuracy scores are useful metrics for researchers as it brings interpretability and separation into error type subcases.

Error mining for syntactic trees. We can also obtain a more detailed picture of which syntactic constructs degrade performance using error mining. After running error mining on all submissions, we examine the subtrees in the input which have highest coverage, i.e. for which the percentage of submissions tagging these forms as suspicious[29] is highest. Tables 6.5, 6.6 and

[29] A form is suspicious if its suspicion score is not null.

deprel	count	Accuracy
list	4,914	17.75
vocative	974	21.91
dislocated	7,832	23.11
reparandum	33	27.27
goeswith	1,453	27.98
parataxis	27,484	28.76
dep	14,496	29.80
advcl	60,719	32.52
csubj	8,229	36.60
discourse	3,862	37.45
ccomp	33,513	41.74
obl	232,097	42.39
appos	35,781	43.59
advmod	180,678	44.84
iobj	16,240	44.96
conj	149,299	45.77
orphan	843	48.49
expl	10,137	50.90
acl	79,168	51.24
cop	45,187	51.78
nsubj	268,686	51.80
xcomp	36,633	56.12
obj	190,140	57.87
nummod	61,459	58.46
aux	95,748	58.47
mark	105,993	59.77
compound	82,314	59.99
nmod	357,367	60.94
flat	62,686	61.28
amod	246,733	61.68
cc	123,866	61.94
clf	1,668	67.47
fixed	27,978	73.08
det	280,978	73.51
case	465,583	74.15

Table 6.4: Macro-average dependency edge accuracy over all submissions sorted from the lowest accuracy to the highest. Count is a number of times a relation was found in all treebanks.

6.7 shows the results when using different views of the data (i.e. focusing only on dependency information, only on POS tags or on both).

Table 6.5 highlights coordination (*conj*, 13 subtrees out of 20) and adverbial clause modifiers (*advcl*, 5 cases) as a main source of low BLEU scores. This mirrors the results shown for single dependency relations (cf. Section 6.5.4) but additionally indicates specific configurations in which these relations are most problematic such as for instance, the combination of an adverbial

rank	subtree	cov.	MSS
1-2	(conj (X))	70-73	1.17
3	(advcl (nsubj))	62	0.91
4	(advcl (advmod))	62	0.95
5	(advmod (advmod))	59	0.77
6	(conj (advcl))	57	0.75
7	(nsubj (conj))	56	0.68
8-11	(conj (X))	52-56	0.87
12	(nmod (advmod))	52	0.56
13	(nsubj (amod))	52	0.75
14-15	(conj (X))	49-50	0.73
16	(parataxis (nsubj))	49	0.75
17	(conj (advmod advmod))	48	0.65
18	(advcl (cop))	48	0.60
19	(advcl (aux))	47	0.59
20	(ccomp (advmod))	47	0.68

Table 6.5: Top-20 of the most frequent suspicious trees (dep-based) across all submissions. In case of *conj*, when tree patterns were similar, they were merged, X serving as a placeholder. Coverage: percentage of submissions where a subtree was mined as suspicious. MSS: mean suspicion score for a subtree.

tree	coverage	MSS
(ADJ (PRON))	70	0.90
(VERB (VERB))	69	1.21
(ADJ (ADJ))	68	0.89
(NOUN (ADV))	67	1.03
(ADJ (ADP))	66	0.77
(VERB (ADJ))	65	0.98
(ADV (ADV))	63	0.87
(NOUN (AUX))	62	0.90
(ADJ (VERB))	60	0.80
(VERB (CCONJ))	60	1.02
(PRON (ADP))	56	0.81
(VERB (VERB VERB))	55	0.89
(NUM (NUM))	55	0.72
(PROPN (NOUN))	53	0.79
(PRON (VERB))	53	0.63
(ADJ (CCONJ))	52	0.65
(VERB (ADV))	52	0.96
(ADJ (SCONJ))	52	0.62
(VERB (ADP))	51	0.76
(VERB (PROPN))	51	0.83

Table 6.6: Most frequent suspicious trees (pos-based) across all submissions.

subtree	cov.	MSS
(VERB~conj (ADV~advmod))	60	0.90
(VERB~conj (PRON~nsubj))	60	0.78
(NOUN~nsubj (ADJ~amod))	55	0.77
(ADV~advmod (ADV~advmod))	54	0.69
(VERB~advcl (ADV~advmod))	53	0.76
(VERB~advcl (NOUN~nsubj))	53	0.70
(VERB~conj (VERB~advcl))	50	0.60
(VERB~advcl (PRON~obj))	48	0.53
(VERB~ccomp (ADV~advmod))	47	0.57
(NOUN~nsubj (NOUN~conj))	46	0.46
(VERB~advcl (NOUN~obl))	46	0.68
(VERB~conj (PRON~obj))	45	0.57
(VERB~advcl (AUX~aux))	44	0.56
(VERB~conj (AUX~aux))	41	0.59
(NOUN~obl (ADJ~amod))	40	0.62
(NOUN~nsubj (VERB~acl))	40	0.46
(VERB~acl (ADV~advmod))	40	0.47
(NOUN~obl (ADV~advmod))	38	0.43
(NOUN~conj (VERB~acl))	38	0.38
(VERB~ccomp (AUX~aux))	38	0.48

Table 6.7: Most frequent suspicious trees (dep-pos-based) across all submissions.

clause modifier with a nominal subject (*nsubj*, 62% coverage) or an adverbial modifier (*advmod*, 62% coverage) or the combination of two adverbial modifiers together (e.g., *down there, far away, very seriously*).

Table 6.6 shows the results for the POS setting. Differently from the dep-based view, it highlights head-dependent constructs with identical POS tags, e.g., (ADV (ADV)), (ADJ (ADJ)), (NUM (NUM)), (VERB (VERB)) and (VERB (VERB VERB)), as a frequent source of errors. For instance, the relative order of two adjectives (ADJ (ADJ)) is sometimes lexically driven and therefore difficult to predict (Malouf, 2000).

Table 6.7 shows a hybrid POS-dep view of the most suspicious forms on a system level, detailing the POS tags most commonly associated with the dependency relations shown in Table 6.5 to raise problem, i.e. coordination, adverbial modifiers and adverbial clauses.

6.6 Using Error Analysis for Improving Models or Datasets

As shown in the preceding section, the error analysis framework introduced in Section 6.3 can be used by evaluation campaign organisers to provide a linguistically informed interpretation of campaign results aggregated over multiple system runs, languages or corpora.

For individual researchers and model developers, our framework also provides a means to have a fine-grained interpretation of their model results which they can then use to guide model improvement, to develop new models or to improve training data. We illustrate this point by giving some examples of how the toolkit could be used to help improve a model or a dataset.

Data Augmentation. Augmenting the training set with silver data has repeatedly been shown to increase performance (Konstas et al., 2017; Elder and Hokamp, 2018). In those approaches, performance is improved by simply augmenting the size of the training data. In contrast, information from the error analysis toolkit could be used to support error-focused data augmentation, i.e., to specifically augment the training data with instances of those cases for which the model underperforms (e.g., for dependency relations with low dependency edge accuracy, for constructions with low suspicion score or for input trees with large depth, length or mean dependency distance). This could be done either manually (by annotating sentences containing the relevant constructions) or automatically by parsing text and then filtering for those parse trees which contain the dependency relations and subtrees for which the model underperforms. For those cases where the problematic construction is frequent, we conjecture that this might lead to a better overall score increase than "blind" global data augmentation.

Language Specific Adaptation. Languages exhibit different word order schemas and have different ways of constraining word order. Error analysis can help identify which language-specific constructs impact performance and how to improve a language-specific model with respect to these constructs.

For instance, a dependency relation with high entropy and low accuracy indicates that the model has difficulty learning the word order freedom of that relation. Model improvement can then target a better modelling of those factors which determine word order for that relation. As it was discussed in Section 5.5.1, in Romance languages, for example, adjectives mostly occur after the noun they modify. However some adjectives are pre-posed. As the pre-posed adjectives rather form a finite set, a plausible way to improve the model would be to enrich the input representation by indicating for each adjective whether it belongs to the class of pre- or post-posed adjectives.

Global Model Improvement. Error analysis can suggest direction for model improvement. For instance, a high proportion of non-projective sentences in the language reference treebank together with lower performance metrics for those sentences suggests improving the ability of the model to handle non-projective structures. Indeed, Yu et al. (2020) showed that the performance of the model of (Yu et al., 2019a) could be greatly improved by extending it to handle non-projective structures.

Treebank Specific Improvement. Previous research has shown that treebanks contain inconsistencies thereby impacting both learning and evaluation (Zeman, 2016). The tree-based metrics and the error mining techniques provided in our toolkit can help identify those dependency relations and constructions which have consistently low scores across different models or diverging scores across different treebanks for the same language. For instance, a case of strong inconsistencies in the annotation of multi-word expressions (MWE) may be highlighted by a low DEA for the *fixed* dependency relation (which should be used to annotate MWE). Such annotation errors could also be detected using lemma-based error mining, i.e., error mining for forms decorated with lemmas. Such mining would then show that the most suspicious forms are decorated with multi-word expressions (e.g., "in order to").

Ensemble Model. Given a model M and a test set T, our toolkit can be used to compute for each dependency relation d present in the test set, the average dependency edge accuracy of that model for that relation (DEA^d_M, the sum of the model's DEA for all d-edge in T normalised

by the number of these edges). This could be used to learn an ensemble model which, for each input, outputs the sentence generated by the model whose score according to this metric is highest. Given an input tree t consisting of a set of edges D, the score of a model M could for instance be the sum of the model's average DEA for the edges contained in the input tree normalised by the number of edges in that tree, i.e., $\frac{1}{|D|} \times \sum_{d \in D} DEA_M^d$.

6.7 Conclusion

We presented a framework for error analysis which supports a detailed assessment of which syntactic factors impact the performance of surface realisation models. We applied it to the results of two SR shared task campaigns and suggested ways in which it could be used to improve models and datasets for shallow surface realisation. We showed that dependency edge accuracy correlates with BLEU which suggests that DEA could be used as an alternative, more interpretable, automatic evaluation metric for surface realisers. We also showed that other tree-based metrics do not correlate with system performance.

More generally, we believe that scores such as BLEU and, to some extent, human ratings do not provide a clear picture of the extent to which SR models can capture the complex constraints governing word order in the world natural languages. We hope that the metrics and tools gathered in this evaluation toolkit can help address this issue.

One of the limitations of the current study was that human evaluation scores were collected while assessing the final sentence in terms of Fluency and Adequacy. For instance, a final sentence may have low human scores due to errors in morphological inflection only. For future research, it is worth collecting human judgments focusing on word order evaluation only and then correlating them with metrics presented in this chapter. Moreover, languages for which human scores were made available, were also limited, and future work should definitely explore to collect judgments for those languages with missing human scores.

7

Evaluating Natural Language Generation Systems

Contents

7.1 Introduction **85**
7.2 Context and Motivation **86**
7.3 Experimental Setup **87**
7.3.1 Data 87
7.3.2 Design 87
7.3.3 Ensuring Quality 88
7.3.4 Correlations 88
7.4 Correlation Analysis Results **88**
7.5 Conclusion **89**

7.1 Introduction

In the previous chapter, we presented an evaluation framework which was specifically tailored to sentence-level evaluation. In this chapter, we focus on the relationship between sentence-level and system-level evaluation as a means to meta-evaluate automatic metrics.

Validity of developed automated methods to assess texts is traditionally measured by their correlations with human ratings. Metrics can be calculated for individual texts as well as for system outputs as a whole. Depending on the setup, one can carry out a meta-evaluation study either at the sentence level by correlating automatic and human scores for each sentence, or at the system level, by correlating ratings (based on automatic metrics and on human judgments) of all systems participating in a task.

This chapter discusses two existing approaches to the correlation analysis between automatic evaluation metrics and human scores in the area of NLG. Our experiments show that depending on the usage of a system- or sentence-level correlation analysis, correlation results between automatic scores and human judgments are inconsistent.

This chapter is based on the following reports:

- Shimorina, A. (2018). Human vs automatic metrics: on the importance of correlation design. *Peer-reviewed, non-archival, presented at the Widening NLP Workshop 2018 at NAACL*, arXiv: 1805.11474

- Shimorina, A., Gardent, C., Narayan, S., and Perez-Beltrachini, L. (2018). WebNLG Challenge: Human Evaluation Results. Technical report, Loria & Inria Grand Est

For reproducibility, the scripts and data are available in this repository: https://gitlab.com/webnlg/webnlg-human-evaluation.

7.2 Context and Motivation

In the machine translation community, the practice to compare system- and sentence-level correlations between automatic and human metrics is well established. From 2008 on, yearly shared tasks have included analyses of metric consistency both on the system- and sentence-level (Callison-Burch et al., 2008, 2009). System-level analysis is motivated by the fact that automatic evaluation metrics such as BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), TER (Snover et al., 2006) were initially created to account for the evaluation of whole systems (i.e. they are corpus-based metrics). For example, while computing BLEU, the brevity penalty is calculated over the whole corpus rather than sentence by sentence; thus outputs are less punished at the sentence level. On the other hand, correlation analysis at the sentence level is motivated by the need to gauge the quality of individual sentences and more generally by the need to have a more fine-grained analysis of the results produced (Kulesza and Shieber, 2004). The common finding in MT is that automatic metrics correlate well with human judgments at the system level but much less so at the sentence level. This in turn prompted the search for alternative automatic metrics which would correlate well with human judgments at the sentence level.

In NLG, there is a lack of such comparative studies. Traditional NLG evaluations and challenges (Reiter and Belz (2009); Gatt and Belz (2010), among others) used only system-level comparisons and reported low to strong correlations depending on the automatic metric used. Reiter and Belz (2009, p. 546) explicitly wrote that they did not compute correlations on individual texts because BLEU-type metrics "are not intended to be meaningful for individual sentences".

Nonetheless, when researchers have one or few systems to evaluate, they resort to sentence-level correlation analysis: e.g., reports of Stent et al. (2005) for paraphrasing, Cahill (2009) for surface realisation, Elliott and Keller (2014) for image caption generation. They usually report low to moderate correlations. In the recent survey on the state of the art in NLG, Gatt and Krahmer (2018) made a comparison of various validation studies, concluding that these studies yielded inconsistent results. However, their survey does not mention that the underlying design of those studies can be different (some of the mentioned studies were system-based, others were sentence-based).

Some observations of the difference between results depending on the correlation evaluation design were made by Novikova et al. (2017a) but they only reported sentence-level correlation results because they also have few systems in their study. Focusing on BLEU only, Reiter (2018) surveyed several NLG validation studies and concluded that BLEU gives poor correlations with human judgments both on the system and sentence level in NLG. Judging from the supplementary material, the structured review of BLEU is based on 6 NLG papers, which reported 19 correlation coefficients, all human evaluations mixed: 6 of them were calculated on the sentence level and 13 on the system level. However, no paper in the survey reported both system- and sentence-level correlations.

In sum, in NLG, researchers tend to report a single type of correlation analysis rather than two. For instance, in recent NLG shared tasks, organisers either reported weak sentence-level

correlations (Dušek et al., 2020) or strong system-level correlations (Mille et al., 2018a, 2019).

In this study, we hope to raise awareness of different design in correlation analysis for NLG evaluation. We present both a system- and a sentence-level correlation analysis on the same NLG data. We show that the results are similar to those obtained for MT systems and we conclude with some recommendations concerning the evaluation of NLG systems.

Terminology note. Strictly speaking, sentence-level analysis should be called text-level analysis in some NLG contexts, since references in NLG corpora can consist of several sentences, as in WebNLG discussed below. Despite that, we stick to the MT tradition and use the term sentence-level correlation in this chapter.

7.3 Experimental Setup

7.3.1 Data

We used the data issued from the WebNLG Challenge (Gardent et al., 2017b) for our experiments. The WebNLG dataset maps data to text, where a data input is a set of triples extracted from DBpedia, and a text is a verbalisation of those triples (Chapters 3 and 4). We sampled 223 data inputs from WebNLG and used the outputs of nine different NLG systems which participated in the WebNLG Challenge[30].

The data inputs were chosen based on different characteristics of the WebNLG corpus: how many RDF triples were in data units (size from 1 to 5), and what the DBpedia category (Building, City, Artist, etc.) was. A sample for each system comprised texts from each category (15 texts); in each category all triple set sizes were covered (5 sizes), and finally we extracted 3 texts per category and size. One should note though that, in such a way, our sample should have had 225 (i.e. $15 * 5 * 3$) texts; however, the count was reduced to 223, as one category (ComicsCharacter) had few data units for a particular size.

Automatic evaluation results (i.e., METEOR, TER, BLEU-4 scores) were calculated for each NLG system both at the system and at the sentence level, and each generated sentence was compared against three references on average.

7.3.2 Design

In total, we evaluated 2230 texts (9 system outputs and gold references) by collecting three judgments per text. Our participants came from English-speaking countries. They were shown data (a set of RDF triples) and a system output (a text), and were asked to answer three questions:

- *Does the text correctly represent the meaning in the data?* (1 - Incorrectly, 2 - Medium, 3 - Correctly)
- *Rate the grammar and the spelling of the text: Is the text grammatical (no spelling or grammatical errors)?* (1 - Ungrammatical, 2 - Medium, 3 - Grammatical)
- *Rate the fluency of the text: Does the text sound fluent and natural?* (1 - Not fluent, 2 - Medium, 3 - Fluent)

[30] https://webnlg-challenge.loria.fr/challenge_2017#participant-submissions

The three questions with a three-point Likert scale rate Semantic adequacy, Grammaticality, and Fluency respectively. One text with its corresponding data entry was shown per page. Each participant had a restriction to give only 30 answers per task. Texts were distributed by five separate tasks, which included outputs produced for the same size of data. We also ensured that each participant evaluated an equal number of texts per team, if possible.

Some rule-based system outputs were empty for a particular data unit, so they were not presented for human evaluation. The lowest score "1" was attributed to those outputs for all assessed parameters.

We collected three judgments per text, and then they were averaged.

7.3.3 Ensuring Quality

We use CrowdFlower[31] to collect human judgments. Apart from using obvious controlling techniques such as the time a contributor spends on a page, restricting a crowdworker to give a limited number of answers per task, we applied several checks to identify if there are untrustworthy workers ("spammers") or not. First, given a sufficient number of answers (say, more than ten), we eliminated contributors whose judgments have always the same pattern for all texts, for instance, "2-3-3" scores. Secondly, we made use of the MACE tool (Hovy et al., 2013) to identify unreliable crowdworkers. MACE allows to detect less trusted annotators in an unsupervised fashion by comparing the probability distributions of answers across annotators. Annotator reliability was calculated independently in three variables (Semantic adequacy, Grammaticality, and Fluency). Based on low ratings, we manually evaluated and eliminated spammers, and afterwards launched another round of collecting judgments to cover missing values. We did not trust MACE blindly, rather it was used as an indicator for examining a potentially bad worker. There were cases when a participant demonstrated a high reliability while assessed in one variable (Fluency), whereas in another variable (Grammaticality) the participant's reliability was low. Those participants were usually kept after examination. In such a way, we did not create a uniform distribution of answers, and tried to preserve a variety in human judgments.

7.3.4 Correlations

To perform correlation analysis both at system and sentence levels, we used Spearman's correlation coefficient. To prevent a possible bias, we excluded human references from the analysis as their automatic scores are equal to 1.0 (BLEU, METEOR) and 0.0 (TER). Thus, for system-level analysis, we have nine data points to build a regression line.

7.4 Correlation Analysis Results

Figure 7.1 shows the results. We are focusing here mostly on the correlation between human and automatic metrics as delineated by the black square in Figure 7.1. At the system level (Figure 7.1a), the only statistically significant correlation ($p < .001$) is between METEOR and semantic adequacy. Similar findings for METEOR were reported in the MT community (Callison-Burch et al., 2009) and in the image caption generation domain (Bernardi et al., 2017). We also found a strong correlation between TER and BLEU, and between grammaticality and fluency judgments.

At the sentence level, on the other hand (Figure 7.1b), all correlations are statistically significant ($p < .001$). The highest correlation between human and automatic metrics is between

[31] https://www.crowdflower.com/, now the company is called Appen, and before it was called FigureEight

	fluency	grammar	semantics	bleu	meteor
grammar	0.97				
semantics	~~0.48~~	~~0.47~~			
bleu	~~0.5~~	~~0.57~~	~~0.65~~		
meteor	~~0.47~~	~~0.47~~	0.91	~~0.59~~	
ter	~~-0.45~~	~~-0.5~~	~~-0.62~~	-0.92	~~-0.45~~

(a) Spearman's ρ at the system level. Crossed squares indicate that statistical significance was not reached ($\alpha = .05$). Human vs. automatic metrics are in the black square.

	fluency	grammar	semantics	bleu	meteor
grammar	0.87				
semantics	0.59	0.6			
bleu	0.43	0.44	0.57		
meteor	0.55	0.55	0.73	0.75	
ter	-0.51	-0.49	-0.59	-0.77	-0.8

(b) Spearman's ρ at the sentence level. All correlations are statistically significant ($\alpha = .001$). Human vs. automatic metrics are in the black square.

Figure 7.1: System- and sentence-level correlation analysis.

METEOR and semantic adequacy ($\rho = 0.73$). For other human/automatic correlation results, the correlation is moderate, ranging from $\rho = 0.43$ to $\rho = 0.59$ in absolute numbers. Automatic metrics show strong correlations with each other ($\rho \geq 0.78$).

In sum, there is a strong discrepancy between system- and sentence-level correlation results. Significance was not reached for most of the system-level correlations. At the sentence level, all correlations are significant, however the correlation between automatic metrics and human scores remains relatively low thereby confirming the findings of the MT community. At the sentence level, statistical significance is easier to achieve, since there are more data points than for the system-level analysis. One possibility to have statistically significant results at the system-level would be to use one-tailed test (instead of two-tailed), as it was done by Reiter and Belz (2009). However, that test is considered less statistically robust.

7.5 Conclusion

We argued that in NLG, as in MT, the specific type (system- vs. sentence-level) of correlation analysis chosen to compare human and automatic metrics strongly impacts the outcome. While system-level correlation analyses have repeatedly been used in NLG challenges, sentence-level correlation is more relevant as it better supports error analysis. Based on our experiment, we showed that, in NLG as in MT, the sentence-level correlation between human and automatic metrics is low which in turn suggests the need for new automatic evaluation metrics for NLG that would better correlate with human scores at the sentence level. We hope that our findings may be helpful to shed more light on the relationship between automatic and human judgments in NLG and to facilitate further comparisons of validation studies with different correlation design.

Our analysis was limited to three automatic metrics, all of them based on word overlap. Recently, a lot of new automatic *learned* metrics have appeared, mainly based on contextual embeddings such as BERT: MoverScore (Zhao et al., 2019), BERTScore (Zhang et al.,

2020), Bleurt (Sellam et al., 2020), among others. Those may be equally integrated in future NLG validation studies.

Further research on new automatic evaluation metrics for NLG may focus on developing metrics targeted for a specific criterion of human evaluation rather than trying accounting for all aspects at once. I.e., a specific metric measuring fluency (using pretrained language models, for instance) or specific metrics for semantic adequacy, etc. The new metrics can also be tailored to a particular NLG input. As we have seen in Chapter 6, dependency trees may benefit from direct exploration of tree parameters. For RDF triples, it would be interesting to see, for example, if a subject and an object of the relation are correctly verbalised as agent and patient respectively, that is they were not swapped. Identifying such cues and also cases with worst performance would enable more meaningful feedback for system developers.

8

Conclusion

The research presented in this thesis has been concerned with some challenges posed in the field of natural language generation. Several areas have drawn our attention: how to perform data collection, how to model NLG systems, and how to evaluate them.

Regarding data collection, we considered how to derive a Russian version of the existing English dataset for data-to-text generation. We showed that translation methods yield significant errors in rendering named entities. Several post-editing techniques were studied to correct falsely translated named entities. This methodology enables to identify the most prominent obstacles in creating new data for generation via machine translation.

As for NLG system development, first we dealt with the issue of handling rare items in data-driven models. We investigated the impact of copying and delexicalisation on standard splits into training, development, and test and on more challenging splits where rare items are never seen during training. Our experiments show that delexicalisation performs better than copying and that copying underperforms for words not seen in the training data. The data partitioning also strongly impacts how rare items are handled by models.

Next, we concentrated on surface realisation, a specific component of NLG systems. We proposed a modular approach for generating sentences from dependency trees, validating it on more than a dozen of different languages. We showed experimentally that word order can be learned without taking into account lexical information. Our model evaluation, based on dependency relations, highlighted cases where the non-consistent head-directionality patterns negatively impact results.

As far as evaluation is concerned, we presented a task-specific framework for error analysis, focused on word order for shallow surface realisation. The framework was applied to the outputs of 174 systems, what permitted us to demonstrate some global results about the state of the art. Our analysis also shows that system performance does not depend on tree characteristics despite being a natural basis for error classification.

Finally, we validated automatic metrics for generation against human judgments. We emphasized that differences in evaluation design—using system-level or sentence-level comparisons—strongly affect results.

Future Directions

As with every research work, there are a lot of open questions and perspectives to explore.

While translating datasets for data-to-text generation in Chapter 3, we witnessed that most errors were related to named entities. A special mechanism, focused on their translation, may

be helpful to ensure better quality. One may think of the use of cross-lingual word embeddings or the extraction of entity mapping from multilingual knowledge bases. In these cases, a neural MT model would be enhanced with an additional representation of entities in two languages. Such setup would also allow to translate datasets not only to one target language, but also to multiple languages, hopefully, with better accuracy.

The methods for treating rare items, presented in Chapter 4, have a restriction that they work for languages with no or few inflection changes. The extension of these methods to other languages would need to explore other strategies, such as using subword representations. Moreover, it would be particularly interesting to see how named entities can be differently realised in generation systems. To this end, a specific module targeted referring expression generation would be useful. Overall, we should strive for methods and approaches beyond English in order to test them in a multilingual setting.

The surface realiser, discussed in Chapter 5, was developed as a stand-alone component. It would be interesting to integrate it into NLG systems which follow the traditional NLG pipeline and to test if an explicit integration of the surface realisation step makes the generation process more robust for NLG applications. Intuitively, using dependency trees as an intermediate representation might allow for better decisions on the microplanning level and for more accurate generation of data. Moreover, as discussed in Section 2.1.2, there is a multitude of other formalisms representing syntax and/or semantics. An explicit incorporation of such representations in modern models may be beneficial to increase models' awareness of language properties.

Regarding NLG evaluation, many issues remain to be solved, for example, the need for better automatic evaluation on the sentence level; more interpretable approaches to evaluation; evaluation methods targeting a particular application; applying evaluation results to drive decisions for creation of better models.

A

Appendix A

Word Order Evaluation

Definitions

- A dependency relation is head-initial (HI) if its head precedes the dependent in the linear order of a sentence. E.g., $walk_{head}$ $slowly_{dep}$.
- A dependency relation is head-final (HF) if its head follows the dependent. E.g., an_{dep} $apple_{head}$.

Column Name Explanation

- count: number of deprel occurrences in the test data;
- found-m: number of found deprels in the model prediction;
- %: found-m in percentage;
- HI: a count of how many relations are head-initial in the test data;
- HI%: HI in percentage;
- HI-m: a count of matches between the model prediction and HI;
- HI-m%: HI-m in percentage w.r.t. HI=100%;
- HF: a count of how many relations are head-final in the test data;
- HF%: HF in percentage;
- HF-m: a count of matches between the model prediction and HF;
- HF-m%: HF-m in percentage w.r.t. HF=100%.

For each language, two tables are supplied: with exact match (head and dependent are in the correct order, and the distance between them is the same in gold and prediction) and approximate match (head and dependent are in the correct order but there is a one-token difference between gold and prediction).

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
nmod	6184	2629	42.5	5866	94.9	2501	42.6	318	5.1	128	40.3
case	4320	1718	39.8	31	0.7	3	9.7	4289	99.3	1715	40
amod	2427	1272	52.4	2425	99.9	1272	52.5	2	0.1	0	0
punct	2046	539	26.3	1621	79.2	450	27.8	425	20.8	89	20.9
cc	1938	797	41.1	477	24.6	105	22	1461	75.4	692	47.4
obj	1923	667	34.7	1919	99.8	667	34.8	4	0.2	0	0
nsubj	1568	704	44.9	1021	65.1	544	53.3	547	34.9	160	29.3
obl	1389	329	23.7	1300	93.6	310	23.8	89	6.4	19	21.3
conj	1219	273	22.4	1219	100	273	22.4	0	0	0	0
mark	764	256	33.5	17	2.2	0	0	747	97.8	256	34.3
acl	555	175	31.5	551	99.3	175	31.8	4	0.7	0	0
parataxis	518	320	61.8	517	99.8	320	61.9	1	0.2	0	0
nummod	378	184	48.7	378	100	184	48.7	0	0	0	0
advmod	319	94	29.5	144	45.1	26	18.1	175	54.9	68	38.9
ccomp	318	93	29.2	318	100	93	29.2	0	0	0	0
advcl	196	24	12.2	165	84.2	23	13.9	31	15.8	1	3.2
det	195	70	35.9	43	22.1	1	2.3	152	77.9	69	45.4
iobj	186	51	27.4	183	98.4	51	27.9	3	1.6	0	0
xcomp	167	40	24	166	99.4	40	24.1	1	0.6	0	0
aux	143	82	57.3	7	4.9	0	0	136	95.1	82	60.3
advmod:emph	127	14	11	34	26.8	4	11.8	93	73.2	10	10.8
dep	101	48	47.5	101	100	48	47.5	0	0	0	0
cop	96	16	16.7	1	1	0	0	95	99	16	16.8
fixed	84	39	46.4	84	100	39	46.4	0	0	0	0
appos	81	16	19.8	68	84	15	22.1	13	16	1	7.7
nsubj:pass	73	38	52.1	52	71.2	33	63.5	21	28.8	5	23.8
csubj	39	7	17.9	39	100	7	17.9	0	0	0	0
flat:foreign	29	12	41.4	2	6.9	0	0	27	93.1	12	44.4
aux:pass	13	7	53.8	0	0	0	0	13	100	7	53.8
total	27396	10514	38.4	18749	68.4	7184	38.3	8647	31.6	3330	38.5

Table A.1: Arabic. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
nmod	6184	3278	53	5866	94.9	3133	53.4	318	5.1	145	45.6
case	4320	1972	45.6	31	0.7	8	25.8	4289	99.3	1964	45.8
amod	2427	1422	58.6	2425	99.9	1422	58.6	2	0.1	0	0
punct	2046	726	35.5	1621	79.2	613	37.8	425	20.8	113	26.6
cc	1938	943	48.7	477	24.6	135	28.3	1461	75.4	808	55.3
obj	1923	920	47.8	1919	99.8	920	47.9	4	0.2	0	0
nsubj	1568	820	52.3	1021	65.1	606	59.4	547	34.9	214	39.1
obl	1389	494	35.6	1300	93.6	472	36.3	89	6.4	22	24.7
conj	1219	408	33.5	1219	100	408	33.5	0	0	0	0
mark	764	330	43.2	17	2.2	0	0	747	97.8	330	44.2
acl	555	226	40.7	551	99.3	226	41	4	0.7	0	0
parataxis	518	339	65.4	517	99.8	339	65.6	1	0.2	0	0
nummod	378	224	59.3	378	100	224	59.3	0	0	0	0
advmod	319	104	32.6	144	45.1	36	25	175	54.9	68	38.9
ccomp	318	133	41.8	318	100	133	41.8	0	0	0	0
advcl	196	46	23.5	165	84.2	45	27.3	31	15.8	1	3.2
det	195	73	37.4	43	22.1	2	4.7	152	77.9	71	46.7
iobj	186	73	39.2	183	98.4	73	39.9	3	1.6	0	0
xcomp	167	55	32.9	166	99.4	55	33.1	1	0.6	0	0
aux	143	82	57.3	7	4.9	0	0	136	95.1	82	60.3
advmod:emph	127	18	14.2	34	26.8	7	20.6	93	73.2	11	11.8
dep	101	51	50.5	101	100	51	50.5	0	0	0	0
cop	96	22	22.9	1	1	0	0	95	99	22	23.2
fixed	84	41	48.8	84	100	41	48.8	0	0	0	0
appos	81	24	29.6	68	84	23	33.8	13	16	1	7.7
nsubj:pass	73	41	56.2	52	71.2	33	63.5	21	28.8	8	38.1
csubj	39	15	38.5	39	100	15	38.5	0	0	0	0
flat:foreign	29	13	44.8	2	6.9	0	0	27	93.1	13	48.1
aux:pass	13	7	53.8	0	0	0	0	13	100	7	53.8
total	27396	12900	47.1	18749	68.4	9020	48.1	8647	31.6	3880	44.9

Table A.2: Arabic. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	24445	13324	54.5	13959	57.1	7541	54	10486	42.9	5783	55.1
nmod	17279	12938	74.9	15995	92.6	12228	76.4	1284	7.4	710	55.3
amod	17271	14351	83.1	1251	7.2	403	32.2	16020	92.8	13948	87.1
case	15839	13705	86.5	6	0	0	0	15833	100	13705	86.6
nsubj	9820	4822	49.1	3072	31.3	1306	42.5	6748	68.7	3516	52.1
obl	9728	3925	40.3	5194	53.4	2242	43.2	4534	46.6	1683	37.1
obj	9139	4673	51.1	6343	69.4	3704	58.4	2796	30.6	969	34.7
advmod	7117	3439	48.3	1587	22.3	346	21.8	5530	77.7	3093	55.9
conj	7099	3806	53.6	7099	100	3806	53.6	0	0	0	0
cc	5854	3755	64.1	190	3.2	52	27.4	5664	96.8	3703	65.4
det	3328	2859	85.9	81	2.4	20	24.7	3247	97.6	2839	87.4
mark	3318	1835	55.3	84	2.5	66	78.6	3234	97.5	1769	54.7
advmod:emph	2544	1832	72	28	1.1	2	7.1	2516	98.9	1830	72.7
cop	2508	1427	56.9	394	15.7	201	51	2114	84.3	1226	58
acl	2492	1168	46.9	2470	99.1	1166	47.2	22	0.9	2	9.1
xcomp	2132	1053	49.4	1971	92.4	1019	51.7	161	7.6	34	21.1
nummod	2051	1125	54.9	889	43.3	399	44.9	1162	56.7	726	62.5
expl:pv	1901	1051	55.3	357	18.8	232	65	1544	81.2	819	53
aux	1749	925	52.9	219	12.5	137	62.6	1530	87.5	788	51.5
dep	1515	784	51.7	1368	90.3	762	55.7	147	9.7	22	15
flat	1468	1234	84.1	1468	100	1234	84.1	0	0	0	0
advcl	1274	365	28.6	790	62	240	30.4	484	38	125	25.8
ccomp	1152	464	40.3	940	81.6	393	41.8	212	18.4	71	33.5
iobj	961	406	42.2	576	59.9	263	45.7	385	40.1	143	37.1
nsubj:pass	856	388	45.3	347	40.5	157	45.2	509	59.5	231	45.4
appos	807	385	47.7	805	99.8	385	47.8	2	0.2	0	0
nummod:gov	767	640	83.4	17	2.2	2	11.8	750	97.8	638	85.1
aux:pass	668	362	54.2	12	1.8	0	0	656	98.2	362	55.2
csubj	607	276	45.5	548	90.3	264	48.2	59	9.7	12	20.3
expl:pass	535	289	54	83	15.5	44	53	452	84.5	245	54.2
fixed	494	460	93.1	494	100	460	93.1	0	0	0	0
flat:foreign	315	127	40.3	63	20	19	30.2	252	80	108	42.9
parataxis	216	48	22.2	154	71.3	35	22.7	62	28.7	13	21
compound	201	133	66.2	153	76.1	99	64.7	48	23.9	34	70.8
det:numgov	116	102	87.9	4	3.4	0	0	112	96.6	102	91.1
det:nummod	70	50	71.4	5	7.1	2	40	65	92.9	48	73.8
csubj:pass	49	26	53.1	45	91.8	26	57.8	4	8.2	0	0
discourse	45	14	31.1	9	20	3	33.3	36	80	11	30.6
vocative	10	4	40	4	40	1	25	6	60	3	50
total	157740	98570	62.5	69074	43.8	39259	56.8	88666	56.2	59311	66.9

Table A.3: Czech. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	24445	16909	69.2	13959	57.1	9688	69.4	10486	42.9	7221	68.9
nmod	17279	14874	86.1	15995	92.6	14072	88	1284	7.4	802	62.5
amod	17271	15056	87.2	1251	7.2	506	40.4	16020	92.8	14550	90.8
case	15839	14745	93.1	6	0	0	0	15833	100	14745	93.1
nsubj	9820	6165	62.8	3072	31.3	1596	52	6748	68.7	4569	67.7
obl	9728	5226	53.7	5194	53.4	2965	57.1	4534	46.6	2261	49.9
obj	9139	5810	63.6	6343	69.4	4520	71.3	2796	30.6	1290	46.1
advmod	7117	4132	58.1	1587	22.3	419	26.4	5530	77.7	3713	67.1
conj	7099	4826	68	7099	100	4826	68	0	0	0	0
cc	5854	4467	76.3	190	3.2	67	35.3	5664	96.8	4400	77.7
det	3328	3009	90.4	81	2.4	21	25.9	3247	97.6	2988	92
mark	3318	2317	69.8	84	2.5	66	78.6	3234	97.5	2251	69.6
advmod:emph	2544	2064	81.1	28	1.1	4	14.3	2516	98.9	2060	81.9
cop	2508	1857	74	394	15.7	231	58.6	2114	84.3	1626	76.9
acl	2492	1536	61.6	2470	99.1	1532	62	22	0.9	4	18.2
xcomp	2132	1412	66.2	1971	92.4	1368	69.4	161	7.6	44	27.3
nummod	2051	1303	63.5	889	43.3	461	51.9	1162	56.7	842	72.5
expl:pv	1901	1297	68.2	357	18.8	234	65.5	1544	81.2	1063	68.8
aux	1749	1216	69.5	219	12.5	141	64.4	1530	87.5	1075	70.3
dep	1515	999	65.9	1368	90.3	962	70.3	147	9.7	37	25.2
flat	1468	1339	91.2	1468	100	1339	91.2	0	0	0	0
advcl	1274	583	45.8	790	62	397	50.3	484	38	186	38.4
ccomp	1152	633	54.9	940	81.6	533	56.7	212	18.4	100	47.2
iobj	961	548	57	576	59.9	352	61.1	385	40.1	196	50.9
nsubj:pass	856	496	57.9	347	40.5	188	54.2	509	59.5	308	60.5
appos	807	537	66.5	805	99.8	537	66.7	2	0.2	0	0
nummod:gov	767	684	89.2	17	2.2	4	23.5	750	97.8	680	90.7
aux:pass	668	475	71.1	12	1.8	0	0	656	98.2	475	72.4
csubj	607	368	60.6	548	90.3	352	64.2	59	9.7	16	27.1
expl:pass	535	374	69.9	83	15.5	44	53	452	84.5	330	73
fixed	494	462	93.5	494	100	462	93.5	0	0	0	0
flat:foreign	315	155	49.2	63	20	26	41.3	252	80	129	51.2
parataxis	216	82	38	154	71.3	64	41.6	62	28.7	18	29
compound	201	140	69.7	153	76.1	106	69.3	48	23.9	34	70.8
det:numgov	116	105	90.5	4	3.4	0	0	112	96.6	105	93.8
det:nummod	70	54	77.1	5	7.1	2	40	65	92.9	52	80
csubj:pass	49	29	59.2	45	91.8	28	62.2	4	8.2	1	25
discourse	45	25	55.6	9	20	5	55.6	36	80	20	55.6
vocative	10	5	50	4	40	2	50	6	60	3	50
total	157740	116314	73.7	69074	43.8	48120	69.7	88666	56.2	68194	76.9

Table A.4: Czech. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	3031	1530	50.5	2187	72.2	1205	55.1	844	27.8	325	38.5
case	1961	1429	72.9	78	4	54	69.2	1883	96	1375	73
nsubj	1951	1383	70.9	87	4.5	50	57.5	1864	95.5	1333	71.5
det	1817	1408	77.5	4	0.2	0	0	1813	99.8	1408	77.7
advmod	1262	613	48.6	305	24.2	108	35.4	957	75.8	505	52.8
obj	1169	892	76.3	1135	97.1	882	77.7	34	2.9	10	29.4
amod	1162	820	70.6	36	3.1	10	27.8	1126	96.9	810	71.9
compound	1147	720	62.8	22	1.9	5	22.7	1125	98.1	715	63.6
obl	1044	555	53.2	963	92.2	533	55.3	81	7.8	22	27.2
conj	856	370	43.2	846	98.8	366	43.3	10	1.2	4	40
aux	809	560	69.2	2	0.2	0	0	807	99.8	560	69.4
mark	769	569	74	0	0	0	0	769	100	569	74
nmod	764	471	61.6	757	99.1	471	62.2	7	0.9	0	0
cc	747	499	66.8	0	0	0	0	747	100	499	66.8
cop	554	396	71.5	36	6.5	20	55.6	518	93.5	376	72.6
nmod:poss	384	313	81.5	0	0	0	0	384	100	313	81.5
advcl	367	163	44.4	285	77.7	133	46.7	82	22.3	30	36.6
xcomp	343	248	72.3	343	100	248	72.3	0	0	0	0
nummod	277	151	54.5	73	26.4	34	46.6	204	73.6	117	57.4
list	251	41	16.3	249	99.2	41	16.5	2	0.8	0	0
flat	247	179	72.5	247	100	179	72.5	0	0	0	0
ccomp	241	115	47.7	233	96.7	115	49.4	8	3.3	0	0
acl:relcl	212	105	49.5	212	100	105	49.5	0	0	0	0
parataxis	199	57	28.6	188	94.5	55	29.3	11	5.5	2	18.2
appos	186	64	34.4	178	95.7	62	34.8	8	4.3	2	25
acl	173	92	53.2	172	99.4	92	53.5	1	0.6	0	0
discourse	122	66	54.1	39	32	16	41	83	68	50	60.2
aux:pass	114	79	69.3	4	3.5	1	25	110	96.5	78	70.9
nsubj:pass	100	61	61	4	4	0	0	96	96	61	63.5
compound:prt	87	61	70.1	87	100	61	70.1	0	0	0	0
obl:tmod	66	28	42.4	60	90.9	28	46.7	6	9.1	0	0
expl	64	39	60.9	8	12.5	0	0	56	87.5	39	69.6
fixed	62	47	75.8	62	100	47	75.8	0	0	0	0
obl:npmod	48	22	45.8	14	29.2	4	28.6	34	70.8	18	52.9
iobj	40	37	92.5	40	100	37	92.5	0	0	0	0
nmod:tmod	37	23	62.2	37	100	23	62.2	0	0	0	0
csubj	23	5	21.7	16	69.6	3	18.8	7	30.4	2	28.6
det:predet	22	18	81.8	0	0	0	0	22	100	18	81.8
vocative	20	14	70	11	55	10	90.9	9	45	4	44.4
nmod:npmod	13	3	23.1	9	69.2	2	22.2	4	30.8	1	25
cc:preconj	9	1	11.1	1	11.1	0	0	8	88.9	1	12.5
flat:foreign	7	0	0	3	42.9	0	0	4	57.1	0	0
dep	1	0	0	0	0	0	0	1	100	0	0
csubj:pass	1	0	0	0	0	0	0	1	100	0	0
total	22759	14247	62.6	9033	39.7	5000	55.4	13726	60.3	9247	67.4

Table A.5: English. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	3031	1962	64.7	2187	72.2	1524	69.7	844	27.8	438	51.9
case	1961	1634	83.3	78	4	57	73.1	1883	96	1577	83.7
nsubj	1951	1667	85.4	87	4.5	58	66.7	1864	95.5	1609	86.3
det	1817	1537	84.6	4	0.2	0	0	1813	99.8	1537	84.8
advmod	1262	725	57.4	305	24.2	130	42.6	957	75.8	595	62.2
obj	1169	986	84.3	1135	97.1	975	85.9	34	2.9	11	32.4
amod	1162	880	75.7	36	3.1	10	27.8	1126	96.9	870	77.3
compound	1147	820	71.5	22	1.9	5	22.7	1125	98.1	815	72.4
obl	1044	698	66.9	963	92.2	667	69.3	81	7.8	31	38.3
conj	856	504	58.9	846	98.8	499	59	10	1.2	5	50
aux	809	714	88.3	2	0.2	0	0	807	99.8	714	88.5
mark	769	623	81	0	0	0	0	769	100	623	81
nmod	764	583	76.3	757	99.1	583	77	7	0.9	0	0
cc	747	597	79.9	0	0	0	0	747	100	597	79.9
cop	554	476	85.9	36	6.5	21	58.3	518	93.5	455	87.8
nmod:poss	384	336	87.5	0	0	0	0	384	100	336	87.5
advcl	367	211	57.5	285	77.7	170	59.6	82	22.3	41	50
xcomp	343	290	84.5	343	100	290	84.5	0	0	0	0
nummod	277	173	62.5	73	26.4	36	49.3	204	73.6	137	67.2
list	251	81	32.3	249	99.2	81	32.5	2	0.8	0	0
flat	247	203	82.2	247	100	203	82.2	0	0	0	0
ccomp	241	162	67.2	233	96.7	161	69.1	8	3.3	1	12.5
acl:relcl	212	135	63.7	212	100	135	63.7	0	0	0	0
parataxis	199	101	50.8	188	94.5	98	52.1	11	5.5	3	27.3
appos	186	85	45.7	178	95.7	83	46.6	8	4.3	2	25
acl	173	112	64.7	172	99.4	112	65.1	1	0.6	0	0
discourse	122	82	67.2	39	32	19	48.7	83	68	63	75.9
aux:pass	114	95	83.3	4	3.5	1	25	110	96.5	94	85.5
nsubj:pass	100	79	79	4	4	1	25	96	96	78	81.2
compound:prt	87	65	74.7	87	100	65	74.7	0	0	0	0
obl:tmod	66	37	56.1	60	90.9	36	60	6	9.1	1	16.7
expl	64	47	73.4	8	12.5	0	0	56	87.5	47	83.9
fixed	62	49	79	62	100	49	79	0	0	0	0
obl:npmod	48	24	50	14	29.2	5	35.7	34	70.8	19	55.9
iobj	40	37	92.5	40	100	37	92.5	0	0	0	0
nmod:tmod	37	31	83.8	37	100	31	83.8	0	0	0	0
csubj	23	6	26.1	16	69.6	4	25	7	30.4	2	28.6
det:predet	22	20	90.9	0	0	0	0	22	100	20	90.9
vocative	20	17	85	11	55	11	100	9	45	6	66.7
nmod:npmod	13	4	30.8	9	69.2	3	33.3	4	30.8	1	25
cc:preconj	9	2	22.2	1	11.1	0	0	8	88.9	2	25
flat:foreign	7	0	0	3	42.9	0	0	4	57.1	0	0
dep	1	0	0	0	0	0	0	1	100	0	0
csubj:pass	1	0	0	0	0	0	0	1	100	0	0
total	22759	16890	74.2	9033	39.7	6160	68.2	13726	60.3	10730	78.2

Table A.6: English. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
case	7228	5091	70.4	118	1.6	36	30.5	7110	98.4	5055	71.1
det	7135	5183	72.6	103	1.4	53	51.5	7032	98.6	5130	73
punct	6306	2560	40.6	3594	57	1512	42.1	2712	43	1048	38.6
nmod	3586	2014	56.2	3495	97.5	1990	56.9	91	2.5	24	26.4
obj	3522	1914	54.3	2631	74.7	1371	52.1	891	25.3	543	60.9
nsubj	2875	1591	55.3	481	16.7	188	39.1	2394	83.3	1403	58.6
amod	2821	1222	43.3	2005	71.1	1049	52.3	816	28.9	173	21.2
obl	2434	983	40.4	1781	73.2	749	42.1	653	26.8	234	35.8
advmod	1913	734	38.4	695	36.3	246	35.4	1218	63.7	488	40.1
mark	1865	866	46.4	9	0.5	1	11.1	1856	99.5	865	46.6
conj	1543	550	35.6	1543	100	550	35.6	0	0	0	0
cc	1440	817	56.7	21	1.5	1	4.8	1419	98.5	816	57.5
flat	1433	1010	70.5	1433	100	1010	70.5	0	0	0	0
acl	977	385	39.4	977	100	385	39.4	0	0	0	0
aux	954	620	65	16	1.7	1	6.2	938	98.3	619	66
advcl	797	183	23	684	85.8	164	24	113	14.2	19	16.8
appos	727	315	43.3	721	99.2	315	43.7	6	0.8	0	0
fixed	708	575	81.2	708	100	575	81.2	0	0	0	0
ccomp	585	203	34.7	476	81.4	177	37.2	109	18.6	26	23.9
nummod	578	360	62.3	153	26.5	51	33.3	425	73.5	309	72.7
cop	533	303	56.8	16	3	1	6.2	517	97	302	58.4
xcomp	256	131	51.2	246	96.1	128	52	10	3.9	3	30
compound	234	142	60.7	234	100	142	60.7	0	0	0	0
iobj	164	91	55.5	79	48.2	42	53.2	85	51.8	49	57.6
csubj	98	29	29.6	70	71.4	23	32.9	28	28.6	6	21.4
parataxis	80	22	27.5	41	51.2	10	24.4	39	48.8	12	30.8
expl:pass	47	31	66	0	0	0	0	47	100	31	66
dep	24	6	25	15	62.5	4	26.7	9	37.5	2	22.2
nsubj:pass	5	1	20	3	60	0	0	2	40	1	50
total	50868	27932	54.9	22348	43.9	10774	48.2	28520	56.1	17158	60.2

Table A.7: Spanish. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
case	7228	5972	82.6	118	1.6	55	46.6	7110	98.4	5917	83.2
det	7135	5895	82.6	103	1.4	62	60.2	7032	98.6	5833	82.9
punct	6306	3586	56.9	3594	57	2158	60	2712	43	1428	52.7
nmod	3586	2688	75	3495	97.5	2658	76.1	91	2.5	30	33
obj	3522	2467	70	2631	74.7	1865	70.9	891	25.3	602	67.6
nsubj	2875	2046	71.2	481	16.7	240	49.9	2394	83.3	1806	75.4
amod	2821	1381	49	2005	71.1	1194	59.6	816	28.9	187	22.9
obl	2434	1382	56.8	1781	73.2	1049	58.9	653	26.8	333	51
advmod	1913	928	48.5	695	36.3	292	42	1218	63.7	636	52.2
mark	1865	1143	61.3	9	0.5	4	44.4	1856	99.5	1139	61.4
conj	1543	810	52.5	1543	100	810	52.5	0	0	0	0
cc	1440	1022	71	21	1.5	4	19	1419	98.5	1018	71.7
flat	1433	1135	79.2	1433	100	1135	79.2	0	0	0	0
acl	977	569	58.2	977	100	569	58.2	0	0	0	0
aux	954	701	73.5	16	1.7	5	31.2	938	98.3	696	74.2
advcl	797	302	37.9	684	85.8	269	39.3	113	14.2	33	29.2
appos	727	428	58.9	721	99.2	428	59.4	6	0.8	0	0
fixed	708	579	81.8	708	100	579	81.8	0	0	0	0
ccomp	585	307	52.5	476	81.4	261	54.8	109	18.6	46	42.2
nummod	578	383	66.3	153	26.5	67	43.8	425	73.5	316	74.4
cop	533	402	75.4	16	3	1	6.2	517	97	401	77.6
xcomp	256	174	68	246	96.1	171	69.5	10	3.9	3	30
compound	234	159	67.9	234	100	159	67.9	0	0	0	0
iobj	164	103	62.8	79	48.2	50	63.3	85	51.8	53	62.4
csubj	98	42	42.9	70	71.4	31	44.3	28	28.6	11	39.3
parataxis	80	34	42.5	41	51.2	16	39	39	48.8	18	46.2
expl:pass	47	36	76.6	0	0	0	0	47	100	36	76.6
dep	24	10	41.7	15	62.5	8	53.3	9	37.5	2	22.2
nsubj:pass	5	2	40	3	60	1	33.3	2	40	1	50
total	50868	34686	68.2	22348	43.9	14141	63.3	28520	56.1	20545	72

Table A.8: Spanish. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	3015	1422	47.2	1776	58.9	811	45.7	1239	41.1	611	49.3
obl	1759	671	38.1	1036	58.9	404	39	723	41.1	267	36.9
advmod	1514	607	40.1	424	28	121	28.5	1090	72	486	44.6
nsubj	1196	771	64.5	137	11.5	20	14.6	1059	88.5	751	70.9
obj	1179	546	46.3	841	71.3	421	50.1	338	28.7	125	37
conj	1134	437	38.5	1134	100	437	38.5	0	0	0	0
nmod:poss	1028	674	65.6	0	0	0	0	1028	100	674	65.6
amod	933	656	70.3	1	0.1	1	100	932	99.9	655	70.3
cc	910	543	59.7	1	0.1	0	0	909	99.9	543	59.7
cop	580	344	59.3	106	18.3	50	47.2	474	81.7	294	62
aux	559	353	63.1	19	3.4	8	42.1	540	96.6	345	63.9
nsubj:cop	523	262	50.1	132	25.2	52	39.4	391	74.8	210	53.7
nmod	500	181	36.2	356	71.2	159	44.7	144	28.8	22	15.3
mark	428	214	50	0	0	0	0	428	100	214	50
acl	359	180	50.1	50	13.9	9	18	309	86.1	171	55.3
advcl	353	95	26.9	252	71.4	82	32.5	101	28.6	13	12.9
nummod	339	165	48.7	89	26.3	38	42.7	250	73.7	127	50.8
det	307	219	71.3	4	1.3	0	0	303	98.7	219	72.3
flat:name	287	216	75.3	287	100	216	75.3	0	0	0	0
case	267	174	65.2	239	89.5	171	71.5	28	10.5	3	10.7
xcomp	234	143	61.1	220	94	136	61.8	14	6	7	50
ccomp	228	78	34.2	227	99.6	78	34.4	1	0.4	0	0
acl:relcl	208	72	34.6	208	100	72	34.6	0	0	0	0
nmod:gobj	204	126	61.8	0	0	0	0	204	100	126	61.8
xcomp:ds	120	64	53.3	111	92.5	61	55	9	7.5	3	33.3
flat	118	79	66.9	118	100	79	66.9	0	0	0	0
appos	100	34	34	100	100	34	34	0	0	0	0
compound:nn	93	62	66.7	2	2.2	2	100	91	97.8	60	65.9
aux:pass	93	63	67.7	1	1.1	0	0	92	98.9	63	68.5
parataxis	77	18	23.4	76	98.7	18	23.7	1	1.3	0	0
fixed	58	36	62.1	58	100	36	62.1	0	0	0	0
nmod:gsubj	54	25	46.3	0	0	0	0	54	100	25	46.3
cop:own	36	25	69.4	33	91.7	25	75.8	3	8.3	0	0
compound:prt	30	16	53.3	16	53.3	11	68.8	14	46.7	5	35.7
csubj:cop	22	10	45.5	21	95.5	10	47.6	1	4.5	0	0
discourse	19	6	31.6	6	31.6	0	0	13	68.4	6	46.2
cc:preconj	17	6	35.3	0	0	0	0	17	100	6	35.3
compound	11	6	54.5	0	0	0	0	11	100	6	54.5
vocative	10	3	30	3	30	1	33.3	7	70	2	28.6
csubj	2	0	0	1	50	0	0	1	50	0	0
dep	1	0	0	0	0	0	0	1	100	0	0
total	18905	9602	50.8	8085	42.8	3563	44.1	10820	57.2	6039	55.8

Table A.9: Finnish. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	3015	1981	65.7	1776	58.9	1178	66.3	1239	41.1	803	64.8
obl	1759	951	54.1	1036	58.9	603	58.2	723	41.1	348	48.1
advmod	1514	797	52.6	424	28	170	40.1	1090	72	627	57.5
nsubj	1196	893	74.7	137	11.5	28	20.4	1059	88.5	865	81.7
obj	1179	712	60.4	841	71.3	555	66	338	28.7	157	46.4
conj	1134	642	56.6	1134	100	642	56.6	0	0	0	0
nmod:poss	1028	764	74.3	0	0	0	0	1028	100	764	74.3
amod	933	713	76.4	1	0.1	1	100	932	99.9	712	76.4
cc	910	682	74.9	1	0.1	0	0	909	99.9	682	75
cop	580	436	75.2	106	18.3	57	53.8	474	81.7	379	80
aux	559	459	82.1	19	3.4	9	47.4	540	96.6	450	83.3
nsubj:cop	523	354	67.7	132	25.2	66	50	391	74.8	288	73.7
nmod	500	241	48.2	356	71.2	213	59.8	144	28.8	28	19.4
mark	428	292	68.2	0	0	0	0	428	100	292	68.2
acl	359	218	60.7	50	13.9	11	22	309	86.1	207	67
advcl	353	166	47	252	71.4	140	55.6	101	28.6	26	25.7
nummod	339	190	56	89	26.3	39	43.8	250	73.7	151	60.4
det	307	249	81.1	4	1.3	1	25	303	98.7	248	81.8
flat:name	287	227	79.1	287	100	227	79.1	0	0	0	0
case	267	186	69.7	239	89.5	181	75.7	28	10.5	5	17.9
xcomp	234	182	77.8	220	94	175	79.5	14	6	7	50
ccomp	228	126	55.3	227	99.6	125	55.1	1	0.4	1	100
acl:relcl	208	114	54.8	208	100	114	54.8	0	0	0	0
nmod:gobj	204	144	70.6	0	0	0	0	204	100	144	70.6
xcomp:ds	120	80	66.7	111	92.5	77	69.4	9	7.5	3	33.3
flat	118	81	68.6	118	100	81	68.6	0	0	0	0
appos	100	49	49	100	100	49	49	0	0	0	0
compound:nn	93	69	74.2	2	2.2	2	100	91	97.8	67	73.6
aux:pass	93	74	79.6	1	1.1	0	0	92	98.9	74	80.4
parataxis	77	40	51.9	76	98.7	40	52.6	1	1.3	0	0
fixed	58	37	63.8	58	100	37	63.8	0	0	0	0
nmod:gsubj	54	29	53.7	0	0	0	0	54	100	29	53.7
cop:own	36	26	72.2	33	91.7	26	78.8	3	8.3	0	0
compound:prt	30	18	60	16	53.3	13	81.2	14	46.7	5	35.7
csubj:cop	22	17	77.3	21	95.5	17	81	1	4.5	0	0
discourse	19	10	52.6	6	31.6	1	16.7	13	68.4	9	69.2
cc:preconj	17	8	47.1	0	0	0	0	17	100	8	47.1
compound	11	6	54.5	0	0	0	0	11	100	6	54.5
vocative	10	4	40	3	30	1	33.3	7	70	3	42.9
csubj	2	0	0	1	50	0	0	1	50	0	0
dep	1	0	0	0	0	0	0	1	100	0	0
total	18905	12267	64.9	8085	42.8	4879	60.3	10820	57.2	7388	68.3

Table A.10: Finnish. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
det	1356	1025	75.6	0	0	0	0	1356	100	1025	75.6
case	1282	960	74.9	0	0	0	0	1282	100	960	74.9
punct	1195	464	38.8	808	67.6	305	37.7	387	32.4	159	41.1
nmod	770	428	55.6	754	97.9	424	56.2	16	2.1	4	25
nsubj	573	343	59.9	26	4.5	2	7.7	547	95.5	341	62.3
obl	563	254	45.1	474	84.2	218	46	89	15.8	36	40.4
amod	480	221	46	365	76	202	55.3	115	24	19	16.5
advmod	458	202	44.1	125	27.3	58	46.4	333	72.7	144	43.2
obj	388	232	59.8	309	79.6	183	59.2	79	20.4	49	62
conj	331	123	37.2	331	100	123	37.2	0	0	0	0
cc	258	159	61.6	0	0	0	0	258	100	159	61.6
mark	231	152	65.8	2	0.9	0	0	229	99.1	152	66.4
aux	187	127	67.9	0	0	0	0	187	100	127	67.9
fixed	178	132	74.2	178	100	132	74.2	0	0	0	0
acl	170	87	51.2	160	94.1	86	53.8	10	5.9	1	10
nummod	146	91	62.3	18	12.3	2	11.1	128	87.7	89	69.5
cop	132	81	61.4	3	2.3	0	0	129	97.7	81	62.8
flat:name	119	92	77.3	119	100	92	77.3	0	0	0	0
appos	118	51	43.2	115	97.5	49	42.6	3	2.5	2	66.7
nmod:poss	113	89	78.8	0	0	0	0	113	100	89	78.8
advcl	94	27	28.7	62	66	18	29	32	34	9	28.1
acl:relcl	75	21	28	75	100	21	28	0	0	0	0
ccomp	60	18	30	51	85	17	33.3	9	15	1	11.1
expl	56	27	48.2	10	17.9	4	40	46	82.1	23	50
nsubj:pass	54	33	61.1	0	0	0	0	54	100	33	61.1
aux:pass	50	38	76	0	0	0	0	50	100	38	76
xcomp	45	25	55.6	42	93.3	25	59.5	3	6.7	0	0
compound	44	14	31.8	34	77.3	12	35.3	10	22.7	2	20
parataxis	32	7	21.9	26	81.2	7	26.9	6	18.8	0	0
iobj	27	10	37	0	0	0	0	27	100	10	37
dep	13	3	23.1	11	84.6	3	27.3	2	15.4	0	0
discourse	5	2	40	1	20	0	0	4	80	2	50
csubj	1	0	0	0	0	0	0	1	100	0	0
total	9604	5538	57.7	4099	42.7	1983	48.4	5505	57.3	3555	64.6

Table A.11: French. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
det	1356	1159	85.5	0	0	0	0	1356	100	1159	85.5
case	1282	1123	87.6	0	0	0	0	1282	100	1123	87.6
punct	1195	653	54.6	808	67.6	448	55.4	387	32.4	205	53
nmod	770	590	76.6	754	97.9	586	77.7	16	2.1	4	25
nsubj	573	445	77.7	26	4.5	4	15.4	547	95.5	441	80.6
obl	563	344	61.1	474	84.2	300	63.3	89	15.8	44	49.4
amod	480	244	50.8	365	76	223	61.1	115	24	21	18.3
advmod	458	256	55.9	125	27.3	70	56	333	72.7	186	55.9
obj	388	289	74.5	309	79.6	225	72.8	79	20.4	64	81
conj	331	182	55	331	100	182	55	0	0	0	0
cc	258	195	75.6	0	0	0	0	258	100	195	75.6
mark	231	176	76.2	2	0.9	0	0	229	99.1	176	76.9
aux	187	152	81.3	0	0	0	0	187	100	152	81.3
fixed	178	143	80.3	178	100	143	80.3	0	0	0	0
acl	170	116	68.2	160	94.1	113	70.6	10	5.9	3	30
nummod	146	92	63	18	12.3	2	11.1	128	87.7	90	70.3
cop	132	101	76.5	3	2.3	0	0	129	97.7	101	78.3
flat:name	119	94	79	119	100	94	79	0	0	0	0
appos	118	72	61	115	97.5	70	60.9	3	2.5	2	66.7
nmod:poss	113	100	88.5	0	0	0	0	113	100	100	88.5
advcl	94	50	53.2	62	66	33	53.2	32	34	17	53.1
acl:relcl	75	41	54.7	75	100	41	54.7	0	0	0	0
ccomp	60	26	43.3	51	85	24	47.1	9	15	2	22.2
expl	56	36	64.3	10	17.9	4	40	46	82.1	32	69.6
nsubj:pass	54	40	74.1	0	0	0	0	54	100	40	74.1
aux:pass	50	45	90	0	0	0	0	50	100	45	90
xcomp	45	35	77.8	42	93.3	35	83.3	3	6.7	0	0
compound	44	16	36.4	34	77.3	14	41.2	10	22.7	2	20
parataxis	32	13	40.6	26	81.2	12	46.2	6	18.8	1	16.7
iobj	27	13	48.1	0	0	0	0	27	100	13	48.1
dep	13	4	30.8	11	84.6	3	27.3	2	15.4	1	50
discourse	5	2	40	1	20	0	0	4	80	2	50
csubj	1	0	0	0	0	0	0	1	100	0	0
total	9604	6847	71.3	4099	42.7	2626	64.1	5505	57.3	4221	76.7

Table A.12: French. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
det	1623	1192	73.4	0	0	0	0	1623	100	1192	73.4
case	1535	1008	65.7	0	0	0	0	1535	100	1008	65.7
punct	1170	532	45.5	952	81.4	446	46.8	218	18.6	86	39.4
nmod	843	435	51.6	823	97.6	432	52.5	20	2.4	3	15
obl	652	264	40.5	502	77	207	41.2	150	23	57	38
amod	570	227	39.8	383	67.2	190	49.6	187	32.8	37	19.8
nsubj	429	252	58.7	111	25.9	69	62.2	318	74.1	183	57.5
advmod	365	139	38.1	70	19.2	15	21.4	295	80.8	124	42
obj	336	194	57.7	295	87.8	171	58	41	12.2	23	56.1
conj	304	103	33.9	304	100	103	33.9	0	0	0	0
cc	262	143	54.6	4	1.5	0	0	258	98.5	143	55.4
aux	192	154	80.2	0	0	0	0	192	100	154	80.2
mark	187	99	52.9	0	0	0	0	187	100	99	52.9
advcl	128	29	22.7	91	71.1	18	19.8	37	28.9	11	29.7
acl:relcl	121	42	34.7	121	100	42	34.7	0	0	0	0
flat:name	116	100	86.2	116	100	100	86.2	0	0	0	0
nummod	114	47	41.2	37	32.5	11	29.7	77	67.5	36	46.8
acl	110	57	51.8	106	96.4	57	53.8	4	3.6	0	0
aux:pass	105	84	80	0	0	0	0	105	100	84	80
cop	105	65	61.9	27	25.7	24	88.9	78	74.3	41	52.6
nsubj:pass	101	50	49.5	35	34.7	13	37.1	66	65.3	37	56.1
expl	74	51	68.9	14	18.9	3	21.4	60	81.1	48	80
xcomp	74	32	43.2	73	98.6	32	43.8	1	1.4	0	0
det:poss	61	44	72.1	1	1.6	0	0	60	98.4	44	73.3
obl:agent	46	23	50	44	95.7	23	52.3	2	4.3	0	0
fixed	43	34	79.1	43	100	34	79.1	0	0	0	0
appos	40	15	37.5	40	100	15	37.5	0	0	0	0
ccomp	35	8	22.9	35	100	8	22.9	0	0	0	0
compound	23	14	60.9	23	100	14	60.9	0	0	0	0
flat	20	17	85	20	100	17	85	0	0	0	0
expl:impers	20	17	85	0	0	0	0	20	100	17	85
iobj	20	11	55	6	30	3	50	14	70	8	57.1
det:predet	15	11	73.3	0	0	0	0	15	100	11	73.3
parataxis	14	3	21.4	11	78.6	3	27.3	3	21.4	0	0
expl:pass	11	8	72.7	1	9.1	0	0	10	90.9	8	80
flat:foreign	6	4	66.7	6	100	4	66.7	0	0	0	0
csubj	3	1	33.3	3	100	1	33.3	0	0	0	0
vocative	3	1	33.3	0	0	0	0	3	100	1	33.3
dep	1	0	0	0	0	0	0	1	100	0	0
dislocated	1	1	100	0	0	0	0	1	100	1	100
total	9878	5511	55.8	4297	43.5	2055	47.8	5581	56.5	3456	61.9

Table A.13: Italian. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
det	1623	1338	82.4	0	0	0	0	1623	100	1338	82.4
case	1535	1240	80.8	0	0	0	0	1535	100	1240	80.8
punct	1170	706	60.3	952	81.4	587	61.7	218	18.6	119	54.6
nmod	843	603	71.5	823	97.6	600	72.9	20	2.4	3	15
obl	652	366	56.1	502	77	295	58.8	150	23	71	47.3
amod	570	262	46	383	67.2	220	57.4	187	32.8	42	22.5
nsubj	429	301	70.2	111	25.9	77	69.4	318	74.1	224	70.4
advmod	365	181	49.6	70	19.2	17	24.3	295	80.8	164	55.6
obj	336	245	72.9	295	87.8	219	74.2	41	12.2	26	63.4
conj	304	148	48.7	304	100	148	48.7	0	0	0	0
cc	262	178	67.9	4	1.5	0	0	258	98.5	178	69
aux	192	175	91.1	0	0	0	0	192	100	175	91.1
mark	187	122	65.2	0	0	0	0	187	100	122	65.2
advcl	128	51	39.8	91	71.1	34	37.4	37	28.9	17	45.9
acl:relcl	121	58	47.9	121	100	58	47.9	0	0	0	0
flat:name	116	101	87.1	116	100	101	87.1	0	0	0	0
nummod	114	50	43.9	37	32.5	11	29.7	77	67.5	39	50.6
acl	110	71	64.5	106	96.4	71	67	4	3.6	0	0
aux:pass	105	98	93.3	0	0	0	0	105	100	98	93.3
cop	105	82	78.1	27	25.7	24	88.9	78	74.3	58	74.4
nsubj:pass	101	66	65.3	35	34.7	17	48.6	66	65.3	49	74.2
expl	74	54	73	14	18.9	3	21.4	60	81.1	51	85
xcomp	74	43	58.1	73	98.6	43	58.9	1	1.4	0	0
det:poss	61	48	78.7	1	1.6	0	0	60	98.4	48	80
obl:agent	46	32	69.6	44	95.7	32	72.7	2	4.3	0	0
fixed	43	34	79.1	43	100	34	79.1	0	0	0	0
appos	40	20	50	40	100	20	50	0	0	0	0
ccomp	35	11	31.4	35	100	11	31.4	0	0	0	0
compound	23	18	78.3	23	100	18	78.3	0	0	0	0
flat	20	17	85	20	100	17	85	0	0	0	0
expl:impers	20	18	90	0	0	0	0	20	100	18	90
iobj	20	12	60	6	30	3	50	14	70	9	64.3
det:predet	15	12	80	0	0	0	0	15	100	12	80
parataxis	14	6	42.9	11	78.6	6	54.5	3	21.4	0	0
expl:pass	11	8	72.7	1	9.1	0	0	10	90.9	8	80
flat:foreign	6	5	83.3	6	100	5	83.3	0	0	0	0
csubj	3	2	66.7	3	100	2	66.7	0	0	0	0
vocative	3	2	66.7	0	0	0	0	3	100	2	66.7
dep	1	0	0	0	0	0	0	1	100	0	0
dislocated	1	1	100	0	0	0	0	1	100	1	100
total	9878	6785	68.7	4297	43.5	2673	62.2	5581	56.5	4112	73.7

Table A.14: Italian. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	1191	419	35.2	886	74.4	317	35.8	305	25.6	102	33.4
det	1185	845	71.3	0	0	0	0	1185	100	845	71.3
case	1180	786	66.6	13	1.1	4	30.8	1167	98.9	782	67
nmod	838	474	56.6	543	64.8	281	51.7	295	35.2	193	65.4
nsubj	708	325	45.9	131	18.5	49	37.4	577	81.5	276	47.8
obl	691	223	32.3	302	43.7	109	36.1	389	56.3	114	29.3
obj	652	249	38.2	306	46.9	128	41.8	346	53.1	121	35
advmod	596	197	33.1	169	28.4	56	33.1	427	71.6	141	33
conj	428	146	34.1	428	100	146	34.1	0	0	0	0
amod	410	303	73.9	19	4.6	2	10.5	391	95.4	301	77
cc	357	190	53.2	1	0.3	0	0	356	99.7	190	53.4
mark	336	147	43.8	16	4.8	4	25	320	95.2	143	44.7
compound	326	232	71.2	319	97.9	231	72.4	7	2.1	1	14.3
aux	291	98	33.7	76	26.1	8	10.5	215	73.9	90	41.9
acl	209	85	40.7	103	49.3	24	23.3	106	50.7	61	57.5
xcomp	199	78	39.2	184	92.5	77	41.8	15	7.5	1	6.7
cop	193	107	55.4	78	40.4	44	56.4	115	59.6	63	54.8
appos	150	82	54.7	149	99.3	81	54.4	1	0.7	1	100
flat	149	110	73.8	149	100	110	73.8	0	0	0	0
ccomp	120	26	21.7	103	85.8	23	22.3	17	14.2	3	17.6
dep	108	23	21.3	108	100	23	21.3	0	0	0	0
parataxis	97	18	18.6	97	100	18	18.6	0	0	0	0
nummod	97	62	63.9	22	22.7	6	27.3	75	77.3	56	74.7
compound:prt	94	44	46.8	53	56.4	15	28.3	41	43.6	29	70.7
advcl	87	8	9.2	52	59.8	3	5.8	35	40.2	5	14.3
csubj	41	8	19.5	19	46.3	1	5.3	22	53.7	7	31.8
iobj	23	8	34.8	1	4.3	1	100	22	95.7	7	31.8
det:nummod	22	14	63.6	2	9.1	0	0	20	90.9	14	70
expl:pv	16	8	50	8	50	5	62.5	8	50	3	37.5
fixed	7	2	28.6	7	100	2	28.6	0	0	0	0
total	10801	5317	49.2	4344	40.2	1768	40.7	6457	59.8	3549	55

Table A.15: Dutch. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	1191	595	50	886	74.4	469	52.9	305	25.6	126	41.3
det	1185	940	79.3	0	0	0	0	1185	100	940	79.3
case	1180	929	78.7	13	1.1	4	30.8	1167	98.9	925	79.3
nmod	838	565	67.4	543	64.8	358	65.9	295	35.2	207	70.2
nsubj	708	396	55.9	131	18.5	56	42.7	577	81.5	340	58.9
obl	691	311	45	302	43.7	142	47	389	56.3	169	43.4
obj	652	321	49.2	306	46.9	171	55.9	346	53.1	150	43.4
advmod	596	273	45.8	169	28.4	81	47.9	427	71.6	192	45
conj	428	192	44.9	428	100	192	44.9	0	0	0	0
amod	410	317	77.3	19	4.6	2	10.5	391	95.4	315	80.6
cc	357	230	64.4	1	0.3	0	0	356	99.7	230	64.6
mark	336	187	55.7	16	4.8	6	37.5	320	95.2	181	56.6
compound	326	255	78.2	319	97.9	254	79.6	7	2.1	1	14.3
aux	291	117	40.2	76	26.1	10	13.2	215	73.9	107	49.8
acl	209	110	52.6	103	49.3	43	41.7	106	50.7	67	63.2
xcomp	199	96	48.2	184	92.5	95	51.6	15	7.5	1	6.7
cop	193	126	65.3	78	40.4	49	62.8	115	59.6	77	67
appos	150	96	64	149	99.3	95	63.8	1	0.7	1	100
flat	149	117	78.5	149	100	117	78.5	0	0	0	0
ccomp	120	44	36.7	103	85.8	40	38.8	17	14.2	4	23.5
dep	108	41	38	108	100	41	38	0	0	0	0
parataxis	97	41	42.3	97	100	41	42.3	0	0	0	0
nummod	97	65	67	22	22.7	7	31.8	75	77.3	58	77.3
compound:prt	94	55	58.5	53	56.4	23	43.4	41	43.6	32	78
advcl	87	13	14.9	52	59.8	7	13.5	35	40.2	6	17.1
csubj	41	14	34.1	19	46.3	4	21.1	22	53.7	10	45.5
iobj	23	9	39.1	1	4.3	1	100	22	95.7	8	36.4
det:nummod	22	16	72.7	2	9.1	0	0	20	90.9	16	80
expl:pv	16	8	50	8	50	5	62.5	8	50	3	37.5
fixed	7	2	28.6	7	100	2	28.6	0	0	0	0
total	10801	6481	60	4344	40.2	2315	53.3	6457	59.8	4166	64.5

Table A.16: Dutch. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
det	1552	1142	73.6	11	0.7	2	18.2	1541	99.3	1140	74
case	1491	992	66.5	7	0.5	3	42.9	1484	99.5	989	66.6
punct	1336	541	40.5	944	70.7	395	41.8	392	29.3	146	37.2
nmod	850	444	52.2	829	97.5	441	53.2	21	2.5	3	14.3
obl	563	242	43	458	81.3	196	42.8	105	18.7	46	43.8
nsubj	517	289	55.9	55	10.6	10	18.2	462	89.4	279	60.4
obj	383	220	57.4	347	90.6	207	59.7	36	9.4	13	36.1
advmod	353	113	32	99	28	22	22.2	254	72	91	35.8
amod	323	135	41.8	229	70.9	117	51.1	94	29.1	18	19.1
flat:name	283	194	68.6	283	100	194	68.6	0	0	0	0
conj	243	71	29.2	242	99.6	71	29.3	1	0.4	0	0
mark	209	94	45	1	0.5	0	0	208	99.5	94	45.2
cc	199	105	52.8	11	5.5	2	18.2	188	94.5	103	54.8
appos	196	94	48	196	100	94	48	0	0	0	0
acl	147	70	47.6	145	98.6	70	48.3	2	1.4	0	0
nummod	137	71	51.8	27	19.7	2	7.4	110	80.3	69	62.7
cop	116	75	64.7	4	3.4	0	0	112	96.6	75	67
acl:relcl	105	36	34.3	103	98.1	35	34	2	1.9	1	50
advcl	102	19	18.6	85	83.3	18	21.2	17	16.7	1	5.9
aux	100	70	70	1	1	0	0	99	99	70	70.7
ccomp	72	23	31.9	56	77.8	22	39.3	16	22.2	1	6.2
xcomp	61	26	42.6	58	95.1	26	44.8	3	4.9	0	0
aux:pass	57	47	82.5	0	0	0	0	57	100	47	82.5
dep	55	20	36.4	39	70.9	15	38.5	16	29.1	5	31.2
nsubj:pass	49	25	51	6	12.2	2	33.3	43	87.8	23	53.5
expl	35	17	48.6	10	28.6	7	70	25	71.4	10	40
obl:agent	31	19	61.3	31	100	19	61.3	0	0	0	0
compound	28	12	42.9	25	89.3	12	48	3	10.7	0	0
nmod:npmod	25	14	56	19	76	14	73.7	6	24	0	0
fixed	24	16	66.7	19	79.2	13	68.4	5	20.8	3	60
parataxis	20	6	30	20	100	6	30	0	0	0	0
nmod:tmod	12	7	58.3	11	91.7	7	63.6	1	8.3	0	0
csubj	11	0	0	9	81.8	0	0	2	18.2	0	0
iobj	7	3	42.9	4	57.1	2	50	3	42.9	1	33.3
flat:foreign	4	1	25	4	100	1	25	0	0	0	0
vocative	3	0	0	3	100	0	0	0	0	0	0
flat	2	1	50	2	100	1	50	0	0	0	0
dislocated	1	0	0	0	0	0	0	1	100	0	0
total	9702	5254	54.2	4393	45.3	2026	46.1	5309	54.7	3228	60.8

Table A.17: Portuguese. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
det	1552	1236	79.6	11	0.7	3	27.3	1541	99.3	1233	80
case	1491	1200	80.5	7	0.5	3	42.9	1484	99.5	1197	80.7
punct	1336	723	54.1	944	70.7	540	57.2	392	29.3	183	46.7
nmod	850	600	70.6	829	97.5	597	72	21	2.5	3	14.3
obl	563	341	60.6	458	81.3	278	60.7	105	18.7	63	60
nsubj	517	373	72.1	55	10.6	17	30.9	462	89.4	356	77.1
obj	383	281	73.4	347	90.6	260	74.9	36	9.4	21	58.3
advmod	353	138	39.1	99	28	25	25.3	254	72	113	44.5
amod	323	141	43.7	229	70.9	121	52.8	94	29.1	20	21.3
flat:name	283	201	71	283	100	201	71	0	0	0	0
conj	243	116	47.7	242	99.6	116	47.9	1	0.4	0	0
mark	209	126	60.3	1	0.5	0	0	208	99.5	126	60.6
cc	199	139	69.8	11	5.5	4	36.4	188	94.5	135	71.8
appos	196	119	60.7	196	100	119	60.7	0	0	0	0
acl	147	93	63.3	145	98.6	93	64.1	2	1.4	0	0
nummod	137	76	55.5	27	19.7	3	11.1	110	80.3	73	66.4
cop	116	93	80.2	4	3.4	0	0	112	96.6	93	83
acl:relcl	105	54	51.4	103	98.1	53	51.5	2	1.9	1	50
advcl	102	28	27.5	85	83.3	25	29.4	17	16.7	3	17.6
aux	100	83	83	1	1	0	0	99	99	83	83.8
ccomp	72	35	48.6	56	77.8	29	51.8	16	22.2	6	37.5
xcomp	61	34	55.7	58	95.1	34	58.6	3	4.9	0	0
aux:pass	57	50	87.7	0	0	0	0	57	100	50	87.7
dep	55	27	49.1	39	70.9	21	53.8	16	29.1	6	37.5
nsubj:pass	49	33	67.3	6	12.2	2	33.3	43	87.8	31	72.1
expl	35	18	51.4	10	28.6	7	70	25	71.4	11	44
obl:agent	31	23	74.2	31	100	23	74.2	0	0	0	0
compound	28	12	42.9	25	89.3	12	48	3	10.7	0	0
nmod:npmod	25	16	64	19	76	16	84.2	6	24	0	0
fixed	24	16	66.7	19	79.2	13	68.4	5	20.8	3	60
parataxis	20	7	35	20	100	7	35	0	0	0	0
nmod:tmod	12	7	58.3	11	91.7	7	63.6	1	8.3	0	0
csubj	11	2	18.2	9	81.8	2	22.2	2	18.2	0	0
iobj	7	3	42.9	4	57.1	2	50	3	42.9	1	33.3
flat:foreign	4	2	50	4	100	2	50	0	0	0	0
vocative	3	0	0	3	100	0	0	0	0	0	0
flat	2	1	50	2	100	1	50	0	0	0	0
dislocated	1	0	0	0	0	0	0	1	100	0	0
total	9702	6447	66.5	4393	45.3	2636	60	5309	54.7	3811	71.8

Table A.18: Portuguese. Approximate match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	20708	16911	81.7	18881	91.2	15916	84.3	1827	8.8	995	54.5
amod	11954	9457	79.1	814	6.8	424	52.1	11140	93.2	9033	81.1
nmod	11100	7836	70.6	10202	91.9	7409	72.6	898	8.1	427	47.6
case	10972	9243	84.2	18	0.2	3	16.7	10954	99.8	9240	84.4
obl	8412	4734	56.3	5358	63.7	3229	60.3	3054	36.3	1505	49.3
nsubj	8084	5221	64.6	2314	28.6	1477	63.8	5770	71.4	3744	64.9
advmod	7767	4554	58.6	1292	16.6	557	43.1	6475	83.4	3997	61.7
conj	5398	2950	54.6	5391	99.9	2947	54.7	7	0.1	3	42.9
cc	4709	3275	69.5	331	7	223	67.4	4378	93	3052	69.7
obj	3584	2329	65	2919	81.4	2051	70.3	665	18.6	278	41.8
parataxis	2131	1167	54.8	1242	58.3	600	48.3	889	41.7	567	63.8
nummod	1890	1235	65.3	551	29.2	203	36.8	1339	70.8	1032	77.1
advcl	1783	898	50.4	1309	73.4	673	51.4	474	26.6	225	47.5
mark	1317	932	70.8	905	68.7	708	78.2	412	31.3	224	54.4
appos	930	616	66.2	922	99.1	614	66.6	8	0.9	2	25
fixed	890	754	84.7	844	94.8	732	86.7	46	5.2	22	47.8
xcomp	841	570	67.8	796	94.6	559	70.2	45	5.4	11	24.4
acl:relcl	715	436	61	713	99.7	434	60.9	2	0.3	2	100
flat:name	707	608	86	707	100	608	86	0	0	0	0
cop	631	387	61.3	230	36.5	152	66.1	401	63.5	235	58.6
nsubj:pass	587	395	67.3	219	37.3	148	67.6	368	62.7	247	67.1
nummod:gov	492	349	70.9	44	8.9	14	31.8	448	91.1	335	74.8
dep	415	283	68.2	404	97.3	282	69.8	11	2.7	1	9.1
aux:pass	373	303	81.2	11	2.9	2	18.2	362	97.1	301	83.1
aux	228	154	67.5	75	32.9	61	81.3	153	67.1	93	60.8
flat:foreign	220	134	60.9	218	99.1	133	61	2	0.9	1	50
acl	186	83	44.6	117	62.9	51	43.6	69	37.1	32	46.4
obl:agent	174	111	63.8	159	91.4	106	66.7	15	8.6	5	33.3
iobj	163	85	52.1	98	60.1	61	62.2	65	39.9	24	36.9
ccomp	101	52	51.5	93	92.1	48	51.6	8	7.9	4	50
compound	57	35	61.4	0	0	0	0	57	100	35	61.4
discourse	47	24	51.1	4	8.5	0	0	43	91.5	24	55.8
nummod:entity	7	6	85.7	7	100	6	85.7	0	0	0	0
flat	3	2	66.7	3	100	2	66.7	0	0	0	0
vocative	2	1	50	1	50	1	100	1	50	0	0
expl	1	0	0	0	0	0	0	1	100	0	0
total	107579	76130	70.8	57192	53.2	40434	70.7	50387	46.8	35696	70.8

Table A.19: Russian. Exact match.

deprel	count	found-m	%	HI	HI%	HI-m	HI-m%	HF	HF%	HF-m	HF-m%
punct	20708	17718	85.6	18881	91.2	16610	88	1827	8.8	1108	60.6
amod	11954	10156	85	814	6.8	483	59.3	11140	93.2	9673	86.8
nmod	11100	9257	83.4	10202	91.9	8701	85.3	898	8.1	556	61.9
case	10972	9898	90.2	18	0.2	3	16.7	10954	99.8	9895	90.3
obl	8412	5817	69.2	5358	63.7	3881	72.4	3054	36.3	1936	63.4
nsubj	8084	6352	78.6	2314	28.6	1727	74.6	5770	71.4	4625	80.2
advmod	7767	5428	69.9	1292	16.6	678	52.5	6475	83.4	4750	73.4
conj	5398	3793	70.3	5391	99.9	3789	70.3	7	0.1	4	57.1
cc	4709	3803	80.8	331	7	246	74.3	4378	93	3557	81.2
obj	3584	2643	73.7	2919	81.4	2298	78.7	665	18.6	345	51.9
parataxis	2131	1469	68.9	1242	58.3	794	63.9	889	41.7	675	75.9
nummod	1890	1356	71.7	551	29.2	251	45.6	1339	70.8	1105	82.5
advcl	1783	1173	65.8	1309	73.4	880	67.2	474	26.6	293	61.8
mark	1317	1052	79.9	905	68.7	763	84.3	412	31.3	289	70.1
appos	930	731	78.6	922	99.1	727	78.9	8	0.9	4	50
fixed	890	763	85.7	844	94.8	737	87.3	46	5.2	26	56.5
xcomp	841	654	77.8	796	94.6	639	80.3	45	5.4	15	33.3
acl:relcl	715	529	74	713	99.7	527	73.9	2	0.3	2	100
flat:name	707	623	88.1	707	100	623	88.1	0	0	0	0
cop	631	470	74.5	230	36.5	168	73	401	63.5	302	75.3
nsubj:pass	587	461	78.5	219	37.3	171	78.1	368	62.7	290	78.8
nummod:gov	492	381	77.4	44	8.9	17	38.6	448	91.1	364	81.2
dep	415	329	79.3	404	97.3	328	81.2	11	2.7	1	9.1
aux:pass	373	334	89.5	11	2.9	3	27.3	362	97.1	331	91.4
aux	228	174	76.3	75	32.9	62	82.7	153	67.1	112	73.2
flat:foreign	220	160	72.7	218	99.1	159	72.9	2	0.9	1	50
acl	186	124	66.7	117	62.9	80	68.4	69	37.1	44	63.8
obl:agent	174	125	71.8	159	91.4	118	74.2	15	8.6	7	46.7
iobj	163	106	65	98	60.1	68	69.4	65	39.9	38	58.5
ccomp	101	70	69.3	93	92.1	64	68.8	8	7.9	6	75
compound	57	42	73.7	0	0	0	0	57	100	42	73.7
discourse	47	29	61.7	4	8.5	0	0	43	91.5	29	67.4
nummod:entity	7	6	85.7	7	100	6	85.7	0	0	0	0
flat	3	2	66.7	3	100	2	66.7	0	0	0	0
vocative	2	1	50	1	50	1	100	1	50	0	0
expl	1	0	0	0	0	0	0	1	100	0	0
total	107579	86029	80	57192	53.2	45604	79.7	50387	46.8	40425	80.2

Table A.20: Russian. Approximate match.

B

Appendix B

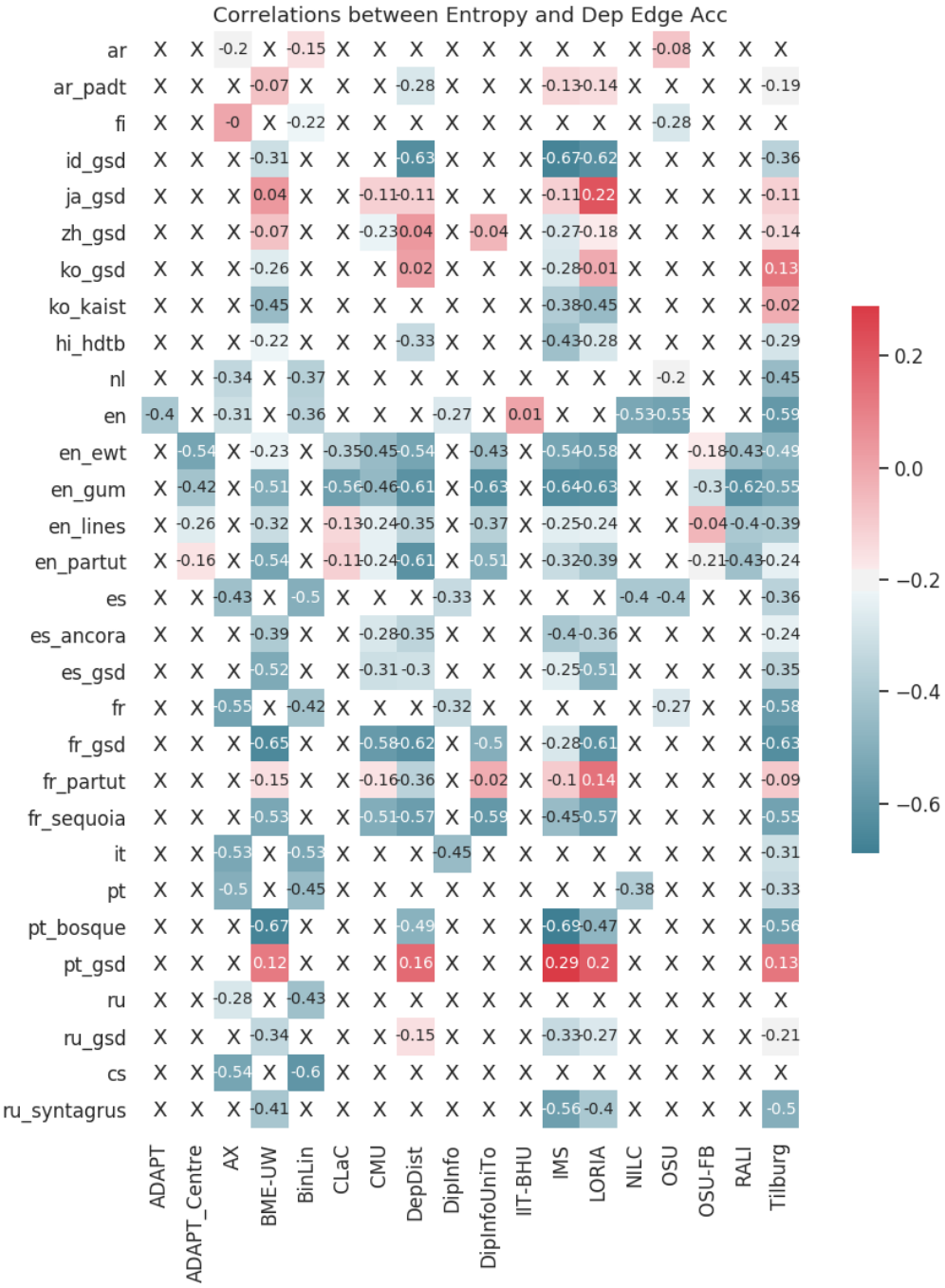


Figure B.1: Spearman ρ coefficients between dependency edge accuracy and entropy. A cross indicates that a team did not make a submission for the treebank. Treebanks are grouped by language families.

Bibliography

Agarwal, S. and Dymetman, M. (2017). A surprisingly effective out-of-the-box char2char model on the e2e nlg challenge dataset. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 158–163. Association for Computational Linguistics.

Aharoni, R. and Goldberg, Y. (2017). Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Aharoni, R. and Goldberg, Y. (2018). Split and rephrase: Better evaluation and stronger baselines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 719–724. Association for Computational Linguistics.

Alexandrescu, A. and Kirchhoff, K. (2006). Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 1–4. Association for Computational Linguistics.

Angeli, G., Liang, P., and Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA. Association for Computational Linguistics.

Baayen, R. H., Piepenbrock, R., and Gulikers, L. (1993). The CELEX lexical database (CD-ROM).

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Balakrishnan, A., Rao, J., Upasani, K., White, M., and Subba, R. (2019). Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Basile, V. and Bos, J. (2011). Towards generating text from discourse representation structures. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 145–150, Nancy, France. Association for Computational Linguistics.

Basile, V. and Mazzei, A. (2018). The dipinfo-unito system for srst 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 65–71. Association for Computational Linguistics.

Beck, D., Haffari, G., and Cohn, T. (2018). Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431.

Belz, A., White, M., Espinosa, D., Kow, E., Hogan, D., and Stent, A. (2011). The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226. Association for Computational Linguistics.

Bernardi, R., Çakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2017). Automatic description generation from images: A survey of models, datasets, and evaluation measures (extended abstract). In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4970–4974.

Bohnet, B., Mille, S., Favre, B., and Wanner, L. (2011). <StuMaBa>: From deep representation to surface. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 232–235, Nancy, France. Association for Computational Linguistics.

Bohnet, B., Wanner, L., Mille, S., and Burga, A. (2010). Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 98–106. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Cahill, A. (2009). Correlating human and automatic evaluation of a German surface realiser. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 97–100, Suntec, Singapore. Association for Computational Linguistics.

Cahill, A. and van Genabith, J. (2006). Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1033–1040, Sydney, Australia. Association for Computational Linguistics.

Callaway, C. B. (2003). Evaluating coverage for large symbolic nlg grammars. In *IJCAI*, pages 811–816.

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., and Schroeder, J. (2008). Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus, Ohio. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 workshop on statistical machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28. Association for Computational Linguistics.

Cao, M. and Cheung, J. C. K. (2019). Referring expression generation using entity profiles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3163–3172, Hong Kong, China. Association for Computational Linguistics.

Cao, Z., Luo, C., Li, W., and Li, S. (2017). Joint copying and restricted generation for paraphrase. In *AAAI*, pages 3152–3158.

Castro Ferreira, T., Gardent, C., Ilinykh, N., van der Lee, C., Mille, S., Moussallem, D., and Shimorina, A. (2020). The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Castro Ferreira, T., Moussallem, D., Kádár, Á., Wubben, S., and Krahmer, E. (2018a). NeuralREG: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1969, Melbourne, Australia. Association for Computational Linguistics.

Castro Ferreira, T., Moussallem, D., Krahmer, E., and Wubben, S. (2018b). Enriching the webnlg corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176, Tilburg University, The Netherlands. Association for Computational Linguistics.

Castro Ferreira, T., van der Lee, C., van Miltenburg, E., and Krahmer, E. (2019). Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Castro Ferreira, T., Wubben, S., and Krahmer, E. (2018c). Surface realization shared task 2018 (sr18): The tilburg university approach. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 35–38. Association for Computational Linguistics.

Cawsey, A. J., Webber, B. L., and Jones, R. B. (1997). Natural Language Generation in Health Care. *Journal of the American Medical Informatics Association*, 4(6):473–482.

Chatterjee, R., Federmann, C., Negri, M., and Turchi, M. (2019). Findings of the WMT 2019 shared task on automatic post-editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 11–28, Florence, Italy. Association for Computational Linguistics.

Chatterjee, R., Negri, M., Rubino, R., and Turchi, M. (2018). Findings of the wmt 2018 shared task on automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 710–725, Belgium, Brussels. Association for Computational Linguistics.

Chen, B. and Cherry, C. (2014). A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367.

Chen, D. L. and Mooney, R. J. (2008). Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135.

Chen, M., Lampouras, G., and Vlachos, A. (2018). Sheffield at e2e: structured prediction approaches to end-to-end language generation. Technical report, E2E Challenge System Descriptions.

Chen, S. (2018). A general model for neural text generation from structured data. Technical report, E2E Challenge System Descriptions.

Chen, W., Chen, J., Su, Y., Chen, Z., and Wang, W. Y. (2020a). Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Chen, Z., Eavani, H., Chen, W., Liu, Y., and Wang, W. Y. (2020b). Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Online. Association for Computational Linguistics.

Cheng, J. and Lapata, M. (2016). Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.

Chisholm, A., Radford, W., and Hachey, B. (2017). Learning to generate one-sentence biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642, Valencia, Spain. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Walther, G., Vylomova, E., Xia, P., Faruqui, M., Kübler, S., Yarowsky, D., Eisner, J., and Hulden, M. (2017). CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., and Hulden, M. (2016). The SIGMORPHON 2016 shared Task—Morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Damonte, M. and Cohen, S. B. (2018). Cross-lingual Abstract Meaning Representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155, New Orleans, Louisiana. Association for Computational Linguistics.

Dang, H. T. and Owczarzak, K. (2008). Overview of the TAC 2008 update summarization task. In *Proceedings of the First Text Analysis Conference, TAC 2008, Gaithersburg, Maryland, USA, November 17-19, 2008*. NIST.

Davoodi, E., Smiley, C., Song, D., and Schilder, F. (2018). The e2e nlg challenge: Training a sequence-to-sequence approach for meaning representation to natural language sentences. Technical report, E2E Challenge System Descriptions.

de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland. European Language Resources Association (ELRA).

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

Deriu, J. and Cieliebak, M. (2018a). End-to-end trainable system for enhancing diversity in natural language generation. Technical report, E2E Challenge System Descriptions.

Deriu, J. M. and Cieliebak, M. (2018b). Syntactic manipulation for generating more diverse and interesting texts. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 22–34, Tilburg University, The Netherlands. Association for Computational Linguistics.

Dethlefs, N. and Cuayáhuitl, H. (2015). Hierarchical reinforcement learning for situated natural language generation. *Natural language engineering*, 21:391–435.

DiMarco, C., Covvey, H. D., Bray, P., Cowan, D., Diciccio, V., Hovy, E., Lipa, J., and Mulholland, D. (2007). The development of a natural language generation system for personalized e-health information. In *12th International Health (Medical) Informatics Congress, Medinfo 2007*, Brisbane, Australia.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Dreyer, M., Smith, J., and Eisner, J. (2008). Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii. Association for Computational Linguistics.

Durrett, G. and DeNero, J. (2013). Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia. Association for Computational Linguistics.

Dušek, O. and Jurčíček, F. (2015). Training a natural language generator from unaligned data. 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 451–461. Association for Computational Linguistics.

Dušek, O. and Jurčíček, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany. Association for Computational Linguistics.

Dušek, O. and Jurčíček, F. (2019). Neural generation for Czech: Data and baselines. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 563–574, Tokyo, Japan. Association for Computational Linguistics.

Dušek, O., Novikova, J., and Rieser, V. (2020). Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123 – 156.

Dyer, W. (2019). Weighted posets: Learning surface order from dependency trees. In *Proceedings of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019)*, pages 61–73, Paris, France. Association for Computational Linguistics.

Elder, H. (2017). Adapt centre submission for the webnlg challenge. Technical report, WebNLG Challenge System Descriptions.

Elder, H., Foster, J., Barry, J., and O'Connor, A. (2019). Designing a symbolic intermediate representation for neural surface realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 65–73, Minneapolis, Minnesota. Association for Computational Linguistics.

Elder, H., Gehrmann, S., O'Connor, A., and Liu, Q. (2018). E2e nlg challenge submission: Towards controllable generation of diverse natural language. Technical report, E2E Challenge System Descriptions.

Elder, H. and Hokamp, C. (2018). Generating high-quality surface realizations using data augmentation and factored sequence models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53. Association for Computational Linguistics.

Elliott, D. and Keller, F. (2014). Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 452–457. Association for Computational Linguistics.

Elsahar, H., Vougiouklis, P., Remaci, A., Gravier, C., Hare, J., Laforest, F., and Simperl, E. (2018). T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Ficler, J. and Goldberg, Y. (2017). Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Filippova, K. and Strube, M. (2009). Tree linearization in English: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 225–228, Boulder, Colorado. Association for Computational Linguistics.

Futrell, R., Mahowald, K., and Gibson, E. (2015). Quantifying word order freedom in dependency corpora. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 91–100, Uppsala, Sweden. Uppsala University, Uppsala, Sweden.

Gage, P. (1994). A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Gardent, C. and Narayan, S. (2012). Error mining on dependency trees. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 592–600, Jeju Island, Korea. Association for Computational Linguistics.

Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017a). Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017b). The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133. Association for Computational Linguistics.

Gatt, A. and Belz, A. (2010). Introducing shared tasks to nlg: The tuna shared task evaluation challenges. In Krahmer, E. and Theune, M., editors, *Empirical Methods in Natural Language Generation*, pages 264–293. Springer-Verlag, Berlin, Heidelberg.

Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Gehrmann, S., Dai, F., Elder, H., and Rush, A. (2018). End-to-end content and plan selection for natural language generation. Technical report, E2E Challenge System Descriptions.

Gerdemann, D. and Hinrichs, E. W. (1990). Functor-driven natural language generation with categorial-unification grammars. In *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*.

Gervás, P. (2011). UCM submission to the surface realization challenge. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 239–241, Nancy, France. Association for Computational Linguistics.

Giménez, J. and Màrquez, L. (2009). On the robustness of syntactic and semantic features for automatic MT evaluation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 250–258, Athens, Greece. Association for Computational Linguistics.

Gkatzia, D. and Mahamood, S. (2015). A snapshot of NLG evaluation practices 2005 - 2014. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 57–60, Brighton, UK. Association for Computational Linguistics.

Gorman, K., McCarthy, A. D., Cotterell, R., Vylomova, E., Silfverberg, M., and Markowska, M. (2019). Weird inflects but OK: Making sense of morphological generation errors. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 140–151, Hong Kong, China. Association for Computational Linguistics.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Gulordava, K. and Merlo, P. (2016). Multi-lingual dependency parsing evaluation: a large-scale analysis of word order properties using artificial data. *Transactions of the Association for Computational Linguistics*, 4:343–356.

Guo, Y., Hogan, D., and van Genabith, J. (2011). DCU at generation challenges 2011 surface realisation track. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 227–229, Nancy, France. Association for Computational Linguistics.

Hajdik, V., Buys, J., Goodman, M. W., and Bender, E. M. (2019). Neural text generation from rich semantic representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2259–2266, Minneapolis, Minnesota. Association for Computational Linguistics.

Hasan, S. A. and Farri, O. (2019). *Clinical Natural Language Processing with Deep Learning*, pages 147–171. Springer International Publishing, Cham.

Hayashi, H., Oda, Y., Birch, A., Konstas, I., Finch, A., Luong, M.-T., Neubig, G., and Sudoh, K. (2019). Findings of the third workshop on neural generation and translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 1–14, Hong Kong. Association for Computational Linguistics.

He, S., Liu, C., Liu, K., and Zhao, J. (2017). Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 199–208. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hokamp, C. (2017). Ensembling factored neural machine translation models for automatic post-editing and quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 647–654, Copenhagen, Denmark. Association for Computational Linguistics.

Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., and Hovy, E. (2013). Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Hovy, E., Lin, C.-Y., and Zhou, L. (2005). Evaluating duc 2005 using basic elements. In *Proceedings of the 5th Document Understanding Conference (DUC)*.

Howcroft, D. M., Belz, A., Clinciu, M.-A., Gkatzia, D., Hasan, S. A., Mahamood, S., Mille, S., van Miltenburg, E., Santhanam, S., and Rieser, V. (2020). Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Jagfeld, G., Jenne, S., and Vu, N. T. (2018). Sequence-to-sequence models for data-to-text natural language generation: Word- vs. character-based processing and output diversity. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 221–232, Tilburg University, The Netherlands. Association for Computational Linguistics.

Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2018). Ms-uedin submission to the wmt2018 ape shared task: Dual-source transformer for automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 822–826, Belgium, Brussels. Association for Computational Linguistics.

Juraska, J., Bowden, K., and Walker, M. (2019). ViGGO: A video game corpus for data-to-text generation in open-domain conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 164–172, Tokyo, Japan. Association for Computational Linguistics.

Juraska, J., Karagiannis, P., Bowden, K., and Walker, M. (2018). A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162. Association for Computational Linguistics.

Kahane, S., Yan, C., and Botalla, M.-A. (2017). What are the limitations on the flux of syntactic dependencies? evidence from UD treebanks. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 73–82, Pisa,Italy. Linköping University Electronic Press.

Katragadda, R. (2009). On alternative automated content evaluation measures. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, Maryland, USA.

King, D. and White, M. (2018). The OSU realizer for SRST '18: Neural sequence-to-sequence inflection and incremental locality-based linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 39–48, Melbourne, Australia. Association for Computational Linguistics.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Komalova, L. R. (2017). Oshibki i netochnosti perevoda (svodnyj referat). *Social'nye i gumanitarnye nauki. Otechestvennaja i zarubezhnaja literatura. Ser. 6, Jazykoznanie: Referativnyj zhurnal*, (4):32–44.

Kondadadi, R., Howald, B., and Schilder, F. (2013). A statistical NLG framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1406–1415, Sofia, Bulgaria. Association for Computational Linguistics.

Konstas, I., Iyer, S., Yatskar, M., Choi, Y., and Zettlemoyer, L. (2017). Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Konstas, I. and Lapata, M. (2013). A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

Kovács, Á., Ács, E., Ács, J., Kornai, A., and Recski, G. (2019). BME-UW at SRST-2019: Surface realization with interpreted regular tree grammars. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 35–40, Hong Kong, China. Association for Computational Linguistics.

Kukich, K. (1983). Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Kulesza, A. and Shieber, S. M. (2004). A learning approach to improving sentence-level mt evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 75–84.

Kunilovskaya, M. A. (2013). Klassifikacija perevodcheskih oshibok dlja sozdanija razmetki v uchebnom parallel'nom korpuse russian learner translator corpus. *Lingua mobilis*, (1 (40)):141–158.

Lampouras, G. and Vlachos, A. (2016). Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112, Osaka, Japan. The COLING 2016 Organizing Committee.

Langkilde-Geary, I. (2002). An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the International Natural Language Generation Conference*, pages 17–24, Harriman, New York, USA. Association for Computational Linguistics.

Lebret, R., Grangier, D., and Auli, M. (2016). Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Leppänen, L., Munezero, M., Granroth-Wilding, M., and Toivonen, H. (2017). Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197, Santiago de Compostela, Spain. Association for Computational Linguistics.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Li, W. (2015). Abstractive multi-document summarization with semantic information extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913, Lisbon, Portugal. Association for Computational Linguistics.

Liang, P., Jordan, M., and Klein, D. (2009). Learning semantic correspondences with less supervision. 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*, pages 91–99, Suntec, Singapore. Association for Computational Linguistics.

Liao, K., Lebanoff, L., and Liu, F. (2018). Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Libovický, J., Helcl, J., Tlustý, M., Bojar, O., and Pecina, P. (2016). Cuni system for wmt16 automatic post-editing and multimodal translation tasks. In *Proceedings of the First Conference on Machine Translation*, pages 646–654, Berlin, Germany. Association for Computational Linguistics.

Lin, B. Y., Zhou, W., Shen, M., Zhou, P., Bhagavatula, C., Choi, Y., and Ren, X. (2020). CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1823–1840, Online. Association for Computational Linguistics.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Liu, D. and Gildea, D. (2005). Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32, Ann Arbor, Michigan. Association for Computational Linguistics.

Liu, H. (2008). Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):159–191.

Liu, H. (2010). Dependency direction as a means of word-order typology: A method based on dependency treebanks. *Lingua*, 120(6):1567–1578.

Liu, T., Wang, K., Sha, L., Chang, B., and Sui, Z. (2018). Table-to-text generation by structure-aware seq2seq learning. In *AAAI*.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Lo, C.-k., Tumuluru, A. K., and Wu, D. (2012). Fully automatic semantic MT evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 243–252, Montréal, Canada. Association for Computational Linguistics.

Lohar, P., Popović, M., and Way, A. (2019). Building English-to-Serbian machine translation system for IMDb movie reviews. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 105–113, Florence, Italy. Association for Computational Linguistics.

Lu, W. and Ng, H. T. (2011). A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Madsack, A., Heininger, J., Davaasambuu, N., Voronik, V., Käufl, M., and Weißgraeber, R. (2018). Ax semantics' submission to the surface realization shared task 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 54–57. Association for Computational Linguistics.

Mager, M., Fernandez Astudillo, R., Naseem, T., Sultan, M. A., Lee, Y.-S., Florian, R., and Roukos, S. (2020). GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Mairesse, F., Gašić, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., and Young, S. (2010). Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561, Uppsala, Sweden. Association for Computational Linguistics.

Mairesse, F. and Young, S. (2014). Stochastic language generation in dialogue using factored language models. *Computational Linguistics*, 40(4):763–799.

Malouf, R. (2000). The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92, Hong Kong. Association for Computational Linguistics.

Marcheggiani, D. and Perez-Beltrachini, L. (2018). Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Marciniak, T. and Strube, M. (2004). Classification-based generation using tag. In Belz, A., Evans, R., and Piwek, P., editors, *Natural Language Generation*, pages 100–109, Berlin, Heidelberg. Springer Berlin Heidelberg.

Martins, A., Smith, N., and Xing, E. (2009). Concise integer linear programming formulations for dependency parsing. 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*, pages 342–350, Suntec, Singapore. Association for Computational Linguistics.

McKeown, K. R. (1982). The text system for natural language generation: An overview. In *20th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Toronto, Ontario, Canada. Association for Computational Linguistics.

Mehay, D. N. and Brew, C. (2007). Bleuâtre: Flattening syntactic dependencies for mt evaluation. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 122–131.

Meteer, M. W. (1991). Bridging the generation gap between text planning and linguistic realization. *Computational Intelligence*, 7(4):296–304.

Miao, Y. and Blunsom, P. (2016). Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mille, S., Belz, A., Bohnet, B., Graham, Y., Pitler, E., and Wanner, L. (2018a). The first multilingual surface realisation shared task (sr'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12. Association for Computational Linguistics.

Mille, S., Belz, A., Bohnet, B., Graham, Y., and Wanner, L. (2019). The second multilingual surface realisation shared task (SR'19): Overview and evaluation results. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 1–17, Hong Kong, China. Association for Computational Linguistics.

Mille, S., Belz, A., Bohnet, B., and Wanner, L. (2018b). Underspecified Universal Dependency structures as inputs for multilingual surface realisation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 199–209, Tilburg University, The Netherlands. Association for Computational Linguistics.

Miller, G. A. (1956). The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63 2:81–97.

Moryossef, A., Goldberg, Y., and Dagan, I. (2019). Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Naik, A., Ravichander, A., Sadeh, N., Rose, C., and Neubig, G. (2018). Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nakanishi, H., Miyao, Y., and Tsujii, J. (2005a). Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102, Vancouver, British Columbia. Association for Computational Linguistics.

Nakanishi, H., Miyao, Y., and Tsujii, J. (2005b). Probabilistic models for disambiguation of an hpsg-based chart generator. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102.

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Narayan, S., Cohen, S. B., and Lapata, M. (2018). Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Narayan, S. and Gardent, C. (2012). Error mining with suspicion trees: Seeing the forest for the trees. In *Proceedings of COLING 2012*, pages 2011–2026, Mumbai, India. The COLING 2012 Organizing Committee.

Narayan, S., Gardent, C., Cohen, S. B., and Shimorina, A. (2017). Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 606–616, Copenhagen, Denmark. Association for Computational Linguistics.

Nayak, N., Hakkani-Tür, D., Walker, M., and Heck, L. (2017). To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *Proc. Interspeech 2017*, pages 3339–3343.

Negri, M., Turchi, M., Chatterjee, R., and Bertoldi, N. (2018). ESCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Nema, P., Shetty, S., Jain, P., Laha, A., Sankaranarayanan, K., and Khapra, M. M. (2018). Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1539–1550, New Orleans, Louisiana. Association for Computational Linguistics.

Neubig, G., Dou, Z.-Y., Hu, J., Michel, P., Pruthi, D., and Wang, X. (2019). compare-mt: A tool for holistic comparison of language generation systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 35–41, Minneapolis, Minnesota. Association for Computational Linguistics.

Nie, F., Yao, J.-G., Wang, J., Pan, R., and Lin, C.-Y. (2019). A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679, Florence, Italy. Association for Computational Linguistics.

Nivre, J., Agić, Ž., Ahrenberg, L., Aranzabe, M. J., Asahara, M., Atutxa, A., Ballesteros, M., Bauer, J., Bengoetxea, K., Bhat, R. A., Bick, E., Bosco, C., Bouma, G., Bowman, S., Candito, M., Cebiroğlu Eryiğit, G., Celano, G. G. A., Chalub, F., Choi, J., Çöltekin, Ç., Connor, M., Davidson, E., de Marneffe, M.-C., de Paiva, V., Diaz de Ilarraza, A., Dobrovoljc, K., Dozat, T., Droganova, K., Dwivedi, P., Eli, M., Erjavec, T., Farkas, R., Foster, J., Freitas, C., Gajdošová, K., Galbraith, D., Garcia, M., Ginter, F., Goenaga, I., Gojenola, K., Gökırmak, M., Goldberg, Y., Gómez Guinovart, X., Gonzáles Saavedra, B., Grioni, M., Grūzītis, N., Guillaume, B., Habash, N., Hajič, J., Hà Mỹ, L., Haug, D., Hladká, B., Hohle, P., Ion, R., Irimia, E., Johannsen, A., Jørgensen, F., Kaşıkara, H., Kanayama, H., Kanerva, J., Kotsyba, N., Krek, S., Laippala, V., Lê Hồng, P., Lenci, A., Ljubešić, N., Lyashevskaya, O., Lynn, T., Makazhanov, A., Manning, C., Mărănduc, C., Mareček, D., Martínez Alonso, H., Martins, A., Mašek, J., Matsumoto, Y., McDonald, R., Missilä, A., Mititelu, V., Miyao, Y., Montemagni, S., More, A., Mori, S., Moskalevskyi, B., Muischnek, K., Mustafina, N., Müürisep, K., Nguyễn Thị, L., Nguyễn Thị Minh, H., Nikolaev, V., Nurmi, H., Ojala, S., Osenova, P., Øvrelid, L., Pascual, E., Passarotti, M., Perez, C.-A., Perrier, G., Petrov, S., Piitulainen, J., Plank, B., Popel, M., Pretkalniņa, L., Prokopidis, P., Puolakainen, T., Pyysalo, S., Rademaker, A., Ramasamy, L., Real, L., Rituma, L., Rosa, R., Saleh, S., Sanguinetti, M., Saulīte, B., Schuster, S., Seddah, D., Seeker, W., Seraji, M., Shakurova, L., Shen, M., Sichinava, D., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Smith, A., Suhr, A., Sulubacak, U., Szántó, Z., Taji, D., Tanaka, T., Tsarfaty, R., Tyers, F., Uematsu, S., Uria, L., van Noord, G., Varga, V., Vincze, V., Washington, J. N., Žabokrtský, Z., Zeldes, A., Zeman, D., and Zhu, H. (2017). Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Novikova, J., Dušek, O., Cercas Curry, A., and Rieser, V. (2017a). Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252. Association for Computational Linguistics.

Novikova, J., Dušek, O., and Rieser, V. (2017b). The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206. Association for Computational Linguistics.

Oraby, S., Harrison, V., Ebrahimi, A., and Walker, M. (2019). Curate and generate: A corpus and method for joint control of semantics and style in neural NLG. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5938–5951, Florence, Italy. Association for Computational Linguistics.

Owczarzak, K. (2009). DEPEVAL(summ): Dependency-based evaluation for automatic summaries. 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*, pages 190–198, Suntec, Singapore. Association for Computational Linguistics.

Owczarzak, K., van Genabith, J., and Way, A. (2007). Dependency-based automatic evaluation for machine translation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on*

Syntax and Structure in Statistical Translation, pages 80–87, Rochester, New York. Association for Computational Linguistics.

Pal, S., Naskar, S. K., Vela, M., and van Genabith, J. (2016). A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, Berlin, Germany. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Parikh, A., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., and Das, D. (2020). ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Perez-Beltrachini, L. and Gardent, C. (2017). Analysing data-to-text generation benchmarks. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 238–242, Santiago de Compostela, Spain. Association for Computational Linguistics.

Perez-Beltrachini, L. and Lapata, M. (2018). Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1516–1527, New Orleans, Louisiana. Association for Computational Linguistics.

Plachouras, V., Smiley, C., Bretz, H., Taylor, O., Leidner, J. L., Song, D., and Schilder, F. (2016). Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 1121–1124, New York, NY, USA. Association for Computing Machinery.

Popović, M. (2018). *Error Classification and Analysis for Machine Translation Quality Assessment*, pages 129–158. Springer International Publishing, Cham.

Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., and Sykes, C. (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7):789 – 816.

Puduppully, R., Dong, L., and Lapata, M. (2019). Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Puduppully, R., Zhang, Y., and Shrivastava, M. (2016). Transition-based syntactic linearization with lookahead features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 488–493, San Diego, California. Association for Computational Linguistics.

Puduppully, R., Zhang, Y., and Shrivastava, M. (2017). Transition-based deep input linearization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 643–654, Valencia, Spain. Association for Computational Linguistics.

Puzikov, Y., Gardent, C., Dagan, I., and Gurevych, I. (2019). Revisiting the binary linearization technique for surface realization. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 268–278, Tokyo, Japan. Association for Computational Linguistics.

Puzikov, Y. and Gurevych, I. (2018a). Binlin: A simple method of dependency tree linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 13–28. Association for Computational Linguistics.

Puzikov, Y. and Gurevych, I. (2018b). E2e nlg challenge: Neural models vs. templates. Technical report, E2E Challenge System Descriptions.

Qader, R., Jneid, K., Portet, F., and Labbé, C. (2018). Generation of company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 254–263, Tilburg University, The Netherlands. Association for Computational Linguistics.

Qader, R., Portet, F., and Labbé, C. (2019). Semi-supervised neural text generation by joint learning of natural language generation and natural language understanding models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 552–562, Tokyo, Japan. Association for Computational Linguistics.

Qi, P., Dozat, T., Zhang, Y., and Manning, C. D. (2018). Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rajkumar, R., Espinosa, D., and White, M. (2011). The OSU system for surface realization at generation challenges 2011. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 236–238, Nancy, France. Association for Computational Linguistics.

Rakhilina, E., Vyrenkova, A., Mustakimova, E., Ladygina, A., and Smirnov, I. (2016). Building a learner corpus for Russian. In *Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition*, pages 66–75, Umeå, Sweden. LiU Electronic Press.

Reiter, E. (2018). A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Reiter, E. and Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4).

Reiter, E. and Dale, R. (2000). *Building natural language generation systems.* Cambridge university press.

Ribeiro, L. F. R., Gardent, C., and Gurevych, I. (2019). Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Robin, J. and McKeown, K. (1996). Empirically designing and evaluating a new revision-based model for summary generation. *Artificial Intelligence*, 85(1-2):135–179.

Schmaltz, A., Rush, A. M., and Shieber, S. (2016). Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas. Association for Computational Linguistics.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Miceli Barone, A. V., and Williams, P. (2017). The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, Copenhagen, Denmark.

Sennrich, R. and Haddow, B. (2016). Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shimorina, A. (2018). Human vs automatic metrics: on the importance of correlation design. *Peer-reviewed, non-archival, presented at the Widening NLP Workshop 2018 at NAACL*, arXiv: 1805.11474.

Shimorina, A. and Gardent, C. (2018). Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands. Association for Computational Linguistics.

Shimorina, A. and Gardent, C. (2019a). LORIA / Lorraine University at multilingual surface realisation 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 88–93, Hong Kong, China. Association for Computational Linguistics.

Shimorina, A. and Gardent, C. (2019b). Surface realisation using full delexicalisation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3086–3096, Hong Kong, China. Association for Computational Linguistics.

Shimorina, A., Gardent, C., Narayan, S., and Perez-Beltrachini, L. (2018). WebNLG Challenge: Human Evaluation Results. Technical report, Loria & Inria Grand Est.

Shimorina, A., Khasanova, E., and Gardent, C. (2019). Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 44–49, Florence, Italy. Association for Computational Linguistics.

Shimorina, A., Parmentier, Y., and Gardent, C. (2021). An Error Analysis Framework for Shallow Surface Realization. *Transactions of the Association for Computational Linguistics*, 9:429–446.

Siddharthan, A., Nenkova, A., and McKeown, K. (2011). Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Singh, S., Sharma, A., Chawla, A., and Singh, A. (2018). Iit (bhu) varanasi at msr-srst 2018: A language model based approach for natural language generation. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 29–34. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Sobrevilla Cabezudo, M. A. and Pardo, T. (2018). Nilc-swornemo at the surface realization shared task: Exploring syntax-based word ordering using neural models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 58–64. Association for Computational Linguistics.

Sobrevilla Cabezudo, M. A. and Pardo, T. (2019). Towards a general Abstract Meaning Representation corpus for Brazilian Portuguese. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 236–244, Florence, Italy. Association for Computational Linguistics.

Song, L., Zhang, Y., and Gildea, D. (2018a). Neural transition-based syntactic linearization. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 431–440, Tilburg University, The Netherlands. Association for Computational Linguistics.

Song, L., Zhang, Y., Song, K., and Liu, Q. (2014). Joint morphological generation and syntactic linearization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Song, L., Zhang, Y., Wang, Z., and Gildea, D. (2018b). A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Sripada, S., Reiter, E., Hunter, J., and Yu, J. (2002). Sumtime-meteo: Parallel corpus of naturally occurring forecast texts and weather data. Technical report.

Stent, A. (2011). ATT-0: Submission to generation challenges 2011 surface realization shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 230–231, Nancy, France. Association for Computational Linguistics.

Stent, A., Marge, M., and Singhai, M. (2005). Evaluating evaluation methods for generation in the presence of variation. In *Computational Linguistics and Intelligent Text Processing, 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005, Proceedings*, pages 341–351.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Takase, S., Suzuki, J., Okazaki, N., Hirao, T., and Nagata, M. (2016). Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas. Association for Computational Linguistics.

Theune, M., Klabbers, E., de Pijper, J. R., Krahmer, E., and Odijk, J. (2001). From data to speech: a general approach. *Natural Language Engineering*, 7(1):47–86.

Tratz, S. and Hovy, E. H. (2009). Bewt-e for tac 2009's aesop task. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, Maryland, USA.

Trisedya, B. D., Qi, J., Zhang, R., and Wang, W. (2018). Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637. Association for Computational Linguistics.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016). Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85. Association for Computational Linguistics.

van Dalen, A. (2012). The algorithms behind the headlines. *Journalism Practice*, 6(5-6):648–658.

van Deemter, K. (2016). Designing algorithms for referring with proper names. In *Proceedings of the 9th International Natural Language Generation conference*, pages 31–35, Edinburgh, UK. Association for Computational Linguistics.

van der Lee, C., Castro Ferreira, T., Krahmer, E., and Wubben, S. (2017). Tilburg university models for the webnlg challenge. Technical report, WebNLG Challenge System Descriptions.

van der Lee, C., Gatt, A., van Miltenburg, E., Wubben, S., and Krahmer, E. (2019). Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

van der Lee, C., Krahmer, E., and Wubben, S. (2018). Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 35–45, Tilburg University, The Netherlands. Association for Computational Linguistics.

Vanderwende, L., Menezes, A., and Quirk, C. (2015). An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30, Denver, Colorado. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Vougiouklis, P., ElSahar, H., Kaffee, L., Gravier, C., Laforest, F., Hare, J. S., and Simperl, E. (2018). Neural wikipedian: Generating textual summaries from knowledge base triples. *J. Web Semant.*, 52-53:1–15.

Wang, Q., Pan, X., Huang, L., Zhang, B., Jiang, Z., Ji, H., and Knight, K. (2018). Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21, Tilburg University, The Netherlands. Association for Computational Linguistics.

Wen, T.-H., Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015). Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

White, M. and Rajkumar, R. (2012). Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255, Jeju Island, Korea. Association for Computational Linguistics.

White, M., Rajkumar, R., and Martin, S. (2007). Towards broad coverage surface realization with ccg. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)*, pages 267–276.

Wiseman, S., Shieber, S., and Rush, A. (2017). Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Wiseman, S., Shieber, S., and Rush, A. (2018). Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.

Xue, N., Bojar, O., Hajič, J., Palmer, M., Urešová, Z., and Zhang, X. (2014). Not an interlingua, but close: Comparison of English AMRs to Chinese and Czech. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1765–1772, Reykjavik, Iceland. European Language Resources Association (ELRA).

Yu, H., Wu, X., Xie, J., Jiang, W., Liu, Q., and Lin, S. (2014). RED: A reference dependency based MT evaluation metric. In *Proceedings of COLING 2014, the 25th International*

Conference on Computational Linguistics: Technical Papers, pages 2042–2051, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Yu, X., Falenska, A., Haid, M., Vu, N. T., and Kuhn, J. (2019a). IMSurReal: IMS at the surface realization shared task 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 50–58, Hong Kong, China. Association for Computational Linguistics.

Yu, X., Falenska, A., Vu, N. T., and Kuhn, J. (2019b). Head-first linearization with tree-structured representation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 279–289, Tokyo, Japan. Association for Computational Linguistics.

Yu, X., Tannert, S., Vu, N. T., and Kuhn, J. (2020). Fast and accurate non-projective dependency tree linearization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1451–1462, Online. Association for Computational Linguistics.

Zeman, D. (2016). Universal annotation of Slavic verb forms. *The Prague Bulletin of Mathematical Linguistics*, 105:143–193.

Zhang, B., Yang, J., Lin, Q., and Su, J. (2018). Attention regularized sequence-to-sequence learning for e2e nlg challenge. Technical report, E2E Challenge System Descriptions.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhang, X. and Lapata, M. (2017). Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

Zhang, Y. (2013). Partial-tree linearization: Generalized word ordering for text synthesis. In *Proceedings of theTwenty-Third international joint conference on Artifi-cial Intelligence*, pages 2232–2238. AAAI Press.

Zhang, Y., Blackwood, G., and Clark, S. (2012). Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 736–746, Avignon, France. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2015). Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.

Zhao, C., Walker, M., and Chaturvedi, S. (2020). Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online. Association for Computational Linguistics.

Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., and Eger, S. (2019). MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Zhou, Y., Liu, C., and Pan, Y. (2016). Modelling sentence pairs with tree-structured attentive encoder. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2912–2922, Osaka, Japan. The COLING 2016 Organizing Committee.