

Investigating Multilingual Approaches for Parsing Universal Dependencies

James Barry
BA, M.Sc

A Thesis Submitted for the Award of Doctor of Philosophy
(Ph.D.)

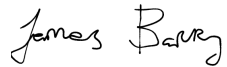
School of Computing
Dublin City University

Supervisors:
Dr. Jennifer Foster
Dr. Joachim Wagner

2022

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy (Ph.D.) is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

The image shows a handwritten signature in black ink that reads "James Barry". The signature is written in a cursive style with a clear, legible font.

ID No.: 16212754

Date: 30/08/2022

Acknowledgements

I would like to extend a sincere thank you to my PhD supervisors, Dr. Jennifer Foster and Dr. Joachim Wagner. Firstly, to Jennifer for agreeing to be my supervisor for my Masters Thesis, where I was scrambling to find a project to work on and discovered that Jennifer was working on Natural Language Processing, an area I was particularly interested in, and generously agreed to supervise me on a project in this area. I was fortunate enough to benefit from Jennifer's guidance throughout this period which made me happy to continue my academic journey under her supervision for a PhD. I have learnt a lot from Jennifer and I will miss working with her once my time as a PhD student has finished. I have always felt very fortunate to discuss research ideas with Jennifer and I sincerely appreciate and value the feedback I receive from her.

I would also like to thank my co-supervisor Joachim Wagner who has provided great supervision to me. I have been fortunate to directly learn from Joachim, where we have worked on many projects together where he has also given me good advice on programming. Joachim has instilled in me good practices for carrying out research and experiments and has always made sure I do things the right way. I have always been impressed by how generous Joachim is with his time when it comes to helping people with their research and his significant experience in the field has helped many students improve their skills as researchers.

A sincere thank you to Prof. Andy Way and Prof. Barbara Plank who served on my Examination Committee and gave me incredibly valuable feedback on my thesis and provided very interesting discussions in the viva. A special thanks to Dr Hyowon Lee for chairing the viva and making sure everything ran smoothly.

A particular highlight of my Ph.D. studies has been meeting people from all over the world at the ADAPT Centre and Dublin City University over the last four and a bit years who have been good colleagues and friends.

I enjoyed getting to meet people when attending academic conferences, particularly at EMNLP (2018) in Brussels where I got to meet fellow participants in the CoNLL 2018 shared task, and then at EMNLP in (2019) in Hong Kong where I met other participants at the DeepLo workshop, and many others at the main conference. Finally, it was great to meet many new people at ACL (2022) in Dublin and LREC (2022) in Marseille. A special thanks to anyone who walked me through their poster presentations, either physical or virtual, and to those who came to our poster sessions and provided fruitful and interesting discussions.

To members of the parsing community and the organisers of the CoNLL 2018 UD Parsing Shared Task and the IWPT 2020 and 2021 Shared Tasks on Parsing into Enhanced Dependencies, Dan Zeman, Djamé Seddah and Gosse Bouma as well as the IWPT Chairs Stephan Oepen and Kenjie Sagae—thank you for all of your contributions to the field of syntactic parsing and for creating opportunities for researchers in this area to participate in shared tasks and also for making us feel part of the community. I would also like to thank all of my collaborators who I have worked on papers with.

I would like to thank the maintainers of software I have found useful throughout the period of my PhD, particularly the developers of the AllenNLP and HuggingFace libraries which I have enjoyed familiarising myself with and which have also helped me to learn about NLP technologies and good programming practices. I would like to extend a sincere thank you to Chris Larkin and members of the Tensorflow Research Cloud who have been very supportive in terms of providing computational resources which have made releasing large language models in Irish possible.

I extend my gratitude to the staff at Dublin City University and the ADAPT Centre, for their help throughout my studies there. A special thanks to Andy Way (again!) and Annalina Caputo for being examiners for my PhD Transfer, and for their valuable feedback.

Finally, thanks to my supportive parents who have always helped me throughout my education and my brother who first got me into programming and has given me good advice in this area.

Contents

List of Tables	vi
List of Figures	viii
1 Introduction	1
1.1 Research Questions	5
1.2 Thesis Outline	10
1.3 Publications	11
2 Background	15
2.1 Dependency Parsing	15
2.1.1 Transition-based Parsing	18
2.1.2 Graph-based Parsing	22
2.2 Neural Networks and their Application to Dependency Parsing	26
2.2.1 Distributed Representations of Words	27
2.2.2 Recurrent Neural Networks	28
2.2.3 Transformer Networks and Self-Attention	30
2.2.4 Neural Graph-based Parsing	36
2.3 Multilingual Dependency Parsing	39
2.3.1 Differences in Annotation Styles	40
2.3.2 Morphologically-Rich Languages	41
2.4 Summary	42
3 Polyglot Training for Cross-lingual Transfer	44
3.1 Background	45
3.1.1 Cross-lingual Transfer	45
3.1.2 Parameter Sharing between Languages	50
3.2 Cross-lingual Parsing with Polyglot Training and Multi-treebank Learning	56
3.2.1 Method	58
3.2.2 Experiments	65
3.3 Summary	71

4	gaBERT a Monolingual Irish BERT Model	75
4.1	Background	76
4.1.1	Multilingual Language Models	77
4.2	gaBERT: an Irish Language Model	79
4.2.1	gaBERT Pretraining Data	80
4.3	Corpus Processing	81
4.3.1	Tokenisation and Segmentation	82
4.4	Hyperparameter Search for the Final Model	83
4.4.1	Corpus Filtering	83
4.4.2	Vocabulary Creation	86
4.4.3	BERT Training	86
4.4.4	Evaluation Measures	87
4.4.5	Results	89
4.4.6	Model Comparison	91
4.4.7	ELECTRA Model	97
4.4.8	Full Model Results	98
4.5	Summary	99
5	Parsing Enhanced Universal Dependency Graphs	101
5.1	Background	102
5.2	The DCU-EPFL Enhanced Dependency Parser at the IWPT 2021 Shared Task	108
5.3	Results and Further Experiments	117
5.3.1	Official Submission	117
5.3.2	Additional Experiments	117
5.4	Summary	122
6	Multiview Learning for Multilingual Dependency Parsing	123
6.1	Background	124
6.2	Methodology	126
6.2.1	Proposed Models	126
6.2.2	Training Procedure	128
6.3	Models	130
6.3.1	Input Features	130
6.3.2	Parsing Components	131
6.4	Experiments	131
6.5	Summary	138
7	Conclusion	141
7.1	Research Questions Revisited	141
7.2	Contributions	144
7.3	Future Work	144
	Bibliography	147

List of Tables

1.1	An example CoNLL-U sentence “ <i>They buy and sell books.</i> ”. The <i>Deps</i> and <i>Miscellaneous</i> columns are omitted for clarity, which contain the enhanced annotations as well as additional miscellaneous information, respectively.	4
3.1	Chosen hyperparameters for our POS tagging and parsing models, <i>both</i> means the feature is common to both the POS tagger and parser.	66
3.2	Source model LAS scores on the development treebanks using silver POS tags.	68
3.3	The number of valid sentences in the Faroese synthetic treebank for each source language after annotation projection and sentence filtering. The original corpus size was 28,862 sentences.	68
3.4	LAS on the target Faroese test treebank. <i>Single</i> refers to using a single synthetic Faroese treebank to train a Faroese model, <i>Multi</i> uses both a multi-treebank POS tagger and a multi-treebank parser with all synthetic Faroese treebanks. The multi-treebank model is tested with each of the five training treebanks (four projected from individual source languages and one using multi-source projection) as proxy treebank. Statistically significant differences between the monolingual and polyglot setting are indicated by † for each result pair, excluding averages.	69
3.5	LAS scores between target models trained on the subset of sentences eligible for multi-source projection (with annotations from the stated source).	70
3.6	Comparison to previous work. LAS on Faroese test set. Note that the first results uses predicted segmentation and tokenization whereas the rest used gold.	71
4.1	Sentence and word counts and plain text file size in megabytes for each corpus after tokenisation and segmentation but before applying sentence filtering.	81
4.2	The number of sentences and words which remain after applying the specific filter.	85
4.3	Chosen hyperparameters for the multitask parser and tagger.	88

4.4	LAS in dependency parsing (UD v2.8) for selected models. Median of five fine-tuning runs. Scores are calculated using the official UD evaluation script (<i>conll18_ud_eval.py</i>).	92
4.5	The number of times the original masked token was predicted (100 test items).	92
4.6	The number of matches, mismatches, copies and gibberish predicted by each model (100 test items).	93
4.7	Examples of cloze test predictions and classifications.	93
4.8	Accuracy of language models segmented by length of context cue where short: 4–10 tokens, medium: 11–20 tokens, and long: 21–77 tokens. . . .	94
4.9	Verbal MWE Identification: (P)recision, (R)ecall and F1 scores of the best performing gaBERT and mBERT model	96
4.10	Full model results on development data. For model name abbreviations, see Table 4.11.	99
4.11	Full model results on test data (os = fine-tuned off-the-shelf model, cp = continued pre-training before fine-tuning).	99
5.1	Chosen hyperparameters for the multitask graph and tree parser.	115
5.2	Evaluation scores on the official test data on the language-specific test files. All runs after Official use Trankit pre-processing and all runs after Trankit use the XLM-R_{Large} model. All numbers inside the parentheses are calculated as the relative error reduction of the particular column and the column it is compared with (which is shown in square brackets). . .	116
5.3	Evaluation scores on the official test data on the language-specific test files submitted by each team. We also include the official reference system (ref.) which copies the gold tree to the enhanced graph as well as (ours best) which is our best post-deadline run, which corresponds to the +Concat+MTL run in Table 5.2. The first and second top-scoring models in each language are specified with black and blue colour, respectively. .	119
6.1	The model types we consider and their components. Single-view models include the SDSV and MDSV settings, and the multi-view model corresponds to the MDMV setting.	131
6.2	Test results based on language groups (part 1/2). For table description, see the second part of the table (2/2).	133
6.3	Test results based on language groups (part 2/2). Results are calculated with the official CoNLL evaluation script (<i>conll18_ud_eval.py</i>). Bolded numbers represent the highest-score among the initial three settings: SDSV, MDSV and MDMV. In the last two columns (+embeds), underlined numbers are used for when one of these settings now achieves the highest score among all settings. Average scores are based on both parts of the results. OOM refers to language groups which could not be run on time; we only report averages for full runs.	134

List of Figures

1.1	A dependency tree for the sentence “ <i>John hit the ball with the bat</i> ”	2
2.1	A rare example of a non-projective English dependency tree for the sentence “ <i>A hearing is scheduled on the issue today.</i> ” Example taken from Kübler et al. (2009).	18
2.2	Types of transitions in the arc-eager transition system (Nivre, 2003). The variables σ and β represent arbitrary sublists of the stack Σ and buffer B . For clarity, the node on top of the stack is shown on the right and the next token (head of the buffer) is shown on the left. Thus, $c = (\sigma w_i, w_j \beta, A)$ represents a configuration with w_i on top of the stack Σ and w_j the next token in the buffer B	20
2.3	Transition-based parsing sequence using the arc-eager transition system for the sentence “ <i>Book the flight through Houston</i> ”. The Arc-set contains the dependency relations which are represented as (w_i, r, w_j) . Example taken from Jurafsky and Martin (2000).	21
2.4	Scaled Dot-Product Attention (left). This process is repeated h times in Multi-Head Attention (right), where the outputs are concatenated and then projected with a feedforward network. Figures taken from Vaswani et al. (2017).	33
2.5	The Transformer encoder block of Vaswani et al. (2017). Image taken from Lu et al. (2019).	35
2.6	Biaffine graph-based parser with a BiLSTM encoder operating over word embeddings. Figure taken from Dozat and Manning (2017).	39
3.1	Annotation projection from the <u>source</u> languages to the target language. Example translations and annotations taken from Tyers et al. (2018). . .	50
3.2	Left: Dependency edges and labels are combined in a single graph and resolved using the CLE algorithm (Chu and Liu, 1965; Edmonds, 1967). The counts of each labelled edge are provided in brackets. Right: The decoded dependency tree for the sentence. Example taken from Tyers et al. (2018).	51

3.3	An example of a dataset or language embedding, in this case <i>English</i> which is concatenated to the token-level input which consists of a word and a character representation and then passed to a BiLSTM encoder. . .	53
3.4	Overview of the machine translation process. An example Faroese sentence <i>Maja býr nú í Malmö</i> (Maja now lives in Malmö) is first translated into Norwegian Bokmål and then from Norwegian Bokmål into the other source languages using pivot translation. The language of the text is shown in bold and the arrow direction shows the direction of translation. Example translations are taken from Tyers et al. (2018).	59
3.5	Overview of the <i>monolingual</i> and <i>polyglot</i> parser experiments using Swedish translations as an example. This process is repeated for all source languages.	61
3.6	Multi-source projection. The source language is listed in brackets. The <i>Filter</i> process involves filtering out sentences where all four sources do not provide a valid projection.	63
3.7	Single versus multi-treebank training. The source language is listed in brackets.	65
4.1	Number of sentences in each corpus per filtering configuration where None: No-filter, Doc: Document-filter, OF-B: OpusFilter-basic, OF-BCL: OpusFilter-basic-char-lang.	85
4.2	Dependency parsing LAS for each filter type. Each box-plot shows five LAS scores obtained by fine-tuning the respective BERT model five times with different initialisation to obtain five different parsers.	90
4.3	Dependency parsing LAS for each vocabulary type. Each box-plot shows five LAS scores obtained by fine-tuning the respective BERT model five times with different initialisation to obtain five different parsers.	91
4.4	Verbal MWE Identification: Precision, Recall and F1 scores for each model across 20 random seed values	96
4.5	Dependency parsing LAS for each model type. Every 100k steps, we show the median of five LAS scores obtained from fine-tuning the respective model five times with different initialisation.	97
5.1	<i>Added subject relations in control and raising</i> : there is an added subject dependency between an embedded verb <i>sell</i> and its raised subject <i>Investors</i> . The extension “:xsubj” (marked in blue) is added in these cases.	103
5.2	<i>Shared heads and dependents in coordination</i> : in a conjoined phrase (e.g. buys and sells), each predicate shares the subject (the store) and the object (cameras). The additional edges are marked in blue.	103
5.3	<i>Insertion of null nodes for elided predicates</i> : \mathcal{E} denotes an elided token which is added to the enhanced graph. The additional edges in the EUD graph are marked in blue and removed tree edges are marked with a red dotted arc.	104

5.4	<i>Co-reference in relative clause constructions</i> : a special <i>ref</i> relation is added to the relative pronoun from its antecedent, and the antecedent is attached as an argument to the main predicate of the relative clause. The additional edges in the EUD graph are marked in blue and removed edges are marked with a red dotted arc. Note that the EUD graph contains a cycle between <i>house</i> and <i>built</i>	105
5.5	<i>Augmenting modifier relations with prepositional or case-marking</i> : each conjunct in this example conjoined noun phrase is attached to the governor of the modifier phrase, e. g. there is an additional <i>nmod</i> relation marked in blue between the noun <i>Tale</i> and the second conjunct <i>sorrow</i> . Note that the lemma of the <i>case</i> and <i>cc</i> dependents are appended to the enhanced dependency labels of their heads.	105
5.6	DCU-EPFL multitask architecture.	111
6.1	Comparison between single-view and multiview architectures for POS tagging. In the single-view architecture, the character and word representations are concatenated and a single classifier is used. In the multiview architecture, sentence-based character and word representations are learned separately and are passed to their own classifiers. The outputs of the MLPs from each module are combined in the meta classifier (which is the top classifier in (b)), which makes the final prediction. . .	126
6.2	Architecture of the Multiview Parser. This figure only shows an example with one dataset. However, a separate single-dataset head is created for each dataset and the input is routed to that head depending on the input source, which is a homogeneous batch of data from that source.	129

Investigating Multilingual Approaches for Parsing Universal Dependencies

James Barry

Abstract

Multilingual dependency parsing encapsulates any attempt to parse multiple languages. It can involve parsing multiple languages in isolation (poly-monolingual), leveraging training data from multiple languages to process any of the included languages (polyglot), or training on one or multiple languages to process a low-resource language with no training data (zero-shot). In this thesis, we explore multilingual dependency parsing across all three paradigms, first analysing whether polyglot training on a number of source languages is beneficial for processing a target language in a zero-shot cross-lingual dependency parsing experiment using annotation projection. The results of this experiment show that polyglot training produces an overall trend of better results on the target language but a highly-related single source language can still be better for transfer. We then look at the role of pretrained language models in processing a moderately low-resource language in Irish. Here, we develop our own monolingual Irish BERT model gaBERT from scratch and compare it to a number of multilingual baselines, showing that developing a monolingual language model for Irish is worthwhile. We then turn to the topic of parsing Enhanced Universal Dependencies (EUD) Graphs, which are an extension of basic Universal Dependencies trees, where we describe the *DCU-EPFL* submission to the 2021 IWPT shared task on EUD parsing. Here, we developed a multitask model to jointly learn the tasks of basic dependency parsing and EUD graph parsing, showing improvements over a single-task basic dependency parser. Lastly, we revisit the topic of polyglot parsing and investigate whether multiview learning can be applied to the problem of multilingual dependency parsing. Here, we learn different views based on the dataset source. We show that multiview learning can be used to train parsers with multiple datasets, showing a general improvement over single-view baselines.

Chapter 1

Introduction

Dependency parsing is one of the long-standing problems in Natural Language Processing (NLP). The goal of dependency parsing is to provide a text with a syntactic analysis which adheres to dependency grammar. This is achieved by providing links, termed dependencies or dependency relations, between a head word and its dependent(s), e. g. a dependency relation between a verb and a subject.¹ Labels can be assigned to the dependencies which specify the type of grammatical relation that holds between a head-dependent pair. For example, consider the sentence “*John hit the ball with the bat*” shown in Figure 1.1.² The arrows indicate a relationship between a head word and its dependent(s) and the labels specify the type of grammatical relation. From this example, we can see that *John* is the nominal subject (*nsubj*) of the verb *hit* and *ball* is the direct object (*dobj*). The preposition *with* indirectly links the verb *hit* to *bat*, indicating that the bat was used to hit the ball. Each word in the sentence has exactly one other word as its head (i. e. only has one incoming arc) except *hit*, the head of the sentence, which we refer to as the root of the sentence and we assign an artificial token *root* as its head. The *root* token is the left-most word in Figure 1.1 and including this token helps from

¹There are several terms we can use for a head word such as “parent” or “governor” and a dependent can also be referred to as a “modifier”, “child” or “argument”.

²Example taken from McDonald and Pereira (2006).

both a linguistic and algorithmic standpoint (Kübler et al., 2009). The resulting syntactic structure is referred to as a *dependency tree* and is the syntactic structure we will mainly be focusing on throughout this project.

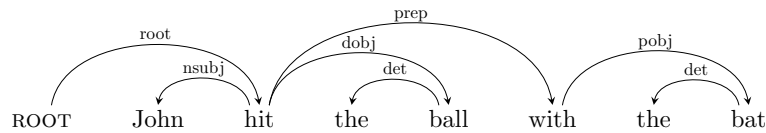


Figure 1.1: A dependency tree for the sentence “*John hit the ball with the bat*”.

Providing direct links between words and their dependents enables us to explicitly show relationships between pairs of words. Taken together, these predicate-argument relationships can provide an insight as to what is happening in a sentence, which can be helpful for other NLP tasks. Among such tasks include Semantic Role Labelling (SRL), where Strubell et al. (2018) were able to outperform previous methods which lack access to explicit syntactic information by modelling syntactic dependencies alongside the main task of SRL. More recently, Mohammadshahi and Henderson (2021) show that directly encoding dependency information to the self attention function in a Transformer (Vaswani et al., 2017) is helpful for SRL. This finding suggests that incorporating syntactic supervision is complementary to unsupervised learning methods which learn syntactic information in an end-to-end manner. In other domains such as aspect-level sentiment analysis, where items and their modifiers may be separated by many intermediate words (e. g. a noun modified by an adjective), dependency information is helpful for making the relationship between such tokens explicit (Sun et al., 2019; Huang et al., 2020). Lastly, by encoding predicted dependency trees with graph-convolutional networks to source-side sentences, Bastings et al. (2017) show that adding such syntactic information is helpful for machine translation (MT).

As outlined by de Lhoneux (2019) and Nivre (2020a), the field of dependency parsing has always enjoyed a multilingual focus, primarily due to the presence of annotated

dependency treebanks in multiple languages (Buchholz and Marsi, 2006; Nivre et al., 2007a). While treebanks were available for multiple languages, the situation where treebanks have been annotated according to different schemes made it difficult to perform reliable comparative analyses between languages and meant that there was limited scope for (a) transferring information across treebanks and languages, and (b) developing multilingual parsers that are able to generalise over the included languages.

Universal Dependencies (UD) (Nivre et al., 2016) is a relatively new project which seeks to address the issue of differing annotation schemes across languages by creating a common set of annotation guidelines for all included treebanks. Specifically, it incorporates the combination of several older projects which aimed to enhance cross-lingual consistency among resources from different languages, namely the Google universal tag set (Petrov et al., 2012), the universal Stanford dependencies (de Marneffe et al., 2014), the universal Google dependency scheme (McDonald et al., 2013) and the Inter-set morphosyntactic tag set of Zeman (2008). UD enables the blending of general and language-specific information by including features such as universal POS tags which contain coarse-grained information about the grammatical category of a word, but also language-specific POS tags which can contain fine-grained information about a word including its gender, number and case. Furthermore, in addition to the core set of universal dependency labels, UD allows language-specific dependency relation subtypes.

The latest UD v2.8 release (Zeman et al., 2021) contains nearly 200 treebanks covering 134 languages that have been annotated following the same guidelines. Each UD treebank contains tokenised sentences with features such as the word form; a lemma, which is the base form of the word; the universal and language-specific POS tags, which represent the grammatical category of the word and possibly more fine-grained information; a set of morphological features, which represent lexical and grammatical properties of the word, and the syntactic dependency annotations which are a set of typed dependency relations specifying the grammatical relationship between a head word and

modifier (as shown in Figure 1.1) (Nivre, 2020b). An example sentence with the UD annotations in CoNLL-U format is shown in Table 1.1.³

ID	Form	Lemma	UPOS	XPOS	Feats	Head	Deprel
1	They	they	PRON	PRP	Case=Nom Number=Plur	2	nsubj
2	buy	buy	VERB	VBP	Number=Plur Person=3 Tense=Pres	0	root
3	and	and	CONJ	CC	_	4	cc
4	sell	sell	VERB	VBP	Number=Plur Person=3 Tense=Pres	2	conj
5	books	book	NOUN	NNS	Number=Plur	2	obj
6	.	.	PUNCT	.	_	2	punct

Table 1.1: An example CoNLL-U sentence “*They buy and sell books.*”. The *Deps* and *Miscellaneous* columns are omitted for clarity, which contain the enhanced annotations as well as additional miscellaneous information, respectively.

Typically, a data-driven dependency parser will use some combination of input features from the *Form*, *Lemma*, *UPOS*, *XPOS*, and *Feats* labels to predict the *Head* and *Deprel* annotations. Dependency parsers are usually evaluated in two settings: the unlabelled attachment score (UAS) refers to the number of times the correct head for each token was predicted, and the labelled attachment score (LAS) refers to the number of times the correct head *and* label were predicted. The treebanks usually consist of training, validation and test portions, though for some languages, there may be no training portions, and various cross-lingual approaches must be used to process them.

In addition to the basic dependency relations described so far, some UD treebanks contain the enhanced dependency annotations (Schuster and Manning, 2016), which enable a more expressive annotation scheme that can explicitly handle certain coordination structures and argument sharing in control and raising. Dependency trees are single-rooted, acyclic, labelled trees where words only have one head. Enhanced Universal Dependencies (EUD) graphs⁴ are an extension of basic UD trees which can contain cycles and also allow words to have more than one head. EUD graphs have been proposed as an alternative to basic UD trees as they are more suitable for represent-

³Example taken from <https://universaldependencies.org/format.html>

⁴<https://universaldependencies.org/u/overview/enhanced-syntax.html>

ing phenomena such as modifier sharing across conjuncts and by their nature, are better able to represent semantic structure than dependency trees. As stated by Schuster and Manning (2016), dependency graphs are more useful in downstream Natural Language Understanding (NLU) tasks than strict surface-structure dependency trees. This is because they are able to better represent certain phenomena such as coordination, argument sharing in control and raising, and relative clauses. As such, we will also be predicting Enhanced Universal Dependency *graphs* in Chapter 5.

In this thesis, the UD treebanks represent the resource from which data-driven parsing models can be trained and evaluated on. Furthermore, the existence of treebanks which have differing data sizes, contain the enhanced dependency annotations, and most importantly, have been annotated with cross-lingual consistency in mind, presents a ripe opportunity to (a) explore techniques for inducing cross-lingual transfer to low-resource languages, (b) ascertain what approaches used in dependency parsing can be used to parse the enhanced annotations, and (c) examine multilingual parser development so that parsers can generalise across any of the included languages. This thesis aims at addressing these three key areas.

1.1 Research Questions

Having introduced the concept of dependency parsing and the UD project—the task and dataset used throughout this thesis, respectively—we will now discuss the specific research questions this Ph.D. thesis seeks to answer. At a general level, all of the research questions which we propose in this thesis stem from the primary research objective:

How can we improve multilingual dependency parsing?

As described by de Lhoneux (2019), multilingual dependency parsing encapsulates any attempt to parse multiple languages. It can involve parsing multiple languages in isolation (poly-monolingual), leveraging training data from multiple languages to process

any of the included languages (polyglot), or training on one or multiple languages to process a low-resource language with no training data (zero-shot). In this thesis, we attempt to improve parsing performance across all three paradigms. We will now list our specific research questions.

RQ1: Can zero-shot cross-lingual dependency parsing be improved by leveraging polyglot training on source languages?

The situation where multiple treebanks have been annotated according to the same universal format has brought renewed attention to multilingual parser development. Among these works are approaches which look at polyglot parsing. Polyglot parsing refers to cases where a single model is trained using multiple languages, where languages have equal status and parameters are shared between languages (de Lhoneux, 2019; Mulcaire et al., 2019b; Kondratyuk and Straka, 2019). The idea behind training a polyglot model is that the model can exploit similarities in morphology and syntax across the included languages, or learn generalisations between the languages which would not be possible by training on monolingual data alone (Ammar et al., 2016; Kondratyuk, 2019; Smith et al., 2018).

Following the release of UD, numerous works have attempted to combine universal treebanks across languages for inducing polyglot parsers: Vilares et al. (2016) showed that training lexicalised bilingual parsers seldom leads to a significant reduction in monolingual performance. Smith et al. (2018) showed that polyglot training was helpful when working with related languages and that low-resource languages, in particular, benefit the most. While previous work has shown promising results when applying polyglot training for low-resource languages which have a small amount of data, the first research question asks whether polyglot training can also help in a *zero-shot* cross-lingual dependency parsing experiment, where there is no labelled data available for the target language, and cross-lingual techniques must be used to transfer information from

source languages to the target language.

Zero-shot dependency parsing can be approached through annotation projection (Yarowsky and Ngai, 2001; Tiedemann and Agić, 2016), where annotations are projected from a source language to a target language using word alignments. Tyers et al. (2018) demonstrate that zero-shot dependency parsing can be achieved by transferring annotations from source translations parsed using models trained on source treebanks to target-language sentences along word alignments and then training a target-language parser on the projected annotations. In Tyers et al. (2018), annotations are transferred from monolingual source parsers, and are then combined in an ensemble where the final tree is selected through arc-voting. Here, there is no cross-lingual interaction until the graph voting step. Given the promising results seen from polyglot training on related languages in Smith et al. (2018), we hypothesise that training on a combination of related source languages in a polyglot fashion will lead to a more robust model capable of providing higher quality dependency annotations to the target language. RQ1 seeks to test whether this is the case by using Faroese as a zero-shot target language, and using Norwegian Nynorsk, Norwegian Bokmål and Swedish as related source languages to transfer annotations from.

RQ2: Does a monolingual language model improve low-resource dependency parsing in the case of Irish?

The recent rise of large-scale language models has played a disproportionately large role towards increasing evaluation scores across a wide variety of NLP tasks (Peters et al., 2018; Devlin et al., 2019). Prior to these models, static word embeddings (Al-Rfou' et al., 2013; Mikolov et al., 2013; Pennington et al., 2014) were learned on unlabelled corpora and then used to enrich lexical features for NLP tasks. The introduction of large-scale neural language models, where contextually-aware word representations are produced as a function of their surrounding context, saw a paradigm shift for modern NLP with

most data-driven NLP models now incorporating these input features as standard.

Along with the release of such models, research has mainly focused on how their representations contribute to various extrinsic NLP tasks such as dependency parsing (Kulmizev et al., 2019; Liu et al., 2019a; Straka et al., 2019) or what these models learn as part of their training procedure (Jawahar et al., 2019; Tenney et al., 2019; Rogers et al., 2020). Another research theme that has emanated from the large-scale release of language models surrounds the development of multilingual language models trained on data from multiple languages (Devlin et al., 2019; Lample and Conneau, 2019; Conneau et al., 2020). With the presence of monolingual and multilingual language models, a natural question that arises is which type of model will work best for a downstream task for a particular language. For high-resource languages such as Finnish (Virtanen et al., 2019), Dutch (de Vries et al., 2019), French (Martin et al., 2020), and Farsi (Farahani et al., 2021), the monolingual variants outperform the more general multilingual models. However, the case is more opaque for lower-resource languages such as Irish, which may benefit from augmenting the relatively small amount of pretraining data with data from other languages. As such, we endeavour to create a monolingual Irish language model, gaBERT, and compare it with state-of-the-art multilingual baselines such as Multilingual BERT (mBERT) (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) as well as with mBERT with continued pre-training on the same Irish corpora used to train our model. In doing so, we seek to answer whether a monolingual language model is better than existing and modified multilingual models for dependency parsing in Irish.

RQ3: How can we leverage existing techniques to parse Enhanced Universal Dependencies?

The first two research questions concern the basic UD annotation, i. e. where the output structure is restricted to a dependency tree. Dependency graphs, on the other hand, are more useful in downstream NLU tasks than strict surface-structure dependency trees

due to their ability to explicitly capture certain relationships between content words (Schuster and Manning, 2016). In light of the demands from downstream applications requiring more semantic inputs, the IWPT 2020 and 2021 Shared Tasks on Multilingual Parsing into Enhanced Universal Dependencies (Bouma et al., 2020, 2021) call for participants to predict EUD graphs for multiple languages, starting from raw text. Given that these tasks are the first of this nature, RQ3 will look at ways of extrapolating existing approaches for dependency parsing to parse EUD graphs. In more detail, it will look at the role of upstream processing, the choice of pre-trained language model, treebank concatenation and multitask learning between a basic dependency parser and an EUD graph parser in contributing to EUD parsing accuracy. Taken together, these insights are used in a final system which is evaluated in the setting of the 2021 IWPT Shared Task (Bouma et al., 2021) and achieves the second highest scoring results among all participants in unofficial scores (Barry et al., 2021).

RQ4: Can Multiview Learning Help Multilingual Dependency Parsing?

Multiview Learning is a branch of machine learning that involves learning multiple *views* of the same data in order to improve generalisation performance (Zhao et al., 2017). Each view is a function that models a different property of the data where the information from each view is then combined such that it should have a synergistic effect. Initial work by Bohnet et al. (2018) has shown that a multiview architecture can be used to achieve state-of-the-art POS and morphological tagging accuracy. In their setup, they learn different views of the data at the character and token level and then combine this information in a *meta* model that takes the output of the two other models. We adopt a similar architectural setup to Bohnet et al. (2018) but treat *dataset source* as a source of multiview data, i. e. data from a different language or treebank represents a different view. We then present a model which learns a single dataset view as well as a shared dataset view and the information from these models is later combined in a meta model.

This model is compared to baselines where a model is trained on the individual dataset and also a single-view baseline where the datasets are simply concatenated. RQ4 will explore the utility of such an approach for multilingual dependency parsing.

1.2 Thesis Outline

The remainder of this thesis is structured as follows:

Chapter 2: Background We first describe dependency syntax and the task of dependency parsing. Concisely, we then describe pre-neural and neural dependency parsing, and describe the components used in modern neural dependency parsers, e. g. recurrent neural networks and Transformer networks and approaches taken by modern parsing models. Chapter 2 is mainly concerned with providing a background to the task of dependency parsing and the components used in typical dependency parsers. As such, the material which is related to each of the research questions will mainly be left to the appropriate chapters.

Chapter 3: Polyglot training and cross-lingual transfer We discuss the literature related to zero-shot cross-lingual transfer and polyglot parsing in Chapter 3. This chapter also details an experiment which analyses the role of polyglot training for zero-shot cross-lingual transfer which aims to answer RQ1: *Can zero-shot cross-lingual dependency parsing be improved by leveraging polyglot training on source languages?*

Chapter 4: Pre-trained language models for low-resource NLP Chapter 4 discusses the role of pre-trained language models⁵ for processing low-resource languages. We then describe an experiment which seeks to answer RQ2: *Does a monolingual language model improve low-resource dependency parsing in the case of Irish?*

⁵These are also referred to as “large language models” (LLMs) or “foundation models”.

Chapter 5: Enhanced Universal Dependencies Chapter 5 concerns the parsing of EUD graphs and describes a number of experiments carried out during the IWPT 2020 and 2021 Shared Tasks on parsing EUD graphs (Bouma et al., 2020, 2021) which are aimed at addressing RQ3: *How can we leverage existing techniques to parse Enhanced Universal Dependencies?* Specifically, we show how better upstream processing, the choice of encoder model, monolingual and multilingual treebank concatenation, and the use of multitask learning between a dependency tree parser and an EUD graph parser contribute towards EUD parsing accuracy.

Chapter 6: Multiview Learning for Multilingual Dependency Parsing Chapter 6 explores the use of multiview learning for multilingual dependency parsing. Within the context of multilingual dependency parsing, different views are created based on private or dataset-specific information, and public or shared-dataset information, where each view has its own parser. The representations from each parser are then passed to a meta view which learns to combine their states. The goal is to ascertain whether multiview learning can be used to combine dataset-specific and shared-dataset information in a positive manner. In doing so, we wish to provide some insight into our final research question RQ4: *Can Multiview Learning Help Multilingual Dependency Parsing?*

1.3 Publications

The work described in this thesis has been published in the following papers:

James Barry, Joachim Wagner, and Jennifer Foster. 2019. **Cross-lingual Parsing with Polyglot Training and Multi-treebank Learning: A Faroese Case Study**. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 163–174, Hong Kong, China. Association for Computational Linguistics.

This paper describes an experiment which looks at the role of using multiple monolingual models versus a single polyglot model for zero-shot cross-lingual transfer and is used to answer RQ1: *Can zero-shot cross-lingual dependency parsing be improved by leveraging polyglot training on source languages?*

James Barry, Joachim Wagner, Lauren Cassidy, Alan Cowap, Teresa Lynn, Abigail Walsh, Micheal J. 'O Meachair, Jennifer Foster (2022). **gaBERT — an Irish Language Model**. In *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, pages 4774–4788, Marseille, France. European Language Resources Association.

This paper describes the creation of a monolingual BERT model for Irish, gaBERT, which is compared to state-of-the-art multilingual baselines across a number of downstream tasks. The experiments in this paper are used to answer RQ2: *Does a monolingual language model improve low-resource dependency parsing in the case of Irish?*

James Barry, Joachim Wagner, and Jennifer Foster. 2020. **The ADAPT Enhanced Dependency Parser at the IWPT 2020 Shared Task**. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 227–235, Online. Association for Computational Linguistics.

James Barry, Alireza Mohammadshahi, Joachim Wagner, Jennifer Foster, and James Henderson. 2021. **The DCU-EPFL Enhanced Dependency Parser at the IWPT 2021 Shared Task**. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 204–212, Online. Association for Computational Linguistics.

These papers describe the *ADAPT-DCU* and *DCU-EPFL* submissions to the 2020

and 2021 Shared Tasks on Parsing Enhanced UD graphs and are used to answer RQ3: *How can we leverage existing techniques to parse Enhanced Universal Dependencies?*

Collaborative work, indirectly related to the thesis research but not described in the thesis have been published as follows:

Joachim Wagner, **James Barry**, and Jennifer Foster. 2020. **Treebank Embedding Vectors for Out-of-Domain Dependency Parsing**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8812–8818, Online. Association for Computational Linguistics.

This paper involves treebank embedding prediction for out-of-domain dependency parsing (Wagner et al., 2020), where I contributed in the form of updating the transition-based parser used in these experiments to work with contextualised ELMo representations (Peters et al., 2018), and analysed the utility of using ELMo representations as a means of comparing sentence similarity between test sentences and training sentences. I also performed statistical significance testing on the outputs.

Henry Elder, Jennifer Foster, **James Barry**, and Alexander O’Connor. 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.

This paper surrounds controlled neural surface realisation using an intermediate representation (Elder et al., 2019). I contributed to this work by performing manual evaluation of the generated outputs between the proposed system and a baseline.

Lauren Cassidy, Teresa Lynn, **James Barry**, and Jennifer Foster. 2022. **TwittIrish: A Universal Dependencies Treebank of Tweets in Modern Irish**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6869–6884, Dublin, Ireland. Association for Computational Linguistics.

This paper details the creation of a user-generated content (UGC) treebank *TwittIrish* of Irish tweets which has been added to the UD collection. In this work, I developed the parser which was used to perform numerous rounds of bootstrapping: where the automatically predicted annotations were corrected by the main annotator and then used in the next round of training.

Chapter 2

Background

This chapter provides the background material necessary to understand the techniques and models used to approach our research questions. We first start out by describing dependency syntax and the theory that underpins dependency parsing and the two main approaches for inducing data-driven dependency parsers: transition-based and graph-based dependency parsing, respectively. We then describe the shift from pre-neural to neural dependency parsing and explain the various components used in modern neural dependency parsers.

2.1 Dependency Parsing

As described in Nivre and Kübler (2006) and Kübler et al. (2009), dependency parsing is the task of assigning a syntactic analysis to lexical items by providing links, i. e. binary asymmetric relations termed dependencies, between the words of a sentence. Dependency parsing is based on the linguistic framework of dependency syntax. The origins of dependency syntax date back to Pāṇini’s Sanskrit grammar in the 4th century and its modern interpretation in the work of Tesnière (1959). Using the English translation provided in Nivre (2006), Tesnière (1959) outlines that linguistic sentences represent

a complete structure made up of individual elements which are the words of sentence. The words are not treated as isolated units, but each word invokes a mental connection to another word, and taken together, the individual connections form the structure of the sentence. The connections take a form of dependence, which involves linking a “superior” term with an “inferior” term; where the superior term is given the name “governor” and the inferior term “subordinate”. For example, in the phrase “Tom walks [...]”, *walks* is the governor and *Tom* is the subordinate. These binary relations (consisting of governor and subordinate items) form a syntactic structure referred to as a dependency tree, e. g. as in Figure 1.1.

Formally, a dependency tree is a directed graph $G = (V, A)$, consisting of a set of V nodes (or vertices) and a set of A arcs, with a linear precedence order $<$ on V . The nodes in V are labelled with word forms and other possible annotations such as POS tags, and the arcs in A are labelled with dependency types. For example, the nodes in V represent the words of a sentence S where $V_S = \{\text{ROOT}, \textit{John}, \textit{hit}, \textit{the}, \textit{ball}, \textit{with}, \textit{the}, \textit{bat}\}$. A contains the labelled dependency relations for the particular dependency graph G . A labelled dependency relation is expressed as (w_i, r, w_j) , where w_i represents a head word, w_j a dependent word, and r specifies the relation type. The relation type r comes from a predefined set of dependency relations R , where, for example, Universal Dependencies includes 37 universal syntactic relations.¹ An example of a labelled dependency relation in Figure 1.1 would be $(\textit{hit}, \textit{dobj}, \textit{ball})$, where the head word *hit* is modified by its direct object *ball*. Thus, dependency parsing involves producing a well-formed dependency graph $G = (V, A)$ given an input sentence S and dependency relation set R .

There are a number of formal conditions for dependency trees which are listed below:

- G is (weakly) **connected**: For every node i there is a node j such that $i \rightarrow j$ or $j \rightarrow i$.

¹<https://universaldependencies.org/u/dep/>

- G is **acyclic**: If $i \rightarrow j$ then not $j \rightarrow^* i$.
- G obeys the **single-head** constraint: If $i \rightarrow j$, then not $k \rightarrow j$, for any $k \neq i$.

The *connectedness* constraint ensures that the syntactic structure is complete; *acyclicity* means the syntactic structure is hierarchical, and the *single-head* constraint ensures that every word has at most one syntactic head. A special root node ROOT can enforce connectedness (Nivre and Kübler, 2006).

While it is too rigid a restriction from a linguistic point of view, another property of dependency trees is projectivity, which refers to whether a dependency tree is projective or non-projective. As stated in Kübler et al. (2009), a dependency arc (w_i, r, w_j) is projective if it satisfies the condition $w_i \rightarrow^* w_k$ for all $i < k < j$ when $i < j$, or $j < k < i$ when $j < i$. This means that an arc in a tree is projective provided there is a directed path from a head word w_i to all of the words between the endpoints of the arc. If every arc $(w_i, r, w_j) \in A$ in the tree $G = (V, A)$ satisfies the above condition then the tree is said to be projective. In some cases, a projective tree is not adequate to analyse a sentence as the sentence must contain crossing dependencies. Figure 2.1 shows such an example of a non-projective English sentence. Here, the preposition *on* modifies *hearing* and this arc crosses with the main verb *is*. While non-projective sentences are relatively rare in English, for some languages such as German or Czech, which exhibit more free word order, they are more frequent and parsing algorithms must be able to account for non-projective analyses of sentences. For instance, McDonald et al. (2005b) showed that despite the proportion of non-projective dependencies in the Czech treebank being less than 2%, their non-projective algorithm outperformed a projective baseline by 1.1%. In contrast, their parser which only produces projective trees was more accurate for English.

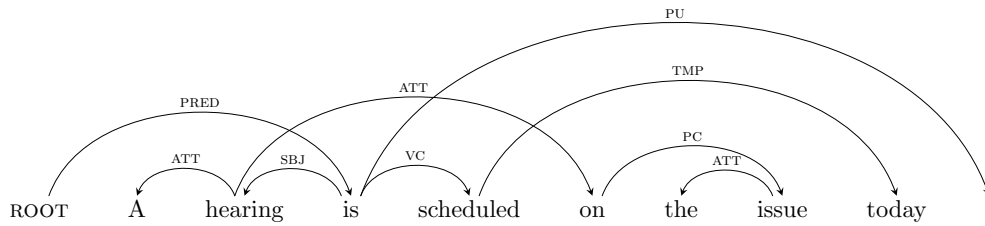


Figure 2.1: A rare example of a non-projective English dependency tree for the sentence “A hearing is scheduled on the issue today.” Example taken from Kübler et al. (2009).

2.1.1 Transition-based Parsing

Transition-based parsing comprises one of two main approaches to dependency parsing, with the other being graph-based or arc-factored parsing (Kübler et al., 2009). Transition-based parsing involves building dependency structures in an incremental fashion whereby a parser performs a sequence of actions or *transitions* until a final parse tree is built. As described by Nivre (2008), a treebank-induced classifier is used to predict the optimal next transition given a predefined list of possible transitions and a current parser state. As the parse tree is built from a series of incremental actions, pure transition-based systems are only capable of providing a single analysis of a dependency tree, though work by Zhang and Clark (2008) shows that adding beam-search to a transition-based parser can improve over standard deterministic parsing. A byproduct of this deterministic approach is that it prunes away parses that are unlikely and makes parsing simple and efficient, e.g. results in a runtime that is linear in the length of the input provided that each transition can be performed in constant time (Nivre, 2003, 2008). One drawback of this greedy deterministic procedure is that the parser must commit to its decisions which can result in error propagation due to an incorrect action earlier on in the sequence. Nevertheless, such parsers (Nivre et al., 2006) have achieved top-performing accuracy on multilingual data such as on the 13 languages in

the 2006 CoNLL shared task (Buchholz and Marsi, 2006).

A seminal work in the field of transition-based dependency parsing is that of Yamada and Matsumoto (2003). The classifier they use is a Support Vector Machine (SVM) (Cortes and Vapnik, 1995). SVMs are able to learn combinations of multiple features when using a polynomial kernel and are also capable of generalising in high-dimensional feature spaces, making them a popular choice of machine learning classifier in pre-neural transition-based parsing. Being able to combine multiple features is important in parsing because combinations of lexical and POS tag information are helpful for predicting syntactic structure (Hall et al., 2007).

The parser of Yamada and Matsumoto (2003) is a type of “shift-reduce” parser (e. g. Nivre (2003)) where most transition-based systems belong to this category. Shift-reduce parsers typically include a configuration consisting of a stack S of tokens currently being processed, a queue or buffer B of input tokens to be processed, and a list of previous parser actions A (Nivre, 2003). As the term implies, shift-reduce parsing involves two actions: *shift* and *reduce*. Shift actions push words from the buffer onto the stack for processing and reduce actions pop words from the stack or buffer when a word is assigned as a dependent. A reduce action can either be *reduce-left* or *reduce-right* depending on the location of the head of the reduced item on the stack/buffer. Reduce actions also involve predicting a dependency label for the reduced word. The types of actions a parser is allowed to take is determined by the transition system, where different transition systems vary with respect to the number of transitions involved, whether words can be reduced from the buffer or only from the stack, or what kind of reordering strategies are used to manipulate the position of words on the stack/buffer. Most transition systems consist of three or four possible operations which can be sub-classified as either push or pop actions. Push actions manipulate the position of words on the stack and buffer and pop actions remove items from the stack or buffer once a word is assigned as a dependent.

Figure 2.2 describes the arc-eager transition system of Nivre (2003). A parser con-

Initial	$[(0, W, \emptyset)]$	
Terminal	$[(0, nil, A)]$	
Transition		Precondition
Shift	$(\sigma, w_i \beta, A) \Rightarrow (\sigma w_i, \beta, A)$	
Left-Arc	$(\sigma w_i, w_j \beta, A) \Rightarrow (\sigma, w_j \beta, A \cup \{(w_j, r, w_i)\})$	$\neg \text{HEAD}(w_i)$
Right-Arc	$(\sigma w_i, w_j \beta, A) \Rightarrow (\sigma w_i w_j, \beta, A \cup \{(w_i, r, w_j)\})$	
Reduce	$(\sigma w_i, \beta, A) \Rightarrow (\sigma, \beta, A)$	$\text{HEAD}(w_i)$

Figure 2.2: Types of transitions in the arc-eager transition system (Nivre, 2003). The variables σ and β represent arbitrary sublists of the stack Σ and buffer B . For clarity, the node on top of the stack is shown on the right and the next token (head of the buffer) is shown on the left. Thus, $c = (\sigma | w_i, w_j | \beta, A)$ represents a configuration with w_i on top of the stack Σ and w_j the next token in the buffer B .

figuration consists of a triple $c = (\Sigma, B, A)$ with stack Σ , buffer B and a set of arcs A for the dependency tree being constructed. At the initial configuration, the stack only contains the dummy ROOT symbol 0, which acts as the head of the entire sentence. The buffer contains the full list of tokens in the sentence W (in their original order) and the list of dependency relations in the tree is empty \emptyset . The parsing algorithm will apply a transition to the partially-parsed configuration until termination where the buffer is empty, the stack only contains the ROOT symbol 0 and A contains the dependency relations of the tree. The transitions involved in the arc-eager transition-system are listed below, where *top* refers to the top word of the stack and *next* is the first word of the buffer.

1. **Shift** pushes *next* from the buffer onto the stack.
2. **Left-Arc** assigns *top* as a dependent of *next* and pops *top* from the stack. This is only allowed provided if *top* is not the artificial root node 0 and has not been already assigned a head.
3. **Right-Arc** assigns *next* as a dependent of *top* and pushes *next* to the stack.

4. **Reduce** pops *top* off the stack (only allowed if *top* has a head).

This transition system is shown for an example sentence in Figure 2.3. Note how **Left-Arc** pops *top* from the stack and how **Right-Arc** pushes *next* to the stack.

Step	Stack (σ)	Buffer (β)	Action	Arc-set (A)
0	[root]	[book, the, flight, through, houston]	Right-Arc	(root, root, book)
1	[root, book]	[the, flight, through, houston]	Shift	
2	[root, book, the]	[flight, through, houston]	Left-Arc	(flight, <i>det</i> , the)
3	[root, book]	[flight, through, houston]	Right-Arc	(book, <i>obj</i> , flight)
4	[root, book, flight]	[through, houston]	Shift	
5	[root, book, flight, through]	[houston]	Left-Arc	(houston, <i>case</i> , through)
6	[root, book, flight]	[houston]	Right-Arc	(flight, <i>nmod</i> , houston)
7	[root, book, flight, houston]	[]	Reduce	
8	[root, book, flight]	[]	Reduce	
9	[root, book]	[]	Reduce	
10	[root]	[]	Done	

Figure 2.3: Transition-based parsing sequence using the arc-eager transition system for the sentence “Book the flight through Houston”. The Arc-set contains the dependency relations which are represented as (w_i, r, w_j) . Example taken from Jurafsky and Martin (2000).

Transition-based parsers make use of an oracle during training which specifies the optimal transition sequence to follow in order to construct the gold parse tree. The transition sequences are used to train a machine learning classifier. This represents a form of data-driven dependency parsing whereby a dependency parser will learn the correct parsing action to take based on the current configuration. At parsing time, the classifier approximates the oracle in a deterministic manner until a parse tree is built (Nivre, 2003, 2004). The most common inference method in transition-based parsing involves greedy deterministic parsing in a left-to-right direction where a classifier chooses the next action based on the parser state and history (Nivre et al., 2007a). Subsequent research has explored different inference methods which can consider a list of several possible actions which are ordered by their probabilities. In this respect, Zhang and Clark (2008) modify a transition-based parser to include beam-search during training and inference and show that this can yield improvements over a purely deterministic transition-based parser. Similarly, Goldberg and Nivre (2012) introduce a “dynamic” oracle which can

consider alternative predictions than that specified by the oracle. This can make the parser more robust to the type of errors made at predict time and helps alleviate some of the problems related to greedy error propagation.

Standard transition-based algorithms (Nivre, 2003, 2004) can only produce projective parse trees (i. e. a nested structure with no crossing arcs). To circumvent this, Nivre and Nilsson (2005) perform pseudo-projective parsing which involves projectivising the training data by performing lifting operations and encoding information about these lifts in the arc labels. A data-driven parser will train on the projectivised data with these additional labels which include information about lifted arcs. At predict time, the parser will predict a projective parse tree but with labels encoding whether a specific arc was lifted. An inverse transformation can then be applied to reconstruct a non-projective tree. Another way of dealing with non-projective dependencies was proposed by Nivre (2009) who added a further *swap* transition. The swap transition updates a configuration with a stack of $[i, j]$ by moving the node i back to the buffer. In this way, the order of i and j on the stack and buffer is reversed. By swapping the order of the nodes, the result is that non-projective trees can be constructed by applying left-arc or right-arc operations on tokens which were previously not located beside each other. The inclusion of a swap operation increases the time complexity from linear to quadratic but estimates by Nivre (2009) show that running time is linear in the range of the data.

2.1.2 Graph-based Parsing

The other main approach in dependency parsing surrounds the use of graph-based methods. In contrast to incrementally building the dependency tree through a series of actions, graph-based parsers search through a space of all possible trees for a given sentence and choose the tree which maximises some score. Graph-based parsers usually consist of two components: a model which is used to score dependency trees, and a parsing algorithm which is used to find the highest-scoring dependency tree given the

scoring model used.

Background In dependency parsing, a model is tasked with learning a mapping from inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$, where \mathcal{X} is typically a set of sentences and \mathcal{Y} a set of parse trees. Following Collins and Roark (2004), we assume:

- Training examples $\mathcal{T} = \{(x_t, y_t)\}_{t=1}^{\mathcal{T}}$ for $t = 1 \dots \mathcal{T}$.
- A function **GEN** which enumerates a set of candidates $\mathbf{GEN}(x)$ for an input x .
- A global representation Φ mapping each $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to a feature vector $\Phi(x, y) \in \mathbb{R}^d$.
- A parameter vector $\bar{\alpha} \in \mathbb{R}^d$.

The components **GEN**, Φ and $\bar{\alpha}$ are used to define a mapping from an input x to an output $F(x)$, as in (2.1):

$$F(x) = \arg \max_{y \in \mathbf{GEN}(x)} \bar{\alpha} \cdot \Phi(x, y) \quad (2.1)$$

The framework above could be used to describe many tasks in NLP but for dependency parsing (x, y) , **GEN** and Φ are the following:

- A training example (x, y) is a pair where x is a sentence and y is a gold dependency parse for that sentence.
- Given an input sentence x , $\mathbf{GEN}(x)$ is a set of possible parses for that sentence.
- The representation $\Phi(x, y)$ contains the features of the dependency trees.

Arc-factored Parsing The simplest type of model is referred to as a first-order model or an arc-factored model, which is a type of edge-based factorisation (Eisner, 1996) that defines the score of a dependency tree as the sum of the scores of all edges in the tree. We will now look at how individual arcs are scored. Following McDonald et al. (2005a),

the i th word of x is denoted as x_i . The generic dependency tree is denoted with y . If y is a dependency tree for sentence x , the notation $(i, j) \in y$ is used to indicate that there is a directed edge from word x_i to word x_j in the tree, meaning that x_i is the head of x_j . Computing the score of the tree involves summing over its individually-scored parts, as in (2.2):

$$s(x, y) = \sum_{(i,j) \in y} s(i, j) = \sum_{(i,j) \in y} \bar{\alpha} \cdot \phi(i, j) \quad (2.2)$$

The component $\phi(i, j)$ is a high-dimensional binary feature representation of the arc involving head word x_i and dependent word x_j . An example feature from the dependency tree in Figure 1.1 would be as in (2.3):

$$\phi(i, j) = \begin{cases} 1 & \text{if } x_i = \text{'hit'} \text{ and } x_j = \text{'ball'} \\ 0 & \text{otherwise.} \end{cases} \quad (2.3)$$

Learning Algorithm The training process is to learn which of the considered features are important for producing the target output. To do so, a learning algorithm is used to set the parameter values $\bar{\alpha}$ given training examples $(\mathcal{X}, \mathcal{Y})$. One such algorithm that can be used to set the parameter values is the discriminative perceptron learning algorithm (Collins, 2002; Collins and Roark, 2004; McDonald et al., 2005a; Zhang and Clark, 2008). The learning procedure used by the perceptron algorithm is shown in Algorithm 1. T is

Algorithm 1 Perceptron algorithm

Input: Training examples (x, y)
Initialisation: Set $\bar{\alpha} = 0$
Output: Parameters $\bar{\alpha}$
// T training iterations, N training examples
for $t = 1 \dots T$, $i = 1 \dots N$ **do**
 Calculate $y' = \arg \max_{y \in \text{GEN}(x)} \bar{\alpha} \cdot \Phi(x, y')$
 if $y' \neq y$ **then** $\bar{\alpha} = \bar{\alpha} + \Phi(x, y) - \Phi(x, y')$
 end if
end for

a parameter which defines the number of passes over the training set and N is the size of

the training set. Initially, all of the parameter values $\bar{\alpha}$ are set to zero. Under the current set of parameters, we will choose the highest-scoring tree $y' = \arg \max_{y \in \mathbf{GEN}(x)}$. If the predicted tree is different to the gold tree, the parameter update will involve adding 1 to each feature seen in the gold tree and subtracting 1 from each incorrect feature in the predicted tree. Intuitively, the perceptron algorithm increases the parameter values for features which were ‘missing’ from the predicted tree y' and down-weights values for features which were ‘incorrect’. If the predicted tree matches the gold tree, i. e. $y = y'$, there is no update to the parameter values $\bar{\alpha}$. Over the course of training, the parameter values $\bar{\alpha}$ will be updated to score trees as they are in the training data.

Decoding Algorithm Thus far, we have described how individual arcs are scored: by multiplying the feature representations with parameter (or weights) values. Finally, we need a way of retrieving the highest-scoring tree from the candidate set of parses (under the current parameters), i. e. $y' = \arg \max_{y \in \mathbf{GEN}(x)} \bar{\alpha} \cdot \Phi(x, y')$. A number of algorithms can be used for finding the maximum spanning tree (MST) in the graph. The dynamic programming algorithm of Eisner (1996) can be used but this method only produces projective trees. This algorithm has a complexity of $O(n^3)$. In contrast, McDonald et al. (2005b) use the Chu-Liu-Edmonds algorithm (CLE) (Chu and Liu, 1965; Edmonds, 1967) for finding the MST in a directed graph. This first-order algorithm has a complexity of $O(n^2)$. Here, each node (token) in the graph greedily selects the incoming edge (dependency relation) with the highest weight. The result is the maximum spanning tree provided there are no cycles. If there are cycles, the algorithm contracts the cycle into a single vertex and picks the incoming edges with the highest weight based on the contracted nodes.

There are a number of reasons why we would consider using graph-based approaches, particularly when considering that there are 134 languages in UD v2.8. A primary reason is that graph-based approaches are very suitable for producing non-projective trees

which is important for languages with less restrictive word order like Czech or German. Another key reason is that graph-based approaches have traditionally been better at dealing with long-range dependencies than transition-based parsers: transition-based methods have high accuracy on short dependency relations but accuracy tends to decline the greater the distance between the head and the dependent (McDonald and Nivre, 2011). In contrast, graph-based methods score entire trees and do not rely on local greedy decisions and so can be more suitable for dealing with long-range dependency relations. More recent work by Dozat et al. (2017) has shown that their graph-based system performs better relative to a transition-based system using swap (Straka et al., 2016) when a test set contains more non-projective dependencies. They also showed that their graph-based system requires fewer examples of non-projective dependencies in the training set to be able to generalise well. We therefore concentrate on graph-based dependency parsing in this thesis but refer the interested reader to de Lhoneux (2019) for analysis of neural transition-based parsers. There are some similarities between this work and that of de Lhoneux (2019) in that both explore the role of polyglot parsing for parsing typologically-diverse languages (Chapter 6). However, in Chapter 3 we look at the role of a polyglot model in an annotation projection setting, whereas the work of de Lhoneux (2019) investigates various parameter sharing strategies when parsing with multiple languages.

2.2 Neural Networks and their Application to Dependency Parsing

Many NLP tasks have experienced success from adopting methods using neural networks and this is also the case for dependency parsing (Chen and Manning, 2014; Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). Traditional dependency parsers relied on many sparse features where feature templates were man-

ually designed with the aim of capturing head-modifier relationships as much as possible (Zhang et al., 2017). These feature templates were difficult to design and it has also been shown by Bohnet (2010) that the feature-index mapping process (i. e. mapping features to their weights) is very costly in terms of parsing time (inference). The adoption of neural networks in dependency parsing saw the shift from using sparse, binary indicator features to using a “core” set of dense features (Chen and Manning, 2014; Kiperwasser and Goldberg, 2016). Rather than trying to design feature combinations, the neural network is tasked with deciding what information is important for parsing. In the following sections, we will describe the components used in neural graph-based dependency parsers.

2.2.1 Distributed Representations of Words

As mentioned above, feature templates have traditionally been used for training parsers (Carreras, 2007; Zhang and Clark, 2008; Koo and Collins, 2010; Hatori et al., 2012). Among these features include lexical items and their POS tags. While a rich feature set incorporating non-local contexts can improve parser performance, the features can be sparse (Chen et al., 2014). Using such feature templates also has the drawback of a high out-of-vocabulary rate for certain features, where features are present in the training set but not the test set. Chen and Manning (2014) made the first advance in modern neural dependency parsing by using a feed-forward neural network (FFNN) to learn features for a transition-based parser. This architecture saw the adoption of dense rather than sparse feature representations, where the features consist of vector representations of a certain number of words (and their dependents), POS tags, and dependency labels. Using dense embeddings to represent features means that the model can leverage the fact that certain features are similar to each other, e. g. based on cosine similarity, where vectors that point in a similar direction can be interpreted as being similar (Mikolov et al., 2013; Kondratyuk and Straka, 2019). For example, certain words are semantically

similar, e. g. nouns such as John or Jane, and such words also have similar syntactic roles (both are often subjects which modify verbs, e.g. [*Jane/John*] *went to the store*).

As described in Bastings (2020), in neural NLP systems, typically a vocabulary $|V|$ is created and each word in the vocabulary will be represented by a one-hot vector. This vector will be the size of the vocabulary and has a 0 in every position except for the position of the word ID in question which will be a 1. This one-hot vector is then multiplied with a word embedding matrix $\mathbf{W}_e \in \mathbb{R}^{d \times |V|}$, with dimension size d rows and vocabulary size $|V|$ columns. This multiplication selects the appropriate column from the word embedding matrix.² The numbers in the vectors can be initialised randomly, with uniform distribution (Glorot and Bengio, 2010) or they can be from a pre-trained embedding (Pennington et al., 2014). Formally, let $S = (w_0, w_1, \dots, w_N)$ be a sentence of length N with w_0 being the artificial root token and the i -th value of S is written as w_i . This embedding lookup procedure is performed for each input token in S to obtain a sequence of embeddings: $X = (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N)$ where \mathbf{x}_i represents the embedding of word w_i . In practice, \mathbf{x}_i may be the concatenation of numerous embeddings, e.g. for a token w_i , we can include a randomly initialised embedding $e(w_i)$, its POS tag embedding $e(t_i)$ and a pre-trained embedding of the word $e(pt_i)$ as shown in (2.4), where \circ denotes vector concatenation:

$$x_i = [e(w_i) \circ e(t_i) \circ e(pt_i)] \quad (2.4)$$

2.2.2 Recurrent Neural Networks

Neural dependency parsers typically include a recurrent neural network (RNN) which encode embeddings of words. RNNs are chosen because they can handle sequential data (such as sentences in natural language) as input. This is useful for tasks which involve recurrence such as in dependency parsing, where the order of words in a sentence can play an important role in determining syntax. Importantly, RNNs are able to learn

²The process is the same for any feature, where the embedding matrix is determined by the dimension size and size of the vocabulary.

context-sensitive representations of the words in a sentence by incorporating information from nearby words. This is because at each time step the RNN updates its hidden state based on all inputs preceding the current input as in (2.5):

$$\text{RNN}(\mathbf{x}_{1:t}) = f(\mathbf{x}_t, \text{RNN}(\mathbf{x}_{1:t-1})) \quad (2.5)$$

One type of RNN that has seen widespread use in NLP is the Long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997), which has been shown to be able to better handle longer-range information than a vanilla RNN. This is because they have a more complex computational unit which is able to control the flow of information across a sequence which can help address the problem of vanishing gradients associated with traditional RNNs. Bidirectional LSTMs (BiLSTMs) (Graves and Schmidhuber, 2005) have been used as the function f in (2.5). In contrast to traditional LSTMs, BiLSTMs enable modelling sequential data in both directions. For this to take place, there is a forward LSTM (LSTM_F) and a backward LSTM (LSTM_B). The forward LSTM moves in a left-to-right direction and enables capturing w_i and the words to its left. The backward LSTM moves from right-to-left and captures w_i and the words to its right. This is shown in (2.6) and (2.7) for the forward and backward LSTMs, respectively. We use x_i to denote the vector representation of word w_i :

$$h_i^f = \text{LSTM}_F(\mathbf{x}_{0:i}) \quad (2.6)$$

$$h_i^b = \text{LSTM}_B(\mathbf{x}_{N:i}) \quad (2.7)$$

At each time step, we obtain a hidden representation h_i which is the concatenation of the hidden states of the forward LSTM $h_i^f \in \mathbb{R}^d$ and backward LSTM $h_i^b \in \mathbb{R}^d$ at position

i , where $\in \mathbb{R}^d$ is the LSTM hidden state size, as in (2.8):

$$h_i = [h_i^f \circ h_i^b] \quad (2.8)$$

As a result, each time step has information about the surrounding context. When considering dependency parsing, the BiLSTM encodings h_i are used as part of a feature function ϕ in either a transition-based or graph-based parser. Initially, the encodings do not contain any information about the task of parsing: Throughout the training process, they are fed into a non-linear scoring function such as a multi-layer perceptron (MLP) which is used to make a prediction and return a loss value. The weights of the whole network can be updated together using back-propagation as in Kiperwasser and Goldberg (2016) to try and minimise the loss function. In this way, the weights of the neural network are updated so that they reflect the task at hand.

2.2.3 Transformer Networks and Self-Attention

The introduction of the Transformer (Vaswani et al., 2017) fundamentally changed the field of NLP with most recent state-of-the-art applications using this architecture as a component to produce contextualised representations which are then passed to subsequent task-dependent modules. This is the case for applications such as sentiment classification, question answering as well as structured prediction tasks such as Named Entity Recognition (NER) and dependency parsing.

As outlined by Vaswani et al. (2017), prior to the Transformer, most NLP applications used RNN encoders to generate contextualised representations (see Section 2.2.2). One drawback of RNNs is their dependence on sequential computation, i. e. computing hidden representations requires passing through all input positions in a sequential fashion, either in their original order or in reversed order. In the Transformer, however, layer outputs can be computed in parallel. Additionally, as inputs to the attention layers in

a Transformer are unordered vectors, this enables all items in the input to interact with other items directly, which is better for handling long-range dependencies as the two items in question are no longer separated through a number of RNN steps. This property of Transformers enables them to model complex relationships between the input tokens, which is useful for many NLP tasks. This point is well articulated by Henderson (2020, p.9) who states that *“Attention-based models are fundamentally different because they use bag-of-vector representations. With BoV representations, attention-based neural network models like Transformer can model the kinds of unbounded structured representations that computational linguists have found to be necessary to capture the generalisations in natural language”*. The Transformer is able to achieve this by using a process called *“self-attention”* to learn dependencies between inputs and this requires a constant number of operations. Self-attention is a type of attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) that relates different positions of a single sequence in order to compute representations for the sequence. Intuitively, self-attention provides a score for how useful a current word w_i is to another word w_j based on some arbitrary task, e. g. language modelling.

The particular attention function used by the Transformer is called *“scaled dot product attention”*. This involves mapping each word in the sequence to a vector representation of that word as a *query*, a *key* and a *value* (q , k and v , respectively) which are all of size d_k . In practice, this is done at the batch level, where the Query, Key and Value matrices, denoted as \mathbf{Q} , \mathbf{K} and \mathbf{V} , respectively, are obtained by multiplying the inputs \mathbf{X} by various learned matrices: $\mathbf{Q} = \mathbf{W}^Q(\mathbf{X})$; $\mathbf{K} = \mathbf{W}^K(\mathbf{X})$; $\mathbf{V} = \mathbf{W}^V(\mathbf{X})$. Given these matrices, the attention weights can be computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V} \quad (2.9)$$

In the Attention function (2.9), a dot product is taken between the Query matrix with

the transpose of the Key matrix to provide the weights. A weight w_{ij} represents the inner product between the i -th query and the j -th key. The output of this multiplication is scaled by a factor of the square root of the model dimension d_k .³ A softmax function is then applied to the weights to normalise them, such that their values are positive and sum to 1. The output is obtained by multiplying the softmax scores by the values \mathbf{V} . The intuition behind multiplying the scores and the values is to preserve tokens which are considered important and to remove unimportant words by multiplying them with large and small numbers, respectively. The attention weights computed above, denote how important a particular query is for a particular key. High weight values indicate that the two words hold an important relationship given a specific task, and low weight values signify that they are not particularly relevant for the task.

Up to this point, we have only described “single-head” scaled dot-product attention. However, in the Transformer, this attention function is repeated h times, where h is the number of attention “heads”. Computing multiple instances of scaled dot-product attention is referred to as “Multi-Head Attention”. In Multi-Head Attention, the q , k and v items are projected h times and different single-head attention modules are applied to the representations in parallel. The individual attention outputs are concatenated and projected to the desired output dimension using a feedforward network. The intuition behind using multiple attention heads is that doing so can counteract overfitting by enabling different attention heads to learn particular phenomena about the input sequence being modelled. The process for computing Multi-Head Attention is shown in (2.10):

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{Head}_1, \dots, \text{Head}_h) \mathbf{W}_0 \\ \text{where Head}_i &= \text{Attention}(\mathbf{Q} \mathbf{W}_i^Q, \mathbf{K} \mathbf{W}_i^K, \mathbf{V} \mathbf{W}_i^V) \end{aligned} \quad (2.10)$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ represent the weight matrices of the i -th attention head. Single-

³This is because with large values of d_k , the dot products have a large magnitude which can cause the softmax function to yield very small gradients. If \mathbf{Q} and \mathbf{K} have a mean of 0 and a variance of 1, their matrix multiplication will have a mean of 0 and a variance of d_k . Scaling the multiplication by the square root of d_k means that the variance will be consistent even when changing the value of d_k .

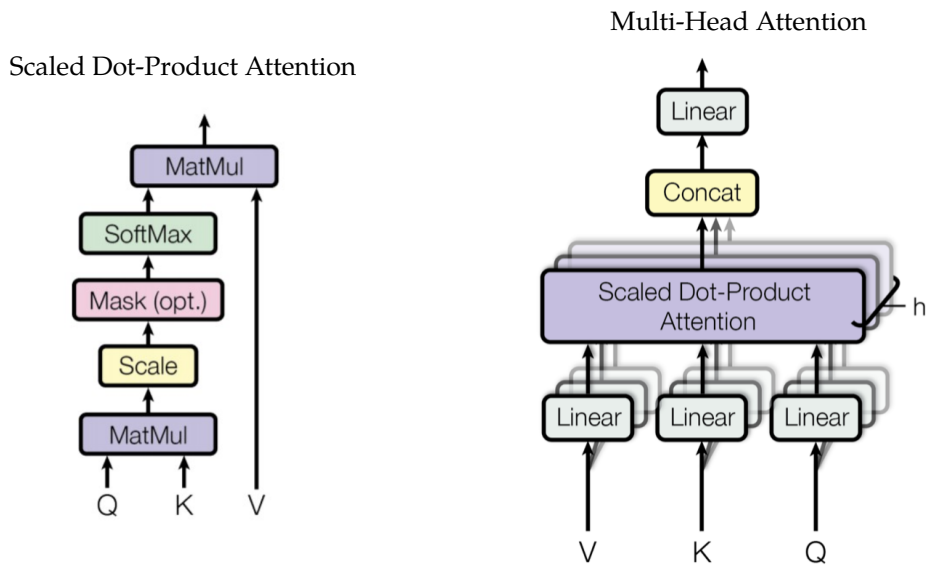


Figure 2.4: Scaled Dot-Product Attention (left). This process is repeated h times in Multi-Head Attention (right), where the outputs are concatenated and then projected with a feedforward network. Figures taken from Vaswani et al. (2017).

Head attention and Multi-Head attention can be visualised as in Figure 2.4. In Multi-Head Attention, the queries, keys and values are projected h times where a different Scaled Dot-Product Attention module is applied to each projection.

Transformer Architecture

Now that we have described the attention mechanism used by the Transformer to relate items in the input sequence, we can widen our focus and explain how it is used in the Transformer architecture. In dependency parsing, as we are not generating output sequences as in Vaswani et al. (2017), who use an encoder-decoder architecture to perform machine translation, we will just describe the encoder part of the Transformer, which generates representations for each token in the sequence. We will also provide more details on Transformer models such as BERT (Devlin et al., 2019) in Section 4.1.

Encoder Input For the first component, the encoder, it takes as input an embedded version of the input, where a word is converted to a numeric ID and then its ID is looked up in an embedding matrix. Embeddings represent an input token in a d -dimensional space, where tokens with similar meanings will be closer in the geometric space. The Transformer solely uses self-attention to relate input words to one another. As such, there is no notion of recurrence and the attention layers will see their input as a set of unordered vectors or, in other terms, a bag-of-vectors. In order to introduce sequential order, a “positional encoding” is added to the input embedding to provide the model with information about the relative position of the tokens in the input sequence. The addition of the position encoding means that tokens will be closer to each other based on similarity as well as as their position in the sentence. Learned position embeddings can also be added to the Transformer but a position encoding based on sin and cos functions was chosen by Vaswani et al. (2017) to allow the model to learn sequence information for sequences larger than what the model was trained on. The positional encoding PE is generated with the functions in (2.11) and (2.12), where a sin function is applied to even positions and a cos function to odd positions:

$$PE(pos, 2i) = \sin(pos/10000^{2i/d_{model}}) \quad (2.11)$$

$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{model}}) \quad (2.12)$$

Encoder Now that we have an input to the encoder—an embedding with an added positional encoding—this representation is then passed to the encoder component. The encoder consists of a stack of N identical layers. Each layer contains two sub-layers or blocks: the first block is a Multi-Head Attention block and the second block is a position-wise fully connected feedforward network block. A residual connection is added to each block followed by a layer normalisation, e. g. the output of each block is $\text{LayerNorm}(x + \text{Block}(x))$. Residual connections are used to avoid problems associ-

ated with vanishing gradients in neural networks. Figure 2.5 shows the encoder block used in the Transformer model.

Similar to using recurrent neural networks for feature extraction, Transformer networks can also be used to obtain a sequence of continuous representations $z = (z_1, \dots, z_n)$ given an input sequence $x = (x_1, \dots, x_n)$, where these representations can be used for a downstream task such as dependency parsing. Currently, Transformers are the model of choice for inducing representations for many NLP tasks, eclipsing previous models such as RNNs in many cases.

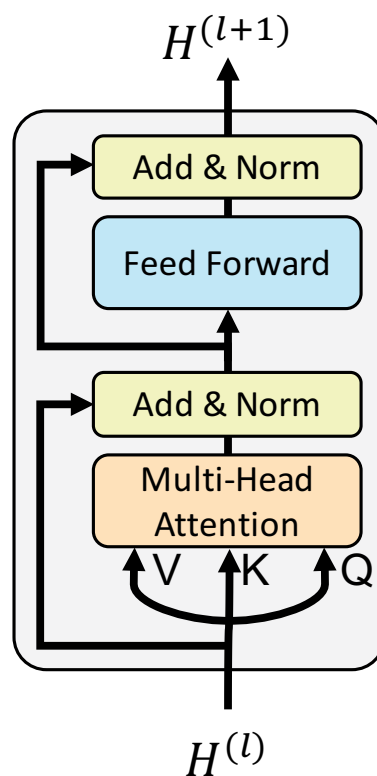


Figure 2.5: The Transformer encoder block of Vaswani et al. (2017). Image taken from Lu et al. (2019).

2.2.4 Neural Graph-based Parsing

Now that we have introduced the task of dependency parsing, and described how to obtain representations of words for a sentence—by embedding them and inputting them to an encoder such as an RNN or a Transformer—we can now describe how these components are combined to perform the task of dependency parsing, where we will focus on recent graph-based neural dependency parsers.

Current state-of-the-art neural dependency parsers such as Kiperwasser and Goldberg (2016) and Dozat and Manning (2017) employ first-order factorisation (Eisner, 1996; McDonald et al., 2005a) as described in Section 2.1.2. Kiperwasser and Goldberg (2016) train a graph-based parser alongside a BiLSTM feature extractor. In their model, given an input sentence $S = (w_0, w_1, \dots, w_N)$ which has been mapped to some vectors $x_{0:N}$, they search for the highest-scoring parse tree Y among the space of valid trees $\mathcal{Y}(S)$ for sentence S . The feature function ϕ they use in their graph-based parser is very simple. Specifically, when processing a sentence, for each token, they concatenate the hidden representations of the token and its potential head (2.13), where i is a head word and j a dependent:

$$\phi(i, j) = h_i \circ h_j \quad (2.13)$$

These representations are passed to an MLP classifier which scores the individual arcs. The scores are then summed to produce the final score for the tree. The overall scoring process used in Kiperwasser and Goldberg (2016) is shown in (2.14):

$$\begin{aligned} \text{parse}(S) &= \arg \max_{Y \in \mathcal{Y}(S)} \text{score}_{\text{global}}(S, Y) \\ &= \arg \max_{Y \in \mathcal{Y}(S)} \sum_{(i,j) \in Y} \text{score}(\phi(i, j)) \\ &= \arg \max_{Y \in \mathcal{Y}(S)} \sum_{(i,j) \in Y} \text{MLP}(h_i \circ h_j) \end{aligned} \quad (2.14)$$

For each token, the parsing algorithm considers all possible edges with the result

that there are n^2 predicted edges and the final tree is decoded using Eisner’s algorithm (Eisner, 1996). Their parser first predicts arcs and afterwards, the label for the chosen arc. The label classifier uses the same feature function $\phi(i, j)$ but these representations are fed into a separate MLP (MLP_L):

$$label(i, j) = \arg \max_{l \in labels} MLP_L(h_i, h_j)[l] \quad (2.15)$$

The graph-based parser of Kiperwasser and Goldberg (2016) considers global tree structure as part of training and inference: during training, the parser uses a margin-based objective which attempts to maximise the margin between the score of the gold parse tree and the highest-scoring incorrect tree. In contrast to this globally-structured dependency parsing objective, Zhang et al. (2017) treat dependency parsing as a head-selection process. Here Zhang et al. (2017) disregard any notion of whether the predicted output should be a dependency tree. They simply train a model which produces for each word a probability distribution of the other words in the sentence being the current word’s head and then choose the highest-scoring result. Each arc is considered independently. Their reasoning behind not using any tree constraints is that the parser should produce trees which are representative of the training set which consists of trees. This assumption is validated where their parser predicts outputs which are trees in around 95% of cases. On outputs that are not trees, a maximum spanning tree algorithm (Section 2.1.2) can be used to ensure well-formedness. This approach is motivated because an $O(n \log n)$ algorithm can be run to check if the tree is projective/well-formed and then a more expensive algorithm can be run on only those outputs which do not meet this criteria.

A more sophisticated neural arc-factored parser was introduced by Dozat and Manning (2017), which differs from previous works such as Kiperwasser and Goldberg (2016) and Zhang et al. (2017) by firstly replacing the traditional MLP-based attention for edge prediction (e.g. passing two vectors through a non-linearity) with a biaffine

attention function. Secondly, they reduce the size of the input representations passed to the biaffine parsing module by first passing the output of the recurrent neural network to feedforward networks. This step makes training both modules faster and less prone to overfitting. Additionally, Dozat and Manning (2017) introduce more regularisation and a larger network with carefully selected hyperparameters. This architecture was used by the winning system of the 2017 CoNLL shared task on UD parsing (Dozat et al., 2017; Zeman et al., 2017) and inspired many systems to use this architecture in the 2018 version of the CoNLL shared task (Zeman et al., 2018) including the winning system (Che et al., 2018), and still reaches state-of-the-art accuracy to this day.

Formally, Dozat and Manning (2017) use a BiLSTM to encode vectors of words and POS tags $X_{[0:N]}$ for a sentence of length N (2.16), producing hidden representations for the sequence $R_{[0:N]}$. In the following, we use the notation r_i to denote a potential head word and r_j to denote a potential dependent word. The outputs of the BiLSTM are passed to MLPs to produce the *arc-dep* and *arc-head* representations (2.17 - 2.18), which allows for learning a specific representation for each word as a head and a dependent. The biaffine classifier is given in (2.19), where first *bilinear attention* is computed between the matrix of head representations $H^{(arc-head)}$ and the dependent representation $h_i^{(arc-dep)}$ using a $(d \times d)$ matrix $U^{(arc)}$. Bilinear attention is used to relate the strength of each potential head-dependent pair by producing a set of scores indicating how likely a current word is of being a dependent of another head word. These outputs are added to another vector which is produced by multiplying the matrix of head representations $H^{(arc-head)}$ with a bias vector \mathbf{b} . Intuitively, this second term produces a vector which signifies how likely a word is to be a head in the first place.

$$R = \text{BiLSTM}(x_0, x_1, \dots, x_N) \quad (2.16)$$

$$h_i^{(arc-dep)} = \text{MLP}^{(arc-dep)}(r_i) \quad (2.17)$$

$$h_j^{(arc-head)} = \text{MLP}^{(arc-head)}(r_j) \quad (2.18)$$

$$s_i^{(arc)} = H^{(arc-head)} U^{(arc)} h_i^{(arc-dep)} + H^{(arc-head)} \mathbf{b} \quad (2.19)$$

The scores for a particular token $s_i^{(arc)}$ can be passed through a softmax function to produce probabilities and the final tree can be decoded using the CLE algorithm (Chu and Liu, 1965; Edmonds, 1967). A similar biaffine classifier is used to score the labels for the chosen arcs. An overview of the biaffine dependency parser is shown in Figure 2.6, where the image shows an example with two input tokens.

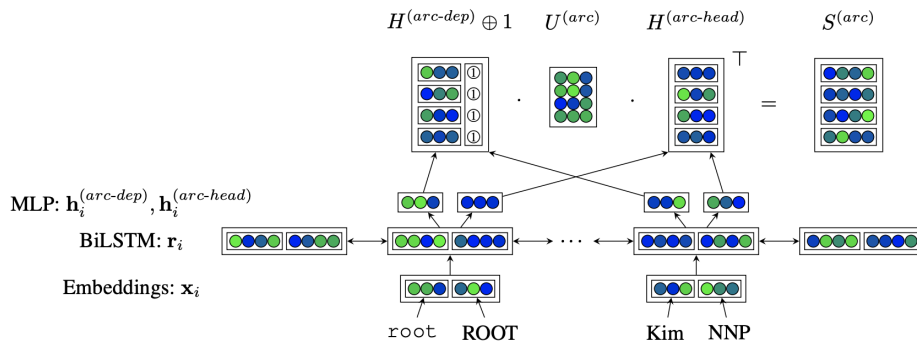


Figure 2.6: Biaffine graph-based parser with a BiLSTM encoder operating over word embeddings. Figure taken from Dozat and Manning (2017).

2.3 Multilingual Dependency Parsing

The dependency parsing community has been oriented towards multilingual goals for well over a decade now. For instance, the CoNLL 2007 shared task (Nivre et al., 2007a) required participants to develop a parsing system which could parse ten different languages. More recently, participants were required to parse 49 and 57 languages in the 2017 and 2018 CoNLL shared tasks on UD parsing, respectively (Zeman et al., 2017, 2018). Parsing multiple languages brings with it many challenges such as differing annotation styles across the languages, and the languages possessing different linguistic

characteristics. These challenges will be discussed in the following sections.

2.3.1 Differences in Annotation Styles

As pointed out by de Lhoneux (2019), multilingual parsing can refer to cases where the same architecture is used to parse multiple languages individually and de Lhoneux introduces the term “polymonolingual” to refer to such cases. Most early works involving multilingual dependency parsing fall into this category. For example, Hall et al. (2007) tune the parameters of the parsing algorithm, the feature model and the learning algorithm of MaltParser (Nivre et al., 2007b) for each of the ten languages separately, which leads to improvements over their general settings.

The situation where parsers have typically been trained on individual languages can be explained by the fact that annotation schemes differed across treebanks from different languages, which hindered the ability for cross-lingual transfer. In early work on multilingual parsing, McDonald et al. (2011) include an experiment on cross-lingual parsing using direct transfer, where the parser is trained only on POS information in a source language which is then ported to a target language. They incorporate data from the following languages: Danish, Dutch, German, Greek, Italian, Portuguese, Spanish, Swedish and English. To avoid issues with different treebank annotation schemes, they focus on unlabelled parsing. The direct transfer experiments of McDonald et al. produce some surprising results: they found that Danish is the worst source for transfer to Swedish while Portuguese is the best. They also found that Greek was the best source for English and Dutch. Some Romance languages transfer well to each other, however, such as Italian and Spanish.

The findings of McDonald et al. highlighted the lack of consistent annotations across treebanks from different languages, and such findings, along with other multilingual initiatives, helped orient the field towards creating truly universal representations in UD (Nivre et al., 2016). UD has alleviated some of these problems by requiring all treebanks

to be annotated in a cross-linguistically consistent fashion. This has paved the way for developing multilingual systems that can be successfully trained on multiple treebanks (Vilares et al., 2016; Smith et al., 2018; Kondratyuk and Straka, 2019; van der Goot et al., 2021; van der Goot and de Lhoneux, 2021).

2.3.2 Morphologically-Rich Languages

One linguistic property that presents a challenge to dependency parsers is agglutination, where certain linguistic phenomena can be expressed with inflection. UD includes languages such as Finnish and Farsi which are described as being morphologically-rich languages (MRLs) (Tsarfaty et al., 2010). In neural parsers, modelling lexical features at the character level has been proposed as a viable strategy for parsing with MRLs (Ballesteros et al., 2015). The intuition behind using character-level models is that orthographically similar words share many parameters. In this way, parsing models produce effective representations for out-of-domain words which are morphological variants of words seen during training (Vania et al., 2018). The top-three performing systems in the 2017 CoNLL shared task all use character-level information (Björkelund et al., 2017; Dozat et al., 2017; Shi et al., 2017) indicating that such information is a necessary ingredient for viable multilingual dependency parsing.

While previous work has identified the importance of character-level information, Vania et al. (2018) ask the question whether character-level information alone is sufficient to learn morphology. In their experiments, they compare character-level models with an oracle approach that has access to morphological features for the task of dependency parsing. Their results show that the character-level models seldom outperform the oracle model which has access to explicit morphological information. Vania et al. show that the character models have difficulty with case syncretism where noun case is ambiguous. They found that a character-level model provided only with morphological case information rivals the oracle model even when morphological case is automatically

predicted.

2.4 Summary

In this chapter, we have described the origins of dependency grammar and provided an overview of dependency syntax by listing some of the formal properties of dependency trees. We then described the two main approaches to dependency parsing: transition-based and graph-based approaches, respectively. We noted that transition-based approaches choose from a predefined set of possible actions to incrementally build the parse tree, whereas graph-based approaches factorise the output structure into parts and score all of these parts individually. The final tree can then be selected with global inference algorithms. We first described earlier “classic” approaches to transition-based and graph-based parsing, and for the graph-based parser, we showed how example features of dependency trees are scored using a learning algorithm such as the discriminative perceptron, which is used to set to weights of the parsing model to score trees as they are in the training set. We then described the shift from these classic approaches to using neural networks for parsing, where the hand-designed feature templates are replaced with a core set of dense features through the use of word embeddings. These embeddings are passed through a neural network architecture to make a prediction and return a loss value, where the whole network is then updated in an end-to-end manner to learn the task of parsing. We provided an overview of two popular components which are used as feature extractors for the dependency parsing module: RNNs and the Transformer. Finally, we introduced the topic of multilingual dependency parsing and mentioned how, prior to UD, differences in annotation schemes were a significant barrier to developing parsers which can parse multiple languages. We then mentioned that UD attempts to address this challenge by incorporating a universal dependency annotation scheme.

In the next chapter we will present an experiment which seeks to leverage the uni-

versal annotation scheme in UD, by analysing the role of a polyglot model trained on all source languages together versus a number of monolingual source models in contributing annotations to a target language using annotation projection.

Chapter 3

Polyglot Training for Cross-lingual Transfer

In the previous chapters we noted that certain treebanks in the UD collection have no training data, and various cross-lingual approaches must be used to process them. We also highlighted that the cross-linguistically consistent annotations in UD makes training parsers on multiple languages (i. e. polyglot parsers) more feasible. In this chapter we wish to see whether low-resource dependency parsing can be improved via a polyglot modelling approach. Specifically, we approach low-resource dependency parsing by integrating two common techniques used for low-resource NLP tasks: (i) a data transfer technique known as annotation projection, which involves transferring annotations from a source to a target language (Yarowsky et al., 2001), and (ii) a polyglot modelling approach, which involves training a model with shared parameters across multiple languages, where languages have equal status (de Lhoneux, 2019; Mulcaire et al., 2019a). Such models have been shown to be helpful for low-resource languages in particular (Smith et al., 2018; Kondratyuk and Straka, 2019; van der Goot and de Lhoneux, 2021). Previous work by Tyers et al. (2018) has shown that regular annotation projection can be improved by combining the projected annotations from multiple source

languages. Here, the source models are trained on individual languages and the cross-lingual interaction occurs when they are merged together in a graph decoding step (Sagae and Lavie, 2006). We adopt the same experimental setup as Tyers et al. (2018) but examine whether it is also beneficial to induce cross-lingual knowledge before the graph-decoding step, by training a polyglot model on the source languages. The intuition is that by training a model on multiple languages together, the model will learn to make generalisations across the included languages, which will result in a better model for transfer to the low-resource target language.

In Section 3.1, we first describe related work on zero-shot cross-lingual transfer as well as polyglot learning. In Section 3.2, we then present a combination of these two methods in a zero-shot cross-lingual dependency parsing experiment and is used to answer *RQ1: Can zero-shot cross-lingual dependency parsing be improved by leveraging polyglot training on source languages?*

3.1 Background

3.1.1 Cross-lingual Transfer

In low-resource scenarios, there may not be enough data for data-driven models to learn the task at hand. In cases where no annotated data is available, zero shot cross-lingual transfer is used. This refers to techniques which develop a model for a target language which has no training data, by making use of source-language annotations (Yarmohammadi et al., 2021). This has typically been approached through data projection, which involves transferring annotations across parallel text, where the parallel text can be created by translating the source language to the target language or vice versa. Then, a model trained on the source language can be used to annotate the data. The annotations can then be transferred back to the target language using word alignments. In other approaches, the target language can be translated to the source language at test time, and

a model trained in the source language can be used to annotate the data (Yarowsky and Ngai, 2001; Tiedemann and Agić, 2016). Another way to induce cross-lingual knowledge is through model transfer, where a model trained on a downstream task for one language is applied to a target language which does not have any training data (Kondratyuk and Straka, 2019; Wu and Dredze, 2019). This chapter will focus on the first approach, i. e. using data transfer techniques to process a low-resource language.

Cross-lingual transfer methods—methods that transfer knowledge from one or more source languages to a target language—have led to substantial improvements for low-resource dependency parsing (Hwa et al., 2005; McDonald et al., 2011; Lynn et al., 2014; Guo et al., 2015; Agić et al., 2016; Rosa and Mareček, 2018) as well as for related tasks such as part-of-speech (POS) tagging (Plank and Agić, 2018). One of the most straightforward approaches for cross-lingual transfer involves direct transfer. Zeman and Resnik (2008) introduced the idea of delexicalised dependency parsing, where a parser can be trained on coarse-grained information such as POS tags in a source language and then be directly applied to a target language provided it has been annotated with POS tags. This relies on the source and target languages sharing some underlying grammatical structure as well as having a compatible set of POS tags but obviates the need to have labelled training data in the target language. McDonald et al. (2011) perform delexicalised dependency parsing using direct transfer and show that this approach outperforms unsupervised approaches for grammar induction. Importantly, this approach can be extended to the multi-source case by training on multiple source languages and predicting a target language.

The idea of annotation projection using word-alignments originates from Yarowsky et al. (2001) who use word alignments to transfer information such as POS tags from source to target languages. This method was later used in dependency parsing by Hwa et al. (2005), who project dependencies to a target language and use a set of heuristics to form dependency trees in the target language. A parser is then trained on the projected

treebank and evaluated against gold-standard treebanks.

Tiedemann and Agić (2016) present a thorough comparison of pre-neural cross-lingual parsing. Various forms of annotation projection methods are compared to delexicalised baselines, and the use of machine translation instead of parallel corpora to produce synthetic treebanks in the target language is explored. In contrast to works which create silver data in the target language that is used to train a target-side model (e. g. Tyers et al. (2018)), Tiedemann and Agić (2016) translate a target sentence and project the source parse tree back to the target during test time.

Agić et al. (2016) leverage massively multilingual parallel corpora such as translations of the Bible and web-scraped data from the Watchtower Online Library website¹ for low-resource POS tagging and dependency parsing using annotation projection. They project weight matrices (as opposed to decoded dependency arcs) from multiple source languages and average the matrices weighted by word alignment confidences. They then decode the weight matrices into dependency trees on the target side, which are then used to train a parser. Agić et al. claim that by utilising dense information from multiple source languages, this helps reduce noise from source-side predictions. Here, the source-side parsing models learn information between source languages independently (i. e. it does not involve polyglot training) and the cross-lingual interaction occurs when projecting the edge scores into multi-source weight matrices. The idea of projecting dense information in the form of a weighted graph has been further extended by Schlichtkrull and Søgaard (2017) who bypass the need to train the target parser on decoded trees and develop a parser which can be trained directly on weighted graphs.

Plank and Agić (2018) use annotation projection for POS tagging. They find that choosing high-quality training instances results in superior accuracy than randomly sampling a larger training set. To this end, they rank the target sentences by the percentage of words covered by word alignments across all source languages and choose

¹<https://wol.jw.org/>

the top- k covered sentences for training.

Meechan-Maddon and Nivre (2019) carry out an evaluation on cross-lingual parsing using language clusters (Scandinavian, West Slavic and Uralic) where each cluster has a low-resource language that is supported by three related source languages. The Scandinavian cluster uses Faroese as the low-resource language supported by Danish, Norwegian Nynorsk and Swedish. The West Slavic cluster includes Upper Sorbian as the target language supported by Czech, Polish and Slovak. The Uralic cluster uses North Saami as the target language supported by Estonian, Finnish and Hungarian. They include three experiments: first, training a *monolingual* model on a small number of sentences in the target language; second, training a *cross-lingual* model on related source languages which is then applied to the target data; and lastly, training a *multilingual* model which includes target data as well as data from the related support languages. They found that training a monolingual model on the target data was always superior to training a cross-lingual model. They found that the best results were achieved by training a model on the various support languages with the included target data, i. e. their multilingual model.

Tyers et al. (2018) describe a method for creating synthetic treebanks for Faroese based on previous work which uses machine translation and word alignments to transfer trees from source language(s) to the target language. Sentences from Faroese are translated into the four source languages: Danish, Swedish, Norwegian Nynorsk and Norwegian Bokmål. The translated sentences are then tokenised, POS tagged and parsed using the relevant source-language model trained on the source-language treebank. The resulting trees are projected back to the Faroese sentences using word alignments. In a subsequent step, the four trees for each sentence are combined into a graph with edge scores one to four (the number of trees that support them), from which a single tree per sentence is produced using the CLE algorithm. The resulting trees make up a synthetic treebank for Faroese which is then used to train a Faroese parsing model. The parser

output is evaluated using the gold-standard Faroese test treebank developed by Tyers et al. (2018). The approach is compared to a delexicalised baseline, which it outperforms by a large margin. It is also shown that for Faroese, a combination of the four source languages (multi-source projection) is superior to individual language projection.

The process of annotation projection used by Tyers et al. (2018) is shown in Figure 3.1, where the source-language sentences are parsed and then projected to the target language (Faroese). In this example, there is a direct alignment between words in the source sentences and the target sentence so annotations can be directly transferred. The trees predicted from each source model are identical apart from Swedish, which predicts an *obj* relation between the words *bor* and *nu*. A parser can either be trained directly on the target language trees, or the trees for a particular sentence from each target treebank can be combined and decoded using global inference algorithms for tree parsing, e.g. the CLE algorithm. This process is referred to as multi-source projection and is shown in Figure 3.2.

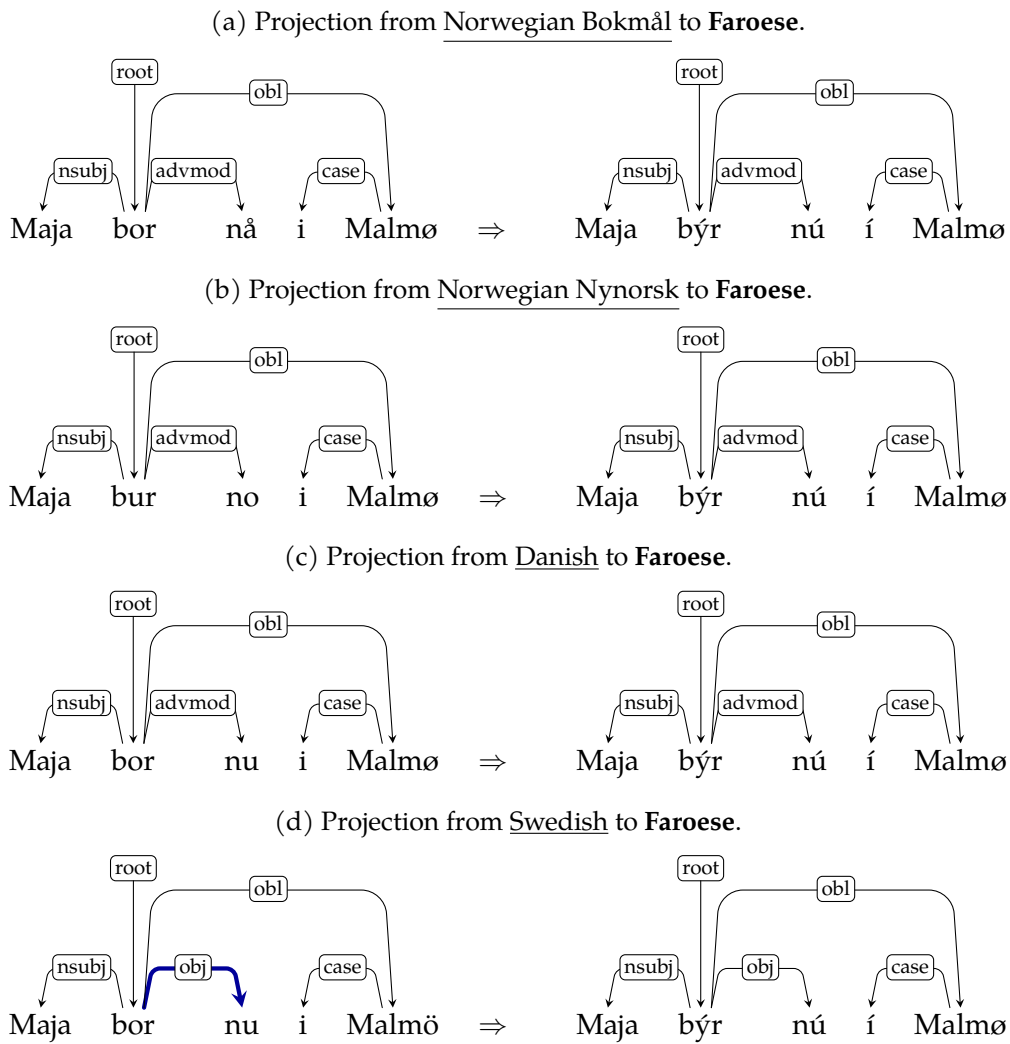


Figure 3.1: Annotation projection from the source languages to the **target** language. Example translations and annotations taken from Tyers et al. (2018).

3.1.2 Parameter Sharing between Languages

With the advent of the UD project and the harmonisation of annotation schemes across numerous languages, some works have tried to determine whether information can be successfully shared between different languages. Among these approaches are those that involve polyglot learning. We adopt the same terminology used in Mulcaire et al.

Multisource Projection.

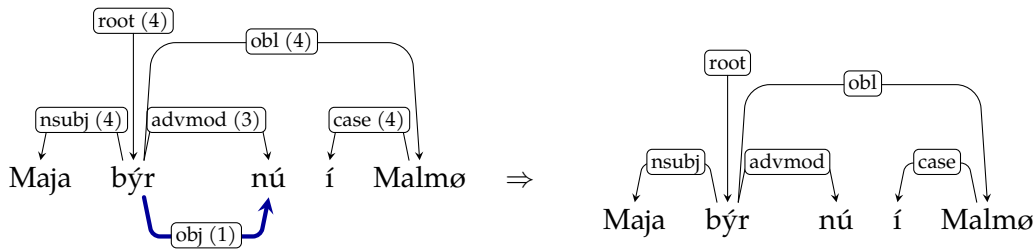


Figure 3.2: Left: Dependency edges and labels are combined in a single graph and resolved using the CLE algorithm (Chu and Liu, 1965; Edmonds, 1967). The counts of each labelled edge are provided in brackets. Right: The decoded dependency tree for the sentence. Example taken from Tyers et al. (2018).

(2019b), who use the term *cross-lingual transfer* to describe methods involving the use of one or more source languages to process a target language. They reserve the term *polyglot learning* for training a single model on multiple languages and where parameters are shared between languages. For the polyglot learning technique applied to multiple treebanks of a single language, we use the term *multi-treebank learning*.

Vilares et al. (2016) train lexicalised, bilingual parsers on the union of treebanks between multiple pairs of languages. They carried out their experiments using either coarse-grained POS tags or language-specific POS tags (UPOS and XPOS in the CoNLL-U format, respectively). Interestingly, they found the version with language-specific tags to work better. They hypothesise that this is because they are combining universal information with language-specific information which has the effect of enabling the parser to learn language-specific constructions and avoids learning spurious similarities between the languages. This finding shows that despite UD making it easier to train parsers on multiple languages, being able to retain idiosyncratic or language-specific information is still important for successful polyglot parsing. Their results show that training bilingual parsers rarely harms performance, e. g. for the majority of cases, there is no statistically significant difference in LAS when compared to the monolingual parser. They also found that for some language pairs, bilingual training can give statistically

significant LAS improvements over the monolingual baseline. In an additional experiment, they show that their approach is vastly superior to monolingual training when parsing code-switched data.

Sato et al. (2017) perform a multilingual dependency parsing experiment using treebanks from French and English. They include two BiLSTM feature extractors in their parsing architecture: one which operates over all treebanks, and another which operates on individual treebanks. The outputs of the BiLSTMs are combined via a gating mechanism which decides what information should be passed through to a dependency parsing component that uses biaffine attention. They also include a domain classification task that tries to classify the domain of the input. The simultaneous training of the shared BiLSTM layers alongside the domain classification task acts as an adversarial mechanism, encouraging the model to find shared parameters and reducing the reliance on features for a particular domain. They found that multilingual training helped for the small French treebanks though a monolingual baseline performed best for the largest treebank. For English, they found that multilingual training actually harmed performance. Similar results were found by Che et al. (2018) who found that concatenating treebanks from related languages harmed monolingual performance.

A method which differs slightly from plain treebank concatenation involves adding a dataset embedding² to the input of the parser which allows the parser to differentiate between the input source (Ammar et al., 2016; Szymne et al., 2018; Wagner et al., 2020; van der Goot et al., 2021). As described by van der Goot et al. (2021), they are continuous representations which encode the source of the dataset, and throughout training, the embedding will capture properties of the data source while still keeping their heterogeneous characteristics. When training a single model with multiple data sources \mathcal{D} , we learn a dataset embedding $e(d)$ for each data source $d \in \mathcal{D}$.

Typically each input token is assigned its own dataset embedding. That is, for each

²In related works, such an embedding may be referred to as a *treebank* or *language embedding*, depending on the abstraction. We will use the more general term *dataset embedding*.

word w_1, \dots, w_n in a sentence of length n , the embedding x_i for the i^{th} word is the concatenation of a word embedding $e(w_i)$, a dataset embedding $e(d_i)$ as well as a character representation obtained by running a BiLSTM over the m characters ch_1, \dots, ch_m of token w_i , as in (3.1):

$$x_i = [e(w_i) \circ e(d_i) \circ \overleftarrow{ch_1} \circ \overrightarrow{ch_m}] \quad (3.1)$$

The token representation x_i is then fed to a contextualised encoder such as a BiLSTM covering the input sequence, producing the hidden representation for the i -th word h_i :

$$h_i = \text{BiLSTM}[x_1, \dots, x_n] \quad (3.2)$$

This process is shown in Figure 3.3 using the English word *Flight* as an example. The example shows this process for just one word, but in practice this process is done for each word in the input sentence. All embeddings are updated during training.

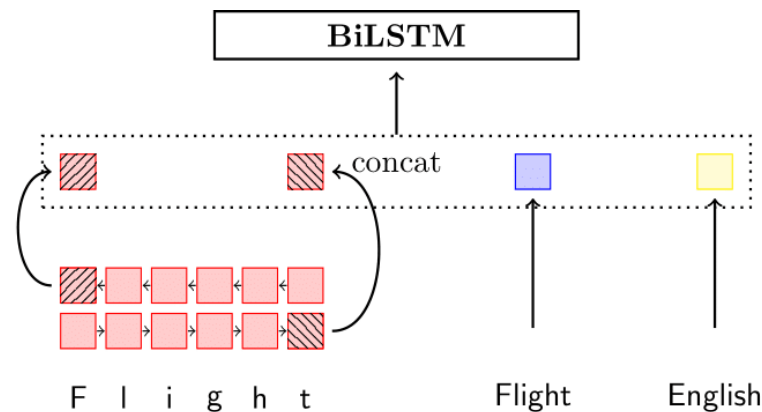


Figure 3.3: An example of a dataset or language embedding, in this case *English* which is concatenated to the token-level input which consists of a word and a character representation and then passed to a BiLSTM encoder.

Dataset embeddings were first used in dependency parsing by Ammar et al. (2016) who include training data from 7 languages in their parsing experiments. They use features such as multilingual Brown clusters and pre-trained multilingual word embed-

dings to learn multilingual representations for parsing. They encode a one-hot vector denoting the language ID of the input to help the parser differentiate between input languages and found this approach to be successful. Initially testing on monolingual data, Stymne et al. (2018) use a treebank embedding which is attached to each input token and marks the treebank that a sentence belongs to. They show that using a treebank embedding outperforms both training on a single treebank and training on a concatenation of treebanks. This idea was extended to the multilingual case by Smith et al. (2018) in the 2018 CoNLL shared task (Zeman et al., 2018). Here, Smith et al. separate the languages into groups based on language family and language-relatedness and train a single model per grouping. For instance, they train 34 parsing models which can parse 82 treebanks. In preliminary experiments, they found that it was better to include a smaller number of related languages than to include a larger amount of less-related languages. They compare these unified models with monolingual baselines and show that training with multiple languages (i. e. polyglot training) helped, whereas for the 64 test sets that were parsed with a polyglot model, only 4 were outperformed by the monolingual baseline. The results also show that this approach is particularly helpful for low-resource languages, which benefit from the training data of other related languages.

More recent work by van der Goot et al. (2021) looks at the feasibility of using dataset embeddings for the morphosyntactic tasks of lemmatisation, morphological tagging and dependency parsing. They analyse the role of dataset embeddings when the test data comes from the same source as the training data: either with gold dataset labels or predicted ones. They also experiment with settings where the test source is not included in the training data (zero shot). They show that dataset embeddings are superior to single dataset training and concatenation for all three tasks when gold information about the training and test source is provided. They are most useful for dependency parsing, followed by lemmatisation, with more modest gains for morphological tagging.

The breakdown between monolingual and multilingual groupings shows that dataset embeddings work best across all tasks and settings except for morphological tagging in multilingual groups, which could be due to the fact that it is harder to transfer useful information for this task which relies on language-specific information. When moving to predicted dataset embeddings, they are only useful for lemmatisation and dependency parsing. When dataset embeddings are used in a zero shot setting, i. e. when the test data is not included in the training group, there is only a marginal increase in dependency parsing LAS, but these gains do not occur in cases where there is no in-language dataset. Therefore, the authors conclude that dataset embeddings are not useful in setups where the test data is from another distribution.

van der Goot and de Lhoneux (2021) build upon the work of Smith et al. (2018) and analyse the role of dataset embeddings when used in a Transformer architecture. They test whether it is better to add a dataset embedding to the input of the encoder (in the same way as the position or segment embedding), concatenate a dataset embedding to the output of the encoder, or in a third setting, combine both approaches. Their results show that the dataset embedding approaches work better than monolingual baselines but also that a lot of the performance seen through the use of dataset embeddings can be achieved through concatenation alone (something that was typically less effective when using BiLSTMs (Che et al., 2018)). They find that adding the dataset embedding to the input embeddings is most helpful, perhaps because this information is passed to the encoder. Their results corroborate previous findings that use BiLSTM architectures (Smith et al., 2018; van der Goot et al., 2021) by showing that such approaches also work well for Transformers. In this work they train the models on closely related language clusters but also try a variant where the same techniques are applied to a model trained on all treebanks together. The results between the models trained on language clusters and all languages are similar, showing positive evidence for Transformers being able to scale to many languages (Kondratyuk and Straka, 2019).

de Lhoneux et al. (2018) compare different parameter sharing strategies in a transition-based parser across pairs of languages. The parameter sharing strategies range between hard-sharing, where all parameters are shared; soft-sharing, where a dataset embedding is used to differentiate between the languages; and no-sharing, where the parameters are trained independently. When considering related languages, all parameter-sharing strategies improved over a monolingual baseline. They found that sharing the later parts of the parser (e. g. the weights of the MLP classifier), was consistently helpful but the results of sharing word and character parameters were more varied depending on the language pairs used. They identify a successful recipe for parameter sharing between related languages which includes soft sharing for word and character parameters and hard sharing of the MLP parameters. The architecture proposed in Chapter 6 uses soft sharing in the encoder and includes a parsing head where all parameters are shared (hard-sharing).

3.2 Cross-lingual Parsing with Polyglot Training and Multi-treebank Learning

We have described how annotation projection (Yarowsky et al., 2001; Tiedemann and Agić, 2016) can be used to create a synthetic treebank for low-resource languages, and how polyglot training (Smith et al., 2018; Kondratyuk and Straka, 2019; van der Goot et al., 2021; van der Goot and de Lhoneux, 2021) can be helpful in processing low-resource languages. The experiment described in this chapter seeks to find out whether these two approaches are complementary for the task of low-resource dependency parsing. Similarly, what role will training a multi-treebank model on the individual silver target treebanks play on inducing a target model? This experiment builds on work by Tyers et al. (2018) who show that in the absence of annotated training data for the target language, a lexicalised treebank can be created by translating a target-language corpus

into a number of related source languages and parsing the translations using models trained on the source-language treebanks.³ These annotations are then projected to the target language using separate word alignments for each source language, combined into a single graph for each sentence and decoded (Sagae and Lavie, 2006), producing a treebank for the target language: Faroese in the case of the experiments of Tyers et al. as well as our own.

To answer these questions, we replicate the experiments of Tyers et al. (2018) and investigate whether additional improvements can be made by making the following two interventions:

1. using a single polyglot parsing model which is trained on the combination of all source languages to create synthetic source treebanks (which are subsequently projected to the target language),
2. training a multi-treebank model on the individually projected treebanks and the treebank produced with multi-source projections.

The former differs from the approach of Tyers et al. (2018), who use multiple discrete monolingual models to parse the translated sentences, whereas in this work we use a single model trained on multiple source treebanks. The latter differs from training on the target treebank produced by multi-source projection in that the information of the individual projections is still available and training data is not reduced to cases where all source languages provide a projection.

In other words, we aim to investigate whether the state-of-the-art approach for Faroese, which relies on cross-lingual transfer, can be improved upon by adopting an approach based on source-side polyglot learning and/or target-side multi-treebank learning. We hypothesise that a polyglot model can exploit similarities in morphology and syntax across the included source languages, which will result in a better model to provide an-

³In this work, *source language* and *target language* always refer to the projection, not the direction of translation.

notations for projection. On the target side, we expect that combining different sources of information will result in a more robust target model.

3.2.1 Method

We outline the process used for creating a synthetic treebank for cross-lingual dependency parsing. We use the following resources: raw Faroese sentences taken from Wikipedia, an MT system to translate these sentences into all source languages (Danish, Swedish, Norwegian Nynorsk and Norwegian Bokmål), a word-aligner to provide word alignments between the words in the target and source sentences, treebanks for the four source languages on which to train parsing models, POS tagging and parsing tools, and lastly, a target language test set. We use the same raw corpus, alignments and tokenised and segmented versions of the source translations⁴ as Tyers et al. (2018) who release all of their data.⁵ In this way, the experimental pipeline is the same as theirs but we predict POS tags and dependency annotations using our own models.

Target Language Corpus We use the target-language corpus built by Tyers et al. (2018) which comprises 28,862 sentences which were extracted from Faroese Wikipedia dumps⁶ using the WikiExtractor script⁷ where the sentences were also further pre-processed to remove any non-Faroese texts and other forms of unsuitable sentences.

Machine Translation As noted by Tyers et al. (2018), popular repositories for developing machine translation systems such as OPUS (Tiedemann, 2016) contain an inadequate amount of sentences to train a data-driven machine translation system for Faroese. For instance, there are fewer than 7,000 sentence pairs between Faroese and Danish, Faroese and English, Faroese and Norwegian and Faroese and Swedish. Consequently,

⁴The original authors tokenise and segment the source translations with UDPipe (Straka and Straková, 2017).

⁵<https://github.com/ftyers/cross-lingual-parsing>

⁶<https://dumps.wikimedia.org/>

⁷<https://github.com/attardi/wikiextractor>

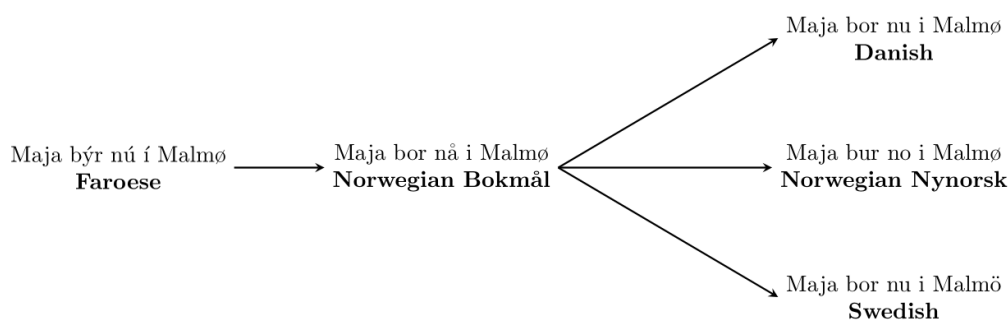


Figure 3.4: Overview of the machine translation process. An example Faroese sentence *Maja býr nú í Malmø* (Maja now lives in Malmø) is first translated into Norwegian Bokmål and then from Norwegian Bokmål into the other source languages using pivot translation. The language of the text is shown in bold and the arrow direction shows the direction of translation. Example translations are taken from Tyers et al. (2018).

to create parallel source sentences, Tyers et al. (2018) use a rule-based machine translation system available in Apertium (Forcada et al., 2011)⁸ to translate from Faroese to Norwegian Bokmål. There also exist translation systems from Norwegian Bokmål to Norwegian Nynorsk, Swedish and Danish in Apertium. As a result, the authors use *pivot translation* from Norwegian Bokmål into the other source languages. The process is illustrated in Fig. 3.4. For a more thorough description of the machine translation process and resource creation in general, see the work of Tyers et al. (2018).

Word Alignments We use word alignments between the Faroese text and the source translations generated by Tyers et al. (2018) who use `fast_align` (Dyer et al., 2013), a word alignment tool based on IBM Model 2.⁹

Source Treebanks We use the Universal Dependencies v2.2 treebanks Nivre et al. (2018a) to train our source parsing models. This is the version used for the 2018 CoNLL

⁸<https://github.com/apertium>

⁹Note that previous related work (Agić et al., 2016) report better results using IBM Model 1 with a more diverse language setup. They claim that IBM Model 2 introduces a bias towards more closely related languages. As we are working with related languages and translations and alignments are largely word-for-word, we expect that this will have less of an impact on our experiments although IBM Model 1 should also be tried in future work.

shared task on Parsing Universal Dependencies (Zeman et al., 2018).

Source Tagging and Parsing Models In order for our parsers to work well with predicted POS tags, we follow the same steps as used in the 2018 CoNLL shared task for creating training and development treebanks with automatically predicted POS tags (henceforth referred to as silver POS). Since we are required to parse translated text which only has lexical features available, we disregard lemmas, language-specific POS (XPOS) and morphological features and only use the word form and universal POS (UPOS) tag as input features to our parsers. We develop our POS tagging and parsing models using the AllenNLP library (Gardner et al., 2018).

We use jackknife resampling to predict the UPOS tags for the training treebanks. We split the training treebank into ten parts, train models on nine parts and predict UPOS for the excluded part. The process is repeated until all ten parts are predicted and they are then combined to recreate the treebank with silver POS tags. Only token features are used to predict the UPOS tag.¹⁰ Finally, we train a model per source language on the full training data to check performance on the respective development set and to POS tag the source language translations before parsing.

We train two variants of parsing models: The first is a monolingual biaffine dependency parser (Dozat and Manning, 2017) trained on the individual source treebanks. The second is a polyglot model trained on all source treebanks using the multilingual parser of Schuster et al. (2019), which is the same graph-based biaffine dependency parser, extended to enable parsing with multiple treebanks. We additionally include a dataset embedding (Section 3.1.2) to the input of the polyglot parser to help the parser differentiate between the source languages. We optimise the model for average development set LAS across the included languages. The process is illustrated in Fig. 3.5.

To ensure that our parser is realistic, we add a pre-trained monolingual word em-

¹⁰We observe slightly lower POS tagging scores on fully annotated test sets than UDPipe, which uses gold lemmas, XPOS and morphological features to predict the UPOS label and so cannot be applied to the translated text without also building predictors for these features.

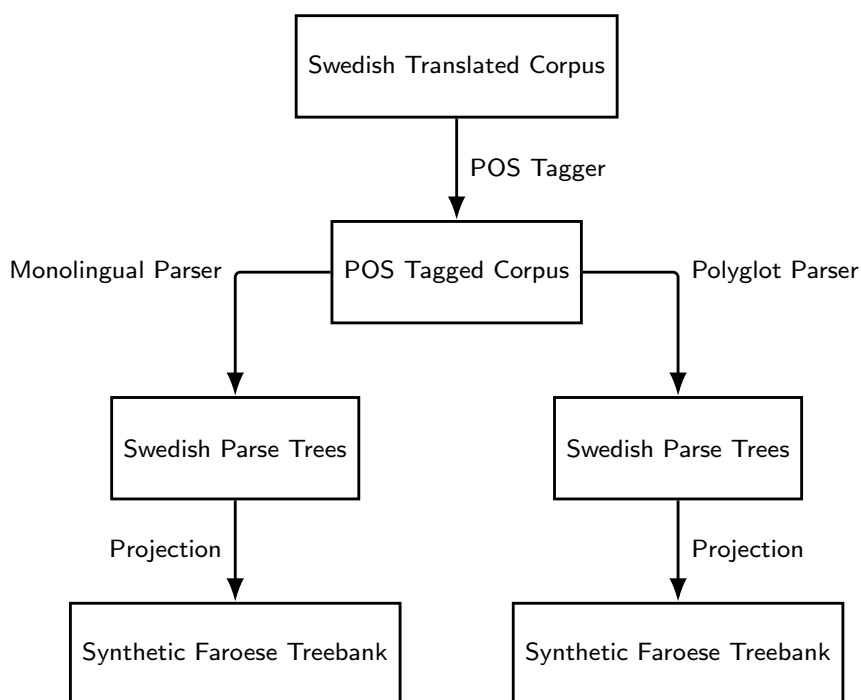


Figure 3.5: Overview of the *monolingual* and *polyglot* parser experiments using Swedish translations as an example. This process is repeated for all source languages.

bedding to each monolingual parser, giving a considerable improvement in accuracy on the development sets of the source languages. We use the precomputed Word2Vec embeddings¹¹ released as part of the 2017 CoNLL shared task on UD parsing (Zeman et al., 2017) which were trained on CommonCrawl and Wikipedia.

In order to use pre-trained word embeddings for the polyglot setting, we need to consider that a polyglot model uses a shared vocabulary across all input languages. In our experiments, we simply use the union of the word embeddings and average the word vector for words that occur in more than one language. This differs from the approach of Smith et al. (2018) who overwrite the word embedding if it was seen in a previous language.¹² Future work should explore cross-lingual word embeddings with

¹¹<https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989>

¹²<https://github.com/UppsalaNLP/uuparser/issues/7>

a limited amount of parallel data or use aligned contextual embeddings as in Schuster et al. (2019).¹³

Synthetic Source Treebanks Source translations are tokenised with UDPipe by Tyers et al. (2018). For each source language, the POS model trained on the full training data (see previous section) is used to tag the tokenised translations. Once the text is tagged, we predict dependency arcs and labels with the parsing models of the previous section, and use annotation projection (described below) to provide syntactic annotations for the target sentences.

Annotation Projection Once the synthetic source treebanks are compiled, i. e. the translations are tokenised, tagged and parsed, the annotations are then projected from the source translations to the target language using the word alignments and projection tool of Tyers et al., resulting in a Faroese treebank induced from that particular source language. In some cases, not all tokens are aligned and Tyers et al. (2018) work around this by falling back to a 1:1 mapping between the target index and the source index. There are also cases where there is a mismatch in length between the source and target sentences and some dependency structures cannot be projected to the target language. The projection setup of Tyers et al. removes unsuitable projected trees containing, for example, more than one root token, a token that is its own head or a token with a head outside the range of the sentence.

Multi-source Projection For multi-source projection, the four synthetic dependency trees (induced from each source language) for a Faroese sentence are projected into a single graph, where the edges are scored according to the number of trees that contain them (Sagae and Lavie, 2006; Nivre et al., 2007a). The dependency structure is first

¹³Pre-trained multilingual language models with a shared vocabulary such as Multilingual BERT (Devlin et al., 2019) would also obviate the need for using aligned cross-lingual word embeddings.

built by voting over the directed edges. Afterward, dependency labels and POS tags are decided using the same voting procedure. The process is illustrated in Fig. 3.6.

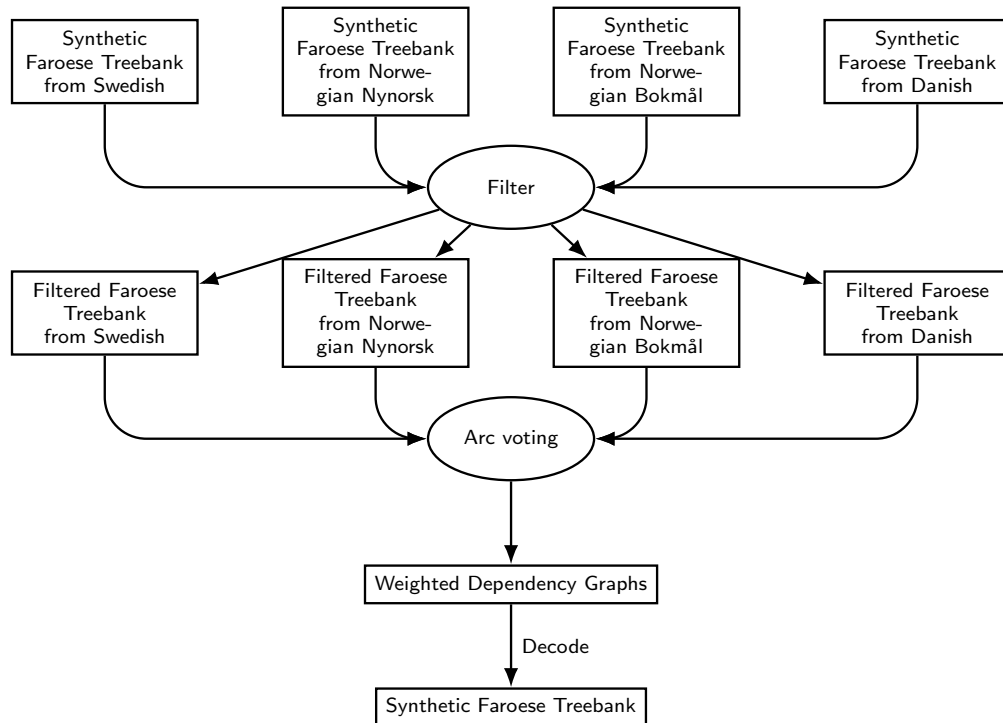


Figure 3.6: Multi-source projection. The source language is listed in brackets. The *Filter* process involves filtering out sentences where all four sources do not provide a valid projection.

Target Tagging and Parsing Models At this stage we have Faroese treebanks to train our POS tagging and parsing models. The Faroese treebanks come in two variants: the result of projection from source trees produced by either a monolingual model or the polyglot model. For each case, we train our POS tagging and parsing models directly on these synthetic treebanks and do not make use of word embeddings as we do not have them for Faroese.

Multi-treebank Target Parsing Since we have several synthetic Faroese treebanks, we have the option of training on a single treebank or using a multi-treebank approach where we train on all target treebanks in the same way as we did for inducing the polyglot source model. The process of training a multi-treebank target model is illustrated in Fig. 3.7. When training a multi-treebank target model, for each target treebank, we add a dataset embedding denoting the source model used to project annotations for that target treebank. This should allow the model to learn information about the source language used to produce that treebank. At predict time, we must include one of these treebank/dataset embeddings as input to the model. As we do not have gold Faroese data in our target training treebanks, we must choose the dataset embedding of one of the synthetic target treebanks. Stymne et al. (2018) introduce the term “proxy treebank” to refer to cases where the test treebank is not in the training set and a dataset embedding from the training set must be used instead.

Without development data to choose the best proxy treebank, one can determine this based on test results. However, simply choosing the best proxy treebank based on test results can be seen as a form of ‘cherry picking’ and cannot be done in settings where there is no gold test data. In order to overcome this limitation, in subsequent work (Wagner et al., 2020), we developed a predictor which takes an out-of-domain test sentence and tries to match it to the most similar sentence in the training set based on a sentence similarity measure. All training set sentences are parsed using a variety of either fixed or interpolated treebank weights, and the best performing treebank weights for the matched training sentence are applied to the test sentence. The same approach can also be done at the treebank level, where given a test treebank, the predictor will try to predict the most similar training treebank.

Another method is also using the average or centroid of the treebank embedding, however, work by Üstün et al. (2020) shows that this method performs poorly for zero-shot languages without language embeddings, and their approach of conditioning lan-

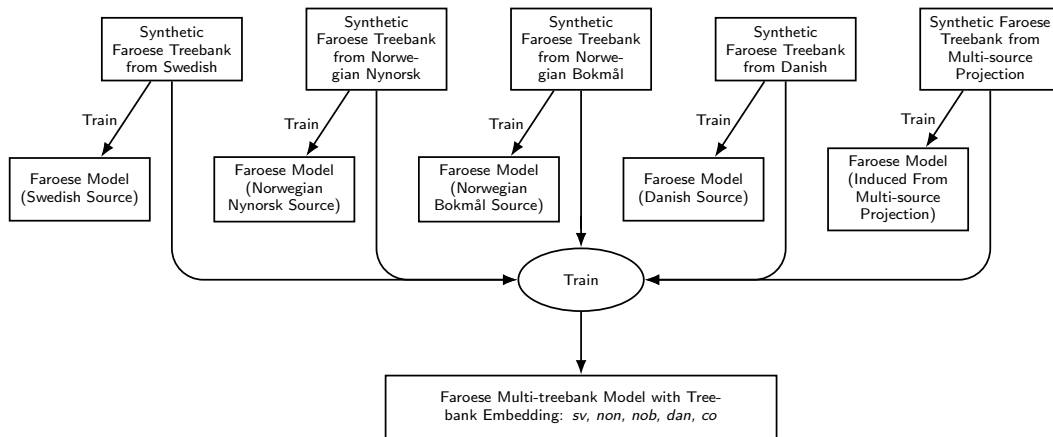


Figure 3.7: Single versus multi-treebank training. The source language is listed in brackets.

guage embeddings on a wide variety of typological features performs better, e. g. using the centroid resulted in average LAS of 9.0 over 30 languages versus their language typology-based approach which had an average LAS of 36.5.

3.2.2 Experiments

In this section, we describe our experiments, which include a replication of the main findings of Tyers et al. (2018), using AllenNLP (Gardner et al., 2018) for POS tagging and parsing instead of UDPipe.¹⁴

Details

The hyper-parameters of our POS tagging and parsing models are given in Table 3.1, which are inspired from other tagging and parsing systems from the 2018 CoNLL shared task (Zeman et al., 2018) as well as a small number of preliminary experiments. For POS tagging, we adopt a standard architecture with a word and character level Bi-LSTM (Graves and Schmidhuber, 2005; Plank et al., 2016) to learn context-sensitive representa-

¹⁴All of the code and scripts to reproduce the experiments can be found at <https://github.com/jbrry/multilingual-parsing>

POS Tagger Architecture	
Parameter	Value
Char-BiLSTM layers	2
BiLSTM layers	2
BiLSTM size	400
Dropout LSTMs	0.33
Dropout MLP	0.33
Dropout embeddings	0.33
Nonlinear act. (MLP)	ELU

Parser Architecture	
Parameter	Value
Char-BiLSTM layers	2
BiLSTM layers	3
BiLSTM size	400
Arc MLP size	500
Label MLP size	100
Dropout LSTMs	0.33
Dropout MLP	0.33
Dropout embeddings	0.33
Nonlinear act. (MLP)	ELU

Embeddings	
Parameter	Size
Word embedding (both)	100
Char embedding (both)	64
POS embedding (parser)	50
Treebank embedding (both)	12

Training	
Parameter	Value
Optimizer	Adam
Learning rate	0.001
Adam epsilon	1e-08
beta1 (both)	0.9
beta2 (parser)	0.9
beta2 (tagger)	0.999

Table 3.1: Chosen hyperparameters for our POS tagging and parsing models, *both* means the feature is common to both the POS tagger and parser.

tions of our words. These representations are passed to a multilayer perceptron (MLP) classifier followed by a softmax function to choose a tag with the highest probability. For both the POS tagging and parsing models, we use a word embedding dimension of size 100 and a character embedding dimension of size 64. POS tag embeddings of dimension 50 are included in the parser. We train our Faroese models for 50 epochs, the same as for

the source models, which tended to converge by this number of epochs. We do not split the synthetic Faroese treebanks into training/development portions though we suspect doing so will help the models to not overfit on the training data. For all experiments we report the LAS produced by the official CoNLL 2018 evaluation script.¹⁵

Results

The development results of our monolingual and polyglot models on the source language treebanks are shown in Table 3.2. The results for the polyglot model are better for three out of four source languages: Danish, Swedish and Norwegian Bokmål, with an LAS of 82.75, 83.85, and 90.29 for the polyglot model vs. 81.10, 80.61, and 89.29 for the monolingual model. For Norwegian Nynorsk, the monolingual model marginally outperforms the polyglot one (88.54 vs. 88.29).

The statistics of the filtered Faroese treebanks obtained via projection with our source parsing models are given in Table 3.3. The treebank sizes are fairly similar regardless of whether source annotations are provided by a monolingual or a polyglot model which is expected because the word alignments are the major factor in determining whether a projection is successful. There is a proportionally lower number of sentences for multi-source projection. This is because this method only uses the intersection of sentences which are present across all synthetic treebanks after filtering. The treebank originating from Norwegian Bokmål has the highest number of valid sentences, suggesting that it could be a good candidate for projection to Faroese. It also has the highest source-language parsing accuracy (90.29 in Table 3.2) and was the language which had a machine translation system available from Faroese in Apertium (i. e. was the language used for pivot translation).

The results of training on our various synthetic Faroese treebanks and predicting the Faroese test set are shown in the first result column of Table 3.4 (SINGLE). In terms

¹⁵https://github.com/ufal/conll2018/blob/master/evaluation_script/conll18_ud_eval.py

TREEBANK	MONOLINGUAL	POLYGLOT
da_ddt	81.10	82.75
sv_talbanken	80.61	83.85
no_nynorsk	88.54	88.29
no_bokmaal	89.29	90.29
average	84.88	86.30

Table 3.2: Source model LAS scores on the development treebanks using silver POS tags.

SOURCE	MONOLINGUAL	POLYGLOT
Danish	13,950	13,944
Swedish	10,894	10,874
Norwegian Nynorsk	13,177	13,194
Norwegian Bokmål	17,345	17,378
Multi-source	6,716	6,833

Table 3.3: The number of valid sentences in the Faroese synthetic treebank for each source language after annotation projection and sentence filtering. The original corpus size was 28,862 sentences.

of monolingual vs. polyglot, we find that projecting from a polyglot model helps with four out of the five possible treebanks, with three of them being statistically significant (Danish, Swedish and Norwegian Bokmål).¹⁶ The polyglot model was outperformed by the monolingual model using Norwegian Nynorsk for projection though the difference is not statistically significant (69.80 vs. 70.27). On the source side, the monolingual Norwegian Nynorsk model also performed slightly better than the polyglot model (Table 3.2). This observation supports the intuition that the quality of the projected annotations can be improved by contributing better source annotations, i. e. improving the source model(s) is one way to improve performance of the target model. This is supported by the fact that the source language with the highest LAS (Norwegian Bokmål) is also the best choice for projection when using a polyglot model (in this single target model setting) with an LAS of 70.51.

¹⁶Statistical significance is tested with `udapi-python` <https://github.com/udapi/udapi-python>. LAS differences are reported as significant if $p < 0.05$.

SOURCE LANGUAGE	SOURCE MODEL	TARGET MODEL	
		SINGLE	MULTI
Danish	Monolingual	61.24	63.40
	Polyglot	65.29 [†]	65.53[†]
Swedish	Monolingual	65.93	66.15
	Polyglot	68.60 [†]	69.69[†]
Norwegian Nynorsk	Monolingual	70.27	71.51
	Polyglot	69.80	71.13
Norwegian Bokmål	Monolingual	67.46	67.94
	Polyglot	70.51 [†]	70.58[†]
Multi-source	Monolingual	68.00	69.80
	Polyglot	68.55	70.07
Average	Monolingual	66.58	67.76
	Polyglot	68.55	69.40

Table 3.4: LAS on the target Faroese test treebank. *Single* refers to using a single synthetic Faroese treebank to train a Faroese model, *Multi* uses both a multi-treebank POS tagger and a multi-treebank parser with all synthetic Faroese treebanks. The multi-treebank model is tested with each of the five training treebanks (four projected from individual source languages and one using multi-source projection) as proxy treebank. Statistically significant differences between the monolingual and polyglot setting are indicated by † for each result pair, excluding averages.

The multi-source approach was not that effective in our case and some individual sources were able to surpass this combination approach. One could argue that this may be due to the lower amount of training data when using the multi-source synthetic treebank. We test this hypothesis by only including those sentences which contributed to multi-source projection in the single-source synthetic treebanks. The results are given in Table 3.5. Comparing the results in Tables 3.4 and 3.5, we see that LAS scores tend to be slightly lower than on the version which included all target sentences, e. g. the average monolingual score drops from 66.58 to 65.43, and the average polyglot score drops from 68.55 to 67.70. This indicates that we did lose some information by filtering out a large number of sentences. However, Norwegian Nynorsk still outperforms the multi-source model for the monolingual setting and both Norwegian models perform better than the multi-source model in the polyglot setting, suggesting that size alone

does not explain the under-performance of the multi-source model. It is also worth noting that polyglot training is superior to all monolingual models which hints that for no_nynorsk (the previously better performing model), the monolingual model was not able to achieve its full potential with the reduced data while the polyglot model was able to provide richer annotations. Another reason why the multi-source model does not work as well in our experiments as it does in those of Tyers et al. (2018) might be because we use pre-trained word embeddings, whereas Tyers et al. (2018) do not. In this way, our monolingual models are stronger to begin with and likely do not benefit from including less accurate parses in the multi-source voting procedure.

The second result column (MULTI) of Table 3.4 shows the effect of training a multi-treebank POS tagger and parser on the Faroese treebanks created by each of the four source languages as well as the treebank which is produced by multi-source projection. This experiment is orthogonal to the experiment using a polyglot model on the source side and so we also test a combination of polyglot source-side parsing and multi-treebank target-side parsing. We see improvements over the single treebank setting for all cases.¹⁷

SOURCE LANGUAGE	MONOLINGUAL	POLYGLOT
Danish	61.13	64.43
Swedish	63.19	67.46
Norwegian Nynorsk	68.72	69.28
Norwegian Bokmål	66.13	68.77
Multi-source	68.00	68.55
Average	65.43	67.70

Table 3.5: LAS scores between target models trained on the subset of sentences eligible for multi-source projection (with annotations from the stated source).

Table 3.6 places our systems in the context of previous results on the same Faroese

¹⁷The multi-treebank tagger closely resembles the dependency parser, where we add a dataset embedding and optimise for average accuracy across the included treebanks. We also tested the effect of training only the dependency parser using multiple treebanks but found that it always helps to also perform multi-treebank training for the POS tagger.

WORK	RESULT
Rosa and Mareček (2018)	49.4
Tyers et al. (2018)	64.4
Our implementation of Tyers et al. (2018)	68.0
Our Best Model	71.5

Table 3.6: Comparison to previous work. LAS on Faroese test set. Note that the first results uses predicted segmentation and tokenization whereas the rest used gold.

test set. The highest-scoring system in the 2018 CoNLL shared task was that of Rosa and Mareček (2018) who achieved a LAS score of 49.4 on the Faroese test set. Note that they use predicted tokenisation and segmentation, whereas our experiments and those of Tyers et al. use gold tokenisation and segmentation, which provides a small artificial boost. Tyers et al. (2018) report an LAS of 64.43 with a monolingual multi-source approach. Our implementation which uses a different parser (AllenNLP versus UDPipe) and pre-trained word embeddings achieves an LAS of 68. Our highest score of 71.51 is achieved through the combination of projecting from strong monolingual source models and then training multi-treebank POS tagging and parsing models on the outputs.

3.3 Summary

In this chapter, we have presented parsing results on Faroese, a low-resource language, using annotation projection from multiple monolingual sources versus a single polyglot model. We also extended the idea of multi-treebank learning to the target treebanks. The results of our experiments show that the use of a polyglot source model helps for three of the four source languages (Danish, Norwegian Bokmål and Swedish) when using single treebank target models. The two source languages that have the lowest LAS when using monolingual parsers, namely Danish and Swedish, see significant improvements when switching to a polyglot model. Our best-performing single target

model is trained on Faroese trees projected by the polyglot model using the Norwegian Bokmål proxy dataset embedding. However, the strongest language with monolingual modelling, Norwegian Nynorsk, does not benefit from switching to a polyglot model. When we filtered the target treebanks to the subset of sentences eligible for multi-source projection, the polyglot model is superior to all five monolingual models, even outperforming the Norwegian Nynorsk model.

We also applied the multi-treebank approach to the target-side POS tagger and parser and see improvements for all settings. The overall best result is with the setting that uses monolingual models to create the source trees that are projected to Faroese and combined in a multi-treebank model, and using the dataset embedding of Norwegian Nynorsk for predicting Faroese trees. This finding is interesting, because when considering single-treebank target modelling, the polyglot model was better overall. Perhaps it is this balance between allowing each source model to fully learn the properties of its own dataset and then fusing this information later on in the multi-treebank target model which makes this setting perform better. In other words, while some sources are weaker, the multi-treebank model can still learn from the single-best source in the monolingual setting, whereas for the polyglot model, while it improves some of the weaker sources, for Norwegian Nynorsk there is a small amount of interference which decreases the upper bound of information which can be passed to the target model from this source. The situation where two or more languages are competing with one another and performance deteriorates with respect to a monolingual model is known as “negative interference” or “negative transfer” (Wang et al., 2019, 2020b), and this is likely a factor at play for Norwegian Nynorsk. It is then essential for future work to address negative interference in multilingual models, while still promoting the positive aspects of sharing across languages. This is a difficult balance and we will try to address this in Chapter 6.

The aim of this chapter was to answer our first research question *RQ1: Can zero-*

shot cross-lingual dependency parsing be improved by leveraging polyglot training on source languages? The experiments in this chapter provide positive evidence for this question, where four out of five sources (including the multi-source projection setting) can be improved by adopting a polyglot model, and the overall best setting when considering single-treebank target models came from the Faroese trees predicted by the polyglot model. However, when using the multi-treebank target model, the overall best setting comes from using monolingual projections. Therefore, we cannot conclude that polyglot modelling provides better annotations for transfer, but that it should help produce an overall trend of better results for some of the weaker sources while some languages, which are already good candidate sources, may suffer from negative interference.

It is worth noting that the approach taken in this work has some drawbacks. Firstly, the annotation projection setup requires aggressive filtering of certain source sentences where there is not a one-to-one mapping between the source and target tokens. Furthermore, the reliance on ‘hard’ alignments and dependency edge predictions means that the selected target annotations are more susceptible to noise from the source-side annotations. Another approach to consider would be that of Agić et al. (2016), who build the target dependency tree for a sentence from a dense set of edge predictions from multiple sources, weighted by alignment probabilities. This multi-source annotation projection setup was also used to good effect by Plank and Agić (2018) who extended it by ranking the target sentences by the percentage of words covered by the word alignments over the considered source languages, and select the top- k instances for training.

Another drawback of the approach taken in this chapter is the choice of MT system and the alignment model. Firstly, the reliance on pivot MT from Norwegian Bokmål to the other source languages means that more noise is introduced into the translations of the other source languages. Recently, there have been advances in developing multilingual MT systems (Tiedemann and Thottingal, 2020), which would likely serve as better MT models to cater for the languages used in our study. Secondly, there have also

been advances in using Transformers in neural word alignment systems (e. g. (Jalili Sabet et al., 2020; Dou and Neubig, 2021)). Thus, it would be important for future work to consider these approaches.

In the next chapter we will also explore low-resource dependency parsing but for Irish, a moderately low-resource language which has training data. Instead of examining the role of monolingual and multilingual (polyglot) source models in an annotation projection setting, we will look at the role of monolingual and multilingual pretrained LMs in contributing features for performing dependency parsing on Irish data.

Chapter 4

gaBERT a Monolingual Irish BERT

Model

The previous chapter involved using a data transfer technique known as annotation projection to process a low-resource language with no training data. In this chapter, we now focus on the role of model transfer, where information is transferred from a pretrained LM for some downstream task. For our case, we focus on dependency parsing in Irish, a moderately low-resource language. In Chapter 2, we mentioned that large language models (LLMs) are a powerful tool for enriching lexical features for NLP tasks. This is particularly the case for low-resource languages such as Irish, which has a comparatively smaller treebank size than other languages in UD, and incorporating external knowledge from LLMs is a useful way to make up for the smaller amount of training data.

While LMs are known for their ability to provide useful features for NLP tasks, it is not always clear whether one should use a monolingual LM or a multilingual LM. Furthermore, as we have seen in Chapter 3, sometimes training on monolingual information is better for dependency parsing. In this chapter, we first introduce some recent popular LMs and discuss the role of monolingual and multilingual LMs for within-language

downstream performance. We then report on the development of a monolingual Irish BERT model, **gaBERT**, and compare this model to a number of multilingual baselines for the tasks of dependency parsing, morphological features classification, POS tagging as well as Multiword Expression (MWE) identification. In doing so, we will provide answers to *RQ2: Does a monolingual language model improve low-resource dependency parsing in the case of Irish?*

4.1 Background

In Chapter 2, we described how contextual embeddings from pre-trained LMS are used in state-of-the-art dependency parsing architectures. In fact, pretrained neural LMs have become a crucial component in almost all NLP applications due to their ability to provide sequences of text with context-sensitive token encodings which are able to generalise well to many NLP tasks (Peters et al., 2018; Devlin et al., 2019). Among such models include ELMo (Embeddings from Language Models) (Peters et al., 2018), which takes unlabelled text as input and obtains contextualised representations using a language-modelling objective. It has been shown by Peters et al. (2018) that this training procedure enables the representations to capture syntactic and semantic information from unlabelled corpora, which can be transferred to downstream tasks.

Another popular language model is BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). BERT differs from ELMo in that the CNN and bidirectional LSTM of ELMo is replaced with a Transformer, which consists of stacked self-attention layers (see Section 2.2.3). A concept called masked language modelling (MLM) is used to update the parameters of the model, which requires the model to predict randomly masked words using the surrounding words as context. Additionally, a classification task termed next sentence prediction (NSP) is used to classify whether two sentences are contiguous or not.

The introduction of Transformer-based LMs such as BERT has led to a remarkable

increase in performance across many common NLP tasks such as Natural Language Inference (Bowman et al., 2015), Question Answering (Rajpurkar et al., 2016) as well as dependency parsing (Kondratyuk and Straka, 2019; Kulmizev et al., 2019),¹ and such models have become a mainstay for NLP.

4.1.1 Multilingual Language Models

Coinciding with the release of Transformer-based LMs (which had originally occurred for English) has been an effort to create multilingual LMs trained on data from many languages (Devlin et al., 2019; Lample and Conneau, 2019; Conneau et al., 2020). The motivation behind training such multilingual LMs is due to their ability to promote cross-lingual transfer among languages. For instance, by training on multiple languages together, the model can learn to make generalisations across languages that would not be possible by training on an individual language. Secondly, training on multiple languages can be particularly helpful for low-resource languages which may not have enough monolingual data to learn useful representations for that language. Prior to multilingual LMs, cross-lingual word embeddings (CLWEs) were the representation of choice to map tokens from different languages to a shared cross-lingual space. This originally occurred with static word embeddings (Mikolov et al., 2013; Ammar et al., 2016; Glavaš et al., 2019; Vulić et al., 2019), but since the introduction of transformer-based representations, such CLWEs have been largely replaced by representations obtained from pretraining LMs on multilingual text (Devlin et al., 2019; Lample and Conneau, 2019; Conneau et al., 2020), because such models have shown superior performance.

The initial BERT model was released for the English language. Shortly after, a multilingual version (mBERT), was released which was trained on 104 languages using Wikipedia data. The training of multilingual LMs such as mBERT involves combining data from multiple languages, where typically a shared subword vocabulary is created

¹<https://github.com/sebastianruder/NLP-progress> is a well-maintained leaderboard ranking for many NLP tasks.

and the model is trained over the combined multilingual data using a language modelling objective. The representations are then fine-tuned on a downstream task using labelled data for a particular language. As stated by Artetxe et al. (2020), the multilingual pretraining gives the model the ability to generalise to other languages, even if there is no data for a particular language in the multilingual training data. This is in spite of the fact that such a pretraining procedure does not involve any cross-lingual supervision such as parallel data or bilingual dictionaries. For this reason, multilingual LMs are a good choice of model for processing languages which may not have that much training data.

While multilingual LMs provide a general purpose resource which can be successfully applied to many languages, a property of multilingual LMs is that they are reasonably capable of processing any of the included languages but for high-resource languages with ample training data, monolingual models can be superior and lower-resource languages benefit the most from multilingual training. This phenomenon is known as the “curse of dimensionality” (Conneau et al., 2020), where the multilingual LM cannot store enough information to cover all of the included languages, and model size must be increased to accommodate more languages. This is especially the case for high-resource languages which have plentiful monolingual training data, such as for Dutch (de Vries et al., 2019), Finnish (Virtanen et al., 2019), and Farsi (Farahani et al., 2021). However, other studies suggest that a multilingual model such as mBERT is a good choice for low-resourced languages (Chau et al., 2020; Rust et al., 2020; Wu and Dredze, 2020).

When using a multilingual LM for a downstream task in a particular language we need to consider whether the language is typologically and etymologically close to those represented in the LM pretraining data, and how much monolingual data in this language was included in the pretraining data (Lauscher et al., 2020). Therefore, models which are trained on typologically distant languages and do not contain monolingual data for a particular language may not be suitable for enabling cross-lingual transfer to

the low-resource language. At the same time, there may not be enough monolingual data to learn a model from scratch. Thus, we are left in a precarious position when performing an NLP task for such languages. In the next section, we train a BERT model from scratch in Irish and by doing so, try to add more clarity on what is the best approach for a moderately low-resource language such as Irish.

4.2 **gaBERT: an Irish Language Model**

As discussed in the previous section, a natural question that arises when approaching a downstream task for a particular language is whether to use a multilingual LM, which benefits from the larger amount of data from other languages, or a monolingual model trained on that language. While previous works have shown that dedicated monolingual LMs perform better than more general multilingual LMs for within-language downstream tasks (de Vries et al., 2019; Virtanen et al., 2019; Farahani et al., 2021), we extend this line of research and investigate whether this is also the case for Irish. Furthermore, we seek to build upon recent progress in data-driven Irish NLP (Lynn et al., 2012, 2015; Walsh et al., 2019; Cassidy et al., 2022), by releasing a monolingual LM **gaBERT**, which we envision will be useful for many downstream applications including grammar checking, dependency parsing, computer-aided language learning, predictive text, search and translation, or be used to bootstrap resources by enabling better initial annotations for a given task to be corrected by a human annotator. As a final benefit, we hope that the release of this model will contribute towards preserving Irish as a living language in the digital age and be an inspiration for developing resources for similar languages.

From a linguistic perspective, the Irish language is an inflected language, sharing linguistic features with other Celtic languages such as verb-subject-object (VSO) word order, initial mutation (lenition and eclipsis) and inflected prepositions. Inflection is common through suffixation, marking tense, number and person, while nouns are in-

flected for number and case. Nouns are either masculine or feminine in grammatical gender, which in turn influences declension-dependent inflections. Its inflected nature has already been shown to impact data-driven NLP tools due to data sparsity (Lynn et al., 2013), as has the frequent use of clefting (fronting), two forms of the verb ‘to be’ and prevalence of variable and discontinuous multiword expressions.

In the following sections, we describe the training data used for gaBERT and detail our hyperparameter search for our final model, where we consider the type of text filtering to apply, the vocabulary size, tokenisation model and Transformer architecture to use.

4.2.1 gaBERT Pretraining Data

Previous open-source models which can be used for processing Irish include mBERT (Devlin et al., 2019) and Wikibert-ga (Pyysalo et al., 2021). Both of these models have been trained on Irish Wikipedia. We suspect this may be a problem as the Irish portion of Wikipedia contains around 56,000 articles (which ranks in 93rd place based on the number of overall articles). This compares to about 6.5 and 2.7 million for English and German, respectively.² As such, Irish may be underrepresented in the training data for large multilingual LMs and we also expect that we can improve upon existing models by using a larger and more diverse collection of Irish corpora. With this in mind, we use the following resources to train gaBERT:

- **CoNLL17**: The Irish data from the CoNLL’17 raw text collection (Ginter et al., 2017) released as part of the 2017 CoNLL Shared Task on UD Parsing (Zeman et al., 2017). This corpus contains a mixture of Wikipedia and CommonCrawl data.
- **IMT**: A collection of Irish texts used in Irish machine translation research (Dowling et al., 2018, 2020), including legal text, general administration and data crawled

²Based on https://meta.wikimedia.org/wiki/List_of_Wikipedias as of 2022-03-09

Corpus	Num. Sents	Num. Tokens	Size (MB)
CoNLL17	1.7M	24.7M	138
IMT	1.4M	22.6M	124
NCI	1.6M	33.5M	174
OSCAR	0.8M	16.2M	89
ParaCrawl	3.1M	67.5M	380
Wikipedia	0.7M	6.8M	38
Overall	9.3M	171.3M	943

Table 4.1: Sentence and word counts and plain text file size in megabytes for each corpus after tokenisation and segmentation but before applying sentence filtering.

from public body websites.

- **NCI:** The New Corpus for Ireland (Kilgarriff et al., 2006), which contains a wide range of texts in Irish, including fiction, news reports, informative texts and official documents.
- **OSCAR:** The unshuffled Irish portion of the 2019 OSCAR corpus (Ortiz Suárez et al., 2019), a subset of CommonCrawl.
- **Paracrawl:** The Irish side of the ga-en bitext pair of ParaCrawl v7 (Bañón et al., 2020), which is a collection of parallel corpora crawled from multi-lingual websites.
- **Wikipedia:** Text from Irish Wikipedia.³

The sentence and word counts in each corpus are listed in Table 4.1 after tokenisation and segmentation but before filtering described below.

4.3 Corpus Processing

In this section, we apply corpus-specific pre-processing, sentence-segmentation and tokenisation, which are detailed in the below paragraphs.

³We use the articles from <https://dumps.wikimedia.org/gawiki/20210520/>

CoNLL17 The CoNLL17 corpus is already tokenised, as it is provided in CoNLL-U format, which we convert to one-sentence-per-line tokenised plain text.

IMT, OSCAR and ParaCrawl The text files from the IMT, OSCAR and ParaCrawl contain raw sentences requiring tokenisation. We describe the tokenisation process for these corpora in Section 4.3.1.

NCI The NCI corpus was provided to us by Foras na Gaeilge in the form of a .vert file containing 33,088,532 tokens in 3,485 documents. We extract the raw text from the first tab-separated column and carry out a number of rule-based conversions. For the types of heuristics used, we refer the reader to the original work (Barry et al., 2022).

Wikipedia For the Wikipedia articles, the Irish Wikipedia dump is downloaded and the WikiExtractor tool⁴ is then used to extract plain text. Article headers are included in the extracted text files. Once the articles have been converted to plain text, they are tokenised using the tokeniser described in Section 4.3.1.

4.3.1 Tokenisation and Segmentation

Raw texts from the IMT, OSCAR, ParaCrawl and Wikipedia corpora are tokenised and segmented with UDPipe (Straka and Straková, 2017). UDPipe’s segmenter is a character-level bidirectional GRU that simultaneously makes end-of-token and end-of-sentence predictions. We trained UDPipe on a combination of the Irish-IDT and English-EWT corpora from version 2.7 of the Universal Dependencies (UD) treebanks (Zeman et al., 2020). We include the English-EWT treebank in the training data to expose the tokeniser to more incidences of punctuation symbols which are prevalent in our pre-training data. This also comes with the benefit of supporting the tokenisation of code-mixed data. We upsample the Irish-IDT treebank by ten times to offset the larger English-EWT treebank

⁴<https://github.com/attardi/wikiextractor>

size. This tokeniser is applied to all corpora apart from the NCI, which is already tokenised by Kilgarriff et al. (2006), and the CoNLL17 corpus as this corpus is already tokenised in CoNLL-U format. After initial corpus pre-processing, all corpora are merged and we use the WikiBERT pipeline (Pyysalo et al., 2020) to create pretraining data.

4.4 Hyperparameter Search for the Final Model

We performed a search over a number of different configurations to obtain our final model. This includes experimenting with four corpus filtering settings, five vocabulary sizes, two tokenisation models and two Transformer models. The following sections detail these approaches.

4.4.1 Corpus Filtering

Previous works involving training language-specific LMs often involve filtering the corpus to ensure high-quality text is used to train the model, and that text in other languages and scripts is not included in the pretraining data (Virtanen et al., 2019; Pyysalo et al., 2021). The WikiBERT pipeline contains a number of filters which dictate whether a document should be kept. As we are working with data sources where there may not be clear document boundaries, or where there are no line breaks over a large number of sentences, document-level filtering may be inadequate for such texts. Consequently, we also experiment with using OpusFilter (Aulamo et al., 2020), which filters individual sentences, thereby giving us the flexibility of filtering noisy sentences while not discarding full documents.

For each filter setting below, we train a BERT model on the data which remains after filtering:

No-filter All collected texts are included in the pre-training data.

Document-filter The default document-level filtering used in the WikiBERT pipeline (Pyysalo et al., 2020). These include filters which ensure that the documents meet the following criteria: Documents must contain at least 3 sentences, a minimum of 20 whitespace ‘tokens’, a minimum of 30 whitespace ‘words’, the average length of sentences must be greater than 4, there must not be more than 10% uppercase words, the number of lines without words must not be more than 20%, the number of punctuation tokens must be below 7.5%, the number of digit tokens must be below 7.5%, and sentences must contain 98% Latin characters.

OpusFilter-basic OpusFilter (Aulamo et al., 2020) with the following filters:

- *LengthFilter*: Filter sentences containing more than 512 words.
- *LongWordFilter*: Filter sentences containing words longer than 40 characters.
- *HTMLTagFilter*: Filter sentences containing HTML tags.
- *PunctuationFilter*: Filter sentences which are over 60% punctuation.
- *DigitsFilter*: Filter sentences which are over 60% numeric symbols.

OpusFilter-basic-char-lang The same filters are used as **OpusFilter-basic** but with additional character script and language ID filters:

- *CharacterScoreFilter*: All alphabetic characters in a sentence must be in Latin script.
- *LanguageIDFilter*: The confidence scores from the language ID tools must be > 0.8 .
We use two language identification tools: `langid.py` (Lui and Baldwin, 2012) and CLD2.⁵

The overall number of sentences and words which remain after applying each filter are shown in Table 4.2. The *Document-filter* removes about 1.2 million sentences, while *OpusFilter-basic* only removes about 200,000 sentences. *OpusFilter-basic-char-lang*

⁵<https://github.com/CLD2owners/cld2>

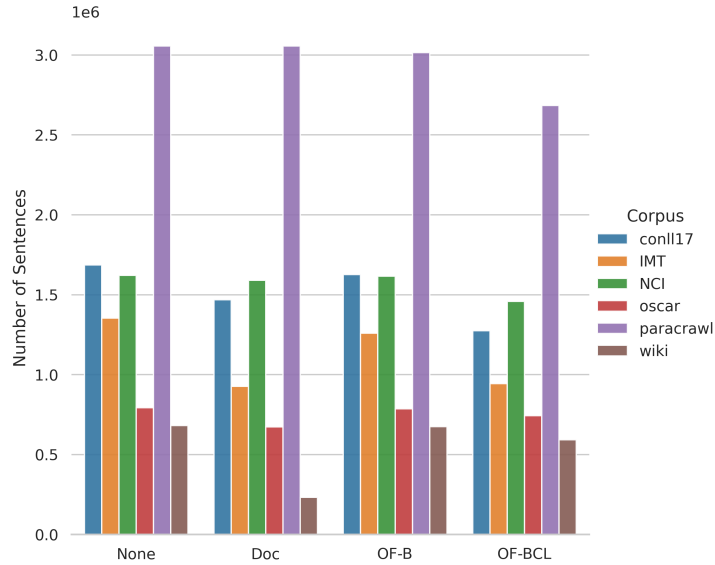


Figure 4.1: Number of sentences in each corpus per filtering configuration where None: No-filter, Doc: Document-filter, OF-B: OpusFilter-basic, OF-BCL: OpusFilter-basic-char-lang.

Filter	Sentences	Tokens
No-filter	9.2M	171.3M
Document-filter	7.9M	161.0M
OpusFilter-basic	9.0M	170.8M
OpusFilter-basic-char-lang	7.7M	161.2M

Table 4.2: The number of sentences and words which remain after applying the specific filter.

removes the most number of sentences. Figure 4.1 shows the breakdown of the number of sentences in each corpus for the specific filtering configuration. *Document-filter* keeps the full ParaCrawl data. This is due to the fact that there are no line breaks in this corpus and the higher proportion of clean data prevents the full document from being filtered. Many Wikipedia articles (where each article is considered as a separate document) are removed by this filter. *OpusFilter-basic* keeps a proportionally higher amount of Wikipedia sentences than the *Document-filter* setting. *OpusFilter-basic-char-lang*, which has additional language and character-script filters, removes a sizeable por-

tion of ParaCrawl sentences compared to the other filters.

4.4.2 Vocabulary Creation

To create a model vocabulary, we experiment with the SentencePiece (Kudo and Richardson, 2018) implementation of byte-pair encoding (Gage, 1994) and WordPiece tokenisers.⁶ Using the model with highest median LAS from the filtering experiments, we try vocabulary sizes of 15K, 20K, 30K, 40K and 50K. We then train a WordPiece tokeniser, keeping the vocabulary size that works best for the SentencePiece tokeniser. We also train a BERT model using the union of the two vocabularies.

4.4.3 BERT Training

We use the original BERT implementation of Devlin et al. (2019). For the development experiments, we train our BERT model for 500K steps with a sequence length of 128. We use whole word masking and the default hyperparameters and model architecture of BERT_{BASE}.⁷ Training for development runs of gaBERT took just under 48 hours on GPU. While a seed for data preparation can be set (we do not change the default 12345), the BERT implementation does not provide an option to set a seed for model initialisation and we did not find code that sets a seed for pretraining internally, suggesting initialisation is non-deterministic.

For the final gaBERT model, we use the best-performing filtering and vocabulary settings which we will describe in Section 4.4.5. We train this model for 900k steps with sequence length 128 and a further 100k steps with sequence length 512. We train on a TPU-v2-8 with 128GB of memory on Google Compute Engine⁸ and use a batch size of 128. Training gaBERT on TPU for 1M steps took around 37.5 hours.

⁶As BERT expects WordPiece tokenisation, a heuristic tool is used to map the SentencePiece vocabulary to WordPiece (<https://github.com/spyysalo/sent2wordpiece>).

⁷We use a lower batch size of 32 in order to train on NVIDIA RTX 6000 GPUs with 24 GB RAM.

⁸TPU access was kindly provided to us through the Google Research TPU Research Cloud.

4.4.4 Evaluation Measures

In this section we describe the various measures used to evaluate our gaBERT model. This includes carrying out an extrinsic evaluation on dependency parsing and MWE identification, as well as an intrinsic evaluation on the MLM outputs of the considered models.

Dependency Parsing The evaluation measure we use to make development decisions is dependency parsing LAS. To obtain this measure, we fine-tune a given BERT model in the task of dependency parsing and measure LAS on the development set of the Irish-IDT treebank in version 2.8 of UD. We report the median of five fine-tuning runs with different random initialisation. For the dependency parser, we use a multitask model which uses a graph-based parser with biaffine attention (Dozat and Manning, 2017) (Section 2.1.2, p. 39) as well as additional classifiers for predicting POS tags and morphological features. Model hyperparameters are given in Table 4.3. For the tagging tasks, the output of the Transformer is first projected through a task-specific Feedforward network and then passed to a classification layer. For dependency parsing, the projected representations from the tagging modules are concatenated to the output of the Transformer before being passed to the parsing module. We use the AllenNLP (Gardner et al., 2018) library to develop our multitask model.

Cloze Test To compile a cloze task test set, 100 strings of Irish text (4–77 words each) containing the pronouns ‘é’ (‘him/it’), ‘í’ (‘her/it’) or ‘iad’ (‘them’) are selected from Irish corpora and online publications. One of these pronouns is masked in each string for the cloze test.⁹

Following Rönnqvist et al. (2019), the models are evaluated on their ability to generate the original masked token, and a manual evaluation of the models is also performed

⁹All the masked tokens exist in the vocabularies of the candidate BERT models and are therefore possible predictions.

Multitask Parser Details	
Encoder	
Word-piece embedding size	768
Word-piece type	average
Dropout	0.33
Tagger (UPOS/XPOS/Feats)	
MLP size	200
Dropout MLP	0.33
Nonlinear act. (MLP)	ELU
Parser	
Arc MLP size	500
Label MLP size	100
Dropout MLP	0.33
Nonlinear act. (MLP)	ELU
Optimiser and Training Details	
Optimizer	AdamW
Learning rate	3e-4
beta1	0.9
beta2	0.999
Num. epochs	50
Patience	10
Batch size	16

Table 4.3: Chosen hyperparameters for the multitask parser and tagger.

wherein predictions are classified into the following exclusive categories:

- **Match:** The predicted token fits the context grammatically and semantically. This may occur when the model predicts the original token or another token which also fits the context.
- **Mismatch** The predicted token is a valid Irish word but is unsuitable given the context.
- **Copy** The predicted token is an implausible repetition of another token in the context.
- **Gibberish** The predicted token is not a valid Irish word. This might occur in the

form of a subword or sequence of punctuation not forming a meaningful word.

MWE Identification task Multiword expressions (MWEs) pose a challenge in many tasks in NLP, including parsing. The task of automatically identifying MWEs has been explored in the series of shared tasks organised by the PARSEME network (Savary et al., 2017), focusing on verbal MWEs, i.e. MWEs headed by a verb, as they pose a particular challenge in terms of automatic identification.

We design an experiment to compare the results of fine-tuning both a gaBERT model and an off-the-shelf mBERT model for the task of identifying MWEs in Irish text. We used the Irish portion of the PARSEME 1.2 shared task data (Walsh et al., 2020), which has been manually annotated with six types of verbal MWEs. The annotations were converted to a modified version of BIO tagging, based on the work of Schneider et al. (2014), and a linear layer for token classification was added for the task of identifying the correct label for each word.

4.4.5 Results

Development Results

This section reports the results of our parameter search for corpus filtering and BERT vocabulary.

Filter Settings The results of training a dependency parser with the gaBERT model produced by each setting are shown in Fig. 4.2. *Document-Filter* has the highest LAS score. As the BERT model requires contiguous text for its next-sentence-prediction task, filtering out full documents may be more appropriate than filtering individual sentences. The two *OpusFilter* configurations perform marginally worse than the *Document-Filter*. In the case of *OpusFilter-basic-char-lang*, the additional character script and language ID filters did not lead to a noticeable change in LAS. Finally, *No-Filter* performs

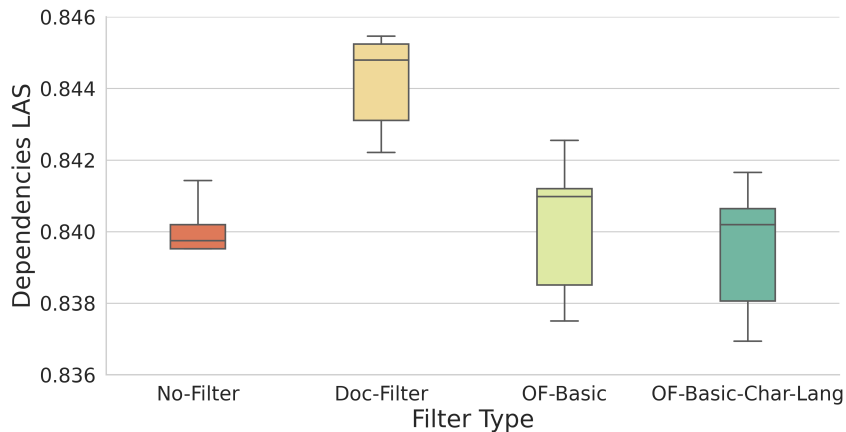


Figure 4.2: Dependency parsing LAS for each filter type. Each box-plot shows five LAS scores obtained by fine-tuning the respective BERT model five times with different initialisation to obtain five different parsers.

in the same range as the two *OpusFilter* configurations but has the lowest median score, suggesting that some level of filtering is beneficial. For all further experiments, we use the version of the training data filtered by *Document-Filter*.

Vocabulary Settings The results of the five runs testing different vocabulary sizes are shown in Fig. 4.3. A vocabulary size of 30K performs best for the SentencePiece tokeniser, which outperforms the WordPiece tokeniser with the same vocabulary size. The union of the two vocabularies results in 32,314 entries, and does not perform as well as the two vocabularies on their own. A manual inspection of the two vocabularies showed that the WordPiece tokeniser created more entries consisting of foreign characters and emojis at the expense of Irish words/word-pieces, which may account for the lower performance of settings using this tokeniser.

To create the final gaBERT model, we use the corpus filtered with Document Filter and the vocabulary produced by the SentencePiece tokeniser with a vocabulary size of 30k. The training process for the released model is described in Section 4.4.3.

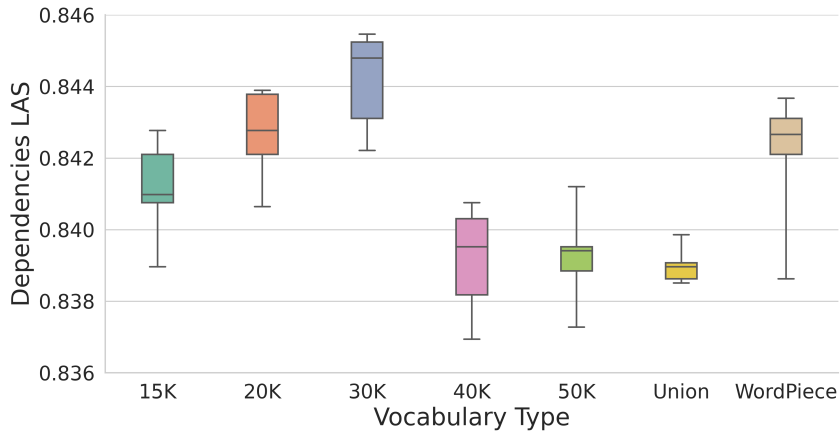


Figure 4.3: Dependency parsing LAS for each vocabulary type. Each box-plot shows five LAS scores obtained by fine-tuning the respective BERT model five times with different initialisation to obtain five different parsers.

4.4.6 Model Comparison

We compare our final gaBERT model with off-the-shelf mBERT and the monolingual Irish WikiBERT-ga model, as well as an mBERT model obtained with continued pre-training on our corpora (mBERT-cp).

Dependency Parsing Table 4.4 shows the results for dependency parsing. The first row (No BERT) is a baseline which does not use a pretrained BERT model but uses a BiLSTM encoder operating over token and character-level features instead. Using mBERT off-the-shelf results in a test set LAS of 80.3, an absolute improvement of 8.9 points over the baseline. The WikiBERT-ga model performs slightly better than mBERT. By training mBERT for more steps on our corpora,¹⁰ LAS can be improved by 2 points. Our gaBERT model has the highest LAS of 84.

The last two rows compare gaBERT on v2.5 of the treebank with the system of Chau et al. (2020), who augment the mBERT vocabulary with the 99 most frequent Irish tokens and fine-tune on Irish Wikipedia. The results are lower for both settings due to the

¹⁰We trained for an additional 600k steps with a sequence length of 128.

Model	LAS		
	UD	Dev	Test
No BERT	2.8	73.4	71.4
mBERT	2.8	81.8	80.3
WikiBERT	2.8	81.9	80.4
mBERT-cp	2.8	84.3	82.3
gaBERT	2.8	85.6	84.0
Chau et al. (2020)	2.5	-	76.2
gaBERT	2.5	-	77.5

Table 4.4: LAS in dependency parsing (UD v2.8) for selected models. Median of five fine-tuning runs. Scores are calculated using the official UD evaluation script (*conll18_ud_eval.py*).

Model	Original Token Prediction
mBERT	16
WikiBERT	53
mBERT-cp	78
gaBERT	75

Table 4.5: The number of times the original masked token was predicted (100 test items).

fewer number of trees in v2.5 of the treebank¹¹ and a manual effort to clean up some inconsistent annotations (McGuinness et al., 2020). Our model outperforms this approach (77.5 vs 76.2 LAS), likely due to our inclusion of a wider variety of corpora as well as our dedicated Irish vocabulary.

Cloze Test Table 4.5 shows the accuracy of each model with regard to predicting the original masked token. mBERT-cp is the most accurate and gaBERT is close behind. Table 4.6 shows the manual evaluation of the tokens generated by each model, accounting for plausible answers deviating from the original token and separately reporting copying of content and production of gibberish. These results echo those of the original

¹¹v2.5 has only 858 trees compared to the 4,005 in v2.8.

Model	Match	Mism.	Copy	Gib
mBERT	41	42	4	13
WikiBERT	62	31	1	6
mBERT-cp	85	12	1	2
gaBERT	83	14	2	1

Table 4.6: The number of matches, mismatches, copies and gibberish predicted by each model (100 test items).

Context Cue	Masked Word	Model	Prediction	Classification
<i>Céard [MASK] na préamhacha raidiciúla sin?</i> 'What [MASK] those radical roots?'	<i>iad</i> 'them'	mBERT-cp	<i>faoi</i> 'about'	match
<i>Agus seo [MASK] an fhadhb mhór leis an bhfógra seo.</i> 'And this [MASK] the big problem with this advert.'	<i>í</i> 'it' (fem.)	WikiBERT	<i>thaitin</i> 'liked'	mismatch
<i>Cheannaigh Seán leabhar agus léigh sé [MASK].</i> 'Seán bought a book and he read [MASK].'	<i>é</i> 'it' (masc.)	gaBERT	<i>leabhar</i> 'a book'	copy
<i>Ní h[MASK] sin aidhm an chláir.</i> '[MASK] is not the aim of the programme.'	<i>##é</i> 'it'(masc.)	mBERT	- minus sign	gibberish

Table 4.7: Examples of cloze test predictions and classifications.

masked token prediction evaluation in so far as they rank the models in the same order.

Table 4.7 provides one example per classification category of masked token predictions generated by the language models during our cloze test evaluation. In the *match* example in Table 4.7, the original meaning ('What are those radical roots?') differs to the meaning of the resulting string ('What about those radical roots?') in which the masked token is replaced by the prediction of mBERT-cp. However, the latter construction is grammatically and semantically acceptable. In the *mismatch* example in Table 4.7, the predicted token is a valid Irish word, but the resulting generated text is nonsensical. Though technically grammatical, the predicted token in the *copy* example in Table 4.7

Model	Short	Medium	Long
mBERT	20.69%	55.56%	41.67%
wikibert	51.72%	58.33%	74.29%
mBERT-cp	75.86%	83.33%	94.29%
gaBERT	79.31%	83.33%	85.71%
gaELECTRA	79.31%	77.78%	88.57%

Table 4.8: Accuracy of language models segmented by length of context cue where short: 4–10 tokens, medium: 11–20 tokens, and long: 21–77 tokens.

results in a string with an unnatural repetition of a noun phrase where a pronoun would be highly preferable (‘Seán bought a book and he read a book.’). In the *gibberish* example in Table 4.7, the predicted token does not form a valid Irish word and the resulting sentence is ungrammatical.

In order to observe the effect that the amount of context provided has on the accuracy of the model, Table 4.8 shows the proportion of matches achieved by each language model when the results are segmented by the length of the context cues. All the models tested are least accurate when tested on the group of short context cues. All except mBERT achieved the highest accuracy on the group of long sentences. This is likely due to the fact that mBERT is exposed to the least amount of Irish data, and has not been further pretrained on Irish data.

A context cue may be considered easy or difficult based on:

- Whether the tokens occur frequently in the training data
- The number of grammatical markers
- The distance of the grammatical markers from the masked token

Two Irish language context cues which vary in terms of difficulty are exemplified in (Examples 4.1 and 4.2).

(4.1) *Bean, agus í cromtha thar thralaí bia agus [MASK] ag ithe a sáithe.*

‘A woman, bent over a food trolley while eating her fill.’

We can consider Example 4.1 to be easy for the task of token prediction due to the following grammatical markers:

- ‘Bean’ is a frequent feminine singular noun.
- ‘í’ is a repetition of the feminine singular pronoun to be predicted.
- The lack of lenition on ‘sáithe’ further indicates that the noun it refers to may not be masculine.

These grammatical markers indicate that the missing pronoun will be feminine and singular. None of the language models tested predicted a plausible token for Example 4.2.

(4.2) *Seo béile aoibhinn fuirist nach dtógann ach timpeall leathuair a chloig chun [MASK] a ullmhú.*

‘This is an easy, delicious meal that only takes about half an hour to prepare.’

This example may be more challenging as the only grammatical marker is the feminine singular noun ‘béile’ which is 11 tokens in distance from the masked token.

MWE Identification MWE identification is a difficult task, and according to the system results of the most recent edition of the PARSEME shared task,¹² it appears to be particularly challenging for Irish, with the majority of systems performing most poorly on the Irish dataset. This may be due to the smaller size of the data, coupled with the relatively high number of MWE labels to classify (Walsh et al., 2020). We attempt a series of fine-tuning experiments varying the learning rate, batch size and initial random seed, and found model performance is sensitive to changes in hyperparameters. Figure

¹²Full results: <http://multiword.sourceforge.net/sharedtaskresults2020/>

Model	P	R	F1
mBERT	0.342	0.245	0.285
gaBERT	0.523	0.361	0.427

Table 4.9: Verbal MWE Identification: (P)recision, (R)ecall and F1 scores of the best performing gaBERT and mBERT model

4.4 shows the results of training twenty models with different random seed values. It is evident that gaBERT outperforms mBERT in precision, recall and F1 scores on the test set.

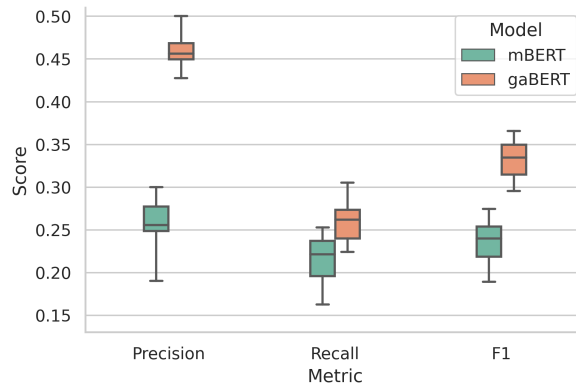


Figure 4.4: Verbal MWE Identification: Precision, Recall and F1 scores for each model across 20 random seed values

Table 4.9 records the Precision (P), Recall (R), and F1 scores for the best-performing gaBERT and mBERT model found during the manual tuning of hyperparameters. gaBERT performs better using these optimised parameters, particularly for precision scores, indicating that the gaBERT model tends to be more correct when classifying MWEs than the mBERT model.

In comparison to other systems submitted to the PARSEME shared task on the Irish data, both models perform well. The best-performing model for MWE identification had an F1 score of 0.306, which the gaBERT model exceeds by 0.121. On a multilingual

level, the averaged F1 score for overall MWE identification of the highest-ranking system was 0.701, and even with the improved F1 score of the best-performing gaBERT model, results for Irish are still below the best system for Hebrew (0.483), which was the language where systems had the second-lowest performance.

4.4.7 ELECTRA Model

In addition to the gaBERT model, we also experiment with using an ELECTRA model (Clark et al., 2020) **gaELECTRA**, trained on the same data as gaBERT. ELECTRA replaces the MLM pre-training objective of BERT with a binary classification task discriminating between authentic tokens and alternative tokens generated by a smaller model for higher training efficiency. We use the default settings of the “Base” configuration of the official implementation¹³ and train on a TPU-v3-8. As with BERT, we train for 1M steps and evaluate every 100k steps. However, we train on more data per step as the batch size is increased from 128 to 256 and a sequence length of 512 is used throughout.

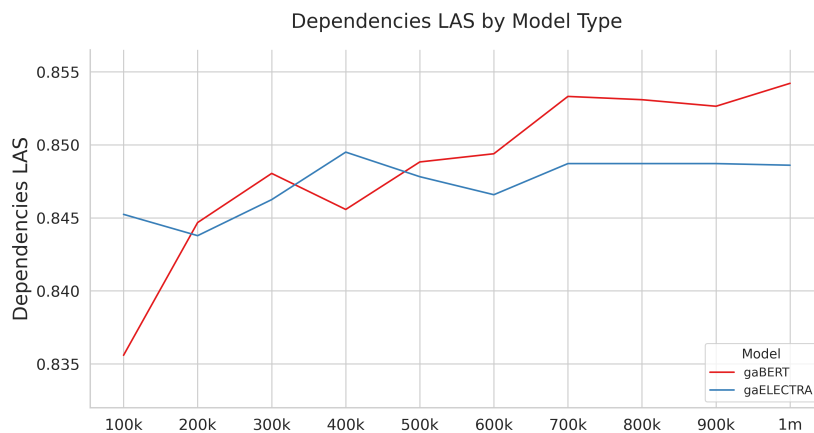


Figure 4.5: Dependency parsing LAS for each model type. Every 100k steps, we show the median of five LAS scores obtained from fine-tuning the respective model five times with different initialisation.

Figure 4.5 shows the development LAS of gaELECTRA and gaBERT for each check-

¹³<https://github.com/google-research/electra>

point. The best gaBERT checkpoint is reached at step 1 million, which may indicate that there are still gains to be made from training for more steps. The highest median LAS for gaELECTRA is reached at step 400k, and we use this checkpoint for all further experiments. It is worth noting that although the two models are compared at the same number of steps, the different pretraining hyperparameters mean they are not trained on the same number of tokens per step.

4.4.8 Full Model Results

This section examines the results produced by each of our models in more detail and also presents the scores of two additional models we examine: The first is another multilingual baseline in XLM-R_{BASE} (Conneau et al., 2020) and the second is the gaELECTRA model of the previous section. Tables 4.10 and 4.11 list the accuracies for predicting universal part of speech (UPOS), treebank-specific part of speech (XPOS) and morphological features, as well as the unlabelled and labelled attachment score (UAS and LAS, respectively) for all models. It should be noted that each row selects the model based on median LAS, so all other metrics are those that this selected model achieved.

In Table 4.11, starting with the multilingual models, mBERT performs worse than XLM-R_{BASE} (80.3 vs. 82.5 LAS), which is a strong multilingual baseline. We expect that the more diverse crawled data found in the XLM-R pretraining data makes it more competitive than mBERT. The XLM-R model is also able to outperform the monolingual WikiBERT model which has an LAS of 80.4. The continued pretraining of mBERT on our data enables us to close the gap between the XLM-R_{BASE} model (82.3 vs. 82.5 LAS). gaBERT is the strongest model for all metrics in terms of test set scores and has an LAS of 84.0. gaELECTRA performs slightly below that of gaBERT at 83.1 LAS but better than the other models. As with gaBERT, this is likely due to the use of a dedicated Irish vocabulary which is absent in the multilingual models, and being exposed to more diverse data than Irish Wikipedia in the case of WikiBERT.

Model	UD	UPOS	XPOS	FEATS	UAS	LAS
mbert-os	2.8	95.7	94.7	89.2	86.9	81.8
xlmr-base-os	2.8	96.4	95.1	90.6	88.3	84.0
wikibert-os	2.8	95.9	94.9	89.4	86.8	81.9
mbert-cp	2.8	97.2	95.8	92.3	88.1	84.3
gabert	2.8	97.1	96.2	93.1	89.2	85.6
gaelectra	2.8	97.3	96.1	92.8	89.1	85.3

Table 4.10: Full model results on development data. For model name abbreviations, see Table 4.11.

Model	UD	UPOS	XPOS	FEATS	UAS	LAS
mbert-os	2.8	95.4	94.3	88.6	86.2	80.3
xlmr-base-os	2.8	96.1	95.1	90.0	87.7	82.5
wikibert-os	2.8	95.7	94.4	88.3	85.9	80.4
mbert-cp	2.8	96.7	95.5	91.7	87.1	82.3
gabert	2.8	97.0	95.7	91.8	88.4	84.0
gaelectra	2.8	96.9	95.5	91.5	87.6	83.1

Table 4.11: Full model results on test data (os = fine-tuned off-the-shelf model, cp = continued pre-training before fine-tuning).

4.5 Summary

We release gaBERT, a BERT model trained on over 7.9M Irish sentences, combining Irish language text from a variety of sources, and evaluate it in dependency parsing, a pronoun cloze test task, and a MWE identification task, showing improvements over three baselines: mBERT, WikiBERT-ga and XML-R_{BASE}. We also experimented with an alternative architecture, ELECTRA, but found this architecture to perform slightly worse than BERT. We showed that multilingual LMs can be further improved by performing continued pretraining on monolingual data. Despite doing so, the dedicated Irish vocabulary in the gaBERT and gaELECTRA models was likely a determining factor in these models outperforming this approach.

The purpose of this chapter was to answer RQ2: *Does a monolingual language model improve low-resource dependency parsing in the case of Irish?* We can conclude by answer-

ing this research question positively: that a monolingual LM trained from scratch in Irish does provide better representations for dependency parsing. Further, we showed that this is in spite of the fact that the data used to train gaBERT was far less than that seen in the training data of the multilingual LMs and other monolingual language models. In total, the gaBERT training data is just over 161 million tokens (less than a gigabyte). In comparison, the English BERT model (Devlin et al., 2019) uses 3.3 billion tokens, the Nordic BERT Danish model¹⁴ uses 1.6 billion tokens, Finnish BERT (Virtanen et al., 2019) uses 3.3 billion tokens, and the French CamemBERT Model (Martin et al., 2020) uses 138 GB of text data, which is over 138 times the amount used to train gaBERT. This shows that collecting a comparably smaller amount of monolingual data and training a model from scratch is worthwhile. In concurrent work, Armengol-Estapé et al. (2021) showed that for another moderately low-resource language in Catalan, their monolingual RoBERTa model also outperformed the baselines of mBERT, WikiBERT-ca (Pyysalo et al., 2021) and XLM-R (Conneau et al., 2020)—the same baselines used in our experiments—across a number of downstream tasks.

In the next chapter, we will turn our attention to the task of parsing EUD graphs, which are an extension of basic UD trees designed to be more useful for shallow natural language understanding tasks due to their better-handling of certain relationships between pairs of content words.

¹⁴https://github.com/certainlyio/nordic_bert

Chapter 5

Parsing Enhanced Universal Dependency Graphs

In the previous chapters, the experiments focused on parsing basic UD trees, whereas in this chapter, we will approach the problem of parsing Enhanced Universal Dependencies (EUD) graphs. EUD is a recent framework which extends on the basic UD tree representation to permit more flexible graph structures in the UD format when compared with strict surface structure dependency trees. Due to their ability to better represent relationships between certain content words, EUD graphs tend to be more useful in shallow natural language understanding tasks (Schuster and Manning, 2016).

In particular, this chapter describes the contribution of the *DCU-EPFL* system (Barry et al., 2021) to the IWPT 2021 shared task on parsing into Enhanced Universal Dependencies (Bouma et al., 2021), which requires participants to parse EUD graphs in 17 languages, starting from raw strings. This differs from the previous CoNLL shared tasks on UD parsing (Zeman et al., 2017, 2018) which focused on predicting syntactic trees. Our main contribution to the task is the introduction of a novel multitask model which jointly predicts basic UD trees and EUD graphs. Due to time constraints, we only submitted our baseline single task graph parser before the official deadline. This system

placed 6th out of 9 participants with a macro-average ELAS¹ score of 83.57 (the winning system of Shi and Lee (2021) is 89.24 by comparison). We then carry out a number of follow-up experiments which help in answering *RQ3: How can we leverage existing techniques to parse Enhanced Universal Dependencies?* Here, we consider the roles of using better upstream preprocessing, a larger pretrained encoding model, monolingual and cross-lingual treebank concatenation, and the use of multitask learning between a tree and a graph parser. We empirically show how our original system can be improved upon by leveraging these approaches, and when incorporating all improvements, our ELAS increases from 83.57 to 88.04, which would have been the second-highest score among all participants. This system also achieves the highest scores for Russian and Italian.

5.1 Background

Despite the recent wave of Deep Learning models and accompanying analyses that show that such models learn information about syntax inherently, certain works have shown that adding further syntactic biases, to pretrained LMs for example, can improve downstream performance (Kuncoro et al., 2020; Mohammadshahi and Henderson, 2020). Such findings speak to the need of utilising hierarchically structured representations such as trees and semantic representations to provide greater supervision about what is taking place in a sentence (Oepen et al., 2019). While dependency trees have typically been used in downstream applications, their structural restrictions may hinder the representation of content words (Schuster and Manning, 2016). As such, there is growing interest in using more general graphs to provide a sentence-level analysis of text, where these graphs are more capable of representing semantic structure than syntactic trees (Oepen et al., 2019).

The Enhanced UD representation aims to fill this gap by enabling more expressive

¹ELAS score corresponds to correctly predicting the labelled enhanced annotations.

graphs in the UD format. EUD graphs capture phenomena such as:

- added subject relations in control and raising constructions (Figure 5.1)
- shared heads and dependents in coordination (Figure 5.2)
- the insertion of null nodes for elided predicates (Figure 5.3)
- co-reference in relative clause constructions (Figure 5.4)
- augmenting modifier relations with prepositional or case-marking information (Figure 5.5)

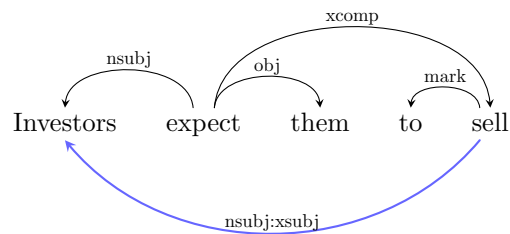


Figure 5.1: *Added subject relations in control and raising*: there is an added subject dependency between an embedded verb *sell* and its raised subject *Investors*. The extension “:xsubj” (marked in blue) is added in these cases.

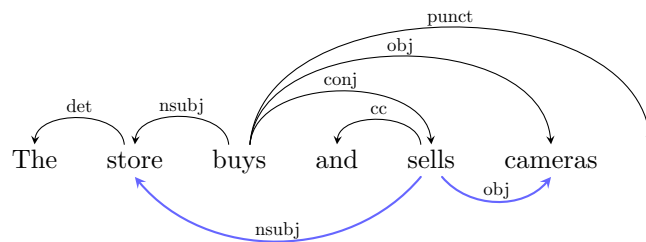


Figure 5.2: *Shared heads and dependents in coordination*: in a conjoined phrase (e. g. *buys and sells*), each predicate shares the subject (the store) and the object (cameras). The additional edges are marked in blue.

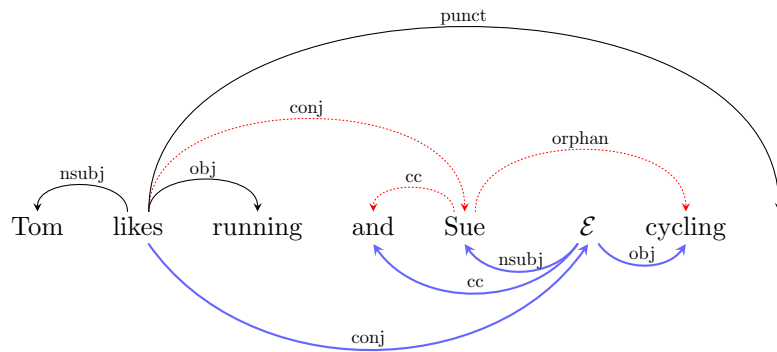


Figure 5.3: *Insertion of null nodes for elided predicates*: ϵ denotes an elided token which is added to the enhanced graph. The additional edges in the EUD graph are marked in blue and removed tree edges are marked with a red dotted arc.

In basic UD trees, these phenomena are expressed implicitly, which can result in there being long paths between related content words, making the resulting analysis less clear (Dehouck et al., 2020). EUD graphs, on the other hand, enable a more flexible analysis which can handle these phenomena explicitly. This is exemplified in some of the figures above, e. g. where both subjects and objects are linked to each predicate in a conjoined phrase (Figure 5.2) or when the antecedent of a relative pronoun is linked to the main predicate of the relative clause (Figure 5.4).

EUD graphs were first created by Schuster and Manning (2016) who build on the Stanford Dependencies (SD) initiative (de Marneffe et al., 2006) and extend certain flavours of the SD dependency graph representations to UD in the form of enhanced UD relations for English. They use a rule-based system that converts basic English UD trees to enhanced UD graphs based on certain dependency structures identified to require enhancement. Nivre et al. (2018b) compare rule-based and data-driven approaches in a cross-lingual setting for bootstrapping EUD representations in Swedish and Italian and show that both techniques are capable of annotating enhanced dependencies in different languages. Bouma et al. (2020) introduced the first shared task on parsing EUD

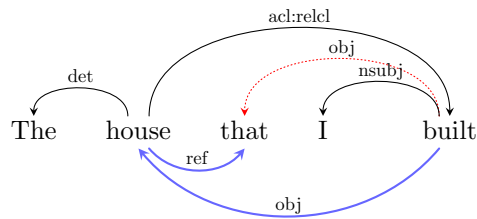


Figure 5.4: *Co-reference in relative clause constructions*: a special *ref* relation is added to the relative pronoun from its antecedent, and the antecedent is attached as an argument to the main predicate of the relative clause. The additional edges in the EUD graph are marked in blue and removed edges are marked with a red dotted arc. Note that the EUD graph contains a cycle between *house* and *built*.

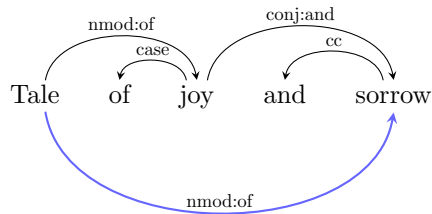


Figure 5.5: *Augmenting modifier relations with prepositional or case-marking*: each conjunct in this example conjoined noun phrase is attached to the governor of the modifier phrase, e. g. there is an additional *nmod* relation marked in blue between the noun *Tale* and the second conjunct *sorrow*. Note that the lemma of the *case* and *cc* dependents are appended to the enhanced dependency labels of their heads.

graphs which brought renewed attention to this research area. This was followed by the 2021 version of the task (Bouma et al., 2021). We review some of the recent approaches used in both shared tasks. We use the categories defined by Shi (2021) to differentiate between the various approaches:

Tree-based Tree-based systems consist of using rule-based approaches to convert UD trees into EUD graphs. These systems rely on the fact that certain information from syntactic trees can be automatically converted to the enhanced format by following a set of rules. Among these approaches include Dehouck et al. (2020) and Ek and Bernardy

(2020). Dehouck et al. (2020) introduce an algorithm that iteratively produces the enhanced representation for a given sentence. Their rule-based system performs well on gold data, achieving an average ELAS of 98.20 on gold development treebanks. This means that the performance of a rule-based system generally depends on the quality of the basic UD annotation, provided the rules cover most of the enhancements. Analysis of their rule-based system showed that there exist some annotation inconsistencies in certain EUD treebanks, hampering their system to an extent.

A similar finding was also made by Guillaume and Perrier (2021), who convert UD annotations into EUD annotations using a Graph Rewriting System with a set of six rules which also have language-specific modifications. Guillaume and Perrier (2021) state that the gold data contains annotations that are not described in the guidelines and consequently, learning-based approaches may have an advantage over rule-based approaches as they can adapt to the trends seen in the data. Nevertheless, rule-based approaches can be used to point out where erroneous annotation may need to be revisited and can be used to annotate new treebanks in situations where learning based approaches might not work well, e. g. in low-resource scenarios. The winning system in official scores for the 2020 version of the task (Kanerva et al., 2020) developed a method for encoding the enhanced representation into the basic annotation so it can be predicted using a standard dependency tree parser. Specifically, they identify four patterns which can be concatenated to the un-modified basic relation: whether the enhanced dependency goes *from* or *to* the head in the basic tree, or *from* or *to* the head of the head in the basic tree. They found that these four patterns can be encoded and decoded on using gold data in an almost lossless fashion (with scores ranging from 97.9 - 99% over the included languages).

Graph-based Graph-based systems predict the output graph directly, typically using a semantic dependency parser. The main difference between a standard dependency

parser and a semantic dependency parser is the use of sigmoid activations on the output of the edge prediction module (Dozat and Manning, 2018). Here, the edge scores are passed through a sigmoid function which assigns probability scores to all other words in the sentence being a head of the current word, and all tokens with a probability above a predefined threshold are included as edges. This is in contrast to dependency parsing, which typically uses a softmax activation on the scores. The choice of sigmoid activations over softmax activations is because in graph parsing, words are not restricted to having just one head.

A benefit of graph-based systems is that they can parse arbitrary graphs, e. g. graphs where words may have numerous heads and contain cycles. However, as outlined by Shi (2021), usually graph-based systems require a post-processing step to make sure the produced graphs are connected and reachable from the root token. Among these approaches include Barry et al. (2020); Grünewald and Friedrich (2020); Wang et al. (2020a). The system of Wang et al. (2020a) uses second order features which model coparent, sibling and grandparent relations and use Mean-Field Variational Inference during decoding. This system had the highest scores in the 2020 version of the task after fixing a validation issue concerning unconnected graphs.

Transition-based The only transition-based system among the participating teams across both tasks was that of Hershovich et al. (2020), who adapt a transition-based system for parsing semantic graphs to parse EUD graphs. The transition-based system they use is the the stack-LSTM architecture of Dyer et al. (2015) and they introduce an extra transition to predict elided tokens (e. g. inserting the elided token into the sentence as in Figure 5.3).

Tree-Graph Integrated Tree-Graph Integrated systems incorporate a tree-parsing component and a graph-parsing component. Here, a standard dependency parser can be used to first produce the tree using standard global inference methods. Then, a second

more general graph parsing component can be used to add additional edges. Among these systems include He and Choi (2020) and Shi and Lee (2021). An advantage of Tree-Graph Integrated systems is that by first producing the dependency tree, the output graph is guaranteed to be connected, which is a requirement of EUD graphs. This is because global inference methods such as the CLE algorithm guarantee a connected tree.

The multitask tree and EUD parser which is introduced in Barry et al. (2021) and described in this chapter is similar to these types of approaches in that it incorporates a dependency tree and graph parser. However, in the aforementioned works, these components are learned separately, and their predictions are later merged. Another difference between our model and these systems is that, despite our multitask parser predicting the basic tree and the enhanced graph simultaneously, the output of the multitask parser is that predicted by the EUD graph parser, which is trained on EUD annotations. In this way, the basic tree parser is used as an auxiliary task to supplement the graph parsing task, but its predictions are not taken into consideration when outputting the final EUD graph.

5.2 The DCU-EPFL Enhanced Dependency Parser at the IWPT 2021 Shared Task

We now describe the *DCU-EPFL* submission to the 2021 shared task (Bouma et al., 2021). The *DCU-EPFL* system relies on a single multilingual Transformer encoder, namely XLM-RoBERTa (XLM-R) (Conneau et al., 2020), which is a multilingual RoBERTa model (Liu et al., 2019b), to obtain contextualised token representations. In the context of dependency parsing, Grunewald et al. (2021) have shown that this model achieves state-of-the-art performance, where the large variant of this model often outperforms monolingual pretrained language models which usually are only released in base sizes. Also,

as we have seen from the experiments in Chapter 4, XLM-R was a stronger model than mBERT. The contextual representations from XLM-R are passed to the enhanced dependency parsing model which can produce arbitrary graphs, including graph structures where words may have multiple heads as well as cyclic graphs. The system is language-generic and is straightforward to apply to new languages with EUD annotations.

As participants are required to produce enhanced graphs from raw text, we take a pipeline approach which involves the following three components:

1. Stanza (Qi et al., 2020) is used for sentence segmentation, tokenisation and the prediction of all UD annotations apart from the enhanced graph.
2. A Transformer-based graph parsing model is used to predict EUD graphs.
3. A post-processor ensures that every predicted graph is a connected graph where all nodes are reachable from the notional root token.

Our official system placed 6th out of 9 teams with a coarse Enhanced Labeled Attachment Score (ELAS) of 83.57. In unofficial post-evaluation experiments, we make four incremental changes to our pipeline approach which are listed below:

1. We replace the Stanza pre-processing pipeline with Trankit (Nguyen et al., 2021) which is an NLP pipeline which uses XLM-R to obtain contextualised features for all preprocessing components, which include sentence segmentation, tokenisation, multiword token expansion, POS taggers as well as a dependency parser.
2. We replace XLM-R_{Base} with XLM-R_{Large}, which has roughly twice as many parameters as the base model.
3. We concatenate treebanks for the languages which have more than one training treebank and also concatenate English treebanks with the Tamil training data, where Tamil is the treebank with the least amount of training sentences.

4. We introduce a novel multitask model which parses the basic UD tree and enhanced graph in tandem.

All of these additional steps improved our evaluation scores, and for our final system, which incorporates all of these modifications, our macro-average ELAS score increases from 83.57 to 88.04, which would have placed second out of all systems.

Official System Submission We first describe our official system, which is the system we submitted prior to the competition deadline. For the official system, we did not include the basic dependency parser in a multitask setup, i. e. we just used the EUD graph parsing head (shown on the left in Figure 5.6). The raw text test files for each language contain a mixture of test data covering multiple treebanks, so participants do not know their exact domain. For our official system, we choose the model trained on the treebank with the most amount of training data in terms of sentences for each language to process the test files. This heuristic corresponds to using Czech-PDT for Czech, Dutch-Alpino for Dutch, English-EWT for English, Estonian-EDT for Estonian and Polish-PDB for Polish.

Pre-processing For sentence segmentation, tokenisation and the prediction of the base UD features (all UD features apart from the enhanced dependency graphs and miscellaneous items in CoNLL-U files), we use the Stanza library (Qi et al., 2020) trained on version 2.7 of the UD treebanks for each treebank released as part of the training data for the shared task.² Note that our graph parser does not presuppose any input features other than the input text but we predict the base features using our pre-processing pipeline for completeness and to enable possible additional post-processing which involves altering enhanced dependency labels with lemma or case information.

²For Arabic, our Stanza Multi-word Token (MWT) expander predicted MWTs with a span of length 1 for two sentences. In the UD guidelines, MWT span lengths must be larger than one. To pass validation, we trained a UDPipe tokenizer with Word2Vec embeddings for Arabic instead.

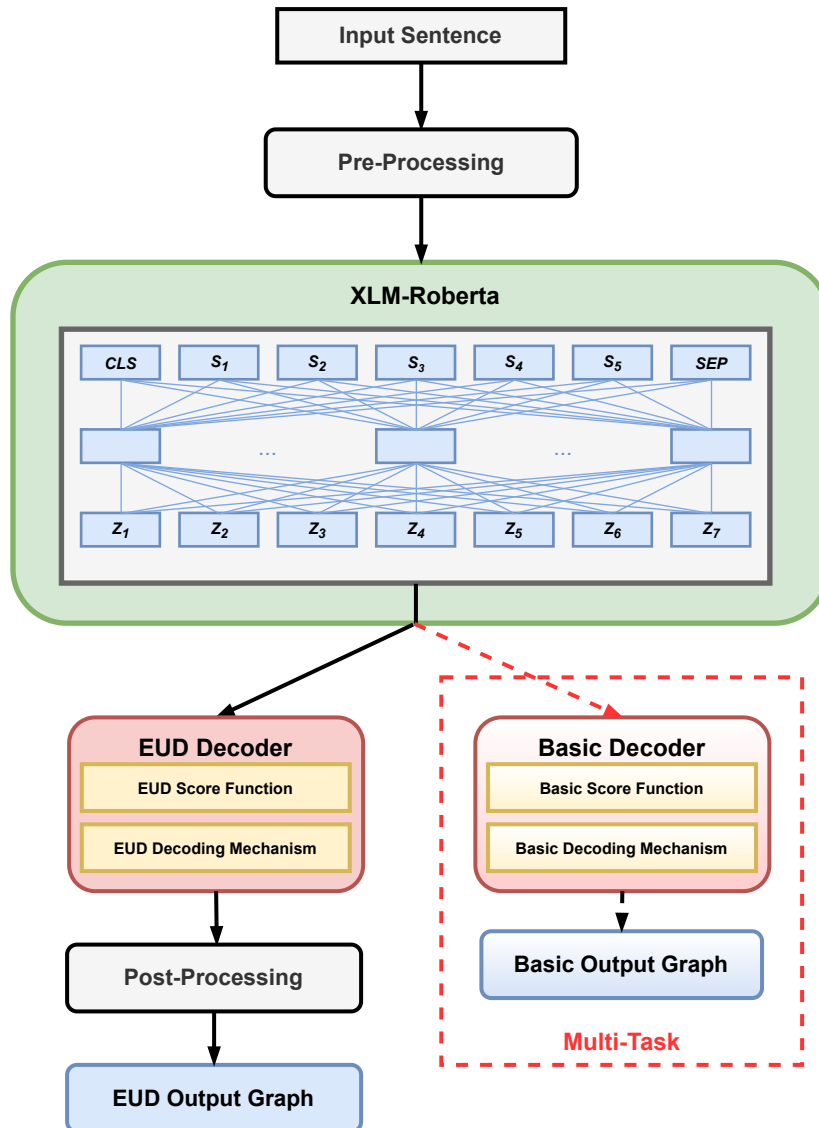


Figure 5.6: DCU-EPFL multitask architecture.

Enhanced UD Parsing For the enhanced UD parser, we use a Transformer encoder in the form of XLM-R (Conneau et al., 2020) with a first-order arc-factored model which utilises the edge and label scoring method of (Kiperwasser and Goldberg, 2016). In initial experiments, we found this model to perform better than biaffine attention (Dozat and Manning, 2017) for the task of EUD parsing. This finding was also made by Lindemann et al. (2019) and Straka and Straková (2019) for the task of semantic parsing across numerous Graphbanks (Oepen et al., 2019). Straka and Straková (2019) suggest that biaffine attention may be less suitable for predicting whether an edge exists between any pair of nodes using a predefined threshold and is perhaps more suited for dependency parsing, where words are “competing” with one another to be classified as the head in a softmax layer. The consistency of these findings across EUD and Graphbank parsing provides evidence that EUD is closer to semantic dependency parsing than basic UD parsing.

Parser Implementation Given a sentence x of length n , our model computes vector representations $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n)$ for the predicted tokens (x_1, x_2, \dots, x_n) . Since the SentencePiece tokenisation of XLM-R differs from the tokenisation used in UD, we track the mapping I from XLM-R’s k -th sub-word unit of the j -th input token produced by Stanza to the sub-word unit’s position $I_{j,k}$ in context of the sentence and we consider the output vector $\mathbf{e}_{I_{j,1}}$ of the first sub-word unit of each word x_j as its vector representation (\mathbf{r}_j):

$$\begin{aligned} \mathbf{E} &= \text{XLMR}(x_1, x_2, \dots, x_n) \\ \mathbf{R} &= \text{Filter}(\mathbf{E}, \mathbf{I}) \end{aligned} \tag{5.1}$$

where $\mathbf{E} = (\mathbf{e}_1, \dots, \mathbf{e}_m)$ are the output vectors of all sub-word units, m being the total number of subword units in the sentence, and $\text{Filter}()$ chooses the first embedding for each token. We add a dummy representation of the same dimensionality for the ROOT

token to the sequence of vectors \mathbf{R} but mask out predictions from this token. Following Kiperwasser and Goldberg (2016), these representations $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n)$ are then passed to the dependency parsing component, where the feature function ϕ is the concatenation of the representations of a potential head word x_h and dependent word x_d , where \circ denotes concatenation:

$$\phi(h, d) = \mathbf{r}_h \circ \mathbf{r}_d \quad (5.2)$$

Edge Prediction We compute scores for all $n(n-1)$ potential edges (h, d) , $h \neq d$, with an MLP:

$$s_{hd}^{(\text{arc})} = MLP^{(\text{arc})}(\phi(h, d)) \quad (5.3)$$

The edge classifier computes scores for all possible head-dependent pairs, and we compute a sigmoid on the resulting matrix of scores to obtain probabilities. We use an edge prediction threshold of 0.5, i. e. we include all edges with a score above 0.5 in the preliminary EUD graph. This enables words to have multiple heads but it can also lead to words receiving no head (if all probabilities are below 0.5). In such cases, we manually select the edge that has the highest probability. This greedy edge-prediction procedure can lead to fragmented graphs, which we address in a post-processing script described in Section 5.2.

Label Prediction To label the graph, we then choose a label for each edge using a separate classifier:

$$s_{hd}^{(\text{label})} = MLP^{(\text{label})}(\phi(h, d)) \quad (5.4)$$

The scores for all possible labels are passed to a softmax layer, which outputs the probability of each label for edge (h, d) and we select the label with the highest probability

for each edge.

Loss Function For edge prediction, sigmoid cross-entropy loss is used, and for label prediction, as we want to select the label for each chosen edge, softmax cross-entropy loss is used (Dozat and Manning, 2018). We interpolate between the loss given by the edge classifier and the loss given by the label classifier (Dozat and Manning, 2018; Wang et al., 2020a) with a constant λ :

$$\mathcal{L} = \lambda \mathcal{L}^{(label)} + (1 - \lambda) \mathcal{L}^{(edge)} \quad (5.5)$$

Training details For the empty nodes which are prevalent in EUD graphs, we added them into the graph, and offset the head indices to account for the new token(s) added to the graph. At test time, we did not predict whether an elided token should be added to the graph. Due to time constraints, we trained using the full lexicalised enhanced dependency labels. We trained each model for 50 epochs and optimised for development ELAS. Early stopping was used if the development ELAS score did not improve for 10 epochs.

Post-processing In the EUD guidelines, the predicted structure must be a connected graph where all nodes are reachable from the notional root.³ After predicting the test files, we use our own graph connection tool (Barry et al., 2020) to make sure that each sentence is a connected graph. Specifically, we repeatedly check for unreachable nodes and the number of unreachable nodes that can be reached from them. We choose the candidate which maximises this number (in the case of ties, we choose the first node in surface order) and make it a child of the notional ROOT, i.e. this node becomes an additional root node. System outputs are then validated at level 2 by the UD validator⁴ to check for errors prior to submission.

³In UD, the notional ROOT is the token with ID 0, whereas a root node is any node that has 0 as its head.

⁴<https://github.com/UniversalDependencies/tools/blob/master/validate.py>

Multitask Parser		
Encoder		
XLM-R _{Base}	Hidden Size	768
XLM-R _{Large}	Hidden Size	1024
EUD Parser		
Arc MLP size		300
Label MLP size		300
Dropout MLP		0.35
Nonlinear act. (MLP)		ELU
Tree Parser		
Arc MLP size		500
Label MLP size		100
Dropout MLP		0.33
Dropout embeddings		0.33
Nonlinear act. (MLP)		ELU
Optimiser and Training Details		
Optimizer		AdamW
Learning rate		3e-4
beta1		0.9
beta2		0.999
Num. epochs		50
Patience		10
Batch size		16

Table 5.1: Chosen hyperparameters for the multitask graph and tree parser.

Model hyperparameters are listed in Table 5.1. The choice of XLM-R encoder (Base or Large) determines the hyperparameters of the encoder part of our model. In our official submission, we use XLM-R_{Base}. For the EUD parser, following Kiperwasser and Goldberg (2016), we use the same dimension size for the edge-scoring and label-scoring components. A dropout value of 0.35 is used for the input embeddings as well as for the encoder and MLP networks. A loss interpolation constant λ of 0.1 is used as in Wang et al. (2020a). For the tree parser, the hyperparameters follow that of Dozat and Manning (2017).

Language	[1]	[2]	[3]	[4]	[5]	[6]
	Official	[1]+Trankit	[2]+XLM-R_{Large}	[3]+Concat	[3]+MTL	[4]+MTL
Arabic	71.01	78.05(+24.2%)	79.51(+6.6%)	-	81.72(+10.8%)	
Bulgarian	92.44	92.47(+0.4%)	93.26(+10.5%)	-	93.59(+4.9%)	
Czech	89.93	90.28(+3.5%)	91.06(+8.1%)	91.43(+4.1%)	91.22(+1.8%)	91.30(-1.5%)
Dutch	81.89	86.51(+25.5%)	87.67(+8.6%)	88.60(+7.5%)	88.60(+7.5%)	89.51(+7.9%)
English	85.70	85.97(+1.8%)	86.94(+6.9%)	87.46(+3.9%)	87.08(+1.1%)	87.28(-1.4%)
Estonian	84.35	84.54(+1.2%)	85.92(+8.9%)	86.68(+5.4%)	86.32(+2.8%)	86.76(+0.6%)
Finnish	89.02	89.34(+2.9%)	90.79(+13.6%)	-	91.16(+4.1%)	
French	86.68	86.80(+0.9%)	89.12(+17.6%)	-	90.38(+11.6%)	
Italian	92.41	92.44(+0.4%)	93.35(+12.1%)	-	93.47(+1.8%)	
Latvian	86.96	86.85(-0.8%)	88.81(+14.9%)	-	89.18(+3.3%)	
Lithuanian	78.04	78.44(+1.8%)	82.09(+16.9%)	-	83.47(+7.7%)	
Polish	89.17	89.30(+1.2%)	90.20(+8.4%)	91.15(+9.7%)	90.46(+2.7%)	90.46(-7.8%)
Russian	92.83	93.06(+3.2%)	93.95(+12.8%)	-	94.09(+2.3%)	
Slovak	89.59	90.81(+11.7%)	92.33(+16.5%)	-	92.73(+5.2%)	
Swedish	85.20	85.98(+5.3%)	88.10(+15.1%)	-	88.64(+4.5%)	
Tamil	39.32	40.64(+2.2%)	48.85(+13.8%)	61.14(+24.0%)	58.60(+19.1%)	62.06(+2.4%)
Ukrainian	86.09	86.30(+1.5%)	89.44(+22.91%)	-	90.91(+13.9%)	
Average	83.57	84.58(+6.2%)	86.55(+12.7%)	84.41	87.74(+8.8%)	84.56

Table 5.2: Evaluation scores on the official test data on the language-specific test files. All runs after **Official** use **Trankit** pre-processing and all runs after **Trankit** use the **XLM-R_{Large}** model. All numbers inside the parentheses are calculated as the relative error reduction of the particular column and the column it is compared with (which is shown in square brackets).

5.3 Results and Further Experiments

In this section, we discuss our official results and then describe a number of post-deadline experiments that improved our submission’s score.

5.3.1 Official Submission

For the official submission, we use the Stanza pre-processing pipeline and our graph parsing model with XLM-R_{Base}. The results are listed in column [1] of Table 5.2. Our official submission placed 6th of 9 participants. The overall scores submitted by each team are listed in Table 5.3. The scores of two teams are close to our overall score: Combo and Unipi placed 4th and 5th with scores of 83.79 and 83.64 compared to our score of 83.57. This grouping is outperformed by the top-three submissions TGIF, ShanghaiTech and RobertNLP by a margin from 3.2 ELAS points (RobertNLP vs. Combo) to 5.7 ELAS points (TGIF vs. DCU-EPFL).

5.3.2 Additional Experiments

We carry out a number of additional post-deadline experiments to try and find potential ways of improving our system. These subsequent experiments are detailed in the following paragraphs.

Trankit Pre-processing First, we replace the Stanza pre-processing pipeline (which uses Word2Vec and FastText embeddings as external input features and a BiLSTM encoder) with Trankit (Nguyen et al., 2021), which uses the Transformer XLM-R as the encoder.⁵ The results from adopting Trankit for sentence segmentation and tokenisation are listed in column [2] of Table 5.2. We notice slight improvements for all languages, with notable exceptions being Arabic, Dutch and Slovak, where the better pre-

⁵We used the Base version of XLM-R in Trankit but higher preprocessing scores would be expected if switching to the Large model.

processing accounts for a 24.2%, 25.5% and 11.7% relative error reduction (RER), respectively.

XLM-R_{Large} Our next modification is to leverage the XLM-R_{Large} model. This model has roughly twice as many parameters as the XLM-R_{Base} model used in our official submission. The results for combining Trankit pre-processing and using XLM-R_{Large} are listed in column [3] of Table 5.2. The larger capacity of the model translates to a large RER particularly for Finnish, French, Latvian, Lithuanian, Swedish, Tamil and Ukrainian of 13.6%, 17.6%, 14.9%, 16.9%, 15.1%, 13.8% and 22.91%, respectively. Given the improvements seen by adopting both Trankit for pre-processing and the larger XLM-R_{Large} model, we now incorporate these modifications into all further experiments.

Treebank Concatenation In our official system, we used just one treebank per language. Our next experiment is to investigate the effect of concatenating all treebanks with enhanced UD annotations for a language. We hypothesise that there could be a positive transfer from learning similar (within-language) treebanks and that it would make our parser more robust to the multiple domains in the test data. This means that for Czech we concatenate the PDT, CAC and FicTree treebanks, for Dutch, Alpino and LassySmall, for English EWT and GUM, and for Estonian EDT and EWT. For Tamil, we concatenate English EWT and GUM training data to Tamil to address the very poor evaluation score of our official submission, taking inspiration from Wang et al. (2020a) who observe substantial positive effects when they add Czech and English data to the Tamil treebank.⁶ The results are listed in column [4] of Table 5.2. Treebank concatenation helps for all languages but most notable is the improvement of over 12 points ELAS or an RER of 24% for Tamil, the language with the least amount of training data in the task.

⁶To keep training times relatively short, we did not include Czech.

Language	combo	dcu-epfl	fastparse	grew	nuig	robertnlp	shanghaitech	tgif	unipi	ref.	ours best
Arabic	76.39	71.01	53.74	71.13	0.0	81.58	82.26	81.23	77.17	67.35	81.72
Bulgarian	86.67	92.44	78.73	88.83	78.45	93.16	92.52	93.63	90.84	85.81	93.59
Czech	89.08	89.93	72.85	87.66	0.0	90.21	91.78	92.24	88.73	78.44	91.30
Dutch	87.07	81.89	68.89	84.09	0.0	88.37	88.64	91.78	84.14	82.48	89.51
English	84.09	85.70	73.00	85.49	65.40	87.88	87.27	88.19	87.11	83.68	87.28
Estonian	84.02	84.35	60.05	78.19	54.03	86.55	86.66	88.38	81.27	76.86	86.76
Finnish	87.28	89.02	57.71	85.20	0.0	91.01	90.81	91.75	89.62	78.26	91.16
French	87.32	86.68	73.18	83.33	0.0	88.51	88.40	91.63	87.43	98.80	90.38
Italian	90.40	92.41	78.32	90.98	0.0	93.28	92.88	93.31	91.81	80.20	93.47
Latvian	84.57	86.96	66.43	77.45	56.67	88.82	89.17	90.23	83.01	79.32	89.18
Lithuanian	79.75	78.04	48.27	74.62	59.13	80.76	80.87	86.06	71.31	75.26	83.47
Polish	87.65	89.17	71.52	78.20	0.0	89.78	90.66	91.46	88.31	81.59	90.46
Russian	90.73	92.83	78.56	90.56	66.33	92.64	93.59	94.01	90.90	79.63	94.09
Slovak	87.04	89.59	64.28	86.92	67.45	89.66	90.25	94.96	86.05	76.42	92.73
Swedish	83.20	85.20	67.26	81.54	63.12	88.03	86.62	89.90	84.91	80.98	88.64
Tamil	52.27	39.32	42.53	58.69	0.0	59.33	58.94	65.58	51.73	75.44	62.06
Ukrainian	86.92	86.09	63.42	83.90	0.0	88.86	88.94	92.78	87.51	77.24	90.91
Average	83.79	83.57	65.81	81.58	30.03	86.97	87.07	89.24	83.64	79.87	88.04

Table 5.3: Evaluation scores on the official test data on the language-specific test files submitted by each team. We also include the official reference system (**ref.**) which copies the gold tree to the enhanced graph as well as (**ours best**) which is our best post-deadline run, which corresponds to the **+Concat+MTL** run in Table 5.2. The first and second top-scoring models in each language are specified with black and blue colour, respectively.

Joint Learning of Basic and Enhanced Dependency Parsing The official reference system submitted by the shared task organisers which copies the gold trees to the enhanced representation performs very well with 79.87 ELAS (see Table 5.3). Thus, there is evidence that the basic tree and enhanced graph contain much mutual information. Previous methods which have leveraged the basic representation for producing EUD graphs have focused on using heuristic rules to convert the basic tree to EUD (Schuster and Manning, 2016; Dehouck et al., 2020; Ek and Bernardy, 2020), using the basic tree as input features to the enhanced parsing model (Barry et al., 2020) or converting the enhanced graph to a richer basic representation (Kanerva et al., 2020). In our final experiment, we try to leverage the information from the basic tree by jointly learning to predict the enhanced graph and the basic tree, testing whether performing basic dependency parsing and EUD parsing in a multitask setup is beneficial for EUD parsing. Given the positive effects seen through concatenation, for those languages where we performed concatenation, we also train multitask models on the concatenated versions of treebanks.

We use our previously described EUD parsing model and integrate with additional basic dependency parsing component (as shown in the right part of Figure 5.6) which is the biaffine parsing model of Dozat and Manning (2017) and train both parsers jointly, where the output of the encoder is passed into each parsing module, which makes its own predictions and returns a loss. The losses of the two components are combined with equal weight. We first try multitask learning on single treebanks and then also using the concatenated versions of the treebanks.

Single Treebanks First, we compare the multitask model to the setting where we are using $XLM-R_{Large}$ and single treebank training. The results are listed in column [5] of Table 5.2. Predicting the basic tree and the enhanced graph in a multitask setting yields improvements for all languages, particularly for Arabic, French and Ukrainian with an

RER of 10.8%, 11.6% and 13.9%, respectively. We also see a large gain for Tamil (19.1% RER), which shows that concatenating data from other treebanks is not the only way to improve parsing scores on this treebank, and that predicting the basic dependency tree can help in this low-resource scenario.

Multitask Model and Treebank Concatenation For the final experiment, we consider multitask learning on concatenated treebanks. The results are listed in column [6] of Table 5.2. Multitask learning on concatenated data is helpful for Dutch, Estonian and Tamil where it provides additional performance gains or an RER of 7.9%, 0.6% and 2.4%, respectively.

It is interesting to note that concatenation alone is more helpful for Czech and English where we see slight performance drops, or an increase in relative error of 1.5% and 1.4% and multitask learning is not helpful when trained on concatenated Polish treebanks (7.8% increase in relative error).

The positive contribution of multitask learning for all languages when not performing treebank concatenation could mean that it would be useful in settings where only one treebank with the enhanced representation is available for a language and the basic tree could be used as auxiliary information to predict the enhanced representation. This is the case for Tamil, which has the smallest amount of training data among all languages.

Comparison to Official Systems Our best unofficial run +Concat+MTL is added to Table 5.3. Compared to the other official runs, the ELAS scores of this run ranks in second place for 13/17 languages and places first for Italian and Russian.

5.4 Summary

In this chapter we have described the DCU-EPFL submission to the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies. We have also provided some answers for *RQ3: How can we leverage existing techniques to parse Enhanced Universal Dependencies?* by showing that, like basic dependency parsing, EUD graph parsing can be improved by leveraging better upstream pre-processing, a larger multilingual encoding model and treebank concatenation. We also show that performing basic dependency parsing and EUD graph parsing in a multitask fashion is beneficial for EUD graph parsing, where the basic dependency parser can provide additional information to the EUD parser.

In the next chapter, we revisit the topic of polyglot parsing, where we test whether multiview learning can be used to help remedy the problem of negative interference in multilingual (polyglot) models.

Chapter 6

Multiview Learning for Multilingual Dependency Parsing

In the previous chapters, we have seen that sometimes monolingual models can be superior to multilingual ones for performing dependency parsing in a certain language. For example, in Chapter 3, we saw that there was negative interference (Wang et al., 2020b) in the polyglot model for one of the source languages, Norwegian Nynorsk, and in Chapter 4, we showed that a monolingual BERT model for Irish was superior to the multilingual models, mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020).

In this chapter, we try to remedy this problem of negative interference in relation to parsing with multiple languages, i. e. polyglot parsing, as well as parsing with multiple datasets from the same language. As a potential solution, we consider *multiview learning*, a branch of machine learning where multiple models learn a separate *view* of the data or problem being modelled, and then in a later stage, the knowledge learned from each system is merged. We treat heterogeneous datasets as a source of multiview data and learn separate views from dependency parsers which are trained on a single dataset and over all included datasets, where there is no sharing in the former and full sharing in the latter. The representations from these two views are then passed to a meta view which

combines the information and makes its own prediction. We hypothesise that the single-dataset view will capture the full properties of the dataset, while the multi-dataset view will learn general/universal information over all datasets. These two sources (specialist and generalist) can then be merged in the meta-model, which will learn to combine the information in a useful manner. We evaluate our multiview architecture against single-view baselines in order to provide answers to *RQ4: Can Multiview Learning Help Multilingual Dependency Parsing?*

6.1 Background

Multiview learning is a branch of machine learning methods that involves using multiple functions where each function models a particular *view* of the data, where all of the functions are jointly optimised in order to improve generalisation performance (Zhao et al., 2017). Multiview learning has its roots in co-training (Blum and Mitchell, 1998) where different learners train on the same dataset but with different features, and the learners are used to label unannotated data for other learners in subsequent rounds.¹ As described by Zhao et al. (2017), a different view can come in the form of using different input features, e. g. in Computer Vision, colour information and texture information are two different kinds of features and thus a separate function could be used to model each one. In NLP, multiple views could correspond to using different lexical features for a task, e. g. modelling the data at the character, word or sentence level, or could also be created by randomly splitting the features (Nigam and Ghani, 2000).

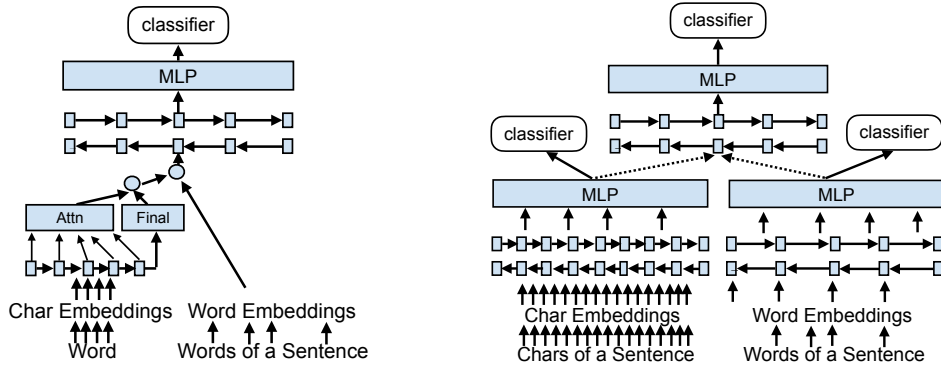
The idea behind using multiple views is that each view can describe a specific statistical property present in the data and assigning an individual learner to this view will result in it being highly tailored towards that property of the data. Having multiple distinct views means that there is now more information for the task which may have been otherwise ignored in a single-view setup where inputs are simply concatenated

¹Typically, only confident predictions on the unlabelled data from each classifier are included.

(Zhao et al., 2017; Lim et al., 2020). Additionally, multiview learning algorithms can be used to improve the consensus among views, thereby reducing overall error (Dasgupta et al., 2001), and jointly optimising multiple views can give rise to a situation where information from one view is used to inform another (Xu et al., 2013).

In the context of POS and morphological features tagging, Bohnet et al. (2018) showed that a multiview architecture can be used to achieve state-of-the-art results where their system had the highest tagging accuracies in the 2018 CoNLL shared task on dependency parsing (Zeman et al., 2018). Specifically, the architecture of Bohnet et al. (2018) incorporates three different views of the data, where each view has its own network configuration and makes its own classification and produces its own loss. The first view is based on *character-level* features which are obtained by running a BiLSTM over the input characters of a sentence, a second view is a *word-level* view which receives a pre-trained word embedding and a randomly initialised embedding as input, and finally, there is a *meta* view which receives as input the MLP projections from the other two networks, and uses those representations to make its own classification. Training is carried out in a synchronised fashion, where all views are jointly optimised. Figure 6.1 shows a comparison between a single-view architecture for POS tagging (Dozat et al., 2017) and the multiview architecture of Bohnet et al. (2018).

We take inspiration from Bohnet et al. (2018) and use a similar multiview learning network configuration. Instead of modelling the same input at the character and word level, we treat *dataset source* as a source of multiview data and incorporate the following views: i) a single-dataset view, where each dataset has its own dedicated parsing head, ii) a multi-dataset view which is a parser that shares parameters over all included datasets, and iii) a *meta* view which is a parser that takes hidden representations from the other two networks and uses those inputs to make its own prediction. The hypothesis is that the single-dataset view will provide dataset-specific information while the multi-dataset view will provide universal information from all included datasets, and



(a) Single-view tagger of Dozat et al. (2017) (b) Multiview tagger of Bohnet et al. (2018)

Figure 6.1: Comparison between single-view and multiview architectures for POS tagging. In the single-view architecture, the character and word representations are concatenated and a single classifier is used. In the multiview architecture, sentence-based character and word representations are learned separately and are passed to their own classifiers. The outputs of the MLPs from each module are combined in the meta classifier (which is the top classifier in (b)), which makes the final prediction.

that the meta view will then make use of this specific and universal information and learn to combine it in a useful manner. The next sections study to what extent this is the case or not.

6.2 Methodology

6.2.1 Proposed Models

In this section we list our baseline models and our proposed multiview model. We assume a dataset $\mathcal{D} = \{(x_d, y_d)\}_{d=1}^{\mathcal{D}}$ for $d = 1 \dots \mathcal{D}$, where x is a sentence and y is a dependency tree for x . Model parameters are denoted by θ . The proposed models can be categorised according to the following two criteria:

- whether they are trained on a single dataset $\mathcal{D}_S = \mathcal{D}_1$ or on concatenated data from K datasets $\mathcal{D}_M = \{\mathcal{D}_1, \dots, \mathcal{D}_K\}$
- whether they are trained using a single view $\theta_S = \theta_1$ or with multiple V views

$$\theta_M = \{\theta_1, \dots, \theta_V\}$$

Single Dataset with Single View (SDSV) The first baseline is training with a single dataset and a single view $M_{\text{SDSV}} = (\theta_S, \mathcal{D}_S)$. This is the standard approach for dependency parsing and indeed for many NLP tasks, where a single model is trained on a single dataset.

Multiple Datasets with Single View (MDSV) The second baseline is trained on the concatenation of data from multiple datasets with a single view $M_{\text{MDSV}} = (\theta_S, \mathcal{D}_M)$. This is also a common approach in NLP, where multiple datasets are concatenated and a single model is trained on them, optionally adding a dataset embedding denoting the source of the dataset (p. 53). This is considered a single-view model because inputs are simply concatenated and a single function is used to learn from the data.

Multiple Datasets with Multiple Views (MDMV) This model is trained on the concatenation of data from multiple datasets using multiple views $M_{\text{MDMV}} = (\theta_M, \mathcal{D}_M)$. In the MDMV model, each view represents a sub-model with its own network configuration, training hyperparameters and loss function. We consider the following views:

- **Single-dataset view:** The single-dataset view is composed of a ‘deck’ of single-dataset parsers, $Y_1^{sd}, \dots, Y_K^{sd}$ for the K datasets in the training data. Each single-dataset head is keyed on the input source, which is a homogeneous batch of data from a particular dataset. Each single-dataset parser consists of a *dataset-specific* BiLSTM encoder and biaffine parser (Dozat and Manning, 2017) (Section 2.1.2).
- **Multi-dataset view:** In the multi-dataset view, there is a *shared* BiLSTM encoder and biaffine dependency parser which are parameterised over all included datasets, i. e. these components are dataset agnostic.

- **Meta view:** The meta view consists of a BiLSTM encoder, which receives the hidden representations of the single- and multi-dataset encoders, and a biaffine dependency parser. These components are parameterised over all included datasets.

We do not consider the setting $(\theta_M, \mathcal{D}_S)$ as the MDMV model requires heterogeneous datasets to learn the separate views, i. e. \mathcal{D}_M is a requirement for this model.

In other words, the single-dataset view consists of multiple private sub-models which are only updated by their own input, the multi-dataset view is a public sub-model that shares parameters over all inputs, and the meta view is a public sub-model which combines information from the other two views and produces the final prediction. Alternatively, using the terminology of de Lhoneux et al. (2018), the single-dataset view corresponds to a “no-sharing” setting, and the multi-dataset and meta views are instances of “hard-sharing”. The architecture of the Multiview parser is shown in Figure 6.2. The input to the multiview parser is a sequence of tokens $x_{[1:n]}$, where these tokens are passed through the XLM-R encoder and then to the particular single-dataset head and the multi-dataset head (Y_i^{sd} and Y^{md} , respectively) and the BiLSTM representations from these two components are then passed to the meta head Y^* .

6.2.2 Training Procedure

We now describe the procedure involved in training our single-view baseline models (SDSV and MDSV) and our proposed multiview model (MDMV).

Single-view Training For our single-view baselines, which differ in whether they are trained on single or multiple datasets (\mathcal{D}_S or \mathcal{D}_M , respectively) the training objective is to minimise the loss \mathcal{L} with respect to the model parameters and the training data:

$$\min_{\theta} \mathcal{L}(\theta_S, \mathcal{D}_S) \tag{6.1}$$

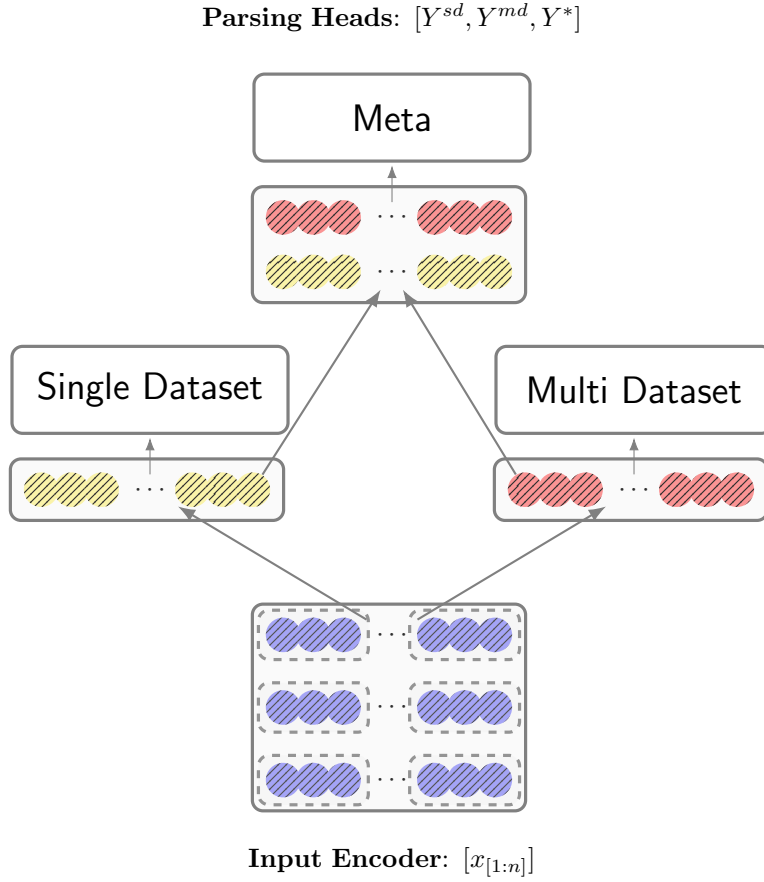


Figure 6.2: Architecture of the Multiview Parser. This figure only shows an example with one dataset. However, a separate single-dataset head is created for each dataset and the input is routed to that head depending on the input source, which is a homogeneous batch of data from that source.

Multi-view Training In multiview training, the model parameters θ are split into multiple parts: the single-dataset parameters $\theta_1^{sd}, \dots, \theta_K^{sd}$ for the K included datasets, the *shared* multi-dataset parameters θ^{md} , and the parameters of the meta-model θ^* . Then, our training objective for the i -th dataset is:

$$\min_{\theta_1^{sd}, \dots, \theta_K^{sd}, \theta^{md}, \theta^*} \sum_{i=1}^K \mathcal{L}_i(\theta_i^{sd}, \theta^{md}, \theta^*, \mathcal{D}_i) \quad (6.2)$$

The training loss from the i -th dataset is composed of the loss from the single-dataset model \mathcal{L}_i^{sd} , the multi-dataset model \mathcal{L}^{md} , and the meta-model \mathcal{L}^* . The losses are summed together with equal weight and are optimised jointly:

$$\mathcal{L}_i = \mathcal{L}_i^{sd} + \mathcal{L}^{md} + \mathcal{L}^* \quad (6.3)$$

Training occurs on all included datasets, and the stopping criterion is based on the highest-averaged LAS on the development datasets. A patience value of 8 is used (Section 5.2).

6.3 Models

6.3.1 Input Features

Given our three included models, we now consider the input features used by each model. For the two single-view baselines, we use lexical features from the pre-trained multilingual language model XLM-R (Conneau et al., 2020), which achieves state-of-the-art performance for multilingual dependency parsing (Grünwald et al., 2021). Concerning the multiview model, for the single- and multi-dataset heads, we follow Ruder and Plank (2018) who share the primary encoder in the context of a multitask setting with multiple learners. Therefore, we also use XLM-R as the encoder for the multiview model. We opt not to use a different Transformer for the single- and multi-dataset views as it would be prohibitively expensive in terms of memory usage to include two different pre-trained encoders in the same forward pass.²

²In preliminary experiments, we replicated the character- and word-level views of Bohnet et al. (2018) but for dependency parsing. We used XLM-R as the encoder for the word view and found that this parsing head achieved much higher performance than the character-level view which only learns on treebank data, where the character-level view did not provide much additional information.

Model Type	Encoder	Parsing Module
Single-view	XLM-R	BiAff
Multi-view	XLM-R	multi = BiLSTM + BiAff single = $K * (\text{BiLSTM} + \text{BiAff})$ meta = BiLSTM + BiAff

Table 6.1: The model types we consider and their components. Single-view models include the SDSV and MDSV settings, and the multi-view model corresponds to the MDMV setting.

6.3.2 Parsing Components

Given that the task we are performing is dependency parsing, the output module is a dependency parsing head which uses biaffine attention to score head and label relationships (Section 2.2.4). At test time, there is also a search for a spanning tree. The biaffine parsing head is common to all setups. However, for the single-view baselines there is just one parsing head, whereas for the multiview model, there are $K + 2$ parsing heads: K single-dataset heads and the multi-dataset and meta heads.³ In order to encourage diversity between the single- and multi-dataset views, and to be able to pass on this information to the meta head, an additional BiLSTM layer is added before the respective parsing heads, which serves as a domain adaptation layer, allowing the component to capture language- or dataset-specific properties. The meta parsing head also has a BiLSTM which takes the concatenated BiLSTM outputs of the below two heads and learns to combine their states as in Bohnet et al. (2018). The different model architectures we consider are listed in Table 6.1.⁴

6.4 Experiments

We implement our multilingual and monolingual experiments using the language groupings selected by Smith et al. (2018) who curated groups of languages based on typolog-

³In the forward pass, there are three parsing heads: the single-dataset parsing head of the current dataset and the multi-dataset and meta heads.

⁴For all single-treebank modules, the components are duplicated L times for all included treebanks.

ical information and language-relatedness. This language grouping was also used by van der Goot et al. (2021) and van der Goot and de Lhoneux (2021) for their experiments on analysing the role of dataset embeddings in parsing with multiple datasets. For each language group we train our three models (SDSV, MDSV and MDMV). Results on the test data of UD v2.8 are shown in the first three columns of Tables 6.2 and 6.3. We discuss each model setting in the following paragraphs.

Dataset Sampling For some language groups the GPU memory of 12GB was too small to use the multiview architecture. This occurred for the language groupings e-s1a and w-s1a, and is likely due to the presence of long sentences in some of the datasets. Also some groups contain up to seven datasets. To streamline the training process for these groups we followed a simple heuristic: for all datasets in these groups, we removed training sentences longer than 150 words and reduced the size of the largest dataset in the group to the size of the second-largest dataset.⁵

⁵The baseline models were also trained on these sampled and filtered datasets for a fair comparison.

Group	Dataset ID	SDSV	MDSV	MDMV	MDSV +embeds	MDMV +embeds
af-de-nl	af_afribooms	88.07	86.79	88.10	87.53	<u>88.36</u>
	de_gsd	85.28	85.90	86.12	<u>86.14</u>	86.02
	nl_alpino	92.84	93.21	93.81	93.32	93.72
	nl_lassysmall	90.51	92.04	92.73	92.19	92.63
e-sla	ru_syntagrus	94.28	91.72	94.5	94.17	<u>94.55</u>
	ru_taiga	84.05	84.08	85.04	84.43	84.92
	uk_iu	92.17	91.98	92.81	91.81	92.78
en	en_ewt	91.01	88.84	91.34	91.03	91.27
	en_gum	91.06	90.43	91.03	91.36	<u>91.55</u>
	en_lines	88.77	88.25	88.97	89.31	<u>89.47</u>
es-ca	ca_ancora	93.66	93.83	94.27	93.89	94.14
	es_ancora	92.61	92.88	93.26	92.85	93.24
finno	et_edt	89.08	89.15	89.49	89.31	89.48
	fi_ftb	93.34	63.32	94.15	93.15	93.99
	fi_tdt	93.38	92.77	93.33	93.47	<u>93.55</u>
	sme_giella	61.27	69.16	70.41	70.47	<u>70.59</u>
fr	fr_gsd	94.00	91.80	93.96	93.74	93.90
	fr_sequoia	94.29	92.10	94.62	93.94	<u>95.34</u>
	fr_spoken	84.68	84.99	86.92	86.38	86.90
indic	hi_hdtb	93.13	92.97	93.00	<u>93.32</u>	93.06
	ur_udtb	82.94	83.98	83.88	<u>84.22</u>	83.98
iranian	fa_seraji	90.51	90.41	90.31	90.17	90.3
	kmr_mg	12.40	61.57	60.71	<u>63.11</u>	54.83
it	it_isdt	94.12	94.10	94.39	93.96	93.92
	it_postwita	84.08	84.23	84.82	84.40	84.00
ko	ko_gsd	88.37	58.59	88.01	87.28	<u>88.53</u>
	ko_kaist	89.20	88.99	89.62	89.33	89.54
nor-ger	da_ddt	89.43	87.43	90.03	88.96	<u>90.20</u>
	no_bokmaal	93.88	93.86	94.41	94.14	94.40
	no_nynorsk	93.24	93.30	93.76	93.79	<u>93.77</u>
	no_nynorskli	76.84	77.36	78.62	78.11	<u>78.64</u>
	sv_lines	90.13	88.45	90.70	90.50	90.66
	sv_talbanken	92.49	91.41	92.89	92.15	<u>92.99</u>
pt-gl	gl_ctg	83.69	82.09	83.19	<u>83.96</u>	83.70
	gl_treegal	83.84	79.23	84.80	83.94	<u>85.48</u>
	pt_bosque	91.39	91.42	91.41	91.28	<u>91.48</u>
sw-sla	hr_set	90.68	89.44	90.87	<u>91.01</u>	90.91
	sl_ssj	95.65	95.26	95.59	95.21	<u>95.65</u>
	sl_sst	77.52	78.46	78.93	78.29	<u>79.52</u>
	sr_set	93.08	93.16	93.43	<u>93.84</u>	93.72
All	average	82.98	84.72	86.64	86.74	-

Table 6.2: Test results based on language groups (part 1/2). For table description, see the second part of the table (2/2).

Group	Dataset ID	SDSV	MDSV	MDMV	MDSV +embeds	MDMV +embeds
turkic	bxr_bdt	10.30	29.89	28.55	29.10	27.21
	kk_ktb	21.21	63.99	64.57	64.40	62.96
	tr_imst	73.93	74.18	73.81	73.91	73.79
	ug_udt	71.02	71.54	71.27	71.51	70.05
w-sla	cs_cac	93.36	93.99	94.19	94.48	OOM
	cs_fictree	94.49	94.85	94.98	94.85	
	cs_pdt	93.20	93.37	93.51	93.53	
	hsb_ufal	07.06	63.78	52.61	65.72	
	pl_lfg	97.14	89.58	97.15	97.09	
	pl_pdb	94.34	94.38	94.33	94.35	
	sk_snk	95.14	92.20	95.33	95.28	
All	average	82.98	84.72	86.64	<u>86.74</u>	-

Table 6.3: Test results based on language groups (part 2/2). Results are calculated with the official CoNLL evaluation script (*conll18_ud_eval.py*). Bolded numbers represent the highest-score among the initial three settings: SDSV, MDSV and MDMV. In the last two columns (+embeds), underlined numbers are used for when one of these settings now achieves the highest score among all settings. Average scores are based on both parts of the results. OOM refers to language groups which could not be run on time; we only report averages for full runs.

Single-Dataset Single View The most typical setting, i.e. training with a single dataset and a single model (SDSV), performs best for 8 out of 51 datasets considered in this experiment. For some of these 8 cases, the difference between this model and the multiview setting is very small, e.g. less than .06 LAS points for *en_gum*, *fi_tdt* and *fr_gsd*. Slightly bigger differences are seen for *ko_gsd* and *gl_ctg* with 0.36 and 0.5 LAS points, respectively.⁶ Training on a single dataset produces very low scores for certain low-resource datasets such as *kmr_mg*, *bxr_bdt*, and *hsb_ufal* which have an LAS of 12.40, 10.30 and 7.06, respectively.

Multiple-Datasets Single View The setting where multiple datasets are concatenated and a single model is trained on the combined data (MDSV) also performs best for 8 of 51 datasets. For some datasets such as *ur_udtb*, this model is only slightly better than the

⁶For Korean, there is a known divergence in annotation style between these two treebanks.

multiview model (83.98 vs. 83.88 LAS). For the low-resource dataset `kmr_mg`, concatenation is far superior to single-dataset training (61.57 vs. 12.4 LAS), where performance on this dataset is improved by training with related datasets. For this dataset as well as `bxr_bdt`, it even outperforms the multiview setting, which highlights that there is likely not enough data for the single-dataset head in the multiview architecture to learn useful information. It is also worth noting that MDSV suffers a dramatic decrease in LAS points for `fi_ftb` (93.34 to 63.32) and `ko_gsd` (88.37 to 58.59) where including other datasets is not helpful due to negative transfer.

Multiple-Datasets Multiple Views For the method which combines multiple datasets with multiple views, i. e. the multiview model (MDMV), we see a general trend of improved scores over the other two methods (35 of 51 datasets). Perhaps what is most interesting is the recovery of LAS points for those cases where simply concatenating data was not helpful: `fi_ftb`, `gl_treegal` and `ko_gsd`, which see a recovery of 30.83, 5.57 and 29.42 LAS points, respectively.

This shows that simply concatenating data is not always a viable strategy for improving parsing scores unless there is a way of differentiating between the input dataset sources, and that multiview learning—where different views come in the form of parsing heads which exhibit different amounts of parameter sharing (i. e. no-sharing and hard-sharing)—is a viable approach for reducing negative interference/transfer. Even when the multiview architecture underperforms the MDSV setting on some low-resource datasets (`kmr_mg` and `bxr_bdt`), the multiview architecture is still able to close the gap between the SDSV and MDSV models by a substantial amount, e. g. the difference in LAS is reduced from 49.17 to 0.86 for `kmr_mg`, and from 19.59 to 1.34 in the case of `bxr_bdt`.

Dataset Embeddings

Concatenating a dataset embedding to the input representations helps dependency parsers to differentiate between the input sources (see Section 3.1.2). When considering a Transformer encoder, van der Goot and de Lhoneux (2021) show that adding a dataset embedding (in the same way as a positional embedding) to the input of the encoder outperforms the approach of plain dataset concatenation. In this next setting, we add a dataset embedding to the XLM-R encoder for the MDSV and MDMV models. While van der Goot and de Lhoneux show that adding a dataset embedding to the encoder is useful for training a single parser on heterogeneous datasets (i. e. our MDSV model), to the best of our knowledge, dataset embeddings have not been used within a multiview architecture. Our next experiment seeks to test whether these two approaches are complementary, i. e. does adding a dataset embedding to the multiview architecture provide additional gains? We suspect that adding a dataset embedding will help the shared encoder to differentiate between the inputs, encouraging more diversity between the views. This also is similar to the strategy which de Lhoneux et al. (2018) found to be helpful (Section 3.1.2), i. e. soft-sharing at the input level and hard-sharing of the MLP classifiers of the network. However, a difference between our approach and theirs is that our setup also includes two other heads which exhibit no-sharing (single-dataset) and hard-sharing (meta). The last two columns in Tables 6.2 and 6.3 consider the MDSV and MDMV models with additional dataset embeddings (+embeds). Underlined numbers represent cases where one of these two settings now achieves the highest score among all settings.

Training Difficulties with the Multiview Model The multiview model requires three parsing heads in the same forward pass, which can require a substantial amount of GPU memory for certain language groups. To train these models, we used a physical batch size of 8 and accumulated gradients for 4 steps, giving an effective batch size of 32 (the

same as in the single-view baselines). For the *w-sla* group, we tried a batch size of 4 but still ran out of memory using a 24GB GPU. Thus, the only way of training this language group would be to reduce the batch size even further, potentially to 1. Unfortunately, this results in extremely long training times, and as a result, we could not include this language group in the MDMV with dataset embeddings setting. As such, this setting only includes 44 datasets and we do not report an average for this setting. Furthermore, the “old” group from van der Goot and de Lhoneux (2021) had to be omitted entirely as the MDMV model could not be trained on our hardware using a batch size of 2.

Multiple-Datasets Single View with Dataset Embeddings First we look at the MDSV model with added dataset embeddings. When compared to the regular MDSV setting, the added dataset embedding gives consistent improvements for many datasets, and for the cases where it is worse, the differences are only minor. Most notable are the cases where there was a large amount of negative interference in the plain concatenation setting (e. g. *ru_syntagrus*, *fi_ftb*, *ko_gsd*, *gl_treegal* and *pl_lfg*), where the added dataset embeddings brings the scores for this setting close to the individual dataset baseline (SDSV).

When compared to the MDMV model, results are similar for many datasets. However, for the low-resource datasets such as *kmr_mg*, *bxr_bdt* and *hsb_ufal*, the MDSV model with added dataset embeddings is better, highlighting a potential weakness of the multiview setting for smaller datasets, where perhaps the single-dataset head is not able to learn useful features to pass on to the meta parsing head.

Multiple-Datasets Multiple Views with Dataset Embeddings Next, we look at the MDMV model with added dataset embeddings. This model performs comparably to the MDSV with dataset embeddings setting, suggesting that the majority of the gains can be obtained by adding dataset embeddings to a single-view architecture. There are some improvements over the regular MDMV setting, e. g. for some English datasets (*en_gum*

and en_lines) as well as for the Galician and Portuguese (gl_treegal and pt_bosque) and Slovenian datasets (sl_ssj and sl_sst). Dataset embeddings do not help in this setting for kmr_mg, where results deteriorate by a large amount compared to the regular MDMV setting. There are also performance degradations for the Turkic datasets (bxr_bdt and kk_ktb). These findings suggest that dataset embeddings are not helpful for alleviating the poor performance of the multiview model when dealing with smaller datasets. We suspect that a more sophisticated way of gating the information passed to the meta head from each component would help the model to disregard weaker information from a particular source. In preliminary experiments, we tried using cross-stitch blocks (Misra et al., 2016; Ruder et al., 2017) to interpolate between the encoder output of each single-dataset head and the multi-dataset head, but found that this did not help performance.

When considering the two new settings with added dataset embeddings, out of the 51 datasets we consider, the original SDSV, MDSV and MDMV models achieve the highest scores for 2, 4, and 22 datasets, respectively, and the MDSV and MDMV models with added dataset embeddings achieve the highest score for 11 and 16 datasets, respectively. This means that a multiview architecture outperforms a single-view architecture for 34 out of 51 datasets.

6.5 Summary

At the beginning of this chapter, we highlighted the need to reduce negative transfer when training on multiple dataset sources (e. g. in cases such as polyglot parsing and even parsing with multiple datasets from the same language). At the same time, we wish to preserve some of the benefits from learning with multiple sources of data.

On one end of the spectrum, we considered training with just a single dataset and a single view. This model performs comparably to the other two models we considered for some datasets. However, in many cases, it is the weakest of the three as it does not benefit from additional information from related datasets. Furthermore, for certain datasets

there is not enough data to learn a useful model, making this approach infeasible. We then considered training with multiple datasets but with a single view. This improved scores for many settings, particularly when there was not enough data for single-dataset training. However, for some settings, there was a large amount of negative interference between datasets and this highlights the need for developing models which are capable of combining dataset-specific and universal information. To address this, we considered multiview learning, where different views are created based on the dataset source. The different views involve a dataset-specific view and a general dataset view, which are then combined in a meta view (Bohnet et al., 2018). We showed that this approach improves on the single dataset setting by including more data from related datasets. Additionally, it addresses the issue of negative interference where single-view concatenation was not helpful, e. g. for certain Finnish, Korean and Portuguese datasets. Taken together, the multiview architecture provides a balance between information which is relevant to a specific dataset and universal information.

However, one limitation of the multiview approach is that, for certain low-resource languages, there appears to be not enough data to learn useful features for the single-dataset view and the multiview model underperforms the plain concatenation setting. The presence of the shared dataset view is able to make up for this to some extent by closing the gap between the two single-view baselines.

We then looked at the role of adding dataset embeddings to our two setups which use multiple datasets, MDSV and MDMV, respectively. Our findings corroborate previous findings (van der Goot and de Lhoneux, 2021) which show that adding dataset embeddings is helpful for the MDSV setting.⁷ This helped alleviate some of the issues of negative interference when performing plain concatenation. Overall, the scores of this setting are comparable to the multiview approach, showing that both are a way of reducing negative interference in models trained on multiple sources of data. However,

⁷One difference between our work and the work of van der Goot and de Lhoneux is that we use XLM-R instead of mBERT.

adding dataset embeddings is more parameter-efficient and having one parsing head instead of three reduces parsing time, making this approach more attractive in many cases. Furthermore, the multiview approach appears to suffer when there is not enough data to learn useful information for the single-dataset parser for a particular dataset. When considering the MDMV setting, we found that adding dataset embeddings was helpful in 21 cases where it improved over the regular MDMV setting but on the whole there was not that much difference to the regular setting and there was even a deterioration in some cases.

The main aim of this chapter was to answer our fourth research question *RQ4: Can Multiview Learning Help Multilingual Dependency Parsing?* The findings of this chapter suggest that multiview learning can be helpful for multilingual dependency parsing, where the multiview setting is the best overall when compared to the regular SDSV and MDSV baselines. However, we showed that adding dataset embeddings can close the gap between the MDSV and MDMV setting, and argued that it is a more parameter-efficient method, which could mean that most practitioners could opt for this approach instead.

Chapter 7

Conclusion

In this chapter, we summarise the findings of the previous chapters and relate these findings to the research questions that we proposed in Chapter 1. We then summarise the main contributions of the works presented in this thesis, and finally, list some possible avenues for future work.

7.1 Research Questions Revisited

In Chapter 3, we approached zero-shot cross-lingual dependency parsing for a low-resource target language in Faroese. We described how zero-shot cross-lingual transfer is usually approached either through data transfer or model transfer. In our experiments, we focused on the data transfer approach, where we transferred annotations from a number of source languages to our zero-shot target language of Faroese. We adopted a common annotation projection setting but diverged from previous work which train monolingual source models (Agić et al., 2016; Tyers et al., 2018) by performing polyglot training on the source languages. We showed that this intervention produces an overall trend of better results on the target language, and achieves the best result overall when considering single-treebank target models. However, when we con-

sidered multi-treebank learning on the target side, the overall best result came from training on monolingual projections and using the proxy dataset embedding learned on Norwegian Nynorsk projections to parse Faroese. These results suggest that, when performing annotation projection, polyglot training will likely lead to a strong model for transfer, and improves the scores of some weaker-performing sources. This could be useful in certain cases, e. g. when the target language is distant to the available source languages. However, if the target language has a closely related source language, then training on a single source would likely suffice. Thus, our answer to *RQ1: Can zero-shot cross-lingual dependency parsing be improved by leveraging polyglot training on source languages?* was not clear-cut: we have shown that polyglot training does improve cross-lingual dependency parsing for 3 of 4 source languages, but when considering multi-treebank learning on the target side, the target model can leverage a well-suited monolingual source.

Continuing with the theme of facilitating parser development for low-resource languages, we then moved to considering model transfer for low resource dependency parsing in Chapter 4. In particular, we considered the popular approach of transfer learning by training a language model on unannotated corpora and then fine-tuning it on labelled data in a downstream task, where we considered the tasks of dependency parsing, POS and morphological tagging, and MWE identification. To develop our model, we collected a wide variety of Irish corpora and then performed a number of development experiments in order to guide us to our final model configuration. We considered differing filtering criteria, vocabulary sizes and tokenisation models as well as Transformer architectures. Our aim was to answer *RQ2: Does a monolingual language model improve low-resource dependency parsing in the case of Irish?* We added to the literature on LLMs for low-resource languages and answered this research question positively, by showing that a monolingual LM does improve dependency parsing (as well as POS/morphological features tagging and MWE identification) scores for Irish, when

compared to off-the-shelf and further-pretrained multilingual models.

We then explored the topic of parsing enhanced UD graphs in Chapter 5, where we described our submission to the 2021 IWPT shared task on EUD parsing (Bouma et al., 2021). We introduced a novel multitask model that performs the tasks of basic dependency tree parsing and EUD graph parsing jointly, yielding performance improvements over a single-task model. We provided answers to *RQ3: How can we leverage existing techniques to parse Enhanced Universal Dependencies?* by showing that EUD parsing accuracy can be improved by providing better upstream tokenisation and segmentation, using a larger pre-trained encoder, monolingual and cross-lingual treebank concatenation, and multitask learning between a basic dependency parser and an EUD graph parser. Our experiments showed that the multitask model was better than the single-task model for all languages. However, performing multitask learning on concatenated treebanks was worse than plain concatenation for certain languages.

In Chapter 6, we looked at the utility of using multiview learning for multilingual dependency parsing. Here, we adapted the setup of Bohnet et al. (2018) for the task of dependency parsing and learned separate views of the data based on dataset source. We showed that multiview learning can improve over the single-view baselines of training on a single dataset with a single parsing model, and training a single parser on multiple datasets, where the multiview model was the best for 35 out of 51 datasets considered. We then integrated another technique shown to work well for polyglot parsing by adding dataset embeddings to the MDSV and MDMV models and showed that added dataset embeddings helped the MDSV setting for 40 out of 51 cases, and in the case of the MDMV model, for 17 out of 38 cases. Overall, a multiview model was the best model for 34 out of 51 datasets.

7.2 Contributions

In this thesis, we carried out four separate experiments related to our overall goal of improving multilingual dependency parsing. The novel contributions arising from these experiments are as follows:

1. We showed that polyglot training is one way of improving the source models in an annotation projection setting, and that multi-treebank learning on the target side can be used to improve scores even further.
2. We released a monolingual Irish language model *gaBERT*, which achieves state-of-the-art performance on dependency parsing, outperforming multilingual models.
3. We developed a multitask parser that improves EUD parsing performance by jointly learning the tasks of basic dependency and EUD graph parsing.
4. We developed a multiview learning architecture for multilingual dependency parsing and showed that learning different views based on the dataset source can help alleviate problems related to negative transfer.

7.3 Future Work

The work presented in this thesis could be built upon in many ways.

In the North Germanic language experiments presented in Chapter 3, our polyglot model uses a BiLSTM encoder operating over word and character features using a shared vocabulary. It is highly likely that a multilingual Transformer with a shared sub-word vocabulary (Devlin et al., 2019; Conneau et al., 2020) would be a better encoder for the source models, and it would be interesting to try these experiments with a multilingual LM. Furthermore, in our experiments, we only considered one North Germanic language group. As such, it would be interesting to see how polyglot training

used alongside annotation projection would work for other language groups, and see if issues related to negative transfer also arise for these groups. The presence of negative transfer in the polyglot model for one source language, Norwegian Nynorsk, motivates the study of how to learn with multiple languages such that universal and language-specific information can be successfully merged. These findings helped motivate our experiments in Chapter 6, where we considered multiview learning as a means to reducing negative transfer, but studying other methods for successful polyglot training would be worthwhile.

In Chapter 4, we released a monolingual BERT model for Irish, gaBERT. The fact that data quality is a major determining factor in LM quality (Scao et al., 2022) means that collecting more high-quality Irish corpora will be paramount to developing future monolingual LMs for Irish. We also wish to explore other model architectures and develop an auto-regressive language model such as GPT (Radford et al., 2019) for generating text in Irish. We also evaluated gaBERT across a number of morphosyntactic tagging and parsing tasks, but would like to compare our model on other NLP tasks such as QA. We were unable to do so due to the lack of existing datasets in these areas but evaluating gaBERT on such tasks would be worthwhile if such data became available. So far, we have just trained a monolingual model for Irish, but in the future, we would also like to train a bilingual model on the related Celtic language of Scots Gaelic.

In Chapter 5, we introduced a novel multitask model to predict basic dependency trees and EUD graphs. Future work could explore other tasks to learn alongside the task of EUD parsing, such as SRL, or Graphbank parsing (Oepen et al., 2019). Similarly, different edge-scoring models could be used by different EUD parsing heads, e. g. where one parser uses the dependency parsing model of Kiperwasser and Goldberg (2016) and the other, the model of Dozat and Manning (2018). In future, we wish to modify our approach to include the predictions from the basic tree parser so that we first predict a directed spanning tree and then combine this with the EUD graph. Doing so would

obviate the need for post-processing involving connecting fragmented graphs (Barry et al., 2020) and would make our model a Tree-Graph Integrated system (He and Choi, 2020; Shi and Lee, 2021). We would also like to perform rule-based delexicalisation to address label sparsity issues and make training EUD parsers more memory-efficient.

In Chapter 6 we approached multilingual parsing through the use of a multiview architecture. This model outperformed the single-view baselines. However, we highlighted some cases where the multiview architecture underperforms relative to the concatenation setting, when one of the included datasets has a small amount of data. Therefore, it is important to work on methods which could help the model to disregard weaker information from one of the sources such that it does not negatively contribute to parsing performance. The use of dataset embeddings saw improvements for the single-view model, where average scores increased to the level of the multiview model. At the same time, the single-view approach is more parameter-efficient. Therefore, it is important to consider other approaches for learning separate views of the data that are also parameter-efficient. Similarly, dataset-specific and general information could be extracted in other ways, e. g. by having separate encoding components.

Bibliography

- Agić, Ž., Johannsen, A., Plank, B., Martínez Alonso, H., Schlueter, N., and Søgaard, A. (2016). Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.
- Al-Rfou', R., Perozzi, B., and Skiena, S. (2013). Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.
- Ammar, W., Mulcaire, G., Ballesteros, M., Dyer, C., and Smith, N. A. (2016). Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.
- Armengol-Estapé, J., Carrino, C. P., Rodriguez-Penagos, C., de Gibert Bonet, O., Armentano-Oller, C., Gonzalez-Agirre, A., Melero, M., and Villegas, M. (2021). Are multilingual models the best choice for moderately under-resourced languages? A comprehensive assessment for Catalan. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4933–4946, Online. Association for Computational Linguistics.
- Artetxe, M., Ruder, S., and Yogatama, D. (2020). On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Associa-*

- tion for Computational Linguistics, pages 4623–4637, Online. Association for Computational Linguistics.
- Aulamo, M., Virpioja, S., and Tiedemann, J. (2020). OpusFilter: A configurable parallel corpus filtering toolbox. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 150–156, Online. Association for Computational Linguistics.
- Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Ballesteros, M., Dyer, C., and Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.
- Bañón, M., Chen, P., Haddow, B., Heafield, K., Hoang, H., Esplà-Gomis, M., Forcada, M. L., Kamran, A., Kirefu, F., Koehn, P., Ortiz Rojas, S., Pla Sempere, L., Ramírez-Sánchez, G., Sarrías, E., Strelec, M., Thompson, B., Waites, W., Wiggins, D., and Zaragoza, J. (2020). ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online. Association for Computational Linguistics.
- Barry, J., Mohammadshahi, A., Wagner, J., Foster, J., and Henderson, J. (2021). The DCU-EPFL enhanced dependency parser at the IWPT 2021 shared task. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 204–212, Online. Association for Computational Linguistics.

- Barry, J., Wagner, J., Cassidy, L., Cowap, A., Lynn, T., Walsh, A., Ó Meachair, M. J., and Foster, J. (2022). gabert - an irish language model. In *Proceedings of the Language Resources and Evaluation Conference*, pages 4774–4788, Marseille, France. European Language Resources Association.
- Barry, J., Wagner, J., and Foster, J. (2020). The ADAPT enhanced dependency parser at the IWPT 2020 shared task. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 227–235, Online. Association for Computational Linguistics.
- Bastings, J. (2020). *A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing*. PhD thesis, University of Amsterdam.
- Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., and Sima'an, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.
- Björkelund, A., Falenska, A., Yu, X., and Kuhn, J. (2017). IMS at the CoNLL 2017 UD shared task: CRFs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.
- Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98*, page 92–100, New York, NY, USA. Association for Computing Machinery.
- Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China. Coling 2010 Organizing Committee.

- Bohnet, B., McDonald, R., Simões, G., Andor, D., Pitler, E., and Maynez, J. (2018). Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia. Association for Computational Linguistics.
- Bouma, G., Seddah, D., and Zeman, D. (2020). Overview of the IWPT 2020 shared task on parsing into enhanced Universal Dependencies. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 151–161, Online. Association for Computational Linguistics.
- Bouma, G., Seddah, D., and Zeman, D. (2021). From raw text to enhanced Universal Dependencies: The parsing shared task at IWPT 2021. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 146–157, Online. Association for Computational Linguistics.
- Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.
- Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Pro-*

- cessing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 957–961, Prague, Czech Republic. Association for Computational Linguistics.
- Cassidy, L., Lynn, T., Barry, J., and Foster, J. (2022). TwittIrish: A Universal Dependencies treebank of tweets in Modern Irish. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6869–6884, Dublin, Ireland. Association for Computational Linguistics.
- Chau, E. C., Lin, L. H., and Smith, N. A. (2020). Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.
- Che, W., Liu, Y., Wang, Y., Zheng, B., and Liu, T. (2018). Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.
- Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.
- Chen, W., Zhang, Y., and Zhang, M. (2014). Feature embedding for dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 816–826, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.
- Chu, Y.-J. and Liu, T.-H. (1965). On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.
- Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. In *8th International Conference on*

Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 111–118, Barcelona, Spain.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Dasgupta, S., Littman, M. L., and McAllester, D. (2001). Pac generalization bounds for co-training. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, NIPS'01*, page 375–382, Cambridge, MA, USA. MIT Press.

de Lhoneux, M. (2019). *Linguistically Informed Neural Dependency Parsing for Typologically Diverse Languages*. PhD thesis, Uppsala University.

de Lhoneux, M., Bjerva, J., Augenstein, I., and Søgaard, A. (2018). Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997, Brussels, Belgium. Association for Computational Linguistics.

- 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).
- de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).
- de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., and Nissim, M. (2019). Bertje: A dutch BERT model. *CoRR*, abs/1912.09582.
- Dehouck, M., Anderson, M., and Gómez-Rodríguez, C. (2020). Efficient EUD parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 192–205, Online. Association for Computational Linguistics.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dou, Z.-Y. and Neubig, G. (2021). Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online. Association for Computational Linguistics.
- Dowling, M., Castilho, S., Moorkens, J., Lynn, T., and Way, A. (2020). A human evalu-

- ation of English-Irish statistical and neural machine translation. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 431–440, Lisboa, Portugal. European Association for Machine Translation.
- Dowling, M., Lynn, T., Poncelas, A., and Way, A. (2018). SMT versus NMT: Preliminary comparisons for Irish. In *Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*, pages 12–20, Boston, MA. Association for Machine Translation in the Americas.
- Dozat, T. and Manning, C. D. (2017). Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net.
- Dozat, T. and Manning, C. D. (2018). Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.
- Dozat, T., Qi, P., and Manning, C. D. (2017). Stanford’s graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.
- Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. 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 334–343, Beijing, China. Association for Computational Linguistics.
- Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American*

- Chapter of the Association for Computational Linguistics: *Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.
- Edmonds, J. (1967). Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.
- Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.
- Ek, A. and Bernardy, J.-P. (2020). How much of enhanced UD is contained in UD? In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 221–226, Online. Association for Computational Linguistics.
- 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.
- Farahani, M., Gharachorloo, M., Farahani, M., and Manthouri, M. (2021). Parsbert: Transformer-based model for persian language understanding. *Neural Process. Lett.*, 53:3831–3847.
- Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O’Regan, J., Rojas, S. O., Pérez-Ortiz, J. A., Sánchez-Martínez, F., Ramírez-Sánchez, G., and Tyers, F. M. (2011). Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25:127–144.
- Gage, P. (1994). A new algorithm for data compression. *C Users J.*, 12(2):23–38.
- Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz,

- M., and Zettlemoyer, L. (2018). AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.
- Ginter, F., Hajič, J., Luotolahti, J., Straka, M., and Zeman, D. (2017). CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.
- Glavaš, G., Litschko, R., Ruder, S., and Vulić, I. (2019). How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 710–721, Florence, Italy. Association for Computational Linguistics.
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feed-forward neural networks. In Teh, Y. W. and Titterton, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.
- Goldberg, Y. and Nivre, J. (2012). A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India. The COLING 2012 Organizing Committee.
- Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.
- Grünewald, S. and Friedrich, A. (2020). RobertNLP at the IWPT 2020 shared task: Surprisingly simple enhanced UD parsing for English. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing*

into Enhanced Universal Dependencies, pages 245–252, Online. Association for Computational Linguistics.

Grünewald, S., Friedrich, A., and Kuhn, J. (2021). Applying occam’s razor to transformer-based dependency parsing: What works, what doesn’t, and what is really necessary. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 131–144, Online. Association for Computational Linguistics.

Guillaume, B. and Perrier, G. (2021). Graph rewriting for enhanced Universal Dependencies. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 175–183, Online. Association for Computational Linguistics.

Guo, J., Che, W., Yarowsky, D., Wang, H., and Liu, T. (2015). Cross-lingual dependency parsing based on distributed representations. 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 1234–1244, Beijing, China. Association for Computational Linguistics.

Hall, J., Nilsson, J., Nivre, J., Eryiğit, G., Megyesi, B., Nilsson, M., and Saers, M. (2007). Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 933–939, Prague, Czech Republic. Association for Computational Linguistics.

Hatori, J., Matsuzaki, T., Miyao, Y., and Tsujii, J. (2012). Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume*

- 1: *Long Papers*), pages 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.
- He, H. and Choi, J. D. (2020). Adaptation of multilingual transformer encoder for robust enhanced Universal Dependency parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 181–191, Online. Association for Computational Linguistics.
- Henderson, J. (2020). The unstoppable rise of computational linguistics in deep learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6294–6306, Online. Association for Computational Linguistics.
- Hershcovich, D., de Lhoneux, M., Kulmizev, A., Pejhan, E., and Nivre, J. (2020). Kopsala: Transition-based graph parsing via efficient training and effective encoding. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 236–244, Online. Association for Computational Linguistics.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Huang, L., Sun, X., Li, S., Zhang, L., and Wang, H. (2020). Syntax-aware graph attention network for aspect-level sentiment classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 799–810, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., and Kolak, O. (2005). Bootstrapping parsers via syntactic projection across parallel texts. *Nat. Lang. Eng.*, 11(3):311–325.
- Jalili Sabet, M., Dufter, P., Yvon, F., and Schütze, H. (2020). SimAlign: High quality word alignments without parallel training data using static and contextualized em-

- beddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.
- Jawahar, G., Sagot, B., and Seddah, D. (2019). What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.
- Jurafsky, D. and Martin, J. H. (2000). *Speech and language processing*.
- Kanerva, J., Ginter, F., and Pyysalo, S. (2020). Turku enhanced parser pipeline: From raw text to enhanced graphs in the IWPT 2020 shared task. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 162–173, Online. Association for Computational Linguistics.
- Kilgarriff, A., Rundell, M., and Uí Dhonnchadha, E. (2006). Efficient corpus development for lexicography: building the New Corpus for Ireland. *Language Resources and Evaluation*, 40:127–152.
- Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.
- Kondratyuk, D. (2019). Multilingual learning using syntactic multi-task training.
- Kondratyuk, D. and Straka, M. (2019). 75 languages, 1 model: Parsing universal dependencies universally. 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 2779–2795, Hong Kong, China. Association for Computational Linguistics.

- Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden. Association for Computational Linguistics.
- Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.
- Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Kulmizev, A., de Lhoneux, M., Gontrum, J., Fano, E., and Nivre, J. (2019). Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. 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 2755–2768, Hong Kong, China. Association for Computational Linguistics.
- Kuncoro, A., Kong, L., Fried, D., Yogatama, D., Rimell, L., Dyer, C., and Blunsom, P. (2020). Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.
- Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.
- Lauscher, A., Ravishankar, V., Vulić, I., and Glavaš, G. (2020). From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

- Lim, K., Lee, J. Y., Carbonell, J., and Poibeau, T. (2020). Semi-supervised learning on meta structure: Multi-task tagging and parsing in low-resource scenarios. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8344–8351.
- Lindemann, M., Groschwitz, J., and Koller, A. (2019). Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.
- Liu, Y., Che, W., Wang, Y., Zheng, B., Qin, B., and Liu, T. (2019a). Deep contextualized word embeddings for universal dependency parsing. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19(1).
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized bert pretraining approach.
- Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*.
- Lui, M. and Baldwin, T. (2012). langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea. 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, Lisbon, Portugal. Association for Computational Linguistics.
- Lynn, T., Cetinoglu, O., Foster, J., Dhonnchadha, E. U., Dras, M., and van Genabith, J. (2012). Irish treebanking and parsing: A preliminary evaluation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC-2012)*, pages 1939–1946, Istanbul, Turkey. European Language Resources Association (ELRA).

- Lynn, T., Foster, J., and Dras, M. (2013). Working with a small dataset - semi-supervised dependency parsing for Irish. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–11, Seattle, Washington, USA. Association for Computational Linguistics.
- Lynn, T., Foster, J., Dras, M., and Tounsi, L. (2014). Cross-lingual transfer parsing for low-resourced languages: An Irish case study. In *Proceedings of the First Celtic Language Technology Workshop*, pages 41–49, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.
- Lynn, T., Scannell, K., and Maguire, E. (2015). Minority Language Twitter: Part-of-Speech Tagging and Analysis of Irish Tweets. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 1–8, Beijing, China. Association for Computational Linguistics.
- Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É., Seddah, D., and Sagot, B. (2020). CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.
- McDonald, R., Crammer, K., and Pereira, F. (2005a). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.
- McDonald, R. and Nivre, J. (2011). Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.
- McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., and Lee, J. (2013). Universal dependency annotation for multilingual parsing. In *Proceed-*

- ings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.
- McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.
- McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005b). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.
- McDonald, R., Petrov, S., and Hall, K. (2011). Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- McGuinness, S., Phelan, J., Walsh, A., and Lynn, T. (2020). Annotating MWEs in the Irish UD treebank. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 126–139, Barcelona, Spain (Online). Association for Computational Linguistics.
- Meechan-Maddon, A. and Nivre, J. (2019). How to parse low-resource languages: Cross-lingual parsing, target language annotation, or both? In *Proceedings of DepLing*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. arXiv 1310.4546v1.
- Misra, I., Shrivastava, A., Gupta, A. K., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003.

- Mohammadshahi, A. and Henderson, J. (2020). Graph-to-graph transformer for transition-based dependency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3278–3289, Online. Association for Computational Linguistics.
- Mohammadshahi, A. and Henderson, J. (2021). Syntax-aware graph-to-graph transformer for semantic role labelling. *ArXiv*, abs/2104.07704.
- Mulcaire, P., Kasai, J., and Smith, N. A. (2019a). Low-resource parsing with crosslingual contextualized representations. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 304–315, Hong Kong, China. Association for Computational Linguistics.
- Mulcaire, P., Kasai, J., and Smith, N. A. (2019b). Polyglot contextual representations improve crosslingual transfer. 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 3912–3918, Minneapolis, Minnesota. Association for Computational Linguistics.
- Nguyen, M. V., Lai, V. D., Pouran Ben Veyseh, A., and Nguyen, T. H. (2021). Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 80–90, Online. Association for Computational Linguistics.
- Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management, CIKM '00*, page 86–93, New York, NY, USA. Association for Computing Machinery.
- Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceed-*

- ings of the Eighth International Conference on Parsing Technologies, pages 149–160, Nancy, France.
- Nivre, J. (2004). Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.
- Nivre, J. (2006). *Inductive dependency parsing*. Springer.
- Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.
- Nivre, J. (2009). Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore. Association for Computational Linguistics.
- Nivre, J. (2020a). Multilingual dependency parsing from universal dependencies to sesame street. In *TDS*.
- Nivre, J. (2020b). Multilingual dependency parsing from universal dependencies to sesame street. In Sojka, P., Kopeček, I., Pala, K., and Horák, A., editors, *Text, Speech, and Dialogue*, pages 11–29. Springer International Publishing.
- Nivre, J., Abrams, M., Agić, Ž., Ahrenberg, L., Antonsen, L., Aranzabe, M. J., Arutie, G., Asahara, M., Ateyah, L., Attia, M., Atutxa, A., Augustinus, L., Badmaeva, E., Balles-teros, M., Banerjee, E., Bank, S., Barbu Mititelu, V., Bauer, J., Bellato, S., Bengoetxea, K., Bhat, R. A., Biagetti, E., Bick, E., Blokland, R., Bobicev, V., Börstell, C., Bosco, C., Bouma, G., Bowman, S., Boyd, A., Burchardt, A., Candito, M., Caron, B., Caron, G., Cebiroğlu Eryiğit, G., Celano, G. G. A., Cetin, S., Chalub, F., Choi, J., Cho, Y., Chun, J., Cinková, S., Collomb, A., Çöltekin, Ç., Connor, M., Courtin, M., Davidson, E.,

de Marneffe, M.-C., de Paiva, V., Diaz de Ilarraza, A., Dickerson, C., Dirix, P., Dobrovoljc, K., Dozat, T., Droганova, K., Dwivedi, P., Eli, M., Elkahky, A., Ephrem, B., Erjavec, T., Etienne, A., Farkas, R., Fernandez Alcalde, H., Foster, J., Freitas, C., Gajdošová, K., Galbraith, D., Garcia, M., Gärdenfors, M., Gerdes, K., Ginter, F., Goenaga, I., Gojenola, K., Gökırmak, M., Goldberg, Y., Gómez Guinovart, X., Gonzáles Saavedra, B., Grioni, M., Grüzītis, N., Guillaume, B., Guillot-Barbance, C., Habash, N., Hajič, J., Hajič jr., J., Hà Mỹ, L., Han, N.-R., Harris, K., Haug, D., Hladká, B., Hlaváčová, J., Hociung, F., Hohle, P., Hwang, J., Ion, R., Irimia, E., Jelínek, T., Johannsen, A., Jørgensen, F., Kaşıkara, H., Kahane, S., Kanayama, H., Kanerva, J., Kayadelen, T., Kettnerová, V., Kirchner, J., Kotsyba, N., Krek, S., Kwak, S., Laippala, V., Lambertino, L., Lando, T., Larasati, S. D., Lavrentiev, A., Lee, J., Lê Hồng, P., Lenci, A., Lertpradit, S., Leung, H., Li, C. Y., Li, J., Li, K., Lim, K., Ljubešić, N., Loginova, O., Lyaševskaya, O., Lynn, T., Macketanz, V., Makazhanov, A., Mandl, M., Manning, C., Manurung, R., Mărănduc, C., Mareček, D., Marheinecke, K., Martínez Alonso, H., Martins, A., Mašek, J., Matsumoto, Y., McDonald, R., Mendonça, G., Miekka, N., Missilä, A., Mititelu, C., Miyao, Y., Montemagni, S., More, A., Moreno Romero, L., Mori, S., Mortensen, B., Moskalevskiy, B., Muischnek, K., Murawaki, Y., Müürisep, K., Nainwani, P., Navarro Horñiacek, J. I., Nedoluzhko, A., Nešpore-Bērzkalne, G., Nguyễn Thị, L., Nguyễn Thị Minh, H., Nikolaev, V., Nitisaroj, R., Nurmi, H., Ojala, S., Olúòkun, A., Omura, M., Osenova, P., Östling, R., Øvreid, L., Partanen, N., Pascual, E., Passarotti, M., Patejuk, A., Peng, S., Perez, C.-A., Perrier, G., Petrov, S., Piitulainen, J., Pitler, E., Plank, B., Poibeau, T., Popel, M., Pretkalniņa, L., Prévost, S., Prokopidis, P., Przepiórkowski, A., Puolakainen, T., Pyysalo, S., Rääbis, A., Rademaker, A., Ramasamy, L., Rama, T., Ramisch, C., Ravishankar, V., Real, L., Reddy, S., Rehm, G., Rießler, M., Rinaldi, L., Rituma, L., Rocha, L., Romanenko, M., Rosa, R., Rovati, D., Roşca, V., Rudina, O., Sadde, S., Saleh, S., Samardžić, T., Samson, S., Sanguinetti, M., Saulīte, B., Sawanakunanon, Y., Schneider, N., Schuster, S., Seddah, D., Seeker, W., Ser-

- aji, M., Shen, M., Shimada, A., Shohibussirri, M., Sichinava, D., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Smith, A., Soares-Bastos, I., Stella, A., Straka, M., Strnadová, J., Suhr, A., Sulubacak, U., Szántó, Z., Taji, D., Takahashi, Y., Tanaka, T., Tellier, I., Trosterud, T., Trukhina, A., Tsarfaty, R., Tyers, F., Uematsu, S., Urešová, Z., Uria, L., Uszkoreit, H., Vajjala, S., van Niekerk, D., van Noord, G., Varga, V., Vincze, V., Wallin, L., Washington, J. N., Williams, S., Wirén, M., Woldemariam, T., Wong, T.-s., Yan, C., Yavrumyan, M. M., Yu, Z., Žabokrtský, Z., Zeldes, A., Zeman, D., Zhang, M., and Zhu, H. (2018a). Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.
- Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).
- Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007a). The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics.
- Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007b). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.
- Nivre, J., Hall, J., Nilsson, J., Eryigit, G., and Marinov, S. (2006). Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the*

- Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City. Association for Computational Linguistics.
- Nivre, J. and Kübler, S. (2006). Dependency parsing tutorial at coling-acl, sydney 2006. <https://cl.lingfil.uu.se/~nivre/docs/ACLslides.pdf>. Accessed: 2021-11-26.
- Nivre, J., Marongiu, P., Ginter, F., Kanerva, J., Montemagni, S., Schuster, S., and Simi, M. (2018b). Enhancing Universal Dependency treebanks: A case study. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 102–107, Brussels, Belgium. Association for Computational Linguistics.
- Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.
- Oepen, S., Abend, O., Hajic, J., Hershcovich, D., Kuhlmann, M., O’Gorman, T., Xue, N., Chun, J., Straka, M., and Uresova, Z. (2019). MRP 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1–27, Hong Kong. Association for Computational Linguistics.
- Ortiz Suárez, P. J., Sagot, B., and Romary, L. (2019). Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In Bański, P., Barbaresi, A., Biber, H., Breiteneder, E., Clematide, S., Kupietz, M., Lungen, H., and Iliadi, C., editors, *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, pages 9 – 16, Cardiff, United Kingdom. Leibniz-Institut für Deutsche Sprache.
- 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.

- Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana, USA. Association for Computational Linguistics.
- Petrov, S., Das, D., and McDonald, R. (2012). A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).
- Plank, B. and Agić, Ž. (2018). Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 614–620, Brussels, Belgium. Association for Computational Linguistics.
- Plank, B., Søgaard, A., and Goldberg, Y. (2016). Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.
- Pyysalo, S., Kanerva, J., Virtanen, A., and Ginter, F. (2020). WikiBERT models: deep transfer learning for many languages. arXiv 2006.01538v1.
- Pyysalo, S., Kanerva, J., Virtanen, A., and Ginter, F. (2021). WikiBERT models: Deep transfer learning for many languages. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 1–10, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.
- Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A python nat-

- ural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Preprint on Paper with Code.
- Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.
- Rönnqvist, S., Kanerva, J., Salakoski, T., and Ginter, F. (2019). Is multilingual BERT fluent in language generation? In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 29–36, Turku, Finland. Linköping University Electronic Press.
- Rosa, R. and Mareček, D. (2018). CUNI x-ling: Parsing under-resourced languages in CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 187–196, Brussels, Belgium. Association for Computational Linguistics.
- Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2017). Sluice networks: Learning what to share between loosely related tasks. *ArXiv*, abs/1705.08142.
- Ruder, S. and Plank, B. (2018). Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Com-*

- putational Linguistics (Volume 1: Long Papers)*, pages 1044–1054, Melbourne, Australia. Association for Computational Linguistics.
- Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., and Gurevych, I. (2020). How good is your tokenizer? on the monolingual performance of multilingual language models. arXiv 2012.15613v2.
- Sagae, K. and Lavie, A. (2006). Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA. Association for Computational Linguistics.
- Sato, M., Manabe, H., Noji, H., and Matsumoto, Y. (2017). Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79. Association for Computational Linguistics.
- Savary, A., Ramisch, C., Cordeiro, S., Sangati, F., Vincze, V., QasemiZadeh, B., Candito, M., Cap, F., Giouli, V., Stoyanova, I., and Doucet, A. (2017). The PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 31–47, Valencia, Spain. Association for Computational Linguistics.
- Scao, T. L., Wang, T., Hesslow, D., Saulnier, L., Bekman, S., Bari, S., Biderman, S. R., ElSahar, H., Phang, J., Press, O., Raffel, C., Sanh, V., Shen, S., Sutawika, L. A., Tae, J., Yong, Z. X., Launay, J., and Beltagy, I. (2022). What language model to train if you have one million gpu hours?
- Schlichtkrull, M. and Søgaard, A. (2017). Cross-lingual dependency parsing with late decoding for truly low-resource languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 220–229, Valencia, Spain. Association for Computational Linguistics.

- Schneider, N., Danchik, E., Dyer, C., and Smith, N. A. (2014). Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.
- Schuster, S. and Manning, C. D. (2016). Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).
- Schuster, T., Ram, O., Barzilay, R., and Globerson, A. (2019). Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. 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 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.
- Shi, T. (2021). *Enhanced Representations and Efficient Analysis of Syntactic Dependencies Within and Beyond Tree Structures*. Thesis, Cornell University.
- Shi, T. and Lee, L. (2021). TGIF: Tree-graph integrated-format parser for enhanced UD with two-stage generic- to individual-language finetuning. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 213–224, Online. Association for Computational Linguistics.
- Shi, T., Wu, F. G., Chen, X., and Cheng, Y. (2017). Combining global models for parsing universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39. Association for Computational Linguistics.

- Smith, A., Bohnet, B., de Lhoneux, M., Nivre, J., Shao, Y., and Stymne, S. (2018). 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.
- Straka, M., Hajič, J., and Straková, J. (2016). UDPipe: Trainable pipeline for processing CoNLL-u files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).
- Straka, M. and Straková, J. (2017). Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.
- Straka, M. and Straková, J. (2019). ÚFAL MRPipe at MRP 2019: UDPipe goes semantic in the meaning representation parsing shared task. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 127–137, Hong Kong. Association for Computational Linguistics.
- Straka, M., Straková, J., and Hajic, J. (2019). Evaluating contextualized embeddings on 54 languages in pos tagging, lemmatization and dependency parsing. *ArXiv*, abs/1908.07448.
- Strubell, E., Verga, P., Andor, D., Weiss, D., and McCallum, A. (2018). Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference*

- on *Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.
- Stymne, S., de Lhoneux, M., Smith, A., and Nivre, J. (2018). Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625. Association for Computational Linguistics.
- Sun, K., Zhang, R., Mensah, S., Mao, Y., and Liu, X. (2019). Aspect-level sentiment analysis via convolution over dependency tree. 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 5679–5688, Hong Kong, China. Association for Computational Linguistics.
- Tenney, I., Das, D., and Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.
- Tesnière, L. (1959). *Éléments de syntaxe structurale*.
- Tiedemann, J. (2016). Opus-parallel corpora for everyone. *Baltic Journal of Modern Computing*, page 384.
- Tiedemann, J. and Agić, Ž. (2016). Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research*, 5.
- Tiedemann, J. and Thottingal, S. (2020). OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.
- Tsarfaty, R., Seddah, D., Goldberg, Y., Kuebler, S., Versley, Y., Candito, M., Foster, J., Rehebein, I., and Tounsi, L. (2010). Statistical parsing of morphologically rich languages

- (SPMRL) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, California, USA. Association for Computational Linguistics.
- Tyers, F., Sheyanova, M., Martynova, A., Stepachev, P., and Vinogrodskiy, K. (2018). Multi-source synthetic treebank creation for improved cross-lingual dependency parsing. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 144–150, Brussels, Belgium. Association for Computational Linguistics.
- Üstün, A., Bisazza, A., Bouma, G., and van Noord, G. (2020). UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.
- van der Goot, R. and de Lhoneux, M. (2021). Parsing with pretrained language models, multiple datasets, and dataset embeddings. In *Proceedings of the 20th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2021)*, pages 96–104, Sofia, Bulgaria. Association for Computational Linguistics.
- van der Goot, R., Üstün, A., and Plank, B. (2021). On the effectiveness of dataset embeddings in mono-lingual, multi-lingual and zero-shot conditions. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 183–194, Kyiv, Ukraine. Association for Computational Linguistics.
- Vania, C., Grivas, A., and Lopez, A. (2018). What do character-level models learn about morphology? the case of dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2583, Brussels, Belgium. Association for Computational Linguistics.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st Interna-*

tional Conference on Neural Information Processing Systems, NIPS'17, pages 6000–6010, USA. Curran Associates Inc.

Vilares, D., Gómez-Rodríguez, C., and Alonso, M. A. (2016). One model, two languages: training bilingual parsers with harmonized treebanks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 425–431, Berlin, Germany. Association for Computational Linguistics.

Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F., and Pyysalo, S. (2019). Multilingual is not enough: Bert for finnish. *ArXiv*, abs/1912.07076.

Vulić, I., Glavaš, G., Reichart, R., and Korhonen, A. (2019). Do we really need fully unsupervised cross-lingual embeddings? 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 4407–4418, Hong Kong, China. Association for Computational Linguistics.

Wagner, J., Barry, J., and Foster, J. (2020). Treebank embedding vectors for out-of-domain dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8812–8818, Online. Association for Computational Linguistics.

Walsh, A., Lynn, T., and Foster, J. (2019). Ilfhocail: A lexicon of Irish MWEs. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 162–168, Florence, Italy. Association for Computational Linguistics.

Walsh, A., Lynn, T., and Foster, J. (2020). Annotating verbal MWEs in Irish for the PARSEME shared task 1.2. In *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pages 58–65, online. Association for Computational Linguistics.

- Wang, X., Jiang, Y., and Tu, K. (2020a). Enhanced Universal Dependency parsing with second-order inference and mixture of training data. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 215–220, Online. Association for Computational Linguistics.
- Wang, Z., Dai, Z., Póczos, B., and Carbonell, J. G. (2019). Characterizing and avoiding negative transfer. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11285–11294.
- Wang, Z., Lipton, Z. C., and Tsvetkov, Y. (2020b). On negative interference in multilingual models: Findings and a meta-learning treatment. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4438–4450, Online. Association for Computational Linguistics.
- Wu, S. and Dredze, M. (2019). Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China. Association for Computational Linguistics.
- Wu, S. and Dredze, M. (2020). Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.
- Xu, C., Tao, D., and Xu, C. (2013). A survey on multi-view learning. *ArXiv*, abs/1304.5634.
- Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France.

- Yarmohammadi, M., Wu, S., Marone, M., Xu, H., Ebner, S., Qin, G., Chen, Y., Guo, J., Harman, C., Murray, K., White, A. S., Dredze, M., and Van Durme, B. (2021). Everything is all it takes: A multipronged strategy for zero-shot cross-lingual information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1950–1967, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Yarowsky, D. and Ngai, G. (2001). Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.
- Yarowsky, D., Ngai, G., and Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.
- Zeman, D. (2008). Reusable tagset conversion using tagset drivers. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).
- Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., and Petrov, S. (2018). CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.
- Zeman, D., Nivre, J., Abrams, M., Ackermann, E., Aepli, N., Aghaei, H., Agić, Ž., Ahmadi, A., Ahrenberg, L., Ajede, C. K., Aleksandravičiūtė, G., Alfina, I., Antonsen, L., Aplonova, K., Aquino, A., Aragon, C., Aranzabe, M. J., Arican, B. N., Arnardóttir, H., Arutie, G., Arwidarasti, J. N., Asahara, M., Aslan, D. B., Ateyah, L., Atmaca,

F., Attia, M., Atutxa, A., Augustinus, L., Badmaeva, E., Balasubramani, K., Balles-
teros, M., Banerjee, E., Bank, S., Barbu Mititelu, V., Barkarson, S., Basmov, V., Batch-
elor, C., Bauer, J., Bedir, S. T., Bengoetxea, K., Berk, G., Berzak, Y., Bhat, I. A., Bhat,
R. A., Biagetti, E., Bick, E., Bielinskienė, A., Bjarnadóttir, K., Blokland, R., Bobicev, V.,
Boizou, L., Borges Völker, E., Börstell, C., Bosco, C., Bouma, G., Bowman, S., Boyd, A.,
Braggaar, A., Brokaitė, K., Burchardt, A., Candito, M., Caron, B., Caron, G., Cassidy,
L., Cavalcanti, T., Cebiroğlu Eryiğit, G., Cecchini, F. M., Celano, G. G. A., Čéplö, S.,
Cesur, N., Cetin, S., Çetinoğlu, Ö., Chalub, F., Chauhan, S., Chi, E., Chika, T., Cho,
Y., Choi, J., Chun, J., Cignarella, A. T., Cinková, S., Collomb, A., Çöltekin, Ç., Connor,
M., Courtin, M., Cristescu, M., Daniel, P., Davidson, E., de Marneffe, M.-C., de Paiva,
V., Derin, M. O., de Souza, E., Diaz de Ilarraza, A., Dickerson, C., Dinakaramani, A.,
Di Nuovo, E., Dione, B., Dirix, P., Dobrovoljc, K., Dozat, T., Droганova, K., Dwivedi,
P., Eckhoff, H., Eiche, S., Eli, M., Elkahky, A., Ephrem, B., Erina, O., Erjavec, T., Eti-
enne, A., Evelyn, W., Facundes, S., Farkas, R., Fernanda, M., Fernandez Alcalde, H.,
Foster, J., Freitas, C., Fujita, K., Gajdošová, K., Galbraith, D., Garcia, M., Gärdenfors,
M., Garza, S., Gerardi, F. F., Gerdes, K., Ginter, F., Godoy, G., Goenaga, I., Gojenola, K.,
Gökırmak, M., Goldberg, Y., Gómez Guinovart, X., González Saavedra, B., Griciūtė,
B., Grioni, M., Grobol, L., Grūzītis, N., Guillaume, B., Guillot-Barbance, C., Güngör,
T., Habash, N., Hafsteinsson, H., Hajič, J., Hajič jr., J., Hämäläinen, M., Hà Mỹ, L., Han,
N.-R., Hanifmuti, M. Y., Hardwick, S., Harris, K., Haug, D., Heinecke, J., Hellwig, O.,
Hennig, F., Hladká, B., Hlaváčová, J., Hociung, F., Hohle, P., Huber, E., Hwang, J.,
Ikeda, T., Ingason, A. K., Ion, R., Irimia, E., Ishola, O., Ito, K., Jelínek, T., Jha, A., Jo-
hannsen, A., Jónsdóttir, H., Jørgensen, F., Juutinen, M., K, S., Kaşıkara, H., Kaasen,
A., Kabaeva, N., Kahane, S., Kanayama, H., Kanerva, J., Kara, N., Katz, B., Kayade-
len, T., Kenney, J., Kettnerová, V., Kirchner, J., Klementieva, E., Köhn, A., Köksal, A.,
Kopacewicz, K., Korkiakangas, T., Kotsyba, N., Kovalevskaitė, J., Krek, S., Krishna-
murthy, P., Kuyrukçu, O., Kuzgun, A., Kwak, S., Laippala, V., Lam, L., Lambertino,

L., Lando, T., Larasati, S. D., Lavrentiev, A., Lee, J., Lê Hồng, P., Lenci, A., Lertpradit, S., Leung, H., Levina, M., Li, C. Y., Li, J., Li, K., Li, Y., Lim, K., Lima Padovani, B., Lindén, K., Ljubešić, N., Loginova, O., Luthfi, A., Luukko, M., Lyashevskaya, O., Lynn, T., Macketanz, V., Makazhanov, A., Mandl, M., Manning, C., Manurung, R., Marşan, B., Mărănduc, C., Mareček, D., Marheinecke, K., Martínez Alonso, H., Martins, A., Mašek, J., Matsuda, H., Matsumoto, Y., Mazzei, A., McDonald, R., McGuinness, S., Mendonça, G., Miekka, N., Mischenkova, K., Misirpashayeva, M., Missilä, A., Mititelu, C., Mitrofan, M., Miyao, Y., Mojiri Ferooshani, A., Molnár, J., Moloodi, A., Montemagni, S., More, A., Moreno Romero, L., Moretti, G., Mori, K. S., Mori, S., Morioka, T., Moro, S., Mortensen, B., Moskalevskiy, B., Muischnek, K., Munro, R., Murawaki, Y., Müürisep, K., Nainwani, P., Nakhlé, M., Navarro Horñiacek, J. I., Nedoluzhko, A., Nešpore-Bērzkalne, G., Nevaci, M., Nguyễn Thị, L., Nguyễn Thị Minh, H., Nikaido, Y., Nikolaev, V., Nitisaraj, R., Nourian, A., Nurmi, H., Ojala, S., Ojha, A. K., Olúòkun, A., Omura, M., Onwuegbuzia, E., Osenova, P., Östling, R., Øvrelid, L., Özateş, Ş. B., Özçelik, M., Özgür, A., Öztürk Başaran, B., Park, H. H., Partanen, N., Pascual, E., Passarotti, M., Patejuk, A., Paulino-Passos, G., Peljak-Łapińska, A., Peng, S., Perez, C.-A., Perkova, N., Perrier, G., Petrov, S., Petrova, D., Phelan, J., Piitulainen, J., Piriinen, T. A., Pitler, E., Plank, B., Poibeau, T., Ponomareva, L., Popel, M., Pretkalniņa, L., Prévost, S., Prokopidis, P., Przepiórkowski, A., Puolakainen, T., Pyysalo, S., Qi, P., Rääbis, A., Rademaker, A., Rama, T., Ramasamy, L., Ramisch, C., Rashel, F., Rasooli, M. S., Ravishankar, V., Real, L., Rebeja, P., Reddy, S., Rehm, G., Riabov, I., Rießler, M., Rimkutė, E., Rinaldi, L., Rituma, L., Rocha, L., Rögnvaldsson, E., Romanenko, M., Rosa, R., Roşca, V., Rovati, D., Rudina, O., Rueter, J., Rúnarsson, K., Sadde, S., Safari, P., Sagot, B., Sahala, A., Saleh, S., Salomoni, A., Samardžić, T., Samson, S., Sanguinetti, M., Saniyar, E., Särg, D., Saulite, B., Sawanakunanon, Y., Saxena, S., Scannell, K., Scarlata, S., Schneider, N., Schuster, S., Schwartz, L., Seddah, D., Seeker, W., Seraji, M., Shen, M., Shimada, A., Shirasu, H., Shishkina, Y., Shohibussirri, M., Sichinava, D.,

Siewert, J., Sigurðsson, E. F., Silveira, A., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Skachedubova, M., Smith, A., Soares-Bastos, I., Spadine, C., Sprugnoli, R., Steingrímsson, S., Stella, A., Straka, M., Strickland, E., Strnadová, J., Suhr, A., Sulestio, Y. L., Sulubacak, U., Suzuki, S., Szántó, Z., Taji, D., Takahashi, Y., Tamburini, F., Tan, M. A. C., Tanaka, T., Tella, S., Tellier, I., Testori, M., Thomas, G., Torga, L., Toska, M., Trosterud, T., Trukhina, A., Tsarfaty, R., Türk, U., Tyers, F., Uematsu, S., Untilov, R., Urešová, Z., Uria, L., Uszkoreit, H., Utká, A., Vajjala, S., van der Goot, R., Vanhove, M., van Niekerk, D., van Noord, G., Varga, V., Villemonte de la Clergerie, E., Vincze, V., Vlasova, N., Wakasa, A., Wallenberg, J. C., Wallin, L., Walsh, A., Wang, J. X., Washington, J. N., Wendt, M., Widmer, P., Williams, S., Wirén, M., Wittern, C., Woldemariam, T., Wong, T.-s., Wróblewska, A., Yako, M., Yamashita, K., Yamazaki, N., Yan, C., Yasuoka, K., Yavrumyan, M. M., Yenice, A. B., Yıldız, O. T., Yu, Z., Žabokrtský, Z., Zahra, S., Zeldes, A., Zhu, H., Zhuravleva, A., and Ziane, R. (2021). Universal dependencies 2.8. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Zeman, D., Nivre, J., Abrams, M., Ackermann, E., Aeppli, N., Aghaei, H., Agić, Ž., Ahmadi, A., Ahrenberg, L., Ajede, C. K., Aleksandravičiūtė, G., Alfina, I., Antonsen, L., Aplonova, K., Aquino, A., Aragon, C., Aranzabe, M. J., Arnardóttir, H., Arutie, G., Arwidarasti, J. N., Asahara, M., Ateyah, L., Atmaca, F., Attia, M., Atutxa, A., Augustinus, L., Badmaeva, E., Balasubramani, K., Ballesteros, M., Banerjee, E., Bank, S., Barbu Mititelu, V., Basmov, V., Batchelor, C., Bauer, J., Bedir, S. T., Bengoetxea, K., Berk, G., Berzak, Y., Bhat, I. A., Bhat, R. A., Biagetti, E., Bick, E., Bielinskienė, A., Bjarnadóttir, K., Blokland, R., Bobicev, V., Boizou, L., Borges Völker, E., Börstell, C., Bosco, C., Bouma, G., Bowman, S., Boyd, A., Brokaitė, K., Burchardt, A., Candito, M., Caron, B., Caron, G., Cavalcanti, T., Cebiroğlu Eryiğit, G., Cecchini, F. M., Celano, G. G. A., Čéplö, S., Cetin, S., Çetinoğlu, Ö., Chalub, F., Chi, E., Cho, Y., Choi, J., Chun,

J., Cignarella, A. T., Cinková, S., Collomb, A., Çöltekin, Ç., Connor, M., Courtin, M., Davidson, E., de Marneffe, M.-C., de Paiva, V., Derin, M. O., de Souza, E., Diaz de Ilaraza, A., Dickerson, C., Dinakaramani, A., Dione, B., Dirix, P., Dobrovoljc, K., Dozat, T., Droганova, K., Dwivedi, P., Eckhoff, H., Eli, M., Elkahky, A., Ephrem, B., Erina, O., Erjavec, T., Etienne, A., Evelyn, W., Facundes, S., Farkas, R., Fernanda, M., Fernandez Alcalde, H., Foster, J., Freitas, C., Fujita, K., Gajdošová, K., Galbraith, D., Garcia, M., Gärdenfors, M., Garza, S., Gerardi, F. F., Gerdes, K., Ginter, F., Goenaga, I., Gojenola, K., Gökırmak, M., Goldberg, Y., Gómez Guinovart, X., González Saavedra, B., Griciūtė, B., Gritti, M., Grobol, L., Grūzītis, N., Guillaume, B., Guillot-Barbance, C., Güngör, T., Habash, N., Hafsteinsson, H., Hajič, J., Hajič jr., J., Hämäläinen, M., Hà Mỹ, L., Han, N.-R., Hanifmuti, M. Y., Hardwick, S., Harris, K., Haug, D., Heinecke, J., Hellwig, O., Hennig, F., Hladká, B., Hlaváčová, J., Hociung, F., Hohle, P., Huber, E., Hwang, J., Ikeda, T., Ingason, A. K., Ion, R., Irimia, E., Ishola, O., Jelínek, T., Johannsen, A., Jónsdóttir, H., Jørgensen, F., Juutinen, M., K, S., Kaşıkara, H., Kaasen, A., Kabaeva, N., Kahane, S., Kanayama, H., Kanerva, J., Katz, B., Kayadelen, T., Kenney, J., Kettnerová, V., Kirchner, J., Klementieva, E., Köhn, A., Köksal, A., Kopacewicz, K., Korkiakangas, T., Kotsyba, N., Kovalevskaitė, J., Krek, S., Krishnamurthy, P., Kwak, S., Laippala, V., Lam, L., Lambertino, L., Lando, T., Larasati, S. D., Lavrentiev, A., Lee, J., Lê Hồng, P., Lenci, A., Lertpradit, S., Leung, H., Levina, M., Li, C. Y., Li, J., Li, K., Li, Y., Lim, K., Lindén, K., Ljubešić, N., Loginova, O., Luthfi, A., Luukko, M., Lyashevskaya, O., Lynn, T., Macketanz, V., Makazhanov, A., Mandl, M., Manning, C., Manurung, R., Mărănduc, C., Mareček, D., Marheinecke, K., Martínez Alonso, H., Martins, A., Mašek, J., Matsuda, H., Matsumoto, Y., McDonald, R., McGuinness, S., Mendonça, G., Miekka, N., Mischenkova, K., Misirpashayeva, M., Missilä, A., Mititelu, C., Mitrofan, M., Miyao, Y., Mojiri Ferooshani, A., Moloodi, A., Montemagni, S., More, A., Moreno Romero, L., Mori, K. S., Mori, S., Morioka, T., Moro, S., Mortensen, B., Moskalevskiy, B., Muischnek, K., Munro, R., Murawaki, Y., Müürisep, K., Nain-

wani, P., Nakhlé, M., Navarro Horñiacek, J. I., Nedoluzhko, A., Nešpore-Bērzkalne, G., Nguyễn Thị, L., Nguyễn Thị Minh, H., Nikaido, Y., Nikolaev, V., Nitisaraj, R., Nourian, A., Nurmi, H., Ojala, S., Ojha, A. K., Olúòkun, A., Omura, M., Onwuegbuzia, E., Osenova, P., Östling, R., Øvreid, L., Özateş, Ş. B., Özgür, A., Öztürk Başaran, B., Partanen, N., Pascual, E., Passarotti, M., Patejuk, A., Paulino-Passos, G., Peljak-Lapińska, A., Peng, S., Perez, C.-A., Perkova, N., Perrier, G., Petrov, S., Petrova, D., Phelan, J., Piitulainen, J., Pirinen, T. A., Pitler, E., Plank, B., Poibeau, T., Ponomareva, L., Popel, M., Pretkalniņa, L., Prévost, S., Prokopidis, P., Przepiórkowski, A., Puolakainen, T., Pyysalo, S., Qi, P., Rääbis, A., Rademaker, A., Rama, T., Ramasamy, L., Ramisch, C., Rashel, F., Rasooli, M. S., Ravishankar, V., Real, L., Rebeja, P., Reddy, S., Rehm, G., Riabov, I., Rießler, M., Rimkutė, E., Rinaldi, L., Rituma, L., Rocha, L., Rögnvaldsson, E., Romanenko, M., Rosa, R., Roşca, V., Rovati, D., Rudina, O., Rueter, J., Rúnarsson, K., Sadde, S., Safari, P., Sagot, B., Sahala, A., Saleh, S., Salomoni, A., Samardžić, T., Samson, S., Sanguinetti, M., Särg, D., Saulīte, B., Sawanakunanon, Y., Scannell, K., Scarlata, S., Schneider, N., Schuster, S., Seddah, D., Seeker, W., Seraji, M., Shen, M., Shimada, A., Shirasu, H., Shohibussirri, M., Sichinava, Dmitry Simionescu, R., Simkó, K., Šimková, M., Simov, K., Skachedubova, M., Smith, A., Soares-Bastos, I., Spadine, C., Steingrímsson, S., Stella, A., Straka, M., Strickland, E., Strnadová, J., Suhr, A., Sulestio, Y. L., Sulubacak, U., Suzuki, S., Szántó, Z., Taji, D., Takahashi, Y., Tamburini, F., Tan, M. A. C., Tanaka, T., Tella, S., Tellier, I., Thomas, G., Torga, L., Toska, M., Trosterud, T., Trukhina, A., Tsarfaty, R., Türk, U., Tyers, F., Uematsu, S., Untilov, R., Urešová, Z., Uria, L., Uszkoreit, H., Utká, A., Vajjala, S., van Niekerk, D., van Noord, G., Varga, V., Villemonte de la Clergerie, E., Vincze, V., Wakasa, A., Wallenberg, J. C., Wallin, L., Walsh, A., Wang, J. X., Washington, J. N., Wendt, M., Widmer, P., Williams, S., Wirén, M., Wittern, C., Woldemariam, T., Wong, T.-s., Wróblewska, A., Yako, M., Yamashita, K., Yamazaki, N., Yan, C., Yasuoka, K., Yavrumyan, M. M., Yu, Z., Žabokrtský, Z., Zahra, S., Zeldes, A., Zhu, H., and Zhuravleva, A. (2020). Univer-

sal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., dePaiva, V., Droганova, K., Martínez Alonso, H., Çöltekin, c., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R., and Li, J. (2017). Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Zeman, D. and Resnik, P. (2008). Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Zhang, X., Cheng, J., and Lapata, M. (2017). Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2008). A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii. Association for Computational Linguistics.

Zhao, J., Xie, X., Xu, X., and Sun, S. (2017). Multi-view learning overview: Recent progress and new challenges. *Inf. Fusion*, 38:43–54.