Dependency parsing as sequence labeling for low-resource languages

Trabajo Fin de Máster

Alberto Muñoz Ortiz

Trabajo de investigación para el

Máster en Lenguajes y Sistemas Informáticos

Universidad Nacional de Educación a Distancia

Dirigido por:

Dra. Dª. Lourdes Araujo Serna

Dr. D. David Vilares Calvo

Octubre 2021

Agradecimientos

Gracias a Lourdes por ser mi tutora y guiarme durante la creación de este trabajo; gracias a David por todo su apoyo y por darme la oportunidad de trabajar juntos.

Financiación Este trabajo está apoyado por una Beca Leonardo 2020 para Investigadores y Creadores Culturales de la FBBVA. La FBBVA no se hace responsable de las opiniones, afirmaciones y contenidos incluidos en el proyecto y/o de los resultados del mismo, que son de la entera responsabilidad de los autores.

Resumen

El procesamiento de lenguaje natural (PLN) ha experimentado claros avances en los últimos años. Sin embargo, la mayoría de mejoras y estudios se han centrado en un selecto grupo de idiomas, siendo el inglés su principal representante, ignorando cómo funcionan estos métodos en idiomas menos privilegiados, que normalmente reciben el nombre de idiomas con pocos recursos.

Este trabajo trata sobre idiomas con pocos recursos, y se centra en una tarea central de PLN conocida como análisis sintáctico de dependencias; ésta consiste en analizar automáticamente la estructura sintáctica de dependencias de una oración, conectando sus palabras mediante relaciones asimétricas binarias entre una palabra gobernante y una palabra subordinada sintácticamente. En concreto, nuestra contribución se encuentra en la intersección entre la velocidad de análisis e idiomas con pocos recursos. En este contexto, recientemente se ha propuesto realizar el análisis de dependencias como una tarea de etiquetado de secuencias. Este enfoque computa un árbol linealizado de n etiquetas dada una frase de longitud n, y otorga una buena relación entre velocidad y precisión. Además, ofrece una forma sencilla de incorporar información sintáctica como una *word embedding* o característica de entrada.

En primer lugar, comparamos el rendimiento de cinco linealizaciones para análisis de dependencias como etiquetado de secuencias en escenarios con pocos recursos. Estas linealizaciones pertenecen a diferentes familias y proponen formular el problema como: (i) seleccionar el gobernante sintáctico para cada palabra, (ii) encontrar una representación de los arcos entre *tokens* utilizando paréntesis equilibrados y (iii) asociar a cada *token* subsecuencias de transiciones de un analizador basado en transiciones. Sin embargo, aún existe poco conocimiento sobre cómo se comportan estas linealizaciones en configuraciones con pocos recursos. En este trabajo, primero estudiamos su

nivel de eficiencia, simulando configuraciones con datos restringidos partiendo de un conjunto diverso de *treebanks* con muchos recursos. Los resultados muestran que las codificaciones de selección del gobernante sintáctico son más eficientes y obtienen mejores resultados en condiciones ideales (*gold*), pero que esta ventaja se desvanece en favor de las estrategias de paréntesis equilibrados cuando la configuración utilizada es más similar a una configuración realista, como la esperada en idiomas con realmente pocos recursos.

En segundo lugar, proponemos un método basado en morfología combinado con aprendizaje translingüe para intentar mejorar el rendimiento del análisis de dependencias en idiomas con pocos recursos. Para ello, primero entrenamos un sistema de flexión morfológica para idiomas objetivo con pocos recursos, y después lo aplicamos a *treebanks* con muchos recursos de idiomas similares para crear un *treebank* flexionado translingüe (o *x-inflected treebank*) que se asemeje al idioma con pocos recursos objetivo. A continuación, utilizamos los *treebanks* flexionados para entrenar los analizadores sintácticos de etiquetado de secuencias en dos escenarios: (i) un escenario *zero-shot* (entrenando un modelo en el *x-inflected treebank* y ejecutándolo sobre el idioma objetivo), y (ii) un escenario *few-shot* (entrenando un modelo utilizando un grupo compuesto por *x-inflected treebank* junto con el *treebank* con pocos recursos y ejecutándolos sobre el idioma objetivo). Nuestro objetivo es comprobar la utilidad del método propuesto en situaciones con distinta disponibilidad de datos anotados. Los resultados muestran que el método propuesto puede ser de ayuda en algunas situaciones, pero se necesita estudiar más en profundidad para entender cómo los distintos factores pueden afectar a los resultados y comprobar si estas tendencias se mantienen usando otros paradigmas, como analizadores basados en transiciones y basados en grafos.

Abstract

Natural Language Processing (NLP) has achieved clear improvements in recent years. However, most improvements and studies have been centered in a selected group of languages, being English its main representative, ignoring how these methods perform on less privileged languages, usually labeled as low-resource languages.

This work is on low-resource languages, and focuses on a core NLP task known as dependency parsing; that consists in analyzing automatically the dependency structure of a sentence, connecting the words of the sentence in pairs by asymmetric relations between a parent word and a syntactically subordinate word. More particularly, our contribution lies in the intersection between fast parsing and low-resource languages. In this context, recent work has proposed to cast dependency parsing as sequence labeling. This approach computes a linearized tree of n labels given a sentence of length n, and provides a good speed/accuracy trade-off. Also, it offers a naïve way to infuse syntactic information as an embedding or feature.

First, we compare the performance of five linearizations for dependency parsing as sequence labeling in low-resource scenarios. These linearizations belong to different families and address the task as: (i) a head selection problem, (ii) finding a representation of the token arcs as bracket strings, or (iii) associating transition subsequences of a transition-based parser to words. Yet, there is little understanding about how these linearizations behave in low-resource setups. Here, we first study their data efficiency, simulating data-restricted setups from a diverse set of rich-resource treebanks. After that, we test whether such differences manifest in truly low-resource setups. The results show that head selection encodings are more data-efficient and perform better in an ideal (gold) framework, but that such advantage greatly vanishes in favour of bracketing formats when the running setup resembles a real-world low-resource configuration.

Second, we propose a morphology-based method combined with cross-lingual learning to try to improve parsing performance for low-resource languages. To do so, we first train a morphological inflection system for low-resource target languages, and then apply it to rich-resource treebanks from similar languages, to create a cross-lingual inflected treebank (or *x-inflected treebank*) that resembles the target low-resource language. Then, we use these inflected treebanks to train sequence labeling parsers in two scenarios: (i) a zero-shot scenario (training on the x-inflected treebank and testing on the target language), and (ii) a few-shot scenario (training on a group of x-inflected treebanks together with the low-resource treebank and testing on the target language). Our goal is to test the usefulness of this method in situations with different availability of annotated data. The results show that the proposed method can be helpful in some situations, but further work is required to understand how different factors affect the results and check if these trends hold when using other parsing paradigms, such as transition-based and graph-based parsers.

Contents

Index of figures

Index of tables

Chapter 1

Introduction

1.1. Motivation

Natural Language Processing (NLP) has achieved clear improvements in recent years, thanks to aspects such as the availability of large annotated datasets, high computational capacities, and the adoption of deep learning models for processing natural texts. However, these improvements have been centered in those languages with enough available data (both raw and annotated), known as rich-resource languages (RRLs), being English the most representative one. This handful of languages represent only a tiny fraction of the around 7 000 known languages in the world. Furthermore, these languages are from a reduced number of families and geographical areas, so they do not account as a meaningful set of the world languages as a whole and lack many existing typological features (Joshi et al., 2020).

In contrast, the vast majority of the world languages can be labeled as low-resource languages (LRLs). This classification is imprecise, as many groups of languages can fit this definition: e.g. less computerized languages, less privileged languages, less studied languages, or languages that do not have enough data to apply statistical methods successfully (Magueresse, Carles, and Heetderks, 2020). For these reasons, methods and techniques that work well for RRLs might not necessary perform the same way on languages with smaller amounts of data or different typology. This, *inter alia*, exemplifies the need to work on NLP besides the privileged group of RRLs (Ruder, 2020).

This work on low-resource languages focuses on dependency parsing (Kübler, McDonald, and Nivre, 2009; Mel'cuk and others, 1988), a core NLP

task. Dependency parsing consists in analyzing the dependency structure of a sentence automatically. It is driven by the assumption that syntactic structure is made of words connected in pairs by asymmetric relations between a parent word (*head*) and a syntactically subordinate word (*dependent*). Each link has a *dependency type* assigned to it (e.g. *subject* or *attribute*), showing the relationship type between both words.

Parsing has been traditionally helpful for different NLP tasks. Among the different existing parsing representations, dependency parsing can be suitable for certain downstream tasks like sentiment analysis (Vilares, Gómez-Rodríguez, and Alonso, 2017), text summarization (Balachandran et al., 2020), machine translation (Aharoni and Goldberg, 2017) or question answering (Cao et al., 2019).

In this context, popular approaches such as graph-based parsers (Martins, Almeida, and Smith, 2013; Dozat, Qi, and Manning, 2017) or transition-based parsers (Ma et al., 2018; Fernández-González and Gómez-Rodríguez, 2019) have achieved a performance comparable to expert human annotators in certain domains like English news (Berzak et al., 2016).

Hence, recent effort has been directed to solve other parsing problems too, such as parsing different domains or multi-lingual scenarios (Sato et al., 2017; Song et al., 2019; Ammar et al., 2016), creating faster models (Volokh, 2013; Chen and Manning, 2014), designing low-resource and cross-lingual parsing techniques (Tiedemann, Agić, and Nivre, 2014; Zhang, Zhang, and Fu, 2019), or infusing syntactic knowledge into models (Strubell et al., 2018; Rotman and Reichart, 2019).

Our contribution is located at the crossroads between fast parsing and low-resource languages. Recent work has proposed different linearizations to cast parsing as a sequence labeling task (Spoustová and Spousta, 2010; Strzyz, Vilares, and Gómez-Rodríguez, 2019; Gómez-Rodríguez, Strzyz, and Vilares, 2020; Li et al., 2018; Kiperwasser and Ballesteros, 2018). Broadly speaking, sequence labeling is a prediction problem where every input token is assigned *one* output label. Many NLP tasks like named entity recognition or part-of-speech (PoS) tagging has been cast as a sequence labeling task (Bohnet et al., 2018; Liu et al., 2018a; Ma and Hovy, 2016). Dependency parsing can also be cast as a sequence labeling task, using encodings or linearizations to represent the dependency tree. Sequence labeling models have a few theoretical advantages over other parsing strategies like being simpler

and faster than other parsing models such as (Ma et al., 2018; Kiperwasser and Goldberg, 2016; Chen and Manning, 2014) while obtaining competitive results. Also, they allow to infuse syntactic information like any other feature or embedding (Ma et al., 2019; Wang et al., 2019) and can be used in any generic sequence labeling framework as any other sequence labeling task such as those mentioned above. In this context, this problem has been studied on English and multilingual setups, but there is a lack of knowledge about how it performs on low-resource languages.

1.2. Proposal and objectives

The purpose of this work is to study the task of dependency parsing as a sequence labeling task in low-resource setups. More particularly, the contribution is twofold.

The first contribution focuses on the data-efficiency aspect, showing how different linearization families require different amounts of data to start performing competitively. In particular, we tested how well the selected encodings performed with different quantities of annotated data, creating six synthetic low-resource setups from a set of rich-resource treebanks and comparing the results. Then, we reproduced the previous experiment, but under real low-resource conditions using a varied set of low-resource treebanks to check whether the trends hold when the models are evaluated on real low-resource languages.

The second contribution is a morphology-based method combined with cross-lingual learning to improve the performance of dependency parsing models for low resource languages. We trained a morphological inflection system in a few low-resource target languages using data obtained from UniMorph, and then applied it to source rich-resource treebanks from similar languages to create a new cross-lingually inflected treebank. We will refer to these treebanks as x-inflected treebanks. Then, we use the x-inflected treebanks to train sequence labeling parsers to test the usefulness of the proposed method in two different scenarios: (i) a zero-shot scenario when no training data is available and (ii) a few-shot scenario when only a few sentences are available.

1.3. Structure of the document

The rest of the work is divided as follows. Chapter 2 describes core concepts to contextualize and better understand this work. Chapter 3 discusses the data efficiency of the existing linearizations for dependency parsing as sequence labeling. Chapter 4 proposes a technique based on using cross-lingual morphological inflection to improve the results of low-resource dependency parsing. Finally, Chapter 5 concludes the thesis and discusses future work.

Chapter 2

State of the art

In this chapter, we introduce the core concepts that are necessary to better understand the thesis, and also to contextualize our work.

First, in section 2.1 we introduce the preliminary concepts about NLP to situate the reader in the project's field. Then, section 2.2 introduces the most relevant concepts about dependency parsing, including some state-of-the-art work. Next, section 2.3 presents the task of sequence labeling and relates it to dependency parsing. After that, section 2.4 introduces the task of morphological inflection, which we will be using in Chapter 4. Finally, section 2.5 comments related work in the field of low-resource parsing.

2.1. Preliminaries

Most tasks in NLP require some degree of preprocessing. In the case of dependency parsing, the minimum required preprocessing steps are segmentation and tokenization.

On the one hand, segmentation delimits the sentences that conform a text. This task is not as trivial as it might seem at first glance; although some languages like English include clear boundaries between sentences, some marks could still be ambiguous, and there are languages like Thai that do not mark sentences boundaries typographically, for instance.

On the other hand, tokenization identifies the elements that constitute a sentence. These elements are usually referred as words or tokens, and that is how we will be referring them in this work, although we are aware that for some languages, such as Japanese, the concept of word is not that well-defined and for instance the concept of phrasal units (*bunsetsu*) has a long

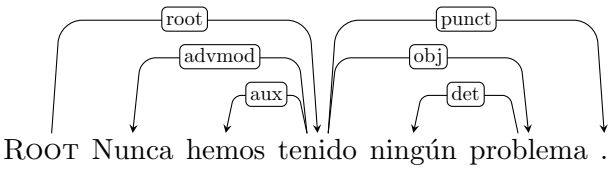

Figure 2.1: Example of a parsed sentence from the $\text{Spanish}_{\text{GSD}}$ UD treebank.

tradition (Murawaki, 2019). Overall, although this is an easy task for most languages, it is more challenging for those that do not use spaces as word separators, like Chinese.

Once the text has been segmented and tokenized, we also can obtain additional information that might benefit the parsing process. For instance, one of the most common tasks is part-of-speech (PoS) tagging. PoS tagging assigns a grammatical class (like a verb or noun) to every token in a sentence. This task must deal with aspects like lexical ambiguity or multi-word expressions, so systems must take into account the context of the whole sentence.

2.2. Dependency parsing

Dependency parsing (Kübler, McDonald, and Nivre, 2009; Mel'cuk and others, 1988) is the task of obtaining the syntactic structure of a natural language sentence automatically. In dependency parsing, the main idea is that the syntactic structure of a sentence is composed of *dependency relations*, directed links from a leading word (*head*) to its syntactically subordinate word (*dependent*), which have attached a *dependency type* that indicates the grammatical relation between both terms. An example of a dependency tree can be seen in Figure 2.2. In the figure, dependency relations are expressed according to the Universal Dependencies (Zeman, Nivre, and others, 2020) project, explained more deeply in 2.2.3, but there are different annotation frameworks that could be also used (Nivre et al., 2007; Bejček et al., 2013).

Given a finite label set $L = \{l_1, ..., l_{|L|}\}$, a *dependency graph* for a sentence $S = w_1, ..., w_n$ is a labeled directed graph $G = (V, A)$ where:

1. $V \subseteq \{0, 1, ..., n\}$ is a set of nodes, where index 0 represents an artificial root node that is usually added.

2. $A \subseteq V \times L \times V$ is a set of labeled directed arcs.

That is, a dependency graph G is a set of labeled dependency relationships between the tokens in S. The tokens are represented by the nodes in V and the relationships are represented by the arcs in A, so an arc (i, l, j) represents a dependency relationship of the type r between the head token i and the dependent token j, where $i, j \in V$ and $l \in L$. The task of a dependency parser consists in assigning an index $0, 1..., n$ to every token of the sentence S.

The goal is that the output tree is well-formed, i.e. a tree that has the following properties:

- **Root**. The tree is rooted at node 0. This artificial node is the governor of the sentence (i.e. it is the only node that does not have a head).
- **Connectedness**. There is a path connecting every two tokens when ignoring the direction of the edge.
- **Single-head**. Every token (except the root node) has one and only one head.
- **Acyclicity**. It does not contain cycles.

A more restrictive type of trees are *projective trees*. A tree is said to be *non-projective* if it contains two arcs $i \rightarrow j$ and $k \rightarrow l$ where $min(i, j) < min(k, l) < max(i, j) < max(k, l)$, and is called projective otherwise. In a more graphical way, a (non-) projective tree is a tree whose dependency arcs can (not) be drawn with none of them crossing. This can be seen in Figure 2.2.

In 2.2.1 and 2.2.2, for completeness, we briefly describe the two dominant paradigms when it comes to train dependency parsers. Then, we introduce the Universal Dependencies project in 2.2.3.

2.2.1. Transition-based parsers

Transition-based parsing algorithms define an abstract state machine where each configuration holds a structured representation together with auxiliary data structures. The system moves between states through shift-reduce actions or transitions until it finds a full parse. Although we are going to focus

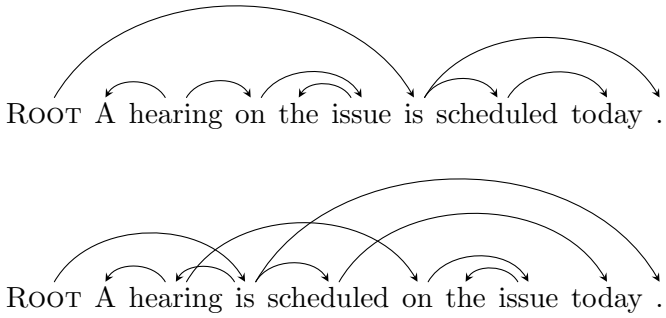

Figure 2.2: Examples of a projective and a non-projective sentence. Dependency labels are omitted.

on dependency parsing, transition-based parsers can be used in other parsing problems such as constituency parsing (Liu et al., 2018b) or semantic parsing (Liu et al., 2018b).

A transition-based dependency parser is defined as follows. Being $S = w_1, ..., w_n$ an input sentence, $\mathbb{P}_w$ is the set of possible well-formed dependency graphs. Nivre (2008) defines a transition system as a quadruple $Q = (C, T, c_s, C_t)$, where C is a set of configurations with at least a partially-built parse P_c, T is a set of transitions, c_s is an initialization function, and $C_t \subseteq C$ is a set of final configurations. Commonly, a (stack-based[1]) configuration is defined by a triplet (σ, β, A), where σ is a stack of partially-processed tokens, β is a buffer of unprocessed tokens and A is a set of dependency arcs for a dependency graph $G = (V, A)$.

A system obtains a parse $P_{cf} \in \mathbb{P}_w$ of a sentence S by applying a sequence of transitions to an initial configuration $c_s(S)$ until a final configuration $c_f \in C_t$ is achieved. To train such systems, an *oracle* is needed. Essentially, an oracle is a function that takes a given configuration and produces the next correct one. The Example 2.2.1 illustrates the parsing process of a sentence.

Example 2.2.1 (**Arc-hybrid algorithm**). The transitions defined for the stack-based *arc-hybrid* algorithm are three:

- Shift (`SH`). $(\sigma, i|\beta, A) \rightarrow (\sigma|i, \beta, A)$. It moves the first word i from the buffer to the stack.

[1]There are algorithms that employ other data structures. For example, the Covington algorithm uses two lists instead of a stack (Nivre, 2008).

- Left-arc (`LA`). $(\sigma|i, j|\beta, A) \rightarrow (\sigma, j|\beta, A \cup \{(j, l, i)\})$. It eliminates the word i from the stack and creates an arc with label l between the first word of the buffer j as head and the word i as a dependent.

- Right-arc (`RA`). $(\sigma|i|j, \beta, A) \rightarrow (\sigma|i, \beta, A \cup \{(i, l, j)\})$. It creates an arc between the second and the first word from the stack (i.e. i and j) and removes the word j from the stack.

Using these transitions, the correct sequence of transitions to obtain a dependency tree for the sentence '*Nunca hemos tenido ningún problema.*' (Figure 2.2) using the arc-hybrid algorithm is represented in Table 2.1.

T	σ	β	A
	[ROOT]	[Nunca,...,.]	$\emptyset$
`SH`	[ROOT, Nunca]	[hemos, ..., .]	$\emptyset$
`SH`	[ROOT, Nunca, hemos]	[tenido, ..., .]	$\emptyset$
`LA`	[ROOT, Nunca]	[tenido, ..., .]	A_1 ={(tenido, *aux*, hemos)}
`LA`	[ROOT]	[tenido, ..., .]	$A_2 = A_1 \cup$ {(tenido, *advmod*, Nunca)}
`SH`	[ROOT, tenido]	[ningún, ..., .]	A_2
`SH`	[ROOT, tenido, ningún]	[problema, .]	A_2
`LA`	[ROOT, tenido]	[problema, .]	$A_3 = A_2 \cup$ {(problema, *det*, ningún}
`SH`	[ROOT, tenido, problema]	[.]	A_3
`RA`	[ROOT, tenido]	[.]	$A_4 = A_3 \cup$ {(tenido, *obj*, problema)}
`SH`	[ROOT, tenido, .]	[ ]	A_4
`RA`	[ROOT, tenido]	[ ]	$A_5 = A_4 \cup$ {(tenido, *punct*, .)}
`RA`	[ROOT]	[ ]	$A_6 = A_5 \cup$ {(ROOT, *root*, tenido)}

Table 2.1: Correct sequence of transitions to obtain the dependency tree for the sentence '*Nunca hemos tenido ningún problema.*' using the arc-hybrid algorithm.

2.2.2. Graph-based parsers

Graph-based parsers use standard algorithms for directed graphs and trees. They define a space of possible dependency graphs for an input sentence, and then assign a score to each possible candidate. The model is defined from an analysis algorithm and a number of restrictions over the graph structure. More formally, we define a graph-based parser using a model $M(\Gamma, \boldsymbol{\lambda}, h)$,

being Γ a set of restrictions on allowed structures, $\boldsymbol{\lambda}$ a set of parameters and h a fixed parsing algorithm.

The assigned score represents how likely is that the selected graph is the correct one. There are different ways of scoring, depending on the algorithm used by the model. After the model is generated, the analysis phase consists in finding the graph with the highest score for the sentence. The score of a dependency tree $G = (V, A) \in G \in \mathcal{G}_S$ for a sentence S is:

$$score(G) = score(V, A) \in \mathbb{R}$$

and represents how likely is for a tree to be the right parse for S. $\mathcal{G}_S$ represents the space of dependency trees for a sentence S. The main property of graph-based parsing systems is that the score is given by:

$$score(G) = f(\psi_1, \psi_2, ..., \psi_q) \text{ for all } \psi_i \text{ in } \Psi_G$$

being f a function over the subgraphs ψ and Ψ_G the set of the relevant subgraphs of G.

A graph-based parsing system must define four things:

1. The definition of Ψ_G for a given dependency tree G.
2. The definition of the parameters $\boldsymbol{\lambda} = \{\lambda_\psi | \text{ for all } \psi \in \Psi_G \text{ for all } G \in \mathcal{G}_S \text{ for all } S\}$.
3. A method for learning $\boldsymbol{\lambda}$ from labeled data.
4. A parsing algorithm $h(S, \Gamma, \boldsymbol{\lambda}) = argmax_{G \in \mathcal{G}_S} score(G)$.

2.2.3. Universal Dependencies

Universal Dependencies[2] (Zeman, Nivre, and others, 2020, UD) is an open community project with over 300 contributors that develops cross-lingual consistent annotation between different treebanks. The annotation scheme used is based on Stanford dependencies (De Marneffe and Manning, 2008), Google universal part-of-speech tags (Petrov, Das, and McDonald, 2011), and the Interset interlingua for morphosyntatic tagsets (Zeman, 2008). Its

[2] https://univeresaldependencies.org

goal is to offer a useful linguistic representation for morphosyntactic research, semantic interpretation and natural language processing across different languages. Thus, it uses simple surface representations that allow easier parallelism between similar structures across such languages, overcoming differences in syntax and morphology.

The 2.7 version that is used for this work contains 183 treebanks for 104 languages, from dozens of linguistic families using several scripts. The treebank sizes range from less than a hundred sentences and a thousand tokens to almost 200 000 sentences and more than 3 million tokens.

The dependency treebanks use a revised version of the CoNLL-X (Buchholz and Marsi, 2006) format, called CoNLL-U. CoNLL-U files are plain text files (UTF-8) with three types of lines: comment lines starting with '`#`', blank lines separating sentences, and word lines that contain the 10 annotation fields of a token separated by single tab characters. Every sentence is composed of one or more word lines. A word line contains the following fields:

1. ID. The word index, an integer starting at 1. Multiword tokens may have a range index (e.g. 2-3), while empty nodes can have a decimal number bigger than 0[3].
2. FORM. The token form.
3. LEMMA. The lemma or stem of FORM.
4. UPOS. A universal part-of-speech tag.[4] Same set of PoS tags for every UD treebank.
5. XPOS. A language-specific part-of-speech tag.
6. FEATS. Universal or language-specific morphological features.[5]
7. HEAD. The index of the current word's head. If the head is the dummy root, it will be 0.
8. DEPREL. The dependency relation between the word and its head. It can be `root`, language-specific or universal.[6]

[3] This is out-of-the-scope of this work.
[4] https://universaldependencies.org/u/pos/index.html
[5] https://universaldependencies.org/ext-feat-index.html
[6] https://universaldependencies.org/u/dep/index.html

```
# sent_id = es-test-001-s295
# text = Nunca hemos tenido ningún problema.
1 Nunca nunca ADV _ Polarity=Neg 3 advmod _ _
2 hemos haber AUX _ Mood=Ind|Number=Plur|Person=1|Tense=Pres|VerbForm=Fin 3 aux _ _
3 tenido tener VERB _ Gender=Masc|Number=Sing|Tense=Past|VerbForm=Part 0 root _ _
4 ningún ninguno DET _ Gender=Masc|Number=Sing|PronType=Neg 5 det _ _
5 problema problema NOUN _ Gender=Masc|Number=Sing 3 obj _ SpaceAfter=No
6 . . PUNCT _ _ 3 punct _ _
```

Figure 2.3: A sentence extracted from the $\text{Spanish}_{\text{GSD}}$ corpus annotated in CoNLL-U format.

9. DEPS. The enhanced dependency graph consisting in a list of HEAD-DEPREL pairs.

10. MISC. Other annotations.

Only FORM, LEMMA and MISC can have space characters. When there is no information available, an underscore ('_') is used, as a synonym of the field being empty. UPOS, HEAD and DEPREL are only allowed to be empty in the case of multiword tokens and empty nodes. An example of an annotated sentence in CoNLL-U format is shown in Figure 2.3.

With respect to the evaluation, the parsing community typically uses two metrics to measure the performance of a parser on a given test set: Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) (Nivre and Fang, 2017). The first evaluates the parser's output considering the number of words that have been assigned a correct syntactic head, ignoring the dependency type; while the second one requires that both elements are correct.

2.3. Sequence labeling

Sequence labeling is a structured prediction problem consisting in assigning a single output label from a fixed set to every input token of a given sequence. This fast and simple arrangement is natural for many NLP tasks, such as PoS tagging (Ma and Hovy, 2016), text chunking (Liu et al., 2018a) or named-entity recognition (Lample et al., 2016).

Traditionally, sequence labeling used classical machine learning techniques like hidden Markov models and conditional random fields, but in recent years, deep learning models have improved the state-of-the-art performance

(Akbik, Blythe, and Vollgraf, 2018; Bohnet et al., 2018), thanks to their ability to learn complex features without the need of 'hand-crafting' them.

In the rest of this section, we explain how dependency parsing can be cast as a sequence labeling task in 2.3.1, and we present our sequence labeling framework in 2.3.2.

2.3.1. Dependency parsing as sequence labeling

To create a linearized tree for dependency parsing it is needed to assign a discrete label (x_i, l_i) to each token w_i, where x_i encodes a subset of the arcs of the tree related to that token and l_i is the dependency type. Two straightforward encodings are the absolute and relative positional encodings. The former encodes the arcs by selecting the token's head using its absolute index as x_i, while the latter encodes the arc through the relative distance between the token and its head as x_i. In this work, we will predict the component labels x_i and l_i semi-independently; this is explained in more detail in 2.3.2, together with the sequence labeling framework.

Also, we will be using five different encodings from three different families: a relative PoS tag index head-selection encoding, two bracketing-based encodings and two mappings from transition-based algorithms (arc-hybrid and Covington). These encodings are preferred over the naïve ones as they have shown to perform better (Strzyz, Vilares, and Gómez-Rodríguez, 2019). They are more thoroughly explained in 3.1.1.

As these labels are discrete elements, only labels seen in the training data can be predicted. Still, it has been previously shown that the coverage is almost complete and that this is not a problem to carry out dependency parsing effectively, at least for rich-resource languages (Strzyz, Vilares, and Gómez-Rodríguez, 2020).

2.3.2. Sequence labeling neural framework

Being w a sequence of words $w_1, w_2, ..., w_n$, $\vec{w}$ is a sequence of word embeddings that will be used as the input to the models. A word embedding is a vector representation of a word that is composed of real numbers. Words with similar meaning are supposed to be close in the vector space. There are two main types of word embeddings: (i) static word embeddings, created using algorithms such as word2vec (Church, 2017) or GloVe (Pennington, Socher, and Manning, 2014), or contextualized word embedding like those

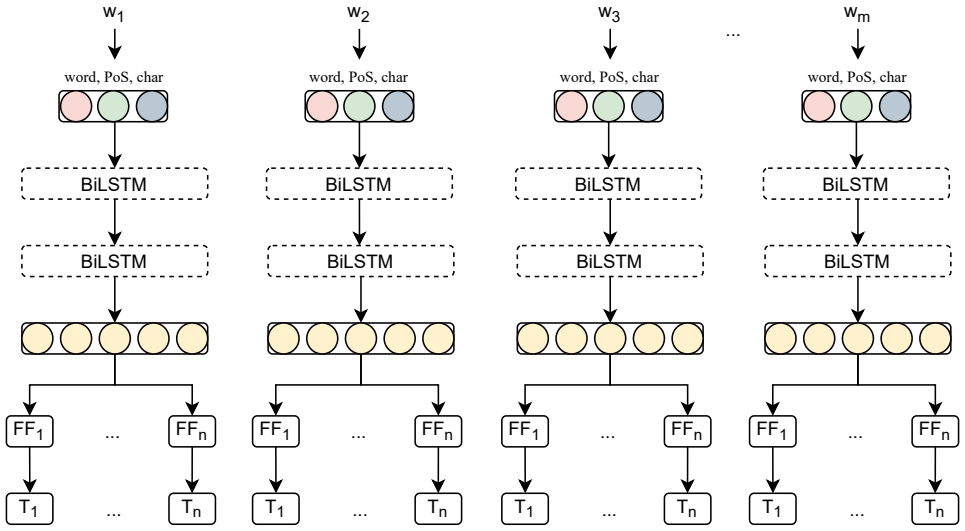

Figure 2.4: Framework used for MTL sequence labeling in this work.

created by ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019). In this work, we will use random initialized (static) word embeddings to represent the input words.

To train our sequence labeling parsers, in this work we will use bidirectional long short-term memory networks (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997, biLSTM). BiLSTMs have showed to be a strong baseline and have been used recently in many NLP tasks (Yang and Zhang, 2018; Reimers and Gurevych, 2017).

Concretely, we use two layers of biLSTMs, and each hidden vector $\vec{h_i}$ from the last biLSTM layer (associated to each input vector $\vec{w_i}$) is fed to separate feed-forward networks that predict the labels that compose the linearization (i.e. x_i and l_i), using softmaxes and a hard-sharing multi-task learning (Caruana, 1997; Ruder, 2017, MTL). Multi-task learning consists in sharing representations between related tasks.

We use a hard-sharing architecture (the hidden layers are shared between all tasks, but the outputs are independent for each of them) as it is a standard and easy form of MTL and it has been used in previous work related to parsing as sequence labeling (Strzyz, Vilares, and Gómez-Rodríguez, 2019; Strzyz, Vilares, and Gómez-Rodríguez, 2020). A diagram of the framework is shown in Figure 2.4.

2.4. Morphological inflection

Many languages express syntactic and semantic information through the variation of word forms using modifications such as adding prefixes or suffixes to a lemma (e.g. look → looking) or vowel mutation (e.g. sing → sang), among others. This word generation process is known as morphological inflection. Automatic morphological inflection has become popular recently, mainly due to the introduction of the SIGMORPHON shared tasks (Cotterell et al., 2016; Cotterell et al., 2017; Cotterell et al., 2018; McCarthy et al., 2019; Nicolai, Gorman, and Cotterell, 2020; Pimentel et al., 2021) and the development of the UniMorph[7] project. It consists in generating a word form from a given a lemma and a set of morphological features. This task is relevant to our work as inflected words carry morphosyntactic information that can help parsers to learn the dependency structure of a sentence.

Morphological inflection has been considered a nearly-solved task, since modern systems obtain accuracy scores higher than 90 %, even when only few annotated examples (in the range of two hundred) are available. However, Goldman, Guriel, and Tsarfaty (2021) recently showed that splitting the training and test data by lemmas instead of by forms (i.e. no lemma that appeared in the training set appears in the test one) drops the scores by an average of 40 points. This is specially relevant in our experiments as most lemmas are unknown to the model, so this could affect its performance as we will see more thoroughly in 4. This work was released in August 2021, contemporaneously to our results.

In the rest of this section, we introduce the UniMorph project in 2.4.1 and present the neural framework for morphological inflection used in this work.

2.4.1. UniMorph

The Universal Morphology project, better known as UniMorph (UM) is a collaborative project focused on handling morphology in world's languages. Its goal is to create a cross-lingual universal schema which allows to identify an inflected word by its lemma and a morphological feature vector, independently of the language.

The UM feature schema (Sylak-Glassman, 2016) is made of 23 dimen-

[7] https://unimorph.github.io/

Lemma	Features	Form
florecer	V;3;PL;SBJV;PST;IPFV	florecesen
	V;1;PL;COND	floreceriamos
	V;1;SG;IND;FUT	florecerei
	V;IND;PST;1;SG;IPFV	florecía
	V;3;SG;IMP;NEG	floreza

Table 2.2: Morphological inflection examples extracted from the Galician UM data.

sions of meaning and over 212 features. The dimensions of meaning cover different morphological categories such as case, tense, number, part-of-speech, gender or person. Every dimension can be represented by a number of values, named features, ranging form 2 for finiteness to 39 for case. Some examples are shown in Table 2.2. The features have a semantic meaning and are an indivisible unit. This schema was built in a top-down fashion, surveying the linguistic typology literature to find common features and identifying the most common morphological features assigned to each part-of-speech tag.

The data has been used in the previously mentioned SIGMORPHON shared tasks. It is obtained from the inflectional paradigms contained in Wiktionary[8], following a process of extraction and normalization (McCarthy et al., 2020). There is also additional data obtained from converting other morphological resources to the UM schema. The current data covers 118 languages from 16 major families and 2 isolated languages, Basque and Haida.

2.4.2. Morphological inflection neural framework

We used the model from Wu, Shapiro, and Cotterell (2018) that has been used as baseline in the SIGMORPHON 2020 shared task 0 (Nicolai, Gorman, and Cotterell, 2020). It is a sequence-to-sequence model that uses a hard monotonic attention mechanism to identify what parts of the input the model should focus on to generate the correct output string.

2.5. Existing approaches to low-resource parsing

Low-resource parsing has been explored from perspectives such as unsupervised parsing, data augmentation, cross-lingual learning, or data-efficiency of models.

[8] https://www.wiktionary.org/

On unsupervised parsing, some authors have worked on generative models to determine whether to continue or stop attaching dependents to a token (Klein and Manning, 2004; Spitkovsky, Alshawi, and Jurafsky, 2010), while others have studied how to use self-training for unsupervised parsing (Le and Zuidema, 2015; Mohananey, Kann, and Bowman, 2020).

On data augmentation, there is work on using self-training to annotate extra data (McClosky, Charniak, and Johnson, 2006) and linguistically motivated approaches to augment treebanks, such as using methods to replace subtrees within a given sentence (Vania et al., 2019; Dehouck and Gómez-Rodríguez, 2020).

On cross-lingual learning, some authors have trained delexicalized parsers in a source rich-resource treebank that are then used to parse a low-resource target language (Søgaard, 2011; McDonald, Petrov, and Hall, 2011). Other works compared lexicalized and delexicalized parsers on low-resource treebanks, depending on factors like the treebank size and the PoS tags performance (Falenska and Çetinoğlu, 2017). Wang and Eisner (2018) created synthetic treebanks that resemble the target language by permuting constituents of distant treebanks. Naseem, Barzilay, and Globerson (2012) and Täckström, McDonald, and Nivre (2013) approached this issue from the model side, training on rich-resource languages in a way that the model learns to detect the source language's aspects that are relevant for the target language. More recently, Mulcaire, Kasai, and Smith (2019) used a LSTM to build a polyglot language model that was used to train a parser on top of it that shows cross-lingual abilities in zero-shot setups.

On data-efficiency, there has been research about the impact of the use of different amounts of data to alleviate the lack of annotated data or the poor quality of it. For example, Lacroix et al. (2016) showed how a transition-based parser with a dynamic oracle can be used without any modifications to parse partially annotated data. This work found that this setup is useful to train low-resource parsers on sentence-aligned texts, from a rich-resource treebank to an automatically translated low-resource language, where only precisely aligned tokens are used for the projection in the target dataset. Lacroix, Wisniewski, and Yvon (2016) studied the effect that preprocessing and post-processing have in annotation projection, concluding that quality should prevail over quantity. Anderson and Gómez-Rodríguez (2020) showed that when distilling a graph-based parser for faster inference time, mo-

dels with smaller treebanks suffered less. Dehouck, Anderson, and Gómez-Rodríguez (2020) also distilled models for Enhanced Universal Dependencies (EUD) parsing with diverse amounts of data, observing that less training data usually means slightly lower performance while having better energy consumption. Garcia, Gómez-Rodríguez, and Alonso (2018) showed that, in Romance languages, picking samples from related languages and adapting them to the target language is useful to train a model that obtains similar performance as one trained on fully but limited manually annotated data. Regarding constituent parsing, Shi, Livescu, and Gimpel (2020) studied the role of the development data in unsupervised parsing. They mentioned that many unsupervised parsers use the score on the development set to update the hyper-parameters, and show that training a counterpart supervised model using a handful of samples from that development set can outperform the results of the unsupervised setup. Finally, there is work describing the impact of the size of the parsing training data on downstream tasks that use syntactic information as part of the input (Sagae et al., 2008; Gómez-Rodríguez, Alonso-Alonso, and Vilares, 2019).

Chapter 3

Data Efficiency of linearizations

Part of the work of this chapter is published in (Muñoz-Ortiz, Strzyz, and Vilares, 2021).

This chapter covers the first part of this work, studying how different dependency parsing linearizations behave when facing different quantities of data. The goal of this experiment is to get a better understanding about the role of encoding selection in dependency parsing as a sequence labeling task in low-resource setups.

First, we simulate data-restricted setups from rich-resource treebanks by dividing the treebanks in smaller sets and compare the obtained results depending on the number of sentences used to train the models. Then, the same encodings are tested on real low-resource treebanks to check if the previous results hold when tested in more realistic conditions. The parsers are tested in three different setups regarding the use of PoS tags as input: using gold PoS tags, using predicted PoS tags and not using PoS tags at all.

This chapter is divided as follows. In 3.1, we present the existing families of encodings to perform parsing as a sequence labeling task. Then, in 3.2, we present the methodology and the results of the two experiments. Finally, a conclusion can be found in 3.3.

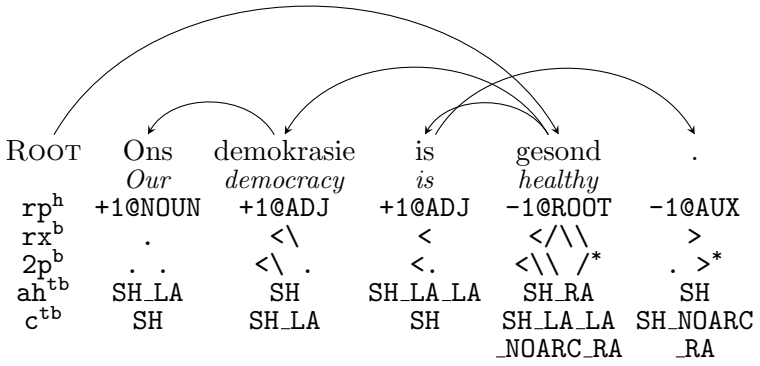

Figure 3.1: Example of the linearizations used in this work in a sentence from the Afrikaans$_{\text{AfriBooms}}$ treebank. Dependency types are omitted for simplicity.

3.1. Description

In this section, we first review the existing families of encodings for parsing as sequence labeling and introduce the five encodings selected for these experiments (3.1.1). Then, the sequence labeling framework used is presented (3.1.2).

3.1.1. Selecting encodings

As seen in 2.3.1, there are multiple ways to linearize dependency trees in the form of tuples (x_i, l_i), where x_i encodes a subset of the arcs related to that token and l_i encodes its dependency type. In this subsection we explain the encodings that are used in this work. We selected five representative linearizations from the three existing encoding families. These are compared in an example sentence in Figure 3.1. In what follows, the encoding families are described along with the chosen linearizations.

Head-selection encodings

Head-selection encodings assign to every word a label x_i that encodes its head. The encoding process can be done by labeling the target word using the absolute index of its head token or by using a relative index given by the distance between the dependent and its head. The former method is

done unequivocally, but for the latter we can use different techniques. For example, considering only words which have a certain tag, using PoS tags (Spoustová and Spousta, 2010; Strzyz, Vilares, and Gómez-Rodríguez, 2019) or head-based tags (Lacroix, 2019). This family of encodings (in particular the relative index distance one) has also been used for seq2seq models, although they did not succeed when it came to sequence labeling models (Li et al., 2018).

In this work, we chose the relative PoS-based encoding ($\texttt{rp}^{\texttt{h}}$), as it has been shown to outperform other linearizations of this family. In this encoding, x_i is portrayed as a tuple (p_i, o_i). When $o_i > 0$, the head of w_i is the o_ith word to the right with a PoS tag p_i. When $o_i < 0$, the head of w_i is the $|o_i|$th word to the left whose PoS tag is p_i.

In addition to its performance, its main advantages are the ability of encoding any non-projective tree and its simplicity in creating a direct connection between the dependent and the head. However, the reliance on external factors (in this case, PoS tags) force us to run an external tagger or having gold information in order to reconstruct the trees.

Bracketing-based encodings

This family of encodings linearizes dependency arcs using a sequence of bracket elements from a set $B = \{\texttt{<},\texttt{\textbackslash},\texttt{/},\texttt{>}\}$ as labels. A balanced pair of brackets `(<,\)` as the labels x_i and x_j means that there is an arc from w_j to w_{i-1}. A balanced pair of brackets `(/,>)` in x_i and x_j represents a right arc from w_{i-1} to w_j. A token can have many outgoing arcs but only one incoming arc, as a word can have many *dependents* but only one *head*. The labels are then composed of several brackets that follow the expression `(<)?((\)*|*(/)*)(>)?`.

This encoding is able to represent trees without the need of PoS tags or any external feature. However, it is incapable of managing crossing arcs in the same direction, though it can handle crossing arcs in opposite directions as left and right are balanced independently. To solve that, Strzyz, Vilares, and Gómez-Rodríguez (2020) propose adding a second independent plane in order to be able to encode any 2-planar graph.

More in detail, a dependency graph (V, E) is k-planar for $k \geq 1$ if its edges can be divided in $E_1, ..., E_k$ planes in a way that edges belonging to the same plane do not cross. For $k = 1$, the formed tree corresponds to

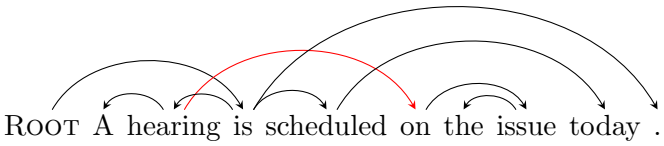

Figure 3.2: Example of a 2-planar sentence. Black and red arcs below to the first and second plane respectively.

a non-crossing dependency graph. For $k \geq 2$, a k-planar graph is a graph where if we assign arcs to two different planes, arcs in the same plane do not cross. Figure 3.2 shows an example of a 2-planar sentence. The vast majority of syntactic structures are 2-planar (Gómez-Rodríguez and Nivre, 2013; Gómez-Rodríguez, 2016), so introducing a second independent plane is an effective way of being able to encode almost any dependency graph. To do so, we need to introduce a different set of balanced brackets for each plane. The arcs belonging to the second plane are marked with * and belong to the set $B^* = \{\texttt{<*}, \texttt{\textbackslash*}, \texttt{/*}, \texttt{>*}\}$. A token w_i can have assigned elements from B and B^* at the same time. As both planes are balanced independently, they only match when they belong to the same plane.

In this work, we consider restricted non-projective ($\texttt{rx}_\texttt{b}$) and 2-planar-bracketing ($\texttt{2p}^\texttt{b}$) encodings. There are many strategies to assign arcs to the second plane (Strzyz, Vilares, and Gómez-Rodríguez, 2020). We chose a so-called second-plane-averse greedy plane assignment, that assigns the arc to the first plane when possible. If not possible, we assign the arc to the second plane, and no plane if both planes are unavailable. This simple but suboptimal assignment was chosen over the other proposed in the paper (second-plane-averse plane assignment based on restriction propagation on the crossing graphs) because, although theoretically it does not secure that all the 2-planar trees are encoded due to its suboptimal assignations, in practice these differences are negligible both in coverage and performance (see Tables 2 and 6 from Strzyz, Vilares, and Gómez-Rodríguez (2020)).

Transition-based encodings

Transition-based parsing algorithms define an abstract state machine where each configuration holds a structured representation together with auxiliary

data structures. The system moves between states through shift-reduce actions or transitions until it finds a full parse. An introduction on how these systems work has been explained in Section 2.2.1.

Gómez-Rodríguez, Strzyz, and Vilares (2020) presents a unified framework to cast transition-based algorithms as a sequence labeling task for any *left-to-right* transition system. These systems have a set of transitions that read words from the input in a strictly left-to-right order, needing n of these transitions to read a sentence of length n. This property can be used to divide the n transition sequences in n labels and assign one label to each word.

These encodings have been shown to perform worse than bracketing-based encodings. However, we include them for completeness and to study its convenience in low-resource setups. In this work, we consider mappings from the projective transition-based algorithm arc-hybrid ($\texttt{ah}^{\texttt{tb}}$)(Kuhlmann, Gómez-Rodríguez, and Satta, 2011) and for the non-projective Covington algorithm ($\texttt{c}^{\texttt{tb}}$) (Covington, 2001).

3.1.2. Framework details

Each input vector $\vec{w}_i$ is composed of a concatenation of a word embedding, a character-level word embedding computed through a char-LSTM, and an optional PoS tag embedding. We use a 2-task MTL setup for every encoding seen in 3.1.1, except for the 2-planar bracketing encoding. One task predicts x_i to each encoding specifics, and the other one predicts the dependency type l_i. For the 2-planar bracketing encoding we use a 3-task MTL setup as it needs to compute the arcs from a second plane; the prediction of x_i is divided in two tasks, one that predicts the arcs from the first plane and another one to predict those from the second one. We used a 3-task setup instead a 2-task one, as it follows previous work and it establishes a fairer comparison regarding label sparsity.

We decided not to use computationally expensive models like BERT (Devlin et al., 2019) for several reasons. First of all, the experiments of this work involve the training of 760 parsing models (see more details in 3.2), which makes the training on BERT or similar models unfeasible. Also, there are not specific-language or multilingual BERT models for all the languages used. This could add uncontrolled variables that may impact the results and the conclusions. In addition to this, there are controversies in literature

about what makes a language help other under a BERT-based framework. For example, Wu and Dredze (2019) concludes that sharing a lot of subword chunks is important, while others state otherwise (Pires, Schlinger, and Garrette, 2019; Artetxe, Ruder, and Yogatama, 2020). Lastly, even if there were language-specific BERT models for every language, the data used for their pretraining would add unwanted noise to our experiments.

3.2. Methodology and experiments

We made two experiments to test how the proposed encodings behave under low-resource setups:

- Experiment 1: Encodings' data-efficiency (3.2.1). We simulate some data-restricted setups to examine which encodings use data more efficiently. Our aim is to investigate if some of these linearizations are learnable with less data or could perform better under the assumption of larger data being available.
- Experiment 2: Encodings' performance on real LRLs data (3.2.2). We test the encodings on real low-resource setups. This experiment aims to check if the results of the previous experiment hold for authentic under-resourced setups and to confirm which sequence labeling encodings perform better under these conditions.

The languages used in the experiments come from a diverse sample of language families and linguistic genera, use different script systems and have different levels of non-projectivity. Table 3.1 shows information about the selected treebanks.

Experimental setups

For both experiments, three experimental setups are considered:

1. Gold PoS tags setup. In this setup, the models are trained and executed considering an ideal framework which uses gold PoS tags as part of the input. Encodings such as $\texttt{rp}^{\texttt{h}}$ use PoS tags to rebuild the linearized tree, so this scenario helps estimate the *optimal* data-efficiency and learnability of these encodings under optimal but unrealistic conditions.

TREEBANK	% NON-PROJECTIVE SENTENCES	FAMILY	SCRIPT
$\text{German}_{\text{HDT}}$	6.76	IE (Germanic)	Latin
$\text{Czech}_{\text{PDT}}$	11.49	IE (West Slavic)	Latin
$\text{Russian}_{\text{SynTagRus}}$	7.53	IE (East Slavic)	Cyrillic
$\text{Ancient Chinese}_{\text{Kyoto}}$	0.01	Sino-Tibetan	Sinographs
$\text{Persian}_{\text{PerDT}}$	14.22	IE (Iranian)	Persian
$\text{Estonian}_{\text{EDT}}$	3.22	Uralic	Latin
$\text{Romanian}_{\text{Nonstandard}}$	5.43	IE (Romance)	Latin
$\text{Korean}_{\text{Kaist}}$	21.70	Korean	Korean
$\text{Ancient Greek}_{\text{PROIEL}}$	37.52	IE (Greek)	Greek
$\text{Hindi}_{\text{HDTB}}$	13.60	IE (Indo-Aryan)	Devanagari
$\text{Latvian}_{\text{LVTB}}$	6.53	IE (Baltic)	Latin
$\text{Afrikaans}_{\text{AfriBooms}}$	22.23	IE (Germanic)	Latin
$\text{Coptic}_{\text{Scriptorium}}$	13.24	Afro-Asiatic	Coptic
$\text{Faroese}_{\text{FarPaHC}}$	0.19	IE (Germanic)	Latin
$\text{Hungarian}_{\text{Szeged}}$	27.10	Uralic	Latin
$\text{Lithuanian}_{\text{HSE}}$	14.07	IE (Baltic)	Latin
$\text{Maltese}_{\text{MUDT}}$	3.86	Semitic	Latin
$\text{Marathi}_{\text{UFAL}}$	6.01	IE (Indo-Aryan)	Devanagari
$\text{Tamil}_{\text{TTB}}$	1.67	Dravidian	Tamil
$\text{Telugu}_{\text{MTG}}$	0.15	Dravidian	Telugu
$\text{Wolof}_{\text{WTB}}$	2.99	Niger-Congo	Latin

Table 3.1: Percentage of non-projective sentences, linguistic family and script for the selected treebanks used in both experiments.

2. Predicted PoS tags setup. As the previous setup does not reflect realistic conditions, this setup executes the models using predicted PoS tags. These are less helpful as their performance lowers, mainly when there is little data available. The $\texttt{rp}^{\texttt{h}}$ encoding is expected to suffer more from this, as it needs the PoS tags to rebuild the tree from the labels, and errors could propagate during decoding. To obtain the predicted PoS tags, we trained taggers for each treebank using the same architecture and the same data used for the parsers.

3. No PoS tags setup. We train and execute the models without using PoS tags as part of the input. The situation is unnatural for the $\texttt{rp}^{\texttt{h}}$ encoding as we still had to run the taggers to get the PoS tags to rebuild the tree. Nevertheless, we include it for completeness and to understand better how different families of encodings suffer from the lack of PoS tags. This is an easy way to get simpler and faster models, as the taggers do not need to be executed. Also, the input vectors and

the models will be smaller, hastening the execution. This setup is most relevant for low-resource languages, as PoS tags might no be available or the taggers are not as accurate as needed in order to help deep learning models (Zhou et al., 2020; Anderson and Gómez-Rodríguez, 2021).

3.2.1. Experiment 1: Encodings' data-efficiency

Dependency parsing linearizations have been tested and compared in English and multilingual setups, but there is no work regarding how different amounts of data can affect the behavior of the encodings. In this experiment, we test the selected encodings in different sets with diverse quantities of data and compare their results.

Data

We chose 11 rich-resource treebanks from UD2.7 (Zeman, Nivre, and others, 2020) with more than 10 000 training sentences: $\text{German}_{\text{HDT}}$, $\text{Czech}_{\text{PDT}}$, $\text{Russian}_{\text{SynTagRus}}$, Classical $\text{Chinese}_{\text{Kyoto}}$, $\text{Persian}_{\text{PerDT}}$, $\text{Estonian}_{\text{EDT}}$, $\text{Romanian}_{\text{Nonstandard}}$, $\text{Korean}_{\text{Kaist}}$, Ancient $\text{Greek}_{\text{PROIEL}}$, $\text{Hindi}_{\text{HDTB}}$ and $\text{Latvian}_{\text{LVTB}}$. We considered training subsets of 100, 500, 1 000, 5 000 and 10 000 samples to simulate data-restricted setup, plus the total training set for comparison. The training sets were shuffled before the division.

Setup

To compare the data-efficiency of the selected encodings we considered the $\texttt{rp}^{\texttt{h}}$ encoding as the reference and as an *a priori* upper bound, since it has showed the strongest performance in previous work for multi-lingual setups (Strzyz, Vilares, and Gómez-Rodríguez, 2019; Gómez-Rodríguez, Strzyz, and Vilares, 2020). Then, we calculate the difference of the average UAS of the 11 treebanks between the $\texttt{rp}^{\texttt{h}}$ and the other encodings for every setup and training set. The goal is to know how the linearizations behave under limited data and how this trend evolves as more data becomes available. Finally, we show the UAS for the models trained on the whole treebanks. We also computed the statistically significant difference between the $\texttt{rp}^{\texttt{h}}$ and the rest of encodings using the p-value ($p < 0.05$) of a paired t-test on the scores distribution, following recommended practices for dependency

parsing (Dror et al., 2018). In this work, UAS is reported over LAS as the linearizations differ on how they encode the dependency arcs and not their types. Table 3.2 shows the PoS taggers average performance.

# Sentences	Average accuracy %
100	57.08
500	81.24
1 000	85.03
5 000	90.90
10 000	92.77
Low-resource	85.03

Table 3.2: Average accuracy of the taggers for the splits of the rich-resource treebanks and the full low-resource treebanks.

Results

Tables 3.3, 3.4 and 3.5 show the difference of the average UAS for each encoding with respect to the $\texttt{rp}^{\texttt{h}}$ for the gold PoS tags, predicted PoS tags and no PoS tags setups, respectively.

For the gold PoS tags setup, we can see how the $\texttt{rp}^{\texttt{h}}$ encoding outperforms both the bracketing ($\texttt{rx}^{\texttt{b}}$ and $\texttt{2p}^{\texttt{b}}$) and the transition-based ($\texttt{ah}^{\texttt{tb}}$ and $\texttt{c}^{\texttt{tb}}$) encodings, for all the training splits. However, this gap narrows as the number of training sentences increases. This suggests that, under an ideal, gold environment, the $\texttt{rp}^{\texttt{h}}$ encodings exploit limited data better than the bracketing and transition-based encodings.

# Sentences	$\texttt{rp}^{\texttt{h}}$	$\texttt{rx}^{\texttt{b}}$	$\texttt{2p}^{\texttt{b}}$	$\texttt{ah}^{\texttt{tb}}$	$\texttt{c}^{\texttt{tb}}$
100	68.34	-2.15	-2.42	-5.82	-9.96
500	76.94	-1.58	-1.5	-5.21	-9.35
1 000	80.29	-1.42	-1.43	-5.16	-8.9
5 000	86.54	-1.16	-1.26	-3.62	-7.04
10 000	88.26	-0.8	-0.72	-3.52	-5.67

Table 3.3: Average UAS difference for the subsets of the rich-resource treebanks under the gold PoS tags setup. Blue (see next tables too) and yellow cells show the UAS increase and decrease with respect to the $\texttt{rp}^{\texttt{h}}$ encoding, respectively.

For the predicted PoS tags setup, the differences narrow and the previous tendency starts to disappear. For the sets of 100, 5 000 and 10 000, the relati-

ve PoS-based encoding slightly outperforms the bracketing-based and clearly outperforms the transition-based encodings. However, both bracketing encodings obtain a higher average UAS than the $\texttt{rp}^{\texttt{h}}$ for the 1 000 sentences set and the $\texttt{2p}^{\texttt{b}}$ also performs better for the 500 sentences set. The $\texttt{rp}^{\texttt{h}}$ is the one encoding that suffers most the degrading quality of the PoS tags. This suggests that the quality of the PoS taggers used can influence significantly its performance. This may also indicate that the use of bracketing-based encodings is preferable in this range as they seem able to take advantage of the available information more than the $\texttt{rp}^{\texttt{h}}$, which suffers from inaccurate PoS tags. Finally, we can observe too that all the scores drop significantly with respect to the previous setup, mainly when little data is available.

# Sentences	$\texttt{rp}^{\texttt{h}}$	$\texttt{rx}^{\texttt{b}}$	$\texttt{2p}^{\texttt{b}}$	$\texttt{ah}^{\texttt{tb}}$	$\texttt{c}^{\texttt{tb}}$
100	41.87	-0.42	-0.19	-1.9	-3.59
500	63.45	-0.01	0.14	-1.96	-5.73
1 000	68.10	0.25	0.17	-2.44	-5.53
5 000	78.56	-0.62	-0.63	-2.53	-5.44
10 000	82.29	-0.37	-0.36	-2.49	-4.44

Table 3.4: Average UAS difference for the subsets of the rich-resource treebanks under the predicted PoS tags setup.

Lastly, the tendency reverses when no PoS tags are used. Bracketing-based encodings obtain better results than the relative PoS-based one for every amount of training samples. The gap narrows as the number of sentences increases. The transition-based encodings also clearly outperform the $\texttt{rp}^{\texttt{h}}$ for the 100 sentences setup, and the $\texttt{ah}^{\texttt{tb}}$ slightly surpasses the relative PoS-based encoding too for the 500 and 1 000 sentences scenario. It is also worth noting that the bracketing-based and the transition-based encodings obtain better UAS for 100 sentences than in the predicted PoS tags scenario (44.79 in and 41.80 in average for the bracketing-based and transition-based encodings respectively in the no PoS tags setup, versus 41.57 and 39.13 in the predicted PoS tags setup), probably due to the poor performance of the taggers with scarce data. The advantage in favour of bracketing linearizations for the predicted and no PoS tags setups for low-resource languages will be further discussed in 3.2.2.

Tables 3.6, 3.7 and 3.8 show the UAS values obtained when training on the full training sets of the rich-resource treebanks for the gold PoS tags, predicted PoS tags, and no PoS tags setups, respectively. The objective is

# Sentences	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
100	35.60	9.06	9.31	7.57	4.83
500	58.63	3.04	2.45	0.99	-2.26
1 000	63.99	3.59	3.42	0.83	-2.24
5 000	75.57	1.47	1.55	-0.19	-3.07
10 000	79.90	1.22	1.54	-0.87	-2.93

Table 3.5: Average UAS difference for the subsets of the rich-resource treebanks under the no PoS tags setup.

to see if some of the encodings are able to perform on par with the relative PoS-based encodings under large quantities of data, since the previous results show that the UAS difference diminishes when the number of training sentences increases. Although the gap between encodings shrinks, rp^{h} still outperforms the rest of the encodings when PoS tags are used in almost every treebank, being the difference statistically significant in most of them when using gold tags. Also, as expected, bracketing-based encodings and even the ah^{tb} obtain higher UAS values than the rp^{h} when no PoS tags are used as part of the input.

	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
grc	83.09	79.89^{--}	81.7^{-}	78.57^{--}	79.86^{--}
lzh	90.21	89.56^{-}	89.24^{-}	89.04^{--}	89.18^{-}
cs	91.61	90.49^{--}	90.91^{--}	88.18^{--}	85.64^{--}
et	85.62	84.79^{-}	84.91^{-}	81.86^{--}	81.11^{--}
de	96.69	95.95^{--}	96.38^{--}	95.15^{--}	86.51^{--}
hi	94.69	94.09^{--}	94.43^{-}	93.05^{--}	85.02^{--}
ko	87.26	86.24^{--}	86.52^{-}	85.68^{--}	84.06^{--}
lv	85.3	83.88^{-}	84.01^{-}	80.88^{--}	81.38^{--}
fa	92.61	92.07^{-}	92.44^{-}	90.45^{--}	87.09^{--}
ro	90.49	89.68^{-}	89.63^{--}	87.39^{--}	86.38^{--}
ru	91.23	90.1^{--}	90.1^{--}	88.19^{--}	84.96^{--}
Avg	**89.89**	88.79	89.12	87.13	84.65

Table 3.6: UAS for the rich-resource treebanks, using the whole training set and the gold PoS tags setup. The red (--) and green cells (++) show that a given encoding performed worse or better than the rp^{h} model, and that the difference is statistically significant. Lime and yellow cells mean that there is no a significant difference between a given encoding and the rp^{h}, appending a $^{+}$ or a $^{-}$ when they performed better or worse than the rp^{h}.

	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
grc	80.2	77.61^{--}	79.21^{-}	76.49^{--}	77.71^{-}
lzh	79.93	79.8^{-}	79.42^{-}	79.41^{-}	79.54^{-}
cs	90.04	88.93^{--}	89.34^{--}	86.67^{--}	84.25^{--}
et	81.07	80.36^{-}	80.34^{-}	77.71^{--}	76.95^{--}
de	95.85	95.14^{--}	95.54^{--}	94.34^{--}	85.79^{--}
hi	92.22	91.76^{-}	92.21^{-}	90.72^{--}	83.24^{--}
ko	84.25	83.44^{-}	83.42^{-}	82.98^{--}	81.25^{--}
lv	70.65	71.98^{++}	71.08^{+}	68.9^{-}	68.97^{-}
fa	90.39	89.8^{-}	90.32^{-}	88.27^{--}	85.28^{--}
ro	87.32	86.64^{-}	86.49^{-}	84.44^{--}	83.5^{--}
ru	88.71	88.13^{-}	88.24^{-}	85.93^{--}	82.96^{--}
Avg	**85.51**	84.87	85.06	83.26	80.86

Table 3.7: UAS for the rich-resource treebanks, using the whole training set and the predicted PoS tags setup.

3.2.2. Experiment 2: Encodings' performance on real LRLs

This experiment aims to check if the results obtained in 3.2.1 for the artificial low-resource datasets that we created hold when tested in real low-resource treebanks. Low-resource treebanks may present other problems in addition to the lack of data, such as the lack of sentence diversity or erroneous annotation, so different encodings might behave differently in these situations.

Data

We used 10 of the smallest treebanks in terms of training sentences (discarding code switching treebanks or small treebanks of rich-resource languages) that had a development set: Lithuanian$_{\text{HSE}}$, Marathi$_{\text{UFAL}}$, Hungarian$_{\text{Szeged}}$, Telugu$_{\text{MTG}}$, Tamil$_{\text{TTB}}$, Faroese$_{\text{FarPaHC}}$, Coptic$_{\text{Scriptorium}}$, Maltese$_{\text{MUDT}}$, Wolof$_{\text{WTB}}$ and Afrikaans$_{\text{AfriBooms}}$. Table 3.9 shows the sizes of the treebanks used, that range between 153 and 1350 training sentences, most being around or between 500 and 1 000.

Setup

We run the second part of the experiment described in 3.2.1 for the low-resource treebanks to check if the trends observed in the previous experiments hold when facing a real low-resource setup, and therefore if the

	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
grc	77.84	77.41^{-}	79.16^{+}	75.64^{-}	76.99^{-}
lzh	79.99	81.02^{+}	80.75^{+}	81.11^{++}	81.42^{++}
cs	88.67	88.2^{-}	88.64^{-}	85.8^{--}	84.23^{--}
et	77.85	79.69^{++}	79.99^{++}	77.15^{-}	76.27^{-}
de	94.51	95.09^{++}	95.41^{++}	94.18^{-}	83.54^{--}
hi	89.43	91.7^{++}	91.98^{++}	90.72^{++}	82.86^{--}
ko	79.39	82.18^{++}	82.15^{++}	81.88^{++}	80.3^{++}
lv	62.56	71.17^{++}	72.38^{++}	66.78^{++}	69.38^{++}
fa	89.14	90.39^{++}	90.48^{++}	88.49^{-}	84.54^{--}
ro	85.28	86.41^{+}	86.94^{++}	84.25^{-}	83.04^{--}
ru	83.35	83.98^{++}	84.5^{++}	83.42^{++}	80.26^{--}
Avg	82.55	84.29	**84.76**	82.67	80.26

Table 3.8: UAS for the rich-resource treebanks, using the whole training set and the no PoS tags setup.

Treebank	# Sentences
$\text{Afrikaans}_{\text{AfriBooms}}$	1 315
$\text{Coptic}_{\text{Scriptorium}}$	1 089
$\text{Faroese}_{\text{FarPaHC}}$	1 020
$\text{Hungarian}_{\text{Szeged}}$	910
$\text{Lithuanian}_{\text{HSE}}$	153
$\text{Maltese}_{\text{MUDT}}$	1 123
$\text{Marathi}_{\text{UFAL}}$	373
$\text{Tamil}_{\text{TTB}}$	400
$\text{Telugu}_{\text{MTG}}$	1 051
$\text{Wolof}_{\text{WTB}}$	1 188

Table 3.9: Number of training sentences for the low-resource treebanks.

conclusions are similar.

Results

Tables 3.10, 3.11, 3.12 show the UAS for each encoding and treebank for the gold PoS tags, the predicted PoS tags and the no PoS tags setups, respectively. These results help to elaborate on the conclusions obtained in the previous experiment.

As in 3.2.1, when gold PoS tags are available, the rp^{h} performs better than the other representations except for Telugu, which appears to be an

	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
af	88.02	85.7^{--}	85.48^{--}	81.84^{--}	78.6^{--}
cop	88.73	88.43^{-}	88.72^{-}	85.5^{--}	84.35^{--}
fo	84.04	83.76^{-}	84.09^{+}	81.78^{--}	79.53^{--}
hu	79.75	76.14^{--}	76.13^{--}	71.66^{--}	64.27^{--}
lt	51.98	50.28^{-}	50.19^{-}	45.0^{-}	46.6^{-}
mt	81.81	81.05^{-}	80.82^{-}	76.78^{--}	74.98^{--}
mr	77.43	76.46^{-}	75.97^{-}	76.94^{-}	73.54^{-}
ta	74.96	73.1^{-}	71.9^{-}	71.74^{-}	66.01^{--}
te	90.01	91.26^{+}	90.43^{+}	90.01^{+}	89.46^{-}
wo	86.19	84.64^{--}	84.51^{--}	80.65^{--}	77.43^{--}
Avg	**80.29**	79.08	78.82	76.19	73.48

Table 3.10: UAS for the low-resource treebanks for the gold PoS tags setup.

outlier.

When using predicted PoS tags, in this case the bracketing-based encodings perform consistently better for most of the treebanks, although the difference is not statistically significant. This reinforces the results from Table 3.4, as most of them are in or close to the range (between 500 and 1 000 training sentences) where the bracketing encodings outperformed the rp^{h} in 3.2.1. Still, the advantage of bracketing encodings is more evident in real low-resource setups than in simulated environments.

Finally, for the no PoS tags setup, the bracketing-based encodings averaged more than 3 points than the relative PoS-head selection encoding, which performance is surpassed even by the encodings of the transition-based family on many of the treebanks.

These results suggest that the bracketing-based encodings are the most suitable linearizations for real low-resource sequence labeling parsing. Furthermore, this might be an indication that we have to be extremely careful when running experiments on artificial low-resource setups, because as observed in this work, this may lead to incomplete or even misleading conclusions. Also, it is worth mentioning that the results obtained in the no PoS tags setup are only slightly worse (less than 1 UAS point in average) than those in the predicted PoS tags setup when using bracketing and transition-based encodings, while being simpler and faster to train and execute.

	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
af	81.84	80.29^{-}	79.9^{-}	77.3^{--}	73.61^{--}
cop	85.77	86.25^{+}	85.92^{+}	83.14^{--}	81.84^{--}
fo	77.04	76.97^{-}	77.52^{+}	75.23^{-}	74.24^{-}
hu	70.52	68.51^{-}	68.77^{-}	64.98^{--}	58.37^{--}
lt	30.28	34.53^{+}	33.11^{+}	31.23^{+}	29.91^{-}
mt	74.6	75.64^{+}	75.07^{+}	71.17^{--}	70.35^{--}
mr	66.99	67.96^{+}	67.23^{+}	68.93^{+}	67.23^{+}
ta	57.11	60.73^{+}	57.57^{+}	58.77^{+}	55.51^{-}
te	86.41	87.93^{+}	87.93^{+}	86.96^{+}	86.69^{+}
wo	76.88	76.4^{-}	76.3^{-}	73.24^{--}	70.84^{--}
Avg	70.74	**71.52**	70.93	69.10	66.86

Table 3.11: UAS for the low-resource treebanks for the predicted PoS tags setup.

	rp^{h}	rx^{b}	2p^{b}	ah^{tb}	c^{tb}
af	79.86	80.78^{+}	80.07^{+}	75.47^{--}	73.76^{--}
cop	84.36	85.76^{+}	85.13^{+}	83.07^{-}	81.28^{--}
fo	73.98	77.08^{++}	77.04^{++}	75.07^{+}	73.67^{-}
hu	63.63	65.21^{+}	64.8^{+}	62.04^{-}	56.17^{--}
lt	26.89	34.62^{++}	35.38^{++}	34.06^{++}	32.92^{+}
mt	70.95	75.5^{++}	75.3^{++}	71.69^{+}	70.32^{-}
mr	64.08	66.75^{+}	67.96^{+}	69.66^{+}	64.56^{+}
ta	52.79	60.03^{++}	56.61^{+}	59.58^{++}	54.95^{+}
te	85.44	88.49^{+}	88.63^{+}	87.1^{+}	86.82^{+}
wo	73.11	77.17^{++}	76.95^{++}	74.01^{+}	70.86^{-}
Avg	67.51	**71.14**	70.79	69.18	66.53

Table 3.12: UAS for the low-resource treebanks for the no PoS tags setup.

3.3. Conclusion

In this chapter, we have studied sequence labeling encodings for dependency parsing in low-resource setups. First, we explore which encodings are more data-efficient under different scenarios that include the use of gold PoS tags, predicted PoS tags and no PoS tags as part of the input. By restricting training data for rich-resource treebanks, we observe that although bracketing encodings are less data-efficient than head-selection ones under ideal conditions, this disadvantage can vanish when the input conditions are not gold and data is limited. Second, we studied their performance under the same

conditions, but on truly low-resource languages. These results show more clearly the advantage of bracketing encodings over the rest of the encoding families when training data is limited and the quality of external factors (such as PoS tags) is poor (as a consequence of the low-resource nature of the problem).

Chapter 4

Cross-lingual morphological inflection

In this chapter, we present the second part of the work. We propose a data-augmentation technique for low-resource parsing, that in this work we will apply only to sequence labeling parsing, building on top of the work of Chapter 3, although it should be applicable to any parsing paradigm. More particularly, the method consists in taking a pair of related languages, a source rich-resource language and a target low-resource language, and training a morphological inflection system or **inflectioner** in the target language to transform a treebank in the (related) source language into (something that should resemble) the target language.

The system inflects the lemmas of the rich-resource treebank as they would be inflected in the low-resource language, using its morphological features. This creates a new hybrid treebank between the two languages that is then used to train a dependency parser for the low-resource language. As both languages are related, the treebanks are expected to share a certain amount of lemmas and morphological features that will ease the transformation process. We are going to refer to these treebanks as cross-lingual inflected treebanks or **x-inflected treebanks**. This process can be seen as a kind of word-level translation using the lemmas and morphological information contained in the treebanks, that allow for automatic cross-lingual dependency annotation.

The main idea behind this technique is that inflected words carry syntactic information, thus training dependency parsers using the x-inflected

treebanks can add some information from the target language into the source treebank. In this chapter, we want to check whether this technique can improve the performance on target low-resource languages. To evaluate the method, we propose two experiments for each target treebank:

1. We test this method on a *zero-shot* setup, i.e. the model is tested on a language without seeing any sentence in that language before. We evaluate two models against the target treebank: one trained with the source treebank and one trained with the x-inflected treebank, and check whether our method helps improving the results.

2. We test this method on a *few-shot* setup, i.e. the model has only seen a few sentences in the language it is being tested on. We evaluate three models against the target treebank: one only trained in the low-resource target treebank, one trained in the target treebank together with some related rich-resource treebanks, and one trained in the target treebank together with some related x-inflected rich-resource treebanks. We compare the results and see if and when our methods can outperform a model trained only in the original low-resource treebank.

This chapter is organized as follows. In 4.1, we describe the process followed to create the x-inflected treebanks. In 4.2, we describe the zero-shot and the few-shot experiments in 4.2.1 and 4.2.2, respectively.

4.1. Description

In this section we describe the process to apply our method and create the x-inflected treebanks. The proposed method requires four steps:

1. Select the treebanks.

2. Train a morphological inflectioner in the target language using UM data.

3. Transform UD morphological features into UM features.

4. X-inflect the source language treebank using the trained system.

Figure 4.1 shows a diagram resuming all the steps required to create the x-inflected treebanks.

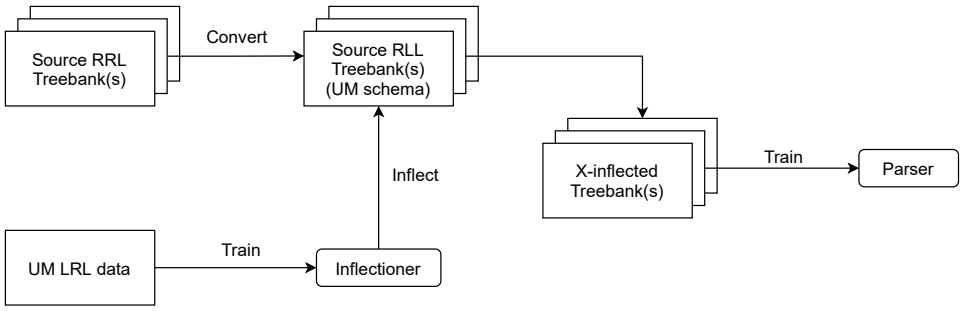

Figure 4.1: High-level architecture to create the x-inflected treebanks.

4.1.1. Selecting the treebanks

Although this technique can be applied to any pair of languages, in practice, the availability of resources in UD and UM to implement it limits the number of options. In what follows, we are going to see the three main aspects that we must take into account when selecting the treebanks:

1. Availability of related rich-resource language. There has to be a rich-resource UD treebank from a language that is similar enough to our target language. To find suitable auxiliary languages, we chose close languages from a phylogenetic point of view, being preferable those languages that share the smallest common phylogenetic group with the target treebank. To better understand this, we show the phylogenetic tree for Germanic languages in Figure 4.2. Faroese shares the West Scandinavian genus with Icelandic and Norwegian, so these two languages would be the first options as source languages. After that, the following languages to be taken into account would be Danish and Swedish, as they and Faroese share the North Germanic genus. After those, any other Germanic language such as English or German would be the next option, and so on. We selected the languages through a manual search of the available resources in both UD and UM together with an exploration of their phylogenetic trees.

2. Availability of data annotated with morphology. We have to take into account some aspects regarding morphology information, both for the target and the source language. On the one hand, for the source language the UD treebank must be annotated with morphological

features (the FEATS field from 2.2.3) so the UD information can be transformed into UM format and their lemmas and feats can be used as a proxy for inflection. On the other hand, for the target language, there must be enough data available in UM in order to train a morphological inflection model with enough accuracy. This narrows the number of languages we can use, as they must have a reasonable number of resources on UM while at the same time being low-resource on UD (for the purpose of our work).

3. Use of morphological inflection. The chosen languages must use morphological inflection at least to a certain degree so their lemmas can be x-inflected into new forms. Intuitively, this method should be more useful for more heavily inflected languages (e.g. Hungarian or Latin), as they encode more syntactic information through inflection, that can be transmitted by our method.

Due to the mentioned constraints, we are mostly restricted to Indo-European and Uralic languages, as most other families do not have feasible combinations of languages that match our requirements. Some others, as Semitic languages, while making an extensive use of morphology inflection to form new words and being present on both UD and UM, they use several scripts that usually do not mark vowels, making transliteration hard and unreliable. Therefore, we leave them out of the scope of this work, but we believe they would be worth studying in future work.

4.1.2. Training a morphological inflection system

To train the systems, we first downloaded the available UM data for the target language, shuffled it and split it 80-10-10 into a training set, a development set and a test set, respectively. In the case of languages containing different files for different dialects, we concatenated all the forms into one file prior splitting. The neural framework used is the one described in 2.4.2. Table 4.1 shows information about the UM data used to train the morphological inflection systems.

4.1.3. Converting UD features to the UM schema

Once we have trained a inflectioner in the target language using the UM data, we need to convert the features of the source treebank from the UD

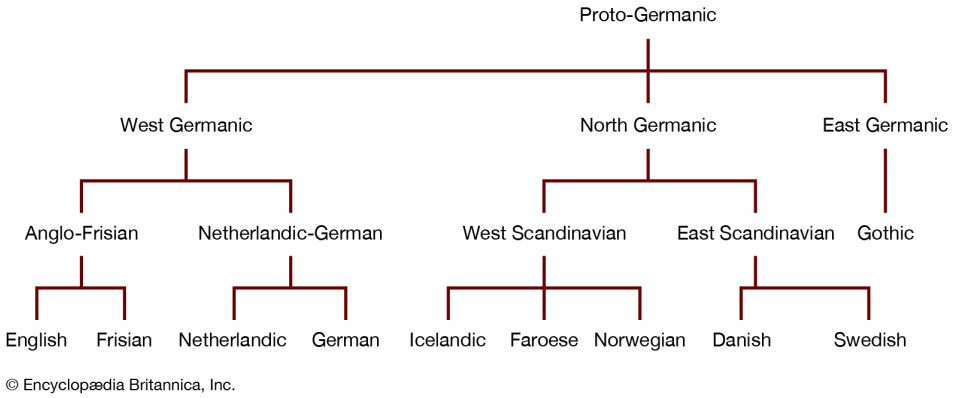

Figure 4.2: Phylogenetic tree for Germanic languages (Encyclopædia Britannica, 1998).

.

schema to the UM schema. Although both UD and UM schemata are universal and cross-lingual, and therefore both could be used for this task, there are several reasons why we have chosen to use UM over UD. First, combining two different resources can help compensating the lack of resources in UD; some resources have few annotated sentences in UD, but they have more annotated data in UM (e.g. Hungarian). Second, UM morphological data is more abundant and complete for most languages than the data of the UD treebanks, containing more inflected forms per lemma. Third, we ran preliminary experiments for our task to select which features to use, and parsing results were considerably worse when using UD features that those obtained using UM, for most of treebanks, even considering that the conversion of the features between schemata is not perfect and might cause some information loss.

To convert the features from the UD feature schema to the UM feature schema, we use the converter described in (McCarthy et al., 2018). In a ideal situation this conversion should be an easy task. The approach used by the converter starts by creating a language-independent mapping between both schemata. Still, annotation errors, disagreements and missing values in both schemata complicate this process, so a post-editing of the output is included to improve the results. The post-editing is an iterative process consisting in analyzing those forms and lemmas that are present in UD and UM, comparing their annotations, and creating rules to refine the mappings

Language	Parts-of-speech	Lemmas	Forms
Galician	Verbs	486	36 801
Livvi	Nouns, verbs, adjectives	23 920	1 003 197
Faroese	Nouns, verbs, adjectives	3 077	45 474
Latin	Nouns, verbs, adjectives	17 214	509 182
Hungarian	Nouns, verbs, adjectives	13 989	490 394
Czech	Nouns, verbs, adjectives	5 125	134 527
Lithuanian	Nouns, verbs, adjectives	1 458	136 998
Slovenian	Nouns, verbs, adjectives	2 535	60 110
Welsh	Verbs	183	10 641
North Sami	Nouns, verbs, adjectives	2 103	62 677

Table 4.1: Information about the UM data for the low-resource languages used to train the inflectioner. Livvi data is composed of several dialects; we mixed all of them in one file.

Form	Lemma	UD Features	UM Features
Omorîră	omorî	Mood=Ind|Number=Plur|Person=3|Tense=Past|VerbForm=Fin	PST;3;IND;PFV;V;PL
fiiul	fiu	Case=Acc,Nom|Definite=Def|Gender=Masc|Number=Sing	NOM/ACC;DEF;SG;N
stăpînului	stăpân	Case=Dat,Gen|Definite=Def|Gender=Masc|Number=Sing	DAT/GEN;DEF;SG;N
de	de	AdpType=Prep|Case=Acc	ADP;ACC
lucrători	lucrătoe	Case=Acc,Nom|Definite=Ind|Gender=Masc|Number=Plur	INDF;NOM/ACC;PL;N

Table 4.2: Morphological features expressed in the UD schema and the UM schema (after conversion) for a sentence extracted from the Romanian$_{\text{Nonstandard}}$ UD treebank.

between schemata. This process changes from language to language, and it is not available for all languages.

The converter translates all the morphological features contained in the UD treebank from the UD schema to the UM schema. An example of conversion for a sentence of the Romanian$_{\text{Nonstandard}}$ UD treebank is shown in Table 4.2.

4.1.4. Transforming the treebank

Once the features of the source treebank are expressed in the UM schema, and the morphological inflection system is trained, it is time to inflect its lemmas using the inflectioner. For this process we only considered those lemmas whose part-of-speech is contained in the UM data of our target language, as indicated in Table 4.1.

Lemma	UM Features	Output form	Original form
omorî	PST;3;IND;PFV;V;PL	Omoraērunt	Omorîră
fiu	NOM/ACC;DEF;SG;N	fius	fiiul
stăpînului	DAT/GEN;DEF;SG;N	stipin	stăpân
de	ADP;ACC	de	de
lucrătoe	INDF;NOM/ACC;PL;N	lucrōtōribus	lucrători

Table 4.3: Cross-lingual inflection of a sentence of the Romanian$_{\text{Nonstandard}}$ treebank using an inflectioner trained in Latin UM data.

We use the morphological features (expressed in the UM schema) and the lemma from the source treebank as input, and the inflectioner generates new inflected forms as an output string that should resemble the inflected term in the target language. These forms replace the original forms in the source UD treebank. Table 4.3 shows an example of an x-inflected sentence.

4.2. Metholodogy and experiments

To test the proposed method, we performed two experiments:

- Experiment 1. We compare the results obtained by a parser trained using the original source treebank and the x-inflected source treebank to see if our approach could improve the results in a zero-shot setup, for a number of target low-resource treebanks.

- Experiment 2. We created groups of languages that include a target low-resource language and a few rich-resource languages from the same phylogenetic group. Then, we train our sequence labeling parsers on three different types of data: (i) the target low-resource treebank, (ii) the target low-resource treebank merged with the related source languages treebanks without x-inflecting them, and (iii) the target low-resource treebank merged with the x-inflected source treebanks.

Regarding the use of PoS tags, we used the predicted PoS tag setup from the previous chapter, described in section 3.2, and the encoding used is the 2-planar bracketing encoding, described in 3.1.1. The sequence labeling framework used to train the parsers is the one used in Chapter 3 and described in 2.3.2.

4.2.1. Experiment 1: Zero-shot setup

We evaluate our method in a zero-shot setup, testing two different models trained on related languages on the same target low-resource treebank: one trained on the original source treebank, and another one trained on the x-inflected treebank. The goal of this experiment is to check whether this method can improve the performance of parsers when there is no available training data, but there is a related language with a more complete UD treebank and enough UM data to train a morphological inflection system in the target language and use it to transform the related rich-resource treebank.

Data

We chose 10 low-resource treebanks: $\text{Galician}_{\text{TreeGal}}$, $\text{Livvi}_{\text{KKPP}}$, $\text{Faroese}_{\text{FarPaHC}}$, $\text{Latin}_{\text{Perseus}}$, $\text{Hungarian}_{\text{Szeged}}$, $\text{Czech}_{\text{CLTT}}$, $\text{Lithuanian}_{\text{HSE}}$, $\text{Slovenian}_{\text{SST}}$ and $\text{Welsh}_{\text{CCG}}$ and North $\text{Sami}_{\text{Giella}}$, and 21 rich-resource treebanks: $\text{Estonian}_{\text{EDT}}$, $\text{Finnish}_{\text{TDT}}$, $\text{Norwegian}_{\text{Bokmaal}}$, $\text{Norwegian}_{\text{Nynorsk}}$, $\text{Icelandic}_{\text{IcePaHC}}$, $\text{Swedish}_{\text{LinES}}$, $\text{Danish}_{\text{DDT}}$, $\text{Polish}_{\text{LFG}}$, $\text{Slovak}_{\text{SNK}}$, $\text{French}_{\text{GSD}}$, $\text{Italian}_{\text{ISDT}}$, $\text{Romanian}_{\text{Nonstandard}}$, $\text{Catalan}_{\text{AnCora}}$, $\text{Spanish}_{\text{AnCora}}$, $\text{Portuguese}_{\text{Bosque}}$, $\text{Latvian}_{\text{LVTB}}$, $\text{Bulgarian}_{\text{BTB}}$, $\text{Croatian}_{\text{SET}}$, $\text{Serbian}_{\text{SET}}$, $\text{Irish}_{\text{IDT}}$ and Scottish $\text{Gaelic}_{\text{ARCOSG}}$. Tables 4.4 and 4.5 show information about the low-resource and rich-resource treebanks, respectively. Target treebanks can be matched with several source treebanks and vice versa, depending on availability of resources. Although Latin and Czech cannot be considered low-resource languages, as they have more rich-resource treebanks in UD, we decided to include these treebanks as a zero-shot setup to provide a sample as diverse as possible taking into account the strict conditions the treebanks must follow. As explained before, the requirements of the resources needed, limits us to the Indo-European and Uralic families, but that still offers diverse linguistic genera and diverse degrees of morphological inflection.

Results

The results for these zero-shot experimental setup are shown in Table 4.6, and show different trends: some languages present only improvements over the corresponding model trained on the original source treebank, some only

Treebank	# Training sent.	# Test sent.
$\text{Galician}_{\text{TreeGal}}$	600	400
$\text{Livvi}_{\text{KKPP}}$	19	106
$\text{Faroese}_{\text{FarPaHC}}$	1 020	301
$\text{Latin}_{\text{Perseus}}$	1 334	939
$\text{Hungarian}_{\text{Szeged}}$	910	449
$\text{Czech}_{\text{CLTT}}$	860	136
$\text{Lithuanian}_{\text{HSE}}$	153	55
$\text{Slovenian}_{\text{SST}}$	2 078	1110
$\text{Welsh}_{\text{CCG}}$	704	953
$\text{North Sami}_{\text{Giella}}$	2 257	865

Table 4.4: Low-resource treebanks used in both experimental setups.

decreases, and some others offer a mix depending on the source treebank. Overall, in the case of UAS we see that the method improves the zero-shot performance for 14 pairs of languages and causes losses for 13; while for LAS the improvements are a bit higher, improving the results for 16 language pairs against 11 pairs that obtained a worse performance.

On the one hand, for two languages, Faroese and Lithuanian, the model trained on every x-inflected treebank obtains a better result than using the original one. We could only pair Lithuanian with one language, but Faroese is paired with different languages, obtaining improvements for all of them. On the other hand, the use of our method decreases the score for all language pairs for Galician and Slovenian. In the case of Galician, the score differences with the model trained on the original source treebanks are small. This might be because the UM Galician data only includes information about verbs, and hence the number of x-inflected forms is smaller than for other languages. In the case of Slovenian, however, these differences are larger, indicating a poor transformation process. The cause of these results is not clear to us, as there are many factors that come into play. Yet, we have a few hypotheses about the poor behaviour in this case, such as: (i) badly annotated UM data or UD features, (ii) an incorrect translation between schemata or (iii) a bad selection of languages. Another possible reason is that morphological inflection systems generalize poorly when facing unseen lemmas, as pointed by Goldman, Guriel, and Tsarfaty (2021). This work is contemporary to ours so we did not take that into account when doing the experiments, but we believe this is worth exploring in future work.

For the rest of the languages, there are pairs that have positive results

Treebank	# Training sentences	Post-editing
$\text{Finnish}_{\text{TDT}}$	12 217	Yes
$\text{Estonian}_{\text{EDT}}$	24 633	No
$\text{Norwegian}_{\text{Bokmaal}}$	15 696	Yes
$\text{Norwegian}_{\text{Nynorsk}}$	14 174	Yes
$\text{Icelandic}_{\text{IcePaHC}}$	34 007	No
$\text{Danish}_{\text{DDT}}$	4 383	Yes
$\text{Swedish}_{\text{LinES}}$	3 176	Yes
$\text{Polish}_{\text{LFG}}$	13 884	Yes
$\text{Slovak}_{\text{SNK}}$	8 483	Yes
$\text{French}_{\text{GSD}}$	14 449	Yes
$\text{Italian}_{\text{ISDT}}$	13 121	Yes
$\text{Romanian}_{\text{Nonstandard}}$	24 121	Yes
$\text{Catalan}_{\text{AnCora}}$	13 304	Yes
$\text{Spanish}_{\text{AnCora}}$	14 305	Yes
$\text{Portuguese}_{\text{Bosque}}$	8 328	Yes
$\text{Latvian}_{\text{LVTB}}$	10 156	Yes
$\text{Bulgarian}_{\text{BTB}}$	8 907	Yes
$\text{Croatian}_{\text{SET}}$	6 914	No
$\text{Serbian}_{\text{SET}}$	3 328	No
$\text{Irish}_{\text{IDT}}$	4 005	Yes
$\text{Scottish Gaelic}_{\text{ARCOSG}}$	1 990	No

Table 4.5: Rich-resource treebanks used in both experimental setups. The availability of a custom post-editing for a language does not assure a good conversion of features from UD to UM, it is just an improvement over the automatic one.

and pairs that have negative results. The case of Hungarian is worth remarking: while the Finnish x-inflected treebank scores 30 points more than the original one, the Estonian transformed treebank performs slightly worse than its original source counterpart. Hungarian is just slightly related to Finnish and Estonian, while Finnish and Estonian are closely related between them. The three languages make a strong use of suffixation to encode morphosyntactic features, which may explain the great boost in performance that the transformation gives to the Finnish trained model, as the x-inflected forms encode more morphosyntactic information. However, this boost does not occur in the case of Estonian. This might be due to differences in annotations between both treebanks or problems in translating schemata, since Estonian does not have a custom post-editing procedure (see Table 4.5). The effect of the conversion between schemata is something we would like to study in

Target	Source	Trans.	UAS	LAS
Galician	Catalan	No	41.21	24.56
		Yes	39.76 (-1.45)	21.44 (-3.12)
	Portuguese	No	57.96	49.21
		Yes	57.77 (-0.19)	47.20 (-2.01)
	Spanish	No	55.99	38.73
		Yes	55.91 (-0.08)	38.03 (-0.70)
Livvi	Finnish	No	36.69	22.58
		Yes	43.55 (+6.86)	28.76 (+6.18)
	Estonian	No	35.01	17.47
		Yes	34.48 (-0.53)	21.10 (+3.63)
Faroese	Danish	No	20.12	8.20
		Yes	27.61 (+7.49)	13.30 (+5.10)
	Icelandic	No	59.79	50.28
		Yes	60.99 (+1.20)	51.35 (+1.07)
	Bokmaal	No	10.24	5.34
		Yes	22.27 (+12.03)	12.18 (+6.84)
	Nynorsk	No	25.49	15.18
		Yes	29.08 (+3.59)	16.32 (+1.14)
	Swedish	No	19.24	5.06
		Yes	25.73 (+6.49)	7.66 (+2.60)
Latin	Catalan	No	16.20	6.74
		Yes	21.00 (+4.80)	9.52 (+2.78)
	French	No	21.10	8.71
		Yes	16.93 (-4.17)	6.74 (-1.97)
	Italian	No	18.71	9.44
		Yes	17.63 (-1.08)	7.63 (-1.81)
	Romanian	No	19.85	8.42
		Yes	29.17 (+9.32)	10.53 (+2.11)
	Spanish	No	16.29	6.80
		Yes	19.00 (+2.71)	7.20 (+0.40)
Hungarian	Finnish	No	25.80	7.11
		Yes	56.28 (+30.48)	33.04 (+25.93)
	Estonian	No	28.67	9.49
		Yes	26.93 (-1.74)	8.82 (-0.67)
Czech	Polish	No	28.11	18.56
		Yes	16.48 (-11.63)	7.68 (-10.88)
	Slovak	No	33.37	26.80
		Yes	35.81 (+2.44)	27.60 (+0.8)
Lithuanian	Latvian	No	29.62	12.08
		Yes	33.96 (+4.34)	15.85 (+3.77)
Slovenian	Bulgarian	No	30.15	14.01
		Yes	26.70 (-3.45)	12.05 (-1.96)
	Croatian	No	35.82	20.29
		Yes	30.09 (-5.73)	15.74 (-4.55)
	Serbian	No	35.85	18.59
		Yes	26.17 (-9.68)	12.89 (-5.7)
Welsh	Irish	No	42.83	14.86
		Yes	34.18 (-8.65)	11.74 (-3.12)
	S. Gaelic	No	25.92	8.79
		Yes	26.93 (+1.01)	10.03 (+1.24)
North Sami	Finnish	No	21.30	6.25
		Yes	18.21 (-3.09)	6.35 (+0.10)
	Estonian	No	26.36	9.03
		Yes	28.12 (+1.76)	10.52 (+1.49)

Table 4.6: Results for Experiment 1. The numbers between parentheses represent the score difference between the model trained on the x-inflected treebank and the original source one.

the future.

4.2.2. Experiment 2: Few-shot setup

Secondly, we test our method in a few-shot setup, combining a low-resource treebank with a group of source languages from the closest phylogenetic group available, as explained in Section 2.2.1. We compare the results obtained on the test set of the low-resource treebank by three different parsers:

- A parser trained only on the low-resource treebank.
- A parser trained on the phylogenetic group, without x-inflecting the source treebanks.
- A parser trained on the phylogenetic group, x-inflecting the source treebanks using our method.

The goal of this experiment is to check whether we are able to outperform sequence labeling parsers trained on a low-resource treebank by joining it with rich-resource treebanks of similar languages that have been transformed using the proposed method. Also, we want to measure if grouping a low-resource treebank with a group of related rich-resource treebanks can improve the parsing results; and although this idea is not new (Vilares, Gómez-Rodríguez, and Alonso, 2016; Ammar et al., 2016), it has not been tested for sequence labeling parsing.

Data

We selected 10 phylogenetic groups of languages consisting of a low-resource and a few rich-resource treebanks:

- Ibero-Romance. Galician (LR), Spanish, Catalan[1] and Portuguese (RR).
- North Germanic. Faroese (LR), Norwegian (Bokmaal), Norwegian (Nynorsk), Swedish and Icelandic (RR).
- Finno-Ugric. Hungarian (LR), Finnish and Estonian (RR).
- West Slavic. Czech (LR), Polish and Slovak (RR).

[1]The classification of Catalan as an Ibero-romance language is still discussed (Porras, 2014; Juge, 2007).

- South Slavic. Slovenian (LR), Bulgarian[2], Croatian and Serbian (RR).
- Romance. Latin (LR), Spanish, Romanian, French, Catalan and Italian (RR).
- Baltic. Lithuanian (LR), Latvian (RR).
- Celtic. Welsh (LR), Irish and Scottish Gaelic[3] (RR).
- Finnic. Livvi (LR), Finnish and Estonian (RR).
- Finno-Permic. North Sami (LR), Finnish and Estonian (RR).

We considered different group sizes and different degrees of phylogenetic closeness for completeness. For the low-resource target treebanks that did not have a development set, we used 20 % of the sentences of the training set to create a development set to train the low-resource parser.

Results

The results of this experiment are shown in Table 4.7. The scores obtained are in line with those obtained in the previous experiment. There are three groups out of ten where the model trained on the x-inflected treebanks outperforms the non-inflected group and the model trained on the low-resource treebank; for two of them (North Germanic and Baltic), all their language pairs obtained improvements when applying our method in the previous experiment. For the other (Finnic), our method also improved all the zero-shot scores except UAS for one pair.

From the seven treebanks where our method is not able to increase the results, in three of them (Finno-Ugric, Romance, and Finno-Permic), the x-inflected group obtains a better score than the original group, but worse than the low-resource treebank alone. This maybe indicates that the x-inflection process might be beneficial, but the grouping is not, as adding the selected treebank does not help the learning process. A couple questions that arise from this are: (i) whether it is a good idea to group all the proposed languages, (ii) and under which criteria should we group the treebanks. The opposite happens for the four remaining cases (Ibero-Romance, West Slavic,

[2]Script converted to latin.
[3]Larger than the Welsh treebank one, but not truly rich-resource.

Group	Model	UAS	LAS
Ibero-Romance	LR treebank	71.30	63.29
	Original	**81.17**	**74.21**
	Transformed	80.88 (-0.29)	73.90 (-0.29)
North Germanic	LR treebank	77.58	71.89
	Original	83.78	80.06
	Transformed	**85.23** (+1.45)	**81.14** (+1.08)
Finno-Ugric	LR treebank	**66.92**	**58.47**
	Original	54.45	43.47
	Transformed	59.70 (-7.22)	48.84 (-9.63)
West Slavic	LR treebank	72.43	66.78
	Original	**76.84**	**71.84**
	Transformed	71.87 (-4.97)	66.88 (-4.96)
South Slavic	LR treebank	59.44	50.29
	Original	**67.89**	**59.47**
	Transformed	55.57 (-12.32)	44.34 (-15.13)
Romance	LR treebank	**51.98**	**38.81**
	Original	33.69	22.80
	Transformed	49.69 (-2.29)	36.16 (-2.65)
Baltic	LR treebank	29.62	12.08
	Original	34.72	18.77
	Transformed	**37.64** (+2.92)	**23.02** (+6.69)
Celtic	LR treebank	78.21	65.23
	Original	**80.35**	**67.92**
	Transformed	76.25 (-4.10)	58.32 (-9.60)
Finnic	LR treebank	9.48	2.55
	Original	40.39	24.53
	Transformed	**53.83** (+13.44)	**38.91** (+14.38)
Finno-Permic	LR treebank	**54.38**	**44.47**
	Original	48.05	38.87
	Transformed	50.25 (-4.13)	39.69 (-4.78)

Table 4.7: Results of Experiment 2. Bold numbers represent the best result for each LR treebank. The numbers between parentheses represent the score difference between the x-inflected group and the best or second best result, depending on which model obtains the best result.

South Slavic and Celtic); the grouping process improves the scores but the x-inflection does not.

For simplicity, we decided to merge all the source treebanks from each phylogenetic group, instead of selecting only those that obtained good zero-shot results in the previous experiment. However, this is a suboptimal stra-

tegy, as seen in the results of this second experiment; it may would be better to mix x-inflected and not-inflected treebanks depending on their results in the zero-shot setup. The grouping process is harder to control, as it is not clear which criteria to follow when using auxiliary languages for transfer learning, and which number of languages would be optimal to group. We plan to study these aspects in future work.

4.3. Conclusion

In this chapter, we proposed a data augmentation technique to improve the performance of dependency parsers for low-resource languages. We trained a morphological inflection system using UM data of a low-resource target language, and then apply it to a UD rich-resource treebank from a similar language to create a cross-lingually inflected treebank more similar to the target language.

Following the work of Chapter 3, we used the x-inflected treebanks to train sequence labeling parsers on two different experiments: one testing its applicability in a zero-shot setup when no UD training data is available, and one testing its applicability in a few-shot setup when only a small UD training treebank is available. For both experiments, we need that the target languages have enough UM data to train a morphological inflectioner and related languages with rich-resource treebanks in UD.

The results show that this method improves the parsing results for some particular situations. However, further work is needed to better understand how different factors determine the usefulness of the methods, and if the observed results are reproduced when testing on models following other parsing paradigms like transition-based and graph-based parsers.

Chapter 5

Conclusion and future work

In this chapter, we discuss the findings of the work and propose some possible future lines of research.

5.1. Conclusion

The field of NLP has significantly advanced in the last decade, thanks to the use of deep learning models and the increase in the availability of data and the computing power. Yet, although these models obtain high accuracy in many tasks, they need annotated data to perform competitively, which is only available for a reduced number of the so-called rich-resource languages. Thus, most of the improvements and studies have been focused on these languages, which are not a representative sample of the human languages. Furthermore, deep learning models often require high volumes of data to be trained. Due to this, most human languages are being ignored by the advances of NLP, keeping millions of people away from their advantages and ignoring a lot of typological features when designing the systems. These aspects, among others, encourage us to study low-resource languages.

In the context of dependency parsing and NLP, this work has made two contributions: (i) studying the behaviour of different linearizations for dependency parsing as sequence labeling in low-resource languages, and (ii) proposing a cross-lingual method that uses morphological inflection for data-augmentation of related (rich-resource) treebanks, given a target low-resource treebank.

Regarding the first part of our work, we compared five different encodings from three different families, considering experiments that use gold

PoS tags, predicted PoS tags and no PoS tags as input. We selected a head selection encoding that uses a relative index distance and a PoS tag to encode the arcs, two bracketing-based encodings, that encode the dependency arcs using bracketing strings, and two mappings from two transition-based parsing algorithms (the arc-hybrid and non-projective Covington). To test this, we carried out two experiments. First, we restricted the training data for rich-resource treebanks, observing that head selection encodings are more efficient under ideal conditions. However, this advantage disappears when input data is not gold. The relative PoS-tag-based head selection encoding needs PoS tags in order to rebuild the dependency tree, hence using predicted PoS tags reduces its accuracy, specially in low-resource setups. Second, we repeated the experiments under real low-resource conditions. The results confirm more clearly the trend observed in the first experiment, reinforcing the utility of bracketing encodings in low-resource situations.

With respect to the second part, we presented a cross-lingual method that uses morphology inflection and it is essentially a data-augmentation approach. The main idea was that given a target low-resource language, a few related (rich-resource) languages are selected and transformed (based on their lemmas and morphological features) into a hybrid treebank, that we named x-inflected treebanks, that ideally should resemble and look similar to the target low-resource language. The results obtained show that this method can improve the results of dependency parsers in some particular low-resource situations, both for zero-shot and the few-shot scenarios. However, due to the large amount of factors that comes into play, further work is needed to understand better in which conditions and situations this method improve the performance of the parsers.

5.2. Future work

As mentioned during this work, we believe there is space for improvement, specially regarding Chapter 4.

First, notice that the proposed method is a data-augmentation approach, that works directly on the treebanks, but it is not really restricted to dependency parsing as a sequence labeling task. Then, next logical step of the work introduced in Chapter 4 would be to study how it performs when used together with other parsing paradigms, such as transition-based or graph-

based parsing. Note that this could be a first attempt to verify whether the trends (i.e. that a x-inflected treebank obtains better or worse performance than the baseline) are consistent across different parsers, and would help discard/confirm a potential reason that would explain the variability of the method across different treebanks and languages, i.e. the suitability of the underlying parsing model.

Another aspect that should be further studied is the behaviour of the morphological inflection system when facing unseen lemmas. As contemporaneously introduced in the work by Goldman, Guriel, and Tsarfaty (2021), the performance of morphological inflection systems drops dramatically when facing lemmas that have not been seen during the training phase, specially for low-resource languages. This problem is of vital importance for the task of morphological inflection, but it takes an extra dimension in our case, as most of the lemmas we are inflecting have not been seen by the models (we are just expecting they are *similar* to some lemmas of the target low-resource language), and some of the UM sets used to train the inflectioners have modest amounts of data. A step in this direction could be splitting the UM data in such a way that lemmas that appear in the training set do not appear in the development and test sets. This could help see if the morphological inflectioner generalizes worse/better and creates x-inflected forms that resemble less/more those of the target language.

A third interesting research line would be the study of this method for other language families, such as Semitic languages, which make an extensive use of morphological inflection and have enough resources in UD and UM to apply this method. However, most of these languages use different scripts, usually *abjads*, a type of alphabetic script that omits some or all vowels. Transliteration between scripts can then lose some information, so it must be carried out very carefully. Related to this last point, UD and UM are projects that are in constant expansion. Therefore, it is likely that in the future more resources will be available for more languages, and more options for combinations could be explored.

Overall, the factor or combination of factors that make the data-augmentation technique successful or not remains an open question. Typological databases such as WALS (Dryer and Haspelmath, 2013) or URIEL (Littell et al., 2017) to measure the similarity of languages might be useful for this. We decided not to rely on them in this work due to data disagree-

ment between resources; suitable language pairs suggested by the WALS or URIEL data did not meet our requirements, and languages that met our requirements do not have enough data in these databases to find suitable partners. However, the growing availability of data in these resources might make this solution feasible in the future.

References

Bibliography

[Aharoni and Goldberg2017] Aharoni, Roee and Yoav Goldberg. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada, July. Association for Computational Linguistics.

[Akbik, Blythe, and Vollgraf2018] Akbik, Alan, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

[Ammar et al.2016] Ammar, Waleed, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

[Anderson and Gómez-Rodríguez2020] Anderson, Mark and Carlos Gómez-Rodríguez. Distilling neural networks for greener and faster 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 2–13, Online, July. Association for Computational Linguistics.

[Anderson and Gómez-Rodríguez2021] Anderson, Mark and Carlos Gómez-Rodríguez. 2021. What taggers fail to learn, parsers need the most. *arXiv preprint arXiv:2104.01083*.

[Artetxe, Ruder, and Yogatama2020] Artetxe, Mikel, Sebastian Ruder, and

Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online, July. Association for Computational Linguistics.

[Balachandran et al.2020] Balachandran, Vidhisha, Artidoro Pagnoni, Jay Yoon Lee, Dheeraj Rajagopal, Jaime Carbonell, and Yulia Tsvetkov. 2020. Structsum: Incorporating latent and explicit sentence dependencies for single document summarization. *ArXiv e-prints*, pages arXiv–2003.

[Bejček et al.2013] Bejček, Eduard, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovskỳ, Anna Nedoluzhko, Jarmila Panevová, et al. 2013. Prague dependency treebank 3.0.

[Berzak et al.2016] Berzak, Yevgeni, Yan Huang, Andrei Barbu, Anna Korhonen, and Boris Katz. Anchoring and agreement in syntactic annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2215–2224, Austin, Texas, November. Association for Computational Linguistics.

[Bohnet et al.2018] Bohnet, Bernd, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 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, July. Association for Computational Linguistics.

[Buchholz and Marsi2006] Buchholz, Sabine and Erwin Marsi. 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, June. Association for Computational Linguistics.

[Cao et al.2019] Cao, Qingxing, Xiaodan Liang, Bailin Li, and Liang Lin. 2019. Interpretable visual question answering by reasoning on dependency trees. *IEEE transactions on pattern analysis and machine intelligence*.

[Caruana1997] Caruana, Rich. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

[Chen and Manning2014] Chen, Danqi and Christopher Manning. 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, Doha, Qatar, October. Association for Computational Linguistics.

[Church2017] Church, Kenneth Ward. 2017. Word2vec. *Natural Language Engineering*, 23(1):155–162.

[Cotterell et al.2018] Cotterell, Ryan, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D McCarthy, Katharina Kann, Sabrina J Mielke, Garrett Nicolai, Miikka Silfverberg, et al. 2018. The conll–sigmorphon 2018 shared task: Universal morphological reinflection. *arXiv preprint arXiv:1810.07125*.

[Cotterell et al.2017] Cotterell, Ryan, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. *arXiv preprint arXiv:1706.09031*.

[Cotterell et al.2016] Cotterell, Ryan, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. The sigmorphon 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

[Covington2001] Covington, Michael A. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, volumen 1. Citeseer.

[De Marneffe and Manning2008] De Marneffe, Marie-Catherine and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

[Dehouck, Anderson, and Gómez-Rodríguez2020] Dehouck, Mathieu, Mark Anderson, and Carlos Gómez-Rodríguez. 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, July. Association for Computational Linguistics.

[Dehouck and Gómez-Rodríguez2020] Dehouck, Mathieu and Carlos Gómez-Rodríguez. Data augmentation via subtree swapping for dependency parsing of low-resource languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3818–3830, Barcelona, Spain (Online), Decenber. International Committee on Computational Linguistics.

[Devlin et al.2019] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June. Association for Computational Linguistics.

[Dozat, Qi, and Manning2017] Dozat, Timothy, Peng Qi, and Christopher D. Manning. 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, August. Association for Computational Linguistics.

[Dror et al.2018] Dror, Rotem, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia, July. Association for Computational Linguistics.

[Dryer and Haspelmath2013] Dryer, Matthew S. and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

[Encyclopædia Britannica1998] Encyclopædia Britannica. 1998. Germanic languages. [Online; accessed October 1, 2021].

[Falenska and Çetinoğlu2017] Falenska, Agnieszka and Özlem Çetinoğlu. Lexicalized vs. delexicalized parsing in low-resource scenarios. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 18–24, Pisa, Italy, September. Association for Computational Linguistics.

[Fernández-González and Gómez-Rodríguez2019] Fernández-González, Daniel and Carlos Gómez-Rodríguez. Left-to-right dependency parsing with pointer networks. 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 710–716, Minneapolis, Minnesota, June. Association for Computational Linguistics.

[Garcia, Gómez-Rodríguez, and Alonso2018] Garcia, Marcos, Carlos Gómez-Rodríguez, and Miguel A Alonso. 2018. New treebank or repurposed? on the feasibility of cross-lingual parsing of romance languages with universal dependencies. *Natural Language Engineering*, 24(1):91–122.

[Goldman, Guriel, and Tsarfaty2021] Goldman, Omer, David Guriel, and Reut Tsarfaty. 2021. (un) solving morphological inflection: Lemma overlap artificially inflates models' performance. *arXiv preprint arXiv:2108.05682*.

[Gómez-Rodríguez, Alonso-Alonso, and Vilares2019] Gómez-Rodríguez, Carlos, Iago Alonso-Alonso, and David Vilares. 2019. How important is syntactic parsing accuracy? an empirical evaluation on rule-based sentiment analysis. *Artificial Intelligence Review*, 52(3):2081–2097.

[Gómez-Rodríguez, Strzyz, and Vilares2020] Gómez-Rodríguez, Carlos, Michalina Strzyz, and David Vilares. A unifying theory of transition-based and sequence labeling parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3776–3793, Barcelona, Spain (Online), Decenber. International Committee on Computational Linguistics.

[Gómez-Rodríguez2016] Gómez-Rodríguez, Carlos. 2016. Restricted Non-Projectivity: Coverage vs. Efficiency. *Computational Linguistics*, 42(4):809–817, 12.

[Gómez-Rodríguez and Nivre2013] Gómez-Rodríguez, Carlos and Joakim Nivre. 2013. Divisible Transition Systems and Multiplanar Dependency Parsing . *Computational Linguistics*, 39(4):799–845, 12.

[Hochreiter and Schmidhuber1997] Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Joshi et al.2020] Joshi, Pratik, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online, July. Association for Computational Linguistics.

[Juge2007] Juge, Matthew. 2007. The position of catalan in the romance language family: Evidence from the algherese dialect. *Research Enhancement Program Final Reports*, 01.

[Kiperwasser and Ballesteros2018] Kiperwasser, Eliyahu and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

[Kiperwasser and Goldberg2016] Kiperwasser, Eliyahu and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

[Klein and Manning2004] Klein, Dan and Christopher Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain, July.

[Kübler, McDonald, and Nivre2009] Kübler, Sandra, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.

[Kuhlmann, Gómez-Rodríguez, and Satta2011] Kuhlmann, Marco, Carlos Gómez-Rodríguez, and Giorgio Satta. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the*

49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 673–682, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Lacroix2019] Lacroix, Ophélie. Dependency parsing as sequence labeling with head-based encoding and multi-task learning. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 136–143, Paris, France, August. Association for Computational Linguistics.

[Lacroix et al.2016] Lacroix, Ophélie, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, June. Association for Computational Linguistics.

[Lacroix, Wisniewski, and Yvon2016] Lacroix, Ophélie, Guillaume Wisniewski, and François Yvon. Cross-lingual dependency transfer : What matters? assessing the impact of pre- and post-processing. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*, pages 20–29, San Diego, California, June. Association for Computational Linguistics.

[Lample et al.2016] Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *HLT-NAACL*.

[Le and Zuidema2015] Le, Phong and Willem Zuidema. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado, May–June. Association for Computational Linguistics.

[Li et al.2018] Li, Zuchao, Jiaxun Cai, Shexia He, and Hai Zhao. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

[Littell et al.2017] Littell, Patrick, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain, April. Association for Computational Linguistics.

[Liu et al.2018a] Liu, Liyuan, Jingbo Shang, Xiang Ren, Frank Xu, Huan Gui, Jian Peng, and Jiawei Han. Empower sequence labeling with task-aware neural language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volumen 32.

[Liu et al.2018b] Liu, Yijia, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430, Brussels, Belgium, October-November. Association for Computational Linguistics.

[Ma et al.2019] Ma, Chunpeng, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. Improving neural machine translation with neural syntactic distance. 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 2032–2037, Minneapolis, Minnesota, June. Association for Computational Linguistics.

[Ma and Hovy2016] Ma, Xuezhe and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August. Association for Computational Linguistics.

[Ma et al.2018] Ma, Xuezhe, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia, July. Association for Computational Linguistics.

[Magueresse, Carles, and Heetderks2020] Magueresse, Alexandre, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. *arXiv preprint arXiv:2006.07264*.

[Martins, Almeida, and Smith2013] Martins, André FT, Miguel B Almeida, and Noah A Smith. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622.

[McCarthy et al.2020] McCarthy, Arya D., Christo Kirov, Matteo Grella, Amrit Nidhi, Patrick Xia, Kyle Gorman, Ekaterina Vylomova, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, Timofey Arkhangelskiy, Nataly Krizhanovsky, Andrew Krizhanovsky, Elena Klyachko, Alexey Sorokin, John Mansfield, Valts Ernštreits, Yuval Pinter, Cassandra L. Jacobs, Ryan Cotterell, Mans Hulden, and David Yarowsky. UniMorph 3.0: Universal Morphology. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3922–3931, Marseille, France, May. European Language Resources Association.

[McCarthy et al.2018] McCarthy, Arya D., Miikka Silfverberg, Ryan Cotterell, Mans Hulden, and David Yarowsky. Marrying Universal Dependencies and Universal Morphology. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 91–101. Association for Computational Linguistics.

[McCarthy et al.2019] McCarthy, Arya D, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sabrina J Mielke, Jeffrey Heinz, et al. 2019. The sigmorphon 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. *arXiv preprint arXiv:1910.11493*.

[McClosky, Charniak, and Johnson2006] McClosky, David, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia, July. Association for Computational Linguistics.

[McDonald, Petrov, and Hall2011] McDonald, Ryan, Slav Petrov, and Keith Hall. 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., July. Association for Computational Linguistics.

[Mel'cuk and others1988] Mel'cuk, Igor Aleksandrovic et al. 1988. *Dependency syntax: theory and practice*. SUNY press.

[Mohananey, Kann, and Bowman2020] Mohananey, Anhad, Katharina Kann, and Samuel R. Bowman. Self-training for unsupervised parsing with PRPN. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 105–110, Online, July. Association for Computational Linguistics.

[Mulcaire, Kasai, and Smith2019] Mulcaire, Phoebe, Jungo Kasai, and Noah A. Smith. 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, November. Association for Computational Linguistics.

[Muñoz-Ortiz, Strzyz, and Vilares2021] Muñoz-Ortiz, Alberto, Michalina Strzyz, and David Vilares. Not all linearizations are equally data-hungry in sequence labeling parsing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 978–988, Varna, Bulgaria, September. INCOMA Ltd.

[Murawaki2019] Murawaki, Yugo. 2019. On the definition of japanese word. *arXiv preprint arXiv:1906.09719*.

[Naseem, Barzilay, and Globerson2012] Naseem, Tahira, Regina Barzilay, and Amir Globerson. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637, Jeju Island, Korea, July. Association for Computational Linguistics.

[Nicolai, Gorman, and Cotterell2020] Nicolai, Garrett, Kyle Gorman, and Ryan Cotterell, editors. 2020. *Proceedings of the 17th SIGMORPHON*

Workshop on Computational Research in Phonetics, Phonology, and Morphology, Online, July. Association for Computational Linguistics.

[Nivre2008] Nivre, Joakim. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

[Nivre and Fang2017] Nivre, Joakim and Chiao-Ting Fang. Universal Dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, Gothenburg, Sweden, May. Association for Computational Linguistics.

[Nivre et al.2007] Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 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, June. Association for Computational Linguistics.

[Pennington, Socher, and Manning2014] Pennington, Jeffrey, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Peters et al.2018] Peters, Matthew E, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Petrov, Das, and McDonald2011] Petrov, Slav, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

[Pimentel et al.2021] Pimentel, Tiago, Maria Ryskinaì, Sabrina MielkeZ, Shijie WuZ Eleanor Chodroff, Brian LeonardB, Garrett Nicolaiá, Yustinus Ghanggo AteÆ, Salam Khalifaè, Nizar Habashè, Charbel El-KhaissiS, et al. 2021. Sigmorphon 2021 shared task on morphological reinflection: Generalization across languages. *SIGMORPHON 2021*, page 154.

[Pires, Schlinger, and Garrette2019] Pires, Telmo, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July. Association for Computational Linguistics.

[Porras2014] Porras, Elber Aguilar. 2014. El catalán:¿ iberorromance o galorromance? *Repertorio Americano*, (24):95–114.

[Reimers and Gurevych2017] Reimers, Nils and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Rotman and Reichart2019] Rotman, Guy and Roi Reichart. 2019. Deep contextualized self-training for low resource dependency parsing. *Transactions of the Association for Computational Linguistics*, 7:695–713, March.

[Ruder2017] Ruder, Sebastian. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

[Ruder2020] Ruder, Sebastian. 2020. Why You Should Do NLP Beyond English. http://ruder.io/nlp-beyond-english.

[Sagae et al.2008] Sagae, Kenji, Yusuke Miyao, Rune Saetre, and Jun'ichi Tsujii. Evaluating the effects of treebank size in a practical application for parsing. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 14–20, Columbus, Ohio, June. Association for Computational Linguistics.

[Sato et al.2017] Sato, Motoki, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 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, Vancouver, Canada, August. Association for Computational Linguistics.

[Schuster and Paliwal1997] Schuster, Mike and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

[Shi, Livescu, and Gimpel2020] Shi, Haoyue, Karen Livescu, and Kevin Gimpel. On the role of supervision in unsupervised constituency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7611–7621, Online, November. Association for Computational Linguistics.

[Søgaard2011] Søgaard, Anders. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Song et al.2019] Song, Linfeng, Yue Zhang, Daniel Gildea, Mo Yu, Zhiguo Wang, et al. Leveraging dependency forest for neural medical relation extraction. 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 208–218.

[Spitkovsky, Alshawi, and Jurafsky2010] Spitkovsky, Valentin I., Hiyan Alshawi, and Daniel Jurafsky. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California, June. Association for Computational Linguistics.

[Spoustová and Spousta2010] Spoustová, Drahomíra and Miroslav Spousta. 2010. Dependency parsing as a sequence labeling task. *The Prague Bulletin of Mathematical Linguistics*, 94:7.

[Strubell et al.2018] Strubell, Emma, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 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, October-November. Association for Computational Linguistics.

[Strzyz, Vilares, and Gómez-Rodríguez2019] Strzyz, Michalina, David Vilares, and Carlos Gómez-Rodríguez. Viable dependency parsing as sequence labeling. 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 717–723, Minneapolis, Minnesota, June. Association for Computational Linguistics.

[Strzyz, Vilares, and Gómez-Rodríguez2020] Strzyz, Michalina, David Vilares, and Carlos Gómez-Rodríguez. Bracketing encodings for 2-planar dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2472–2484, Barcelona, Spain (Online), Decenber. International Committee on Computational Linguistics.

[Sylak-Glassman2016] Sylak-Glassman, John. 2016. The composition and use of the universal morphological feature schema (unimorph schema). *Johns Hopkins University.*

[Täckström, McDonald, and Nivre2013] Täckström, Oscar, Ryan McDonald, and Joakim Nivre. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia, June. Association for Computational Linguistics.

[Tiedemann, Agić, and Nivre2014] Tiedemann, Jörg, Željko Agić, and Joakim Nivre. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140.

[Vania et al.2019] Vania, Clara, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. 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 1105–1116, Hong Kong, China, November. Association for Computational Linguistics.

[Vilares, Gómez-Rodríguez, and Alonso2016] Vilares, David, Carlos Gómez-Rodríguez, and Miguel A. Alonso. 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, August. Association for Computational Linguistics.

[Vilares, Gómez-Rodríguez, and Alonso2017] Vilares, David, Carlos Gómez-Rodríguez, and Miguel A Alonso. 2017. Universal, unsupervised (rule-based), uncovered sentiment analysis. *Knowledge-Based Systems*, 118:45–55.

[Volokh2013] Volokh, Alexander. 2013. Performance-oriented dependency parsing.

[Wang and Eisner2018] Wang, Dingquan and Jason Eisner. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337, Brussels, Belgium, October-November. Association for Computational Linguistics.

[Wang et al.2019] Wang, Yufei, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. How to best use syntax in semantic role labelling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5338–5343, Florence, Italy, July. Association for Computational Linguistics.

[Wu and Dredze2019] Wu, Shijie and Mark Dredze. 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, November. Association for Computational Linguistics.

[Wu, Shapiro, and Cotterell2018] Wu, Shijie, Pamela Shapiro, and Ryan Cotterell. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium, October-November. Association for Computational Linguistics.

[Yang and Zhang2018] Yang, Jie and Yue Zhang. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System*

Demonstrations, pages 74–79, Melbourne, Australia, July. Association for Computational Linguistics.

[Zeman2008] Zeman, Daniel. Reusable tagset conversion using tagset drivers. In *LREC*, volumen 2008, pages 28–30.

[Zeman, Nivre, and others2020] Zeman, Daniel, Joakim Nivre, et al. 2020. Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[Zhang, Zhang, and Fu2019] Zhang, Meishan, Yue Zhang, and Guohong Fu. Cross-lingual dependency parsing using code-mixed TreeBank. 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 997–1006, Hong Kong, China, November. Association for Computational Linguistics.

[Zhou et al.2020] Zhou, Houquan, Yu Zhang, Zhenghua Li, and Min Zhang. Is pos tagging necessary or even helpful for neural dependency parsing? In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 179–191. Springer.

Appendix A

Publications

Muñoz-Ortiz, A., Strzyz, M., Vilares, D. (2021). Not All Linearizations Are Equally Data-Hungry in Sequence Labeling Parsing. In *Procceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 978-988, Varna, Bulgaria, September. INCOMA Ltd.