Multi-Lingual Dependency Parsing : Word Representation and Joint Training for Syntactic Analysis

Mathieu Dehouck

École Doctorale Sciences Pour L'Ingénieur

Thèse de Doctorat
Spécialité : Informatique et Applications

préparée au sein de l'équipe Magnet, du laboratoire Cristal
et du centre de recherche Inria Lille - Nord Europe
financée par l'Université de Lille

Mathieu Dehouck

Multi-Lingual Dependency Parsing : Word Representation and Joint Training for Syntactic Analysis

Parsing en Dépendances Multilingue :
Représentation de Mots et Apprentissage Joint
pour l'Analyse Syntaxique

sous la direction de Dr. Marc TOMMASI
et l'encadrement de Dr. Pascal DENIS

Soutenue publiquement à **Villeneuve d'Ascq**, le **20 mai 2019** devant le jury composé de:

Mme Sandra KÜBLER	Indiana University Bloomington	Rapportrice
M. Alexis NASR	Université d'Aix Marseille	Rapporteur
Mme Hélène TOUZET	CNRS	Présidente du jury
M. Philippe BLACHE	CNRS	Examinateur
M. Carlos GÓMEZ RODRÍGUEZ	Universidade da Coruña	Examinateur
M. Pascal DENIS	Inria	Encadrant
M. Marc TOMMASI	Université de Lille	Directeur

Multi-Lingual Dependency Parsing : Word Representation and Joint Training for Syntactic Analysis

Parsing en Dépendances Multilingue : Représentation de Mots et Apprentissage Joint pour l'Analyse Syntaxique

Mathieu DEHOUCK

20 Mai 2019

Contents

List of Figures

List of Tables

List of Algorithms

Abstract Syntactic analysis is a key step in working with natural languages. With the advances in supervised machine learning, modern parsers have reached human performances. However, despite the intensive efforts of the dependency parsing community, the number of languages for which data have been annotated is still below the hundred, and only a handful of languages have more than ten thousands annotated sentences. In order to alleviate the lack of training data and to make dependency parsing available for more languages, previous research has proposed methods for sharing syntactic information across languages. By transferring models and/or annotations or by jointly learning to parse several languages at once, one can capitalise on languages grammatical similarities in order to improve their parsing capabilities. However, while words are a key source of information for mono-lingual parsers, they are much harder to use in multi-lingual settings because they vary heavily even between very close languages. Morphological features on the contrary, are much more stable across related languages than word forms and they also directly encode syntactic information. Furthermore, it is arguably easier to annotate data with morphological information than with complete dependency structures. With the increasing availability of morphologically annotated data using the same annotation scheme for many languages, it becomes possible to use morphological information to bridge the gap between languages in multi-lingual dependency parsing.

In this thesis, we propose several new approaches for sharing information across languages. These approaches have in common that they rely on morphology as the adequate representation level for sharing information. We therefore also introduce a new method to analyse the role of morphology in dependency parsing relying on a new measure of morpho-syntactic complexity.

The first method uses morphological information from several languages to learn delexicalised word representations that can then be used as feature and improve mono-lingual parser performances as a kind of distant supervision. The second method uses morphology as a common representation space for sharing information during the joint training of model parameters for many languages. The training process is guided by the evolutionary tree of the various language families in order to share information between languages historically related that might share common grammatical traits. We empirically compare this new training method to independently trained models using data from the Universal Dependencies project and show that it greatly helps languages with few resources but that it is also beneficial for better resourced languages when their family tree is well populated. We eventually investigate the intrinsic worth of morphological information in dependency parsing. Indeed not all languages use morphology as extensively and while some use morphology to mark syntactic relations (via cases and persons) other mostly encode semantic information (such as tense or gender). To this end, we introduce a new measure of morpho-syntactic complexity that measures the syntactic content of morphology in a given corpus as a function of preferential head attachment. We show through experiments that this new measure can tease morpho-syntactic languages and morpho-semantic languages apart and that it is more predictive of parsing results than more traditional morphological complexity measures.

Résumé L'analyse syntaxique est une étape cruciale du traitement de la langue. Suite aux récentes avancées dans le domaine de l'apprentissage automatique, les parsers (analyseurs syntaxiques) atteignent des résultats comparables à ceux d'experts humains. Cependant, en dépit des efforts de la communauté, le nombre de langues ayant des données annotées est encore relativement faible et seules une vingtaine de langues ont plus de 10000 phrases annotées. Afin de lutter contre le manque de données d'apprentissage et rendre l'analyse syntaxique en dépendances accessible à plus de langues, des chercheurs ont proposé des méthodes pour partager de l'information syntaxique entre différentes langues. En transférant modèles et/ou annotations ou en apprenant à analyser plusieurs langues en même temps, l'on peut profiter des similarités grammaticales des différentes langues et ainsi améliorer leurs analyses respectives. Par contre, alors que les mots sont une source d'information importante pour l'analyse monolingue, ils sont bien moins facilement utilisables dans un contexte multilingue du fait de le grande variabilité même entre des langues proches. Les traits grammaticaux (personne, genre, mode, cas...) sont biens plus stables que les mots et ils encodent directement de l'information syntaxique. Il est également plus simple d'annoter du texte juste avec les traits grammaticaux qu'avec la structure en dépendances complète. D'autant plus qu'avec l'augmentation de nombre langues ayant des données annotées suivant les mêmes règles d'annotation, il devient possible d'utiliser l'information morphologique comme pont entre les langues pour l'analyse syntaxique multilingue en dépendances.

Dans cette thèse, nous présentons de nouvelles méthodes pour partager de l'information entre plusieurs langues. Elles ont en commun le fait d'utiliser la morphologie comme espace de représentation pour partager l'information. Nous présentons également une nouvelle mesure de la complexité morphosyntaxique nous permettant d'étudier le rôle de la morphologie dans l'analyse en dépendances.

La première méthode utilise de l'information morphologique de plusieurs langues pour induire des représentations de mots délexicalisées qui peuvent être utilisées ensuite pour améliorer les résultats de parsers monolingues. La seconde méthode traite la morphologie comme un espace de travail commun à toutes les langues pour y partager de l'information lors de l'apprentissage simultané de modèles d'analyse syntaxique. L'apprentissage y est guidé par l'arbre phylogénique des différentes familles de langues, ce qui permet de partager de l'information entre les langues historiquement liées susceptibles de partager des trait grammaticaux. Nous montrons par le biais d'expériences avec les données du projet Universal Dependencies que cette nouvelle méthodes d'apprentissage est bien plus efficace que l'apprentissage de modèles indépendants pour les langues ayant très peu de ressources, et qu'elle est aussi bénéfiques pour les langues mieux dotées dès que leurs branches sont biens fournies. Nous finissons avec une étude de la valeur intrinsèque de la morphologie pour l'analyse syntaxique. Dans les faits, alors que certaines langues utilisent la morphologie pour encoder de l'information syntaxique (avec les cas et les personnes), d'autres encodent surtout de l'information sémantique (comme le temps ou le mode). Ainsi nous introduisons une nouvelle mesure de la complexité morphosyntaxique qui quantifie l'information syntaxique contenue dans la morphologie en termes d'attachement préférentiel au gouverneur. Nous montrons par une série d'expériences que cette nouvelle mesure est capable de discriminer les langues morphosyntaxiques des langues morphosémantiques et qu'elle prédit mieux la qualité de l'analyse syntaxique d'une langue que les mesures plus traditionnelles de complexité morphologique.

Chapter 1

Introduction

Sentences mean more than the sum of their words meanings. This is in part due to their internal structure. Thus, computational methods to revealing the syntactic structure of natural language sentences (also called **syntactic parsers** or simply **parsers**) are at the core of many natural language processing systems (machine translation, summarisation, information retrieval...).

Dependency trees are a formalism used to represent sentences internal syntactic structure. In a dependency tree, each word (but one) is attached to the word that it modifies. For example, an adjective that modifies a noun attaches to that noun. Similarly, a subject or an object that modifies a verb, whether it is a noun, a proper noun or a pronoun, attaches to that verb. The set of all those directed binary relations reveals the structure of the sentence.

Because dependency trees are more flexible than phrase-structures (especially when considering languages with free word order) while preserving most of the relevant syntactic information, they have been the subject of many recent research works in the field of natural language processing. While early **parsers** were based on sets of rules explicitly coding dependency grammars, most modern works focus on the **data-driven** approach to dependency parsing, meaning that instead of relying on an explicit grammar to parse sentences, they make use of annotated data (also called treebanks) in order to **learn** a parsing model, that can be seen as a representation of a latent grammar (which has been used to annotate the data).

Building on recent advances in machine learning and word representation, modern state-of-the-art parsers achieve results comparable to human [SSO+14] with an accuracy of 90% [DQM17]. However, this is only true for well resourced languages with several thousands of annotated sentences to learn from. Furthermore, as data annotation is a lengthy error prone task, there are still only a handful of languages with more than 10000 annotated sentences. And most modern languages do not have annotated data at all[1]. In order to remedy this problem, and capitalising on the fact that languages share grammatical properties (similar word order and sentence structure), different methods have been proposed to perform **cross-lingual** or **multi-lingual dependency parsing**.

The idea behind cross-lingual dependency parsing, is to use annotated data from one or more source languages to learn a parser for a target language. Proposed methods include training delexicalised dependency parsers [MPH11], adapt-

[1] As of January 2019, the Universal Dependencies Project [NAA+18a] hosts treebanks for 76 languages, most of which are Indo-European languages. As there are treebanks for dead languages (Latin, Old Greek, Old French, Sanskrit...), this represents less than 1% of the estimated 7000 languages currently spoken on Earth according to Ethnologue [SF18].

ing existing treebanks to better fit target languages [AWY16] and using parallel corpora to transfer annotation [WP12]. However, those methods are asymmetrical and benefit only the target language. For example, one could use parallel corpora to adapt an English model to the related Scots language, but that would not improve the parsing capabilities of the English model regarding new English sentences.

The idea behind multi-lingual dependency parsing is indeed to use data from several languages in order to improve the parsing capabilities for each language. The basic intuition is that since languages have a lot of similarities, either because of common history, contact or chance, using data from several languages should help learning common patterns and improve overall parsing accuracy. For example, instead of learning an independent parsing model for each Scandinavian language (Danish, Swedish and both Norwegians), one could realise that because of their common ancestry, they are very similar and thus models could share information in order to learn on up to four times as much data as any single independent model. However, much less work has been conducted in this area compared to cross-lingual parsing.

The biggest obstacle to both cross-lingual and multi-lingual dependency parsing is lexicalisation (using word forms). Because, even very close languages with very similar grammars may have different spelling conventions (Spanish ll and ñ correspond to Portuguese lh and nh) or even use different writing systems altogether (German uses the Latin alphabet while Yiddish uses a variant of the Hebrew abjad), it is difficult to share information between languages at the word level. Many methods thus relied solely on parts-of-speech (word classes such as nouns, verbs, pronouns and adjectives) that are more easily identifiable across languages [LFDT14]. However, parts-of-speech are too shallow for delexicalised parsers to achieve state-of-the-art results. Thus more recently, following the advances in learning word representations, people have started to use cross-lingual word embeddings and clusters [AMB+16], but a lot of work still needs to be done for those methods to reach full usability and they tend to require a lot of data to give good results, data that are not available for all languages.

There is yet another alternative between shallow parts-of-speech and full lexicalisation, namely morphological information. **Morphological features** are grammatical categories that are expressed at the word level such as tense, mood, gender, number, case or person. Those features are appealing in a multi-lingual setting as they tend to be more stable than word form across related languages (in an evolutionary sens). For example, despite their very different forms, French and Spanish verbs conjugate for mostly the same tenses, persons and moods. Morphological features are also less numerous than word forms, which in turn makes it easier for parsers to learn with less data. Furthermore, some languages use morphology directly to encode syntactic information via cases or persons for example. Overall, morphological information adds an extra layer of the top of parts-of-speech allowing parsers to learn finer patterns, it is less sparse than word forms thus reducing the amount of data needed to learn, it can directly encode syntactic information, and it remains mostly stable across related languages.

Another obstacle that has slowed down research in cross-lingual and multi-lingual dependency parsing was the lack of consistently annotated data in different languages. In the early days of parsing, very few languages (mostly European languages) had annotated treebanks, and even when treebanks were available, they often followed different annotation guide lines, use different parts-of-speech sets,

different representation for morphological information and different dependency formalisms. This made both multi-lingual training and results comparison hard if not impossible.

Since 2015, the Universal Dependencies project has started to support the creation of annotated corpus with morphological information and dependency structure in a cross-lingually consistent manner for as many languages as possible. This initiative has made studying multi-lingual parsing easier and as the treebanks are now consistently annotated with morphological information, it becomes possible to study the role of morphological information in a multi-lingual learning setting.

Our thesis is that it is beneficial to share information across languages for learning models of natural languages syntax, and especially (but not only) so when a limited amount of annotated data is available. Indeed, languages share a lot of structures amongst which syntactic ones, either because of shared history such as common ancestry or prolonged contact, or because languages follow general trends (sometimes called linguistic universals). As we abstract away from word forms, similarities between syntactic structures of different languages become more apparent and therefore easier to capture. Morphological attributes reveal shared structures across languages at a finer level than raw parts-of-speech and also already encode syntactic information in certain languages. We will therefore focus on the use of morphological information as a mean to share syntactic information across languages.

Amongst several available parsing frameworks that we will present in Chapter 2, we chose to work with graph-based parsing models as they have proven to achieve state-of-the-art performances, as re-emphasised by Dozat et al. [DQM17], and give full power to the data representation compared to transition-based parsers where data representation is only one parameters alongside the transitions set, the oracle used for training and the whole possibilities offered by reinforcement learning models. Using well established online learning techniques and mostly linear models, allows us to investigate the impact of using morphology and sharing information between languages on dependency parsers.

Our first contribution follows the path of previous works on learning representations for dependency parsing. We investigate the ability of morphological information represented by **delexicalised word embeddings** to complement traditional feature vectors. Delexicalised word embeddings are word vectors that only encode morphological information and ignore lexical and semantic information in order to be more easily shareable across languages. We then show that using multi-lingual morphological information helps learning representation of morphological features that will then be used by mono-lingual parsers, acting as a form of distant supervision.

Our second contribution is to use morphological features as a common space to share syntactic information between related languages. As languages evolve over time, we can imagine that languages that have diverged more recently share more common features with each other than with more distantly related languages. Following the idea of language evolution and divergence, we tie parsing models parameters of several languages together so that they learn from each other's data until they need to diverge. The moment at which models can diverge is determined by the **phylogenetic tree** of the language families in order to mimic

model evolution. This way, closely related languages models will share a longer training time than models of distant languages. We call this training procedure **phylogenetic training**. It is interesting to note that this generic procedure can adapt to many different learning frameworks such as large margin classifier or neural networks. Furthermore, it also gives models that can be used to parse languages without training data at all (also called **zero-shot parsing**) as soon as we know where they sit in the tree. We empirically show that phylogenetic training is really beneficial for languages with few training data but that it also helps improving results for well resourced languages when their family tree is well populated. We also compare it to another multi-lingual parsing technique inspired by model propagation in similarity graph and we show that phylogenetic training is a viable option for zero-shot parsing.

Our third contribution is to look at the intrinsic value of morphological information for dependency parsing. Indeed not all languages have the same use of morphology. While some have a rather impoverished morphology, some have a very productive one and they use it to encode semantic and/or syntactic information. We also mentioned that as morphological features are less numerous than word forms, they reduce data sparsity. By a series of parsing experiments using various sources of information (word forms, morphological features, learned representations...) we show that morphology plays indeed a double role in dependency parsing. For all languages, morphological information reduces data sparsity and thus improves parsing accuracy. Furthermore, for a subset of morphologically rich languages that we call *morpho-syntactic languages* as opposed to *morpho-semantic languages*, morphology directly encodes syntactic information and thus improves parsing accuracy beyond mere sparsity reduction, even when morphological features have been poorly predicted. However, traditional measures of **morphological complexity** are unable to predict those parsing results. We therefore, introduce a new measure of **morpho-syntactical complexity** called **Head Part-of-speech Entropy** (HPE) that focuses on the head attachment preferences of morphological features and that is able to tease morpho-syntactic languages and morpho-semantic languages apart. We eventually use this new measure of morpho-syntactic complexity to investigate the weight of annotation choices on parsing performances.

1.1 Outline

The remain of this thesis is organised as follow:

In Chapter 2, we introduce the problem of syntactic dependency parsing as a natural language processing task and present different approaches to solving it. We focus on the graph-based dependency parsing framework and thus describe algorithms used to retrieve trees from weighted directed graphs, also called tree inference. As we work with trees and graphs throughout this thesis, we also presents quickly a few concepts from graph theory. We end this preliminary chapter with a presentation of the machine learning tools used in this work, and most importantly online learning algorithms for structured prediction.
The reader familiar with dependency parsing, tree inferences and discriminative learning for structure prediction can easily skip this chapter.

In Chapter 3, we discuss the problem of representing lexical information (words) in a machine friendly format. We first present some linguistic tools used to analyse

words and to encode information in natural language processing such as parts-of-speech, morphological features and lemmas. Then we look at more recent approaches to word representation based on statistical machine learning that automatically induce representations for words from raw text.
The reader familiar with linguistic and/or statistical word representations can easily skip this chapter.

In Chapter 4, we focus on the problem of cross-lingual and multi-lingual dependency parsing. It presents the Universal Dependency project, which is our main source of cross-lingual annotated data. It presents the data and discusses a few issues that arise when using this type of cross-lingual data. In this chapter, we also review related works on cross-lingual and multi-lingual dependency parsing.

In Chapter 5, we focus on the representation of the input data and not on the learning process itself. Following the trend of learning word representation, we propose a simple method to learn cross-lingual delexicalised morphological representation. We show via a series a experiments that even in a mono-lingual parsing setting, using cross-lingually induced morphological representation helps improving parsing results over mono-lingually induced representations. This shows that even distant multi-lingual supervision can be useful and that morphology is a relevant mean to share information between languages.

In Chapter 6, we turn to the actual problem of training multi-lingual parsers. We present a multi-task learning framework that make use of several tasks diverging history to guide their simultaneous learning process. We instantiate this framework with both a linear parser and a neural parser, leveraging on languages evolutionary history as represented by a phylogenetic tree to share information between them. We show that learning several parsing models side by side using the phylogenetic tree is beneficial, especially for poorly resourced languages and dense branches of the tree.

In Chapter 7, we venture in computational linguistics territory and investigate the actual role played by morphological information in dependency parsing. Morphological analysis can be used as a bridge between languages. It also reduces data sparsity as there are much less morphological attributes than word forms for example. But morphology can also encode syntactic information by itself. As such, it should play different roles in dependency parsing. In this chapter, we try to disentangle data sparsity reduction from syntax encoding using morphological complexity measures and various word representations. We show that while morphological complexity measures alone do not take syntactic information into consideration and thus are poorly predictive of parsing results, our newly proposed measure of morpho-syntactic complexity can tell languages that rely on morphology to encode syntax from others.

In Chapter 8, we conclude this thesis by discussing current limitations of our methods and opening directions for future works.

In Chapter 9, we present a few extra result tables that did not find their room in the body of this thesis.

Chapter 2

Preliminaries

Dependency parsing as a natural language processing task takes sequences of linguistic tokens as inputs (sentences) and outputs linguistic structures over those inputs (dependency trees) that match a set of rules (a grammar) nowadays mostly learned automatically from annotated data and represented by weight models. An automated system addressing this task is called a dependency parser. Parsers are made up of three key components : 1) a data representation mechanism whose goal is to encode input sentences into a mathematical representation also called a feature vector; 2) a scoring function encoding grammatical rules that can either take the shape of hard-coded rules or of parameters learned from annotated data; and 3) a tree inference that is used to retrieve the dependency structure of an encoded input sentence scored by the scoring function.

In this work we investigate different methods to improve automatic syntactic analysis when one has access to data from several languages. To do so, we rely on concepts from Natural Language Processing, Machine Learning, Mathematics and Linguistics. In this chapter, we introduce important concepts from each field on which we build our work up.

Because dependency trees are graphs and that we also use graphs as learning tools later in this work, we start by introducing a few concepts from graph theory in Section 2.1. We then turn to dependency parsing proper.

Section 2.2 introduces the concept of syntactic analysis using dependency trees, as well as the problem of automatically predicting those dependency trees. It presents some broad classes of parsers such as transition-based parsers and graph-based parsers. Most importantly, this section presents the Eisner's and Chu-Liu-Edmonds' algorithms for retrieving trees in directed graphs that we use in subsequent chapters. In this section, we also discuss the evaluation of parsers. This section roughly corresponds to the inference part (3) of a parser.

In this work, as it the case for most modern parsers, the actual structural constraints are not hard-coded in the shape of grammars but are automatically learned from annotated data. Section 2.3 introduces the necessary background in machine learning and more specifically structure prediction and online learning. Amongst other, we describe there the Perceptron algorithm and the Passive-Aggressive algorithm used in this work. This section roughly corresponds to the scoring function part (2) of a parser.

For most of the parsing history, inputs (sentences, edges or parser states) were represented by shallow long sparse feature vectors typically encoding the identity of words involved, pairs of words, parts-of-speech and so on. Only recently, following advances in word representation, have parsers started to use deeper more

contextualised representations for their inputs. As word representation is a pivotal aspect of our work and that it has evolved independently from dependency parsing, we will discuss it, from both a linguistic and a natural language processing perspective in a following chapter.

Similarly, as the development of consistently annotated multi-lingual training data is somewhat orthogonal to the problem of parsing itself, we keep the description of the training data used in this work as well as the most important related work on multi-lingual dependency parsing for next chapter.

2.1 Snippet of Graph Theory

Graph theory deals with sets of objects structured by pairwise relations. Those structured sets are called graphs. It is a basic framework used for representing syntactic structures as well as language relationship.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a set of vertices (or nodes) $v \in \mathcal{V}$ that may be linked to each other by edges $e \in \mathcal{E}$ that represent relations between those vertices $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Two vertices v_i and v_j are said to be adjacent if they are linked by an edge e_{ij}. If the relation represented by $\mathcal{E}$ is symmetrical, we say that the graph is undirected and we have that $e_{ij} = e_{ji}$ for all pairs of adjacent vertices. Figure 2.1 depicts a simple undirected graph. It has five vertices and four edges.

If however, the relation represented by $\mathcal{E}$ is asymmetrical, the graph is directed. In that case, e_{ij} is the edge going from i to j and e_{ji} goes in the other direction. In general, the existence of an edge e_{ij} does not imply the existence of an edge e_{ji} going in the other direction. Figure 2.2 shows a directed graph.

Some definitions allow self loops (edges going from a vertex to itself), but as this is mostly irrelevant for the structure we are considering in this thesis, we will assume that self loops are not allowed.

A graph can also be weighted. In that case, a graph is a triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ with $\mathcal{W} = \{w_{ij} \in \mathbb{R}, \forall e_{ij} \in \mathcal{E}\}$ being the weights of the edges of $\mathcal{G}$. Edges not in $\mathcal{E}$ are often given a weight of 0. Figure 2.3 shows a weighted graph.

A path is a sequence $p = \{p_t \in \mathcal{E} | t \in \mathbb{N}_{|p|}\}$ of edges of length $|p|$, such that if $p_t = e_{ij}$ and $p_{t+1} = e_{i'j'}$ then $j = i'$. We also assume that $\forall t, t' \in \mathbb{N}^2_{|p|-1}$, such that $t < t'$, $p_t = e_{ij}$ and $p_{t'} = e_{i'j'}$ then $i \neq j'$. A path cannot go through an vertex more than once, except maybe the first/last vertex of the path, in that case the path is also called a cycle. We say that two vertices v_i and v_k are connected if there exists a path $p \in \mathcal{E}^*$ such that p goes from u to v. A connected component is a maximal subset of $\mathcal{V}$ such that any two vertices in it are connected.

We say that $\mathcal{G}$ is a tree if it is connex (it has only one connected component) and it has no cycle. It means that between any two vertices, there is exactly one path, no more nor less. Figure 2.4 represents a tree.

In a tree, a single vertex can be promoted to be the *root* of the tree. In a rooted tree, for a given edge e_{ij}, the vertex v_i is closer to the tree root than v_j. We say that v_i is the parent of v_j and that v_j is the child of v_i. The transitive closure of the child relation is called descendant relation. Thus the descendants of v_i are either v_i's children or are children of another descendant of v_i. Conversely, all the vertices between v_j and the root of the tree are the ancestors of v_j.

There are several ways to represent a graph. The adjacency matrix A of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is a square matrix representing the edges of the graph. The

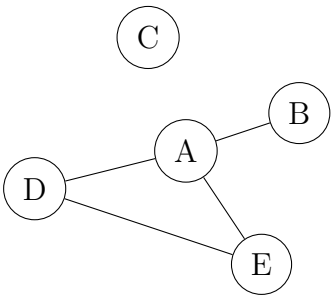


Figure 2.1: A simple graph.

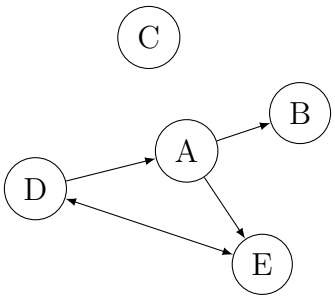


Figure 2.2: A simple directed graph.

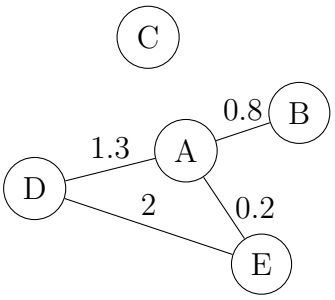


Figure 2.3: A weighted graph.

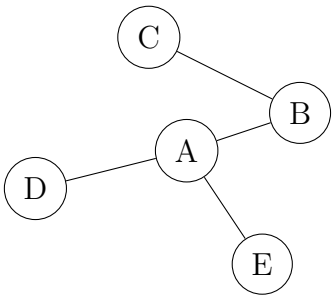


Figure 2.4: A simple tree.

matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined as:

$$A_{ij} = \begin{cases} \mathcal{W}_{ij} \text{ if } e_{ij} \in \mathcal{E}, \\ 0 \text{ otherwise.} \end{cases}$$

If $\mathcal{G}$ is unweighted then we can set $\mathcal{W}_{ij}$ to 1 in the above definition. We see that given this definition, the adjacency matrix of an undirected graph is always symmetrical while it needs not be the case for general directed graphs.

Linked to the adjacency matrix is the matrix of degree. The degree of a vertex v_i is the number of adjacent vertices v_j in $\mathcal{G}$. If $\mathcal{G}$ is weighted then the degree of v_i is the sum of its edges weights. The matrix of degree $D \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined as:

$$D_{ii} = \sum_{j|e_{ij}\in\mathcal{E}} \mathcal{W}_{ij},$$

$$D_{ij} = 0 \text{ if } i \neq j.$$

If $\mathcal{G}$ is directed then we define the out-degree and the in-degree to be the sum of the outgoing edges weights and the sum of the incoming edges weights respectively.

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

A and D are the adjacency matrix and the degree matrix of the graph of Figure 2.1.

2.2 Dependency Parsing

We call a sentence $x = x_1 x_2 \dots x_n$, a sequence of words, numbers and punctuation symbols of length $|x| = n$. In the following, we will use *word* as a generic term for words, numbers and punctuation symbols.

A dependency relation is a directed relation between two words x_i and x_j such that one is the governor (or head or parent) and the other is the dependent (or child). Typically, those relations encode syntactic/semantic relations, such as the fact that a determiner or an adjective attach to a noun or that noun used as a direct object attaches to a verb.

A dependency graph $\mathcal{G}$ over a sentence x is a graph whose vertices $\mathcal{V} = \{x_i | i \in \mathbb{N}_n\}$ are the words of x in the sequential order and whose edges $\mathcal{E} = \{e_{ij} | (i,j) \in \mathbb{N}_n^2, i \neq j\}$ are dependency relations between those words.

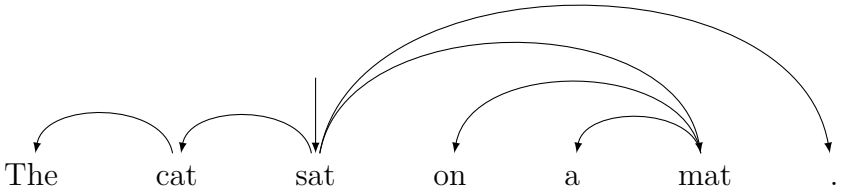


Figure 2.5: Dependency structure for the sentence *"The cat sat on a mat."*

Different linguistic theories of syntactic dependency can lead to different dependency graph for a single sentence x. Some are arbitrary graphs while others are constrained. The most widely studied dependency graphs in NLP are by far dependency trees. Meaning that each word has at most one governor and that only one word has no governor. This word is the *root* of the tree. Figure 2.5 gives an example of such a dependency tree.

Dependency relations can further be typed. In that case we say that the dependency structure is labeled. Typical labels for dependency relations are syntactic roles like *Subject* and *Object*. But again, types can vary from theory to theory. Figure 2.6 gives the labeled version of the structure from Figure 2.5.

An annotation scheme is the translation of a dependency theory into a set of rules used to derive trees for sentences. Whilst most rules of a scheme are directly derived from a given theory, conventions are often necessary to keep the structures constrained. For example, most theories have subjects and objects governed by verbs. However, in sentences with several possible governors like in *"Peter bought and ate apples"*, where *Peter* could depend on both *bought* and *ate*, rules might be necessary to break ties, especially if one wants to work with trees.

Dependency trees can be further qualified as projective or non-projective. A tree is projective if it can be drawn above the sentence without any crossing edges. More formally, for any dependency relation e_{ij} on x, if for all k such that $i < k < j$ (resp. $j < k < i$), x_k is a descendant of x_i, then e_{ij} is projective. If all the dependencies of a tree are projective then the tree itself is projective.

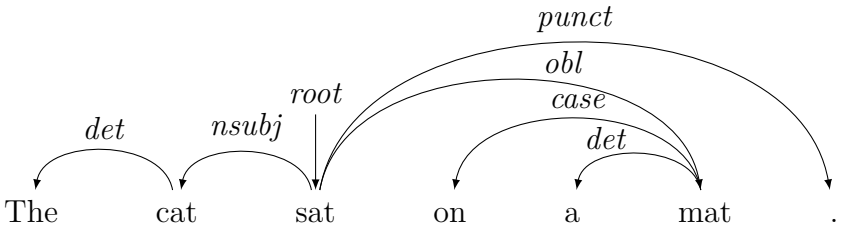


Figure 2.6: Labeled dependency structure for the sentence *"The cat sat on a mat."*

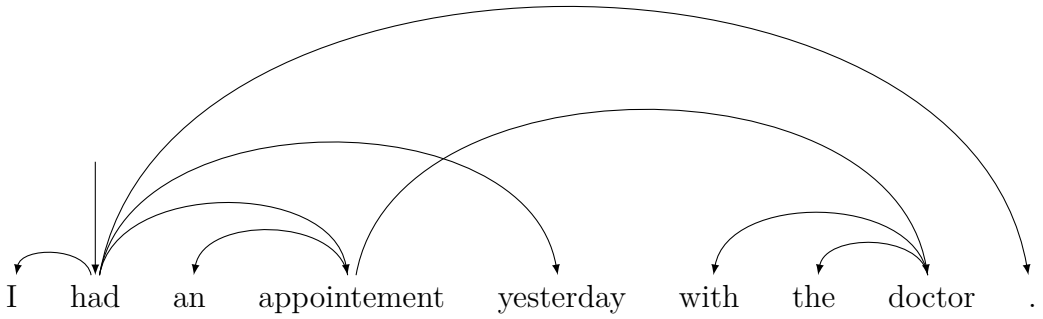


Figure 2.7: Dependency structure for the sentence *"I had an appointement yesterday with the doctor."*

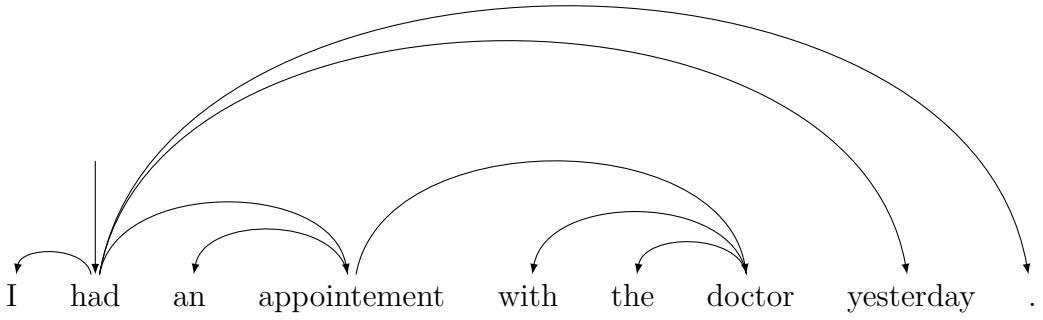


Figure 2.8: Dependency structure for the sentence *"I had an appointement with the doctor yesterday."*

This is an important distinction for two reasons. First, from a linguistic perspective, projectivity links with word order and typology. Languages that do not rely on word order to express syntactic roles tend to be less projective than those that have stricter word orders. For example, words can move rather freely in the sentence in classical Latin because Latin relies on declension for marking syntactic roles. In English or French, core roles (subject, direct and indirect objects) are marked by position in the sentence. But while other phrases are more free to move, words inside them are still fixed. There even exist languages that are strictly projective, like Japanese for example with a very strict word order, where phrases can only move so far as the scope of their governor allows.

Let us take the sentence: *"I had and appointment yesterday with the doctor."* We could also write: *"I had and appointment with the doctor yesterday."* or again: *"Yesterday I had and appointment with the doctor."* but not **"I had and appointment with yesterday the doctor."* or **"I had and appointment yesterday the with doctor."* The first sentence is non-projective (Figure 2.7) while the second one is (Figure 2.8).

Second, from a computational perspective, we shall see that some parsing algorithms have restrictions on the kind of structure they can retrieve, the most common one being projectivity. Thus some parsers will only be able to produce projective trees and thus will be more suited to some languages than others.

As a way of encoding syntactic information, dependency trees are a useful tool for downstream tasks such as automated translation or information retrieval.

Given the above definition of a dependency tree, we shall now define the problem of finding those trees in such a way addressable by computers.

2.2.1 Dependency Parsing as a NLP Task

The problem of dependency parsing is to find the best dependency structure (from now on, we will assume that this structure is a tree) for a sentence given a set of rules that we will define later. From a computational point of view, an algorithm to perform dependency parsing is an algorithm $\mathcal{A}$ of type $\mathcal{R} \to (\mathcal{X} \to \mathcal{Y})$, where $\mathcal{R}$ is the set of sets of rules constraining the dependency structures, $\mathcal{X}$ is the set of sentences and $\mathcal{Y}$ is the set of dependency structures. Let $r \in \mathcal{R}$ be a specific set of rules, then the instance $\mathcal{A}_r$ is an algorithm that takes a sentence x as input and outputs a tree y. An example of such a rule would be that an adjective appearing at the left of a noun attaches to that noun, as it is the case in French.

The sets of rules that constraint dependency trees, can be given in very different ways. The first dependency parsers used explicit hand crafted hard coded rules in the shape of dependency grammars [KMN09]. However, this requires a huge amount of error prone work and human expertise, and because human languages evolve fast, are ambiguous and full of idiosyncrasies, such hand crafted rule sets are not able to generalise well to the broad range of language uses. Thus moderns parsers rely on implicit rules given in the shape of annotated example sentences sets called treebanks. Because of the implicit nature of those rules, parsers use machine learning techniques to learn them from annotated examples. We will describe those annotated examples in greater details in Section 4.2.

Assuming one has access to a mechanism that would tell how good a tree y is for a given sentence x and a rule set r, like a scoring function for example, then we would just need to search through the space of trees and pick the best:

$$\hat{y} = \underset{y \in \mathcal{Y}_x}{\operatorname{argmax}} \, Score_r(x, y).$$

The problem here, is that the space of possible trees grows exponentially with the size of the sentence and thus makes it impossible in practice to search through all trees for reasonable sentence sizes. Cayley's formula gives n^{n-1} rooted trees for a sentence of length n, meaning that for a sentence of 10 words, there are already a billion trees to search. Thus one needs to resort to more clever algorithms to effectively find a good parse tree for a sentence.

Over the years, different approaches have been proposed to the problem of dependency parsing. They all have in common to factor the score of a tree in some way in order to make the search tractable. The two most successful and widely used today are the so called graph-based dependency parsing and transition-based dependency parsing. We review them in the next sections. Other approaches like those based on grammar formalism are less trendy today but are worth mentioning, so we quickly present some afterward.

2.2.2 Graph-based Parsing

In graph-based parsing, we use graph-theoretic tools to retrieve the best dependency structure given a sentence.

To make the problem tractable, in practice we factor a tree score as the sum of the scores of its sub-parts. The most common factorisation is edge factorisation,

also called first order factorisation. Each possible edge receives a score and the score of a tree is just the sum of its edges scores. Edges can be seen as sub-trees of size 1 or sub-trees containing only one edge, thus the name of first order factorisation. Then the problem becomes:

$$\hat{y} = \underset{y \in \mathcal{Y}_x}{\operatorname{argmax}} \sum_{e \in y} score(x, e).$$

Any function that can assign a score to an edge can be used here, but in practice, linear functions are used most of the time. Thus, given a function $\boldsymbol{\phi}$ turning an edge e into a vector $\boldsymbol{\phi}(e) \in \mathbb{R}^d$ of length d. The vector $\boldsymbol{\phi}(e)$ is also called the feature vector of e. The score of e is then given by:

$$score_\theta(x, e) = \boldsymbol{\theta} \cdot \boldsymbol{\phi}(e),$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the weight vector corresponding to the linear scoring function, and is to be learned from some data. We will see in Section 2.3 how this weight vector is learned, but for now we assume that it exists.

Typically, the feature vector $\boldsymbol{\phi}(e)$ of an edge e encodes information about the parts-of-speech and forms of the two ends of the edge. It also often encodes information about words surrounding the two ends and the signed[1] length of the relation. Those vectors used to be one-hot encoded, thus they were very long (several millions of dimensions) and very sparse (less than a hundred active dimensions at a time). Those big one-hot vectors are not very good at sharing information between dimensions, thus they have been complemented with smaller, denser representations. With the advances in representation learning and neural networks, they even tend to be completely replaced by small, dense representation in very modern models as we shall see later. Table 2.1 gives the detail of the feature vector template used in the MSTparser of McDonald et al. [MCP05a]. Those features have been widely used, adapted and extended in subsequent works on graph-based dependency parsing.

Once a score has been given to each possible edge (or sub-structure), we can use tree inferences to find the best scoring tree for a given sentence. The two most common tree inferences used are the Chu-Liu-Edmonds' algorithm [CL65] which is a purely graph-based method and the Eisner's algorithm [Eis96] originating in grammar parsing and dynamic programming. We review those two algorithms in the next sections.

Chu-Liu-Edmonds' Algorithm

The problem of finding a tree covering all the vertices of a weighted graph that maximises (or minimises) a cost function is an old problem in computer science. An algorithm solving for that problem is called a spanning tree algorithms. Basic spanning tree algorithms like Kruskal's and Prim's algorithms are fast greedy algorithms but only apply to undirected graphs, while in dependency structures, edge direction is semantically important.

The fact that we work with directed graphs has an important algorithmic implication. We cannot use purely greedy approach. Chu-Liu-Edmonds' algorithm

[1] Because relations are directed (a noun depending on a verb is not the same as a verb depending on a noun), to distinguish between edges going to the left and to the right, their length is signed so that an edge with a positive length goes in the direction of the text while a edge with a negative length runs backward.

Uni-gram
H-form, H-pos
H-form
H-prefix, H-pos
H-prefix
H-pos
D-form, D-pos
D-form
D-prefix, D-pos
D-prefix
D-pos

Bi-gram
H-form, H-pos, D-form, D-pos
H-form, H-pos, D-form
H-form, H-pos, D-pos
H-form, D-form, D-pos
H-pos, D-form, D-pos
H-form, D-form
H-prefix, H-pos, D-prefix, D-pos
H-prefix, H-pos, D-prefix
H-prefix, H-pos, D-pos
H-prefix, D-prefix, D-pos
H-pos, D-prefix, D-pos
H-prefix, D-prefix
H-pos, D-pos

Tri-gram
H-1-pos, H-pos, D-pos
H-pos, H+1-pos, D-pos
H-pos, D-1-pos, D-pos
H-pos, D-pos, D+1-pos

Tetra-gram
H-1-pos, H-pos, D-1-pos, D-pos
H-pos, H+1-pos, D-1-pos, D-pos
H-1-pos, H-pos, D-pos, D+1-pos
H-pos, H+1-pos, D-pos, D+1-pos

In Between
H-pos, B-pos, D-pos

Table 2.1: Feature templates for the one-hot edge feature vector for dependency parsing adaptated from [MCP05a]. H stands for the head of a relation, D for the dependent, ±1 selects the word before/after the given word. B is any word in between the Head and the Dependent of the relation. On top of forms and POS, prefixes of length 5 are used as back-offs for unseen words. All those templates are further conjoined with the binned signed length of the relation.

(CLE for short) is a maximum spanning tree algorithm for directed graphs. CLE algorithm alternates between two stages. In a first greedy stage, it picks the best incoming edge for each vertex. Because the greedy stage might have created cycles, then a second stage contracts cycles and update the scores of the edges involved in them, leading to a new graph on which to apply the algorithm anew. This is repeated recursively until the greedy stage does not create any more cycle, then all the contractions are unfolded to make the final tree. We give pseudo-code for the algorithm in Algorithm 1.

Data: a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$
Result: a dependency tree $\mathcal{G}_T$
begin
  $\mathcal{E}' = \{e_{ij} \mid e_{ij} \in \mathcal{E},\ \mathcal{W}_{ij} = \max_{i' \in \mathbb{N}_n} \mathcal{W}_{i'j}\}$
  $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \mathcal{W})$
  if $\mathcal{G}'$ *has no cycles* **then**
    return $\mathcal{G}'$
  Find any $\mathcal{E}_C$ that is a cycle in $\mathcal{G}'$
  $\mathcal{G}_C, ep = \text{contract}(\mathcal{G}', \mathcal{E}_C)$
  $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T) = \text{CLE}(\mathcal{G}_C)$
  $v_j = ep(e_{ic})$
  Find $v_k \in \mathcal{E}_C$ such that $e_{kj} \in \mathcal{E}_C$
  Add e_{ij} to $\mathcal{E}_T$
  Add e_{il} to $\mathcal{E}_T$ where $v_i = ep(e_{cl})$ for each $e_{cl} \in \mathcal{E}_T$
  Add e_{il} to $\mathcal{E}_T$ for each $e_{il} \in \mathcal{E}_C$ except e_{kj}
  return $\mathcal{G}_T$

Algorithm 1: Chu-Liu-Edmonds algorithm, the *contract* routine is given in Algorithm 2.

Let us explain how the algorithm works in more details. In the first stage, for each vertex in graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, pick the best incoming edge (the best parent) with regard to the attachment scores. This is the greedy stage. Then check if the resulting graph is a well formed tree. If that is the case, then it is the maximum spanning tree, we return it and the algorithm stops. If on the contrary, the resulting graph is not a well formed tree, then there must be at least one cycle $\mathcal{C}$ in it.

The second stage, takes any cycle $\mathcal{C}$ and contracts it. Create a new graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{W}')$ where $\mathcal{V}' = \mathcal{V}/\mathcal{C} \cup \{v_\mathcal{C}\}$. $v_\mathcal{C}$ is a vertex that will represents the contracted cycle $\mathcal{C}$. We need to remember the vertices that were in $\mathcal{C}$ to be able the reconstruct the tree for the original graph at the end. $\mathcal{E}' = \{e_{ij} | (i,j) \in \mathcal{V}' \times \mathcal{V}'\}$, edges in $\mathcal{C}$ are removed and edges going from (to) $\mathcal{C}$ to (from) other vertices are replaced by edges going from (to) $v_\mathcal{C}$. The new weights (scores) $\mathcal{W}'$ are such that for the edges out of $\mathcal{C}$ the weight does not change, and for the freshly created edges, $\mathcal{W}_{ic} = \max_{j \in \mathcal{C}}(\mathcal{W}_{ij} - \mathcal{W}_{a(j)j} + \sum_{j \in \mathcal{C}} \mathcal{W}_{a(j)j})$ and $\mathcal{W}_{cj} = \max_{i \in \mathcal{C}} \mathcal{W}_{ij}$, where $\alpha(j)$ is the antecedent of j in the cycle such that $e_{\alpha(j)j}$ is in the cycle. Then we recurse on the newly defined graph $\mathcal{G}'$. The recursion stops when the greedy phase does not create cycles any more.

Once the recursion is done, because there is no cycle in the graph, there is an edge e_{ic} that goes toward the cycle $\mathcal{C}$. This edge correspond to an incoming edge e_{ij} of maximum score in the original graph $\mathcal{G}$ such that v_j in in $\mathcal{C}$. Restoring the edges of $\mathcal{C}$ in place of v_c makes that v_j has now two parents, v_i and v_k the one in

Data: a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ and a cycle $\mathcal{C}$
Result: a weighted directed graph $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{E}_C, \mathcal{W}_C)$ and a tracker ep
begin
  $\mathcal{V}_C = \mathcal{V}/\mathcal{C}$
  $\mathcal{E}_C = \mathcal{E}/\mathcal{C}$
  for $v_j \in \mathcal{V}_C$ *such that* $e_{ij} \in \mathcal{E}$ *for some* $v_i \in \mathcal{C}$ **do**
    Add e_{cj} to $\mathcal{E}_C$
    $ep(e_{cj}) = \text{argmax}_{v_i \in \mathcal{C}} \mathcal{W}_{ij}$
    $v_i = ep(e_{cj})$
    $\mathcal{W}_{Ccj} = \mathcal{W}_{ij}$
  for $v_i \in \mathcal{V}_C$ *such that* $e_{ij} \in \mathcal{E}$ *for some* $v_j \in \mathcal{C}$ **do**
    Add e_{ic} to $\mathcal{E}_C$
    $ep(e_{ic}) = \text{argmax}_{v_j \in \mathcal{C}} \mathcal{W}_{ij} - \mathcal{W}_{a(j)j}$
    $v_j = ep(e_{ic})$
    $\mathcal{W}_{Cic} = \mathcal{W}_{ij} - \mathcal{W}_{a(j)j} + \text{score}(\mathcal{C})$
    where $a(j)$ is the predecessor of v_j in $\mathcal{C}$
    and $\text{score}(\mathcal{C}) = \sum_{e_{i,j} \in \mathcal{C}} \mathcal{W}_{ij}$
  Add v_c to $\mathcal{V}_C$
  return $\mathcal{G}_C, ep$

Algorithm 2: The *contract* routine for Chu-Liu-Edmonds algorithm.

$\mathcal{C}$. By removing the edge e_{kj} from the graph, the cycle $\mathcal{C}$ ceases to be a cycle. By breaking all the cycles one by one, we end up with a tree.

The complexity of the basic algorithm is $O(n^3)$ because each recursion performs a greedy edges selection over $O(n^2)$ edges and there need to be a recursion per cycle with at most $n-1$ such cycles in the graph. Better implementations have since reached $O(n^2)$ for complete graphs. The proof of optimality is available in the original papers [CL65, Edm67]

Eisner's Algorithm

Eisner's algorithm [Eis96] is a dynamic programming algorithm that retrieves the best projective tree over a sentence given a set of edge scores. The idea behind the algorithm is that assuming projectivity and horizontal independence of the scoring function (meaning that the score of an edge does not depend on its potential siblings), the score of the left part of a tree is independent from the score of the right part of the tree and thus can be computed separately. While projectivity, is enforced by the algorithm design, horizontal independence is a property of the tree scoring function. For example, when the scores are computed for edges independently of any other structure before hand, and then passed down to the algorithm, horizontal independence is met.

More formally, given a sentence x and a tree y over it. Let x_i be a word of the sentence and an inner node of the tree and let x_j and x_k be two dependents of x_i with $j < k$ (there could be more but the explanation is easier with only two dependents). Horizontal independence means that the score of the sub-tree yielded by x_j (the left sub-tree) does not depend on the score of the sub-tree yielded by x_k (the right sub-tree).

By nature, edge factorisation meets horizontal independence because all the edges are scored independently from each other, but it is not the only possible scor-

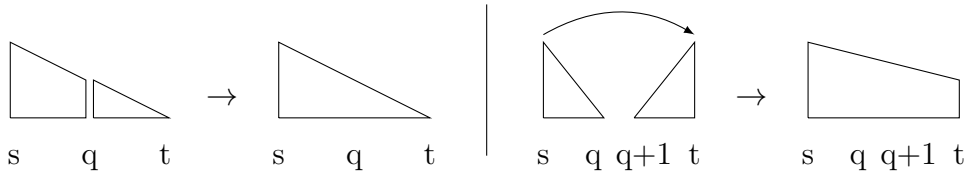


Figure 2.9: Illustration of Eisner algorithm tree merging. On the left a complete tree and an incomplete tree are merged into a bigger complete tree. On the right two facing complete trees and a new edge are merged into a new incomplete tree.

ing function that meet this requirement. A function that would consider strings of ancestors from the root, also called vertical dependency would meet this requirement as well. However, as an example, a scoring function that would take valency (the number of dependents) into account would not meet horizontal independence.

The algorithm pseudo-code is given in Algorithm 3. Given a sentence x of length n, a complete weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ has been created where $\mathcal{V} = \{x_i | i \in \mathbb{N}_{|x|}\}$ is set of all the words in the sentence, $\mathcal{E} = \{e_{ij} | (i,j) \in \mathbb{N}^2_{|x|} | i \neq j\}$ is the set of edges between different words, and $\mathcal{W}_{ij}$ is the weight (score) of edge e_{ij}. Let us define $E \in \mathbb{R}^{n \times n \times 2 \times 2}$ the table that will store the scores of already computed sub trees, and set all the scores to 0. The two first indices represent the boundaries of each sub tree in the sentence. The third index distinguishes for left trees where the root is the rightmost vertex and right trees where the root is the leftmost vertex. The last index distinguishes for complete and incomplete trees.

A tree is said incomplete if it is the result of the merger of two facing complete trees with a linking edge at the top. A tree is complete when it results from the merger of an incomplete tree and a complete tree going in the same direction. Figure 2.9 gives a depiction of those merging process. We use l (respectively r) for left (respectively right) and c (respectively i) for complete (respectively incomplete) for the two last indices.

Trees (complete or incomplete) spanning a single word ($s = t$) have a score of zero because they do not hold any edge. Then, for each span length from 1 to n, left and right incomplete trees are made up of smaller facing complete trees. Only then, complete trees are made from incomplete trees and smaller complete trees of the same direction. Eventually, the score of the best tree is the score of the right complete tree spanning the whole sentence stored in cell $E[0][n][r][c]$.

Eisner's algorithm only computes the score of the best projective tree but does not compute the tree itself. In order to retrieve the tree, one can add an extra table to store the indices q corresponding to the best merging point for each cell of E. The direction of the edges thus retrieved is directly stored in E. Then one just needs to backtrack the tree following the q table.

The complexity of Eisner's algorithm is $O(n^3)$ because of the filling of the four tables of size $O(n^2)$ requires for each cell at most n scores to be compared. The backtracking takes $O(n)$ because one just needs to check one merging q index per edge and there are exactly $n - 1$ such edges.

2.2.3 Transition-based Parsing

Transition-based parsing [NH05], also called shift-reduce parsing relies on a very different factoring strategy. Instead of assigning scores to edges and then relying

Data: a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$
Result: a projective dependency tree $\mathcal{G}_T$
begin
  $n = |\mathcal{V}|$
  Instantiate $E \in \mathbb{R}^{n \times n \times 2 \times 2}$
  Initialise $E[s][t][d][c] = 0, \quad \forall s, t, d, c$
  for $m \in [1...n]$ **do**
    for $s \in [1...n]$ **do**
      $t = s + m$
      if $t > n$ **then**
        Break

      $E[s][t][l][i] = \max_{s \leq q < t} E[s][q][r][c] + E[q+1][t][l][c] + \mathcal{W}_{ts}$
      $E[s][t][r][i] = \max_{s \leq q < t} E[s][q][r][c] + E[q+1][t][l][c] + \mathcal{W}_{st}$

      $E[s][t][l][c] = \max_{s \leq q < t} E[s][q][l][c] + E[q][t][l][i]$
      $E[s][t][r][c] = \max_{s < q \leq t} E[s][q][r][i] + E[q][t][r][c]$
  return

Algorithm 3: The Eisner algorithm for Dependency Parsing. Freely adapted from Kubler et al. [KMN09].

on a maximum spanning tree inference to retrieve the best structure, transition-based parsers use abstract state machines in order to build the tree. The score of a tree is thus factored by the scores of transitions between states.

The simplest machines are composed of a buffer that holds the remaining words and a stack that keeps trees predicted so far. At each time step, one can choose between two possibilities, either shifting the next word in the buffer to the top of the stack, or reducing the stack by adding an arc (a left-arc or a right-arc) between the two trees at the top of the stack making it one. A sequence of transitions from the state with an empty stack and a buffer containing a sentence to the state with an empty buffer and a single tree on the stack effectively constructs a dependency structure over the original sentence. The score of a state is the score of the sequence of transitions that lead to that state, thus the score of a tree is the score of the sequence of transitions that lead to it. When several sequences lead to the same state one can either pick the max or the sum of the sequences scores.

From a machine learning perspective, the goal is to predict the best transition at each time step given a representation of the current machine state. Because it might be hard to predict the cost of preferring a decision against another at some time step, the training of such systems is based on concepts from reinforcement learning. This is in part because several sequences of action can lead to the same tree and because the actual worthiness of an action is oftentimes only known at the end of the parsing process.

Because of their greedy way of selecting transitions, those basic shift-reduce parsers are fast, they run in linear time, but are also inexact (meaning that the final tree given by the algorithm might not be the tree with the highest score given that abstract machine) by nature and they can only build projective trees and need heuristics to render non-projectivity. This is the traditional comparison line with graph-based parser. Graph-based parsers are exact and can deal with non-

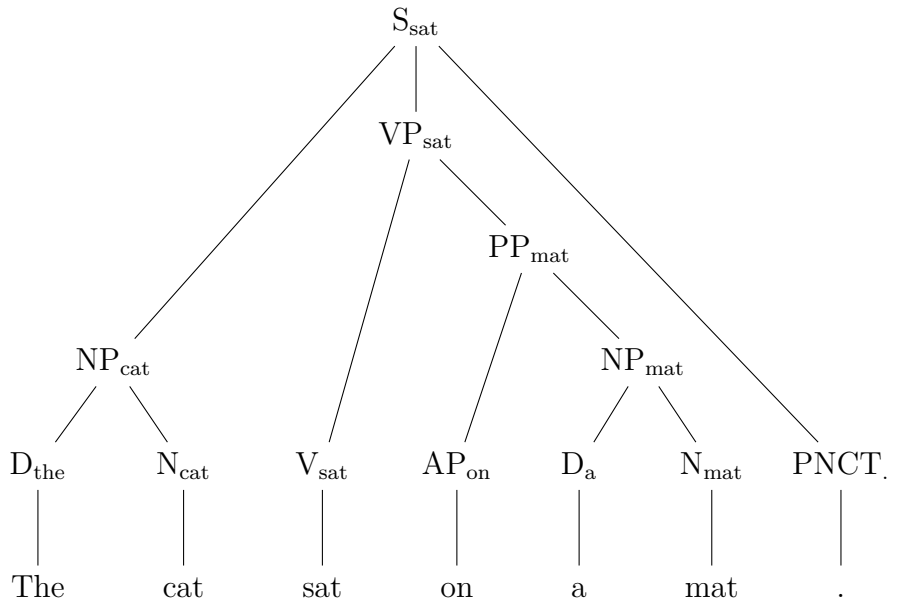


Figure 2.10: Constituency structure for the sentence *"The cat sat on a mat."*

projectivity but are slower and rely on very local features as parsing information[2]. Shift-reduce parsers are fast and can use all the parsing history as information source for feature representation but they are inexact and cannot handle (natively) non-projectivity.

Modern shift-reduce parsers propose a wider range of transitions like the swapping of words, thus handling non-projectivity but making the search intractable in some cases. They also use beam search techniques to soften their greediness. Furthermore, it is especially hard to know what are the best transitions to predict once parsing errors have been produced and the training stack does not look like the gold stack anymore. However, it is very important for parsers to be able to handle parsing errors and to keep producing sensical outputs even after a mistake. Therefore, there is a whole area of research dedicated to designing training oracles for transition systems, whose aim is to design better training procedures to learn to handle parsing errors.

2.2.4 Other Approaches

Dependency trees are just one amongst several formalisms used to represent syntactic information. Another important formalism is constituency grammar, in which the syntactic structure of a sentence is represented in the shape of a derivation tree. In a constituency tree, the leaves are the actual words of the sentence and the inner nodes are non-terminal symbols from a grammar. Figure 2.10 gives an example of constituency tree for our example sentence.

There are links between constituency grammars and dependency grammars. A constituency grammar can be fully lexicalised or can be equipped with a set of head selection rules. If in a constituency grammar, each derivation rule is associated with a head selection rule, then a derivation tree can also embed dependency relations. For example in figure 2.10, rule VP $\rightarrow$ VB PP, would be associated with the head selection rule $h(\text{VP}) = h(\text{VB})$. Different heuristics have been used to produce

[2]Researchers have proposed extensions to edge factored graph-based parsing that allow one to use more complex features at the cost of relaxing inference.

dependency structures from constituency structures [EB07]. However constituency parsing has a greater time complexity than dependency parsing, therefore transition and graph-based methods are favoured. Still it is worth noting that the Penn treebank [MMS93], one of the first annotated corpus widely used in dependency parsing was originally annotated with constituency trees and only later turned into a dependency parsing resource using such heuristics.

In fact, Eisner's algorithm in also an off-shoot of those parsing techniques, as it can be seen as a parsing algorithm for bi-lexical dependency grammars [KMN09].

Other grammar based parsing techniques not directly linked to constituency grammar have also been proposed with different parsing algorithms. For example, methods have been proposed that see parsing as a constraint satisfaction problem where the grammar defines legal structures and the parser tries to break as few constraints as possible [Fot06].

Another proposed approach is to see dependency parsing as a sequence tagging task. Indeed, a dependency tree can be turned into a sequence of label over the original sentence. Each word can be labeled with its governors index (or relative offset), its dependency relation type, the POS of its governor or any other relevant information. This method has been applied by Spoustová and Spousta [SS10] on Czech and English as a proof of concept.

With the increasing application of the sequence-to-sequence neural learning paradigm to more and more problems, first papers applying this method to dependency parsing appeared in August 2018 [LCHZ18].

Finally, several models have been proposed to perform unsupervised dependency parsing [LZ15]. In unsupervised dependency parsing the goal is to perform dependency parsing without any annotated data. While it is interesting as it directly applies to any text in any language, results are still well below those of supervised systems. We just mention this but we do not go any further because this is out of the scope of this thesis.

2.2.5 Evaluation

So far, we have seen different algorithms to perform dependency parsing, but we have not yet said how those methods are evaluated. Indeed it is an important aspect of the development of a parser or a new parsing algorithm to be able to assess its quality, to see how well it works and how to compare to other approaches. We will see here some evaluation metrics used to evaluate dependency parsers given some annotated corpora.

Maybe the simplest measure of a parser's quality is the percentage of trees it predicts correctly. However, this score is usually very low (hardly ever above 50%), highly biased and does not tell much about the quality of the parser. The reason for this is quite simple. Trees are complex structures composed of several edges, and because of the combinatorial nature of the structure the prediction occurs at the edge level. By considering perfectly reconstructed trees only, we completely disregard the fact that prediction is done at the edge level and we treat equally a tree that would have missed one edge and a tree that would be completely off. Furthermore, the longer a sentence, the more likely it is that an edge will be mispredicted, thus the score is biased toward parsers that focus on short sentences.

To avoid those problems, edge prediction scores are preferred. The unlabeled accuracy score (UAS) is the percentage of well predicted edges (good governor and dependent). It is higher than the percentage of perfect trees and it distinguishes

trees that are slightly wrong from trees that are completely off. It also reflects better the learning process focused on edges. It is still slightly biased toward shorter sentences since dependency parsing is easier for shorter sentences.

The labeled accuracy score (LAS) is the labeled counter part of the UAS. An edge is considered well predicted if it has the correct governor and dependent as well as the correct relation label. We give greater details about those labels in the following chapter when discussing the kind of data used in this thesis, but in general those are labels of the kind depicted in Figure 2.6. As such, LAS is lower than UAS because it is stricter.

More recently, the content-word labeled accuracy score (CLAS) has been proposed. The idea is to consider only content words when computing the LAS. Typically, determiners, adpositions and conjunction would be ignore. It was already common practice to ignore punctuation in the UAS. With the advances of the Universal Dependencies project and availability of more and more corpora in many languages, it has become important to be able to compare parsing performances across languages, however LAS and UAS are not suited for that. For example, French and English use lots of articles and adpositions, which are rather easy[3] to assign a governor. Finnish does not use article, and uses very few adpositions thanks to its extensive case system. This renders comparison of French and Finnish unfair since an French sentence contains more "small" words than its Finnish equivalent. Focusing on content words only (nouns, verbs, adjectives and adverbs) makes the cross-lingual comparison easier.

We have seen ways to automatically produce dependency trees for sentences given a scoring function and how to evaluate their quality given some annotated standard. We have said earlier that those scoring function can take different forms. In this work, we are interested in functions learned from annotated data. Next section therefore presents the problem of learning such a function from data and some algorithms available to solve it.

2.3 Machine Learning and Structured Prediction

In terms of machine learning, dependency parsing falls under the category of structured prediction problems. This means that the different predictions done by the models are interdependent. For example, in dependency parsing, there is the constraint that the final output be a tree. Therefore, if we have already predicted a dependency relation between a governor x_i and a dependent x_j, we cannot predict that x_i depends on x_j or any dependant of x_j because that would create a cycle. Likewise, as x_j already has a governor, we cannot predict another as it would also break the tree constraint that each vertex has at most one governor. Figure 2.11 gives an example of a partially parsed sentence. In this example, the word *sat* is already governing *cat* which governs *the* in turns, thus *sat* cannot be governed by *cat* nor *the* as it would break the acyclicity constraint required for the output to be a tree.

In principle, predicted variables could be either discrete or continuous, thus making structured prediction problems sharing with both classification and regression. Indeed, for dependency parsing output variables are discrete (head indices, presence or absence of an edge...) and thus one could train models to perform

[3] A French article always attaches to the first noun to its right. When there is no such noun, then it attaches to the substantivised adjective or participle.

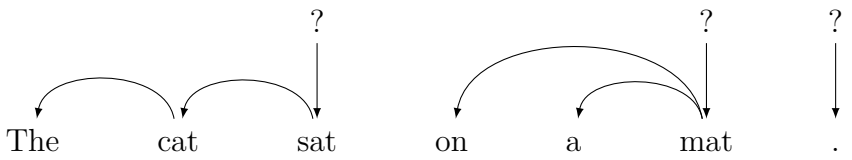


Figure 2.11: A partial dependency structure for the sentence *"The cat sat on a mat."* The tokens *sat*, *mat* and . do not have a governor yet. However, *sat* cannot depend on *The* nor *cat* and *mat* cannot depend on *on* nor *a*.

discrete predictions. However, we have seen in the previous section that tree inferences such as Eisner's algorithm were taking in weighted graphs, in which case models are asked to provide continuous weights/scores to edges rather than discrete classes.

The interdependence between output variables calls for specialised learning algorithms. In the next subsection, we describe the problem of supervised structured prediction for dependency parsing in more details. Then we introduce the notions of large margin classifier and online learning and some important algorithms used to learn dependency parsing models. Eventually, we discuss about the recent trend of neural parsers.

2.3.1 Structured Prediction

As we saw in section 2.2, given a sentence x and a tree scoring function, we want to find the highest scoring tree $\hat{y}$ over x. The problem is that the space of possible trees $\mathcal{Y}_x$ grows exponentially with the length of x and thus becomes quickly too big to be effectively traversed. This is the curse of structured prediction.

In order to make the search in the solution space tractable, the actual structures (trees) are broken down into sub-parts (factors) in way such that the number of potential sub-parts does not grow exponentially in the size of the input sentence anymore but only polynomially. Then the scoring function has to score factors instead of complete trees. The structured prediction arises from the fact that some of those potential edges are incompatible with each other.

The graph-based dependency parsing based on edge scoring presented above is a factorisation. It breaks the structure into parts (its edges) and the score of the structure is simply the sum of its edges scores. The Eisner's and Chu-Liu-Edmonds algorithms from section 2.2 are examples of inferences used in graph-based dependency parsing. Those inferences guarantee the optimality of the retrieved structure. However, running those inferences is time consuming (Eisner algorithm has a cubic complexity [Eis96]) which might be a problem when training on large datasets or with limited computing power. A way to lighten the computation is to relax part of the output structure constraints.

Let n be the length of a sentence we want to parse. In an edge factored graph-based setting, we need to score every $O(n^2)$ possible edge. Applying Eisner algorithm to those scores to retrieve the best tree adds an extra $O(n^3)$ to the computation. From the two basic properties of a tree, acyclicity and connexity, follows that each word but one (the root) has a unique parent word. However, acyclicity and connexity do not follow from the unique parent property. By relaxing those constraints and just enforcing that each word has a unique parent, the problem becomes strictly local and faster. For each word, we just need to scan through the other $n-1$ words to find its parent, only adding an extra $O(n^2)$ to the com-

putation, at the cost of loosing the acyclicilty and connexity guarantee. We can even go further by seeing a tree as a back of edges. Each possible edge is either in or out of the tree. Then deciding for each edge if it belongs in the tree does not cost more than scoring it in the first place thus not adding extra complexity to the computation, but at the cost of possibly having words with multiple parents and so on.

It is worth noting that there is an hierarchy between those three problems. If a model is able to solve the unstructured edge binary classification problem, it will be able to solve the more constrained parent selection problem. The opposite is not true though. Likewise, a model able to solve the parent selection problem will be able to solve the more constrained tree retrieval problem. Again, the opposite is not true. Because of this, modern neural parsers have been trained to solve the head selection problem, with the tree inference only being used to ensure an actual tree structure at test time [DQM17].

To receive scores, factors need to be mathematically represented. This is done with feature vectors. A feature vector is a longer or shorter vector in which each dimension encodes a specific information about the object the vector represents. Historically, those were very long sparse vectors representing the object as the results of many handcrafted feature functions of the type given in Table 2.1. But nowadays, with the use of word embeddings and neural networks, feature vectors can be much smaller and denser.

If some feature function considers the entire structure, for example by considering the maximum depth of the tree, then one would have to consider each tree one by one to know the value of that feature and the total score of the tree. But because it is impossible to look at all the possible trees in a reasonable amount of time for a given (reasonably sized) sentence, we have to resort to factorisation. There is therefore a direct link between the type of factorisation used and the type of information available to score the factors. However, as factors become more complex and have access to more information, their number increases and the inference also gets more involved and might even become intractable.

Given a sentence, we have inferences that can retrieve the best possible structure given a scoring of its potential sub-parts. The mathematical representation of those sub-parts can only encode information available inside the sub-parts itself and thus depends on the factorisation used. We now look at how the scoring function is learned from annotated data given some sub-parts representation.

2.3.2 Learning Scoring Functions

Instead of scoring complete trees, we will score sub-parts independently and use some inference mechanism to retrieve the best tree from its sub-parts scores. The sub-parts scoring function can be any computable function mapping feature vectors onto real numbers. The shape of the actual scoring function is also called the parameterisation of the model. For example, if we assume a linear scoring function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ such that the score of a part e is the dot product between its feature vector $\boldsymbol{\phi}(e)$ and the weight vector $\boldsymbol{\theta}$:

$$Score(e, x) = f_\theta(\boldsymbol{\phi}(e)) = \boldsymbol{\phi}(e) \cdot \boldsymbol{\theta},$$

then any $\boldsymbol{\theta} \in \mathbb{R}^d$ defines a different linear function of that family and $\boldsymbol{\theta}$ is called the parameters of the function.

The goal is to learn the parameter vector $\boldsymbol{\theta}$ that best fits the data. Similarly, if instead of a linear function, f_θ was parameterised by a neural network for example, then the parameters $\boldsymbol{\theta}$ would be the weights and bias associated with each neuron and the goal would also be to find the parameters $\boldsymbol{\theta}$ that would best fit the data.

By *best fits the data*, we mean that the model should do as few errors in term labeled or unlabeled accuracy score (LAS/UAS as defined in section 2.2) as possible on the training data. However, a model has to score factors (edges) and the model accuracy on a data set is a very distant and weak feedback for the model to learn from. It would be better to have actual feedback at the factor level. But since we only care for the actual structures and not the intermediate scores, training samples are not annotated with edge scores. Furthermore, many different scores and thus scoring functions can lead to the same final structures. Thus we need to learn the scoring function only relying on structure feedback and not on direct score feedback. Hopefully, because this is a common problem in machine learning and specially in NLP, there are a lot of algorithms to solve that kind of problem.

In fact, many of the most popular learning algorithms have received structured extensions such as the structured-SVM [TJHA05] rendering structured prediction possible in the support vector machine framework or the structured Perceptron [Col02].

There also exist algorithms targeting specific types structured problems. This is the case of the well known Hidden Markov Models (HMM) and its extension to general graphs, the Conditional Random Fields (CRF) [Smi11]. Those algorithms are designed for graph (especially sequence) labeling, where the graph structure induces constraints on nodes labels.

2.3.3 Large Margin Classifiers

Large margin classifiers are a class of classification algorithms that have been adapted to structured prediction in which the SVM belongs. For training large ma ring classifiers, we do not have and do not need factors scores, but we require that the actual true structure have a score higher by a given margin than the score of any other possible structure. The margin violation is also called the loss. For a sentence x whose true tree is y and for any structure y' that is not y, let $m > 0$ be the margin, we define the loss as:

$$loss(x, y, y') = max(0, m - (Score(x, y) - Score(x, y'))).$$

If y scores higher than y' by more than the margin m then the loss if 0. But if y does not scores higher than y' by more than m, this also includes the case where y' actually scores higher then y, the loss becomes equal to the amount by which the margin is violated. It is also called the hinge loss because of its shape. If the margin constraint is respected then the true structure scores higher than any other possible structure and the parsing accuracy will be high.

Large margin classifiers, do not learn to minimise the parsing error, they instead learn to minimise the total loss over the training data. Because the hinge loss is used in place of the actual objective function we wish to optimise, it is called a surrogate function. Other surrogate losses have been proposed and different problems use different surrogate losses to train their models.

We have just defined surrogate losses that allow to train our parsing models. However, the problem is once more that we can not go through all the possible structures to make sure the margin criterion is respected. Thus, approximations

have been proposed. A basic solution, is to consider the structure that is violating the margin constraint the most. If the best scoring structure is not the true one, then the inference automatically finds it and it is the one that violates the margin constraint the most. Under certain factorisations, it is also possible to find the k best scoring structures, allowing to ensure the margin constraint even when the true structure already scores higher than any other. There actually is an extension of the Chu-Liu-Edmonds algorithm that find the k best scoring trees [Hal07].

While we can learn parsing model this way, there is a problem with the hinge loss in our context of structured prediction, namely the loss is computed the same way for any structure while not all the structures are as bad compared to the true one. This becomes a real problem when the model is not able to score the desired structure higher than some others, because the feature vector is not expressive enough for example. In that case, the model and the inference in turn will favour an erroneous structure which will decrease the parsing accuracy. In such a case, we might prefer the model to favour structures that are only slightly wrong to structures that are completely off. Hence the inclusion of a term depending on the actual number of mistakes in a structure in structured losses, so that the margin constraint becomes stronger as the number of mistakes increases. This favours choosing structures with as few mistakes as possible when the model is unable to pick the actual true structure.

Structured-SVM has been applied to dependency parsing [YM03]. However, it is computation intensive as it requires to perform inference for each training sample in order to do one update of the model. This can be highly impractical for big treebanks containing tens of thousands of sentences. Thus people proposed to use similar algorithms but designed for handling one sample at a time instead of the whole data set.

2.3.4 Online Learning

Even though we have access to a complete training set of examples, because of the tree inference that needs to be performed for each example at each model update, batch learning is prohibitive. Instead people prefer online learning where at each step one predicts the structure for an example and updates its model according to the mistakes made. We now review two online learning algorithms that have been used for structured prediction, namely the structured Perceptron [Ros58, Col02] and the Passive Aggressive algorithm.

Let $\mathcal{D}$ be a training set of sentence-tree pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the set of sentences and $\mathcal{Y}$ is the set of trees. We note $\mathcal{Y}_x \subset \mathcal{Y}$ the set of possible trees over a sentence x. Let $\mathcal{E}_x$ be the set of all possible edges over sentence x.

Let $\boldsymbol{\phi} : \mathcal{E}_x \to \mathbb{R}^d$ be a function that maps an edge from a sentence to a feature vector. We define the edge scoring function as the linear function $score_\theta : \mathcal{E}_x \to \mathcal{R}$ parameterised by a model $\boldsymbol{\theta} \in \mathbb{R}^d$ such that:

$$score_\theta(x, e_{ij}) = \boldsymbol{\phi}(e_{ij}) \cdot \boldsymbol{\theta}.$$

Because we work with first order factorisation, the score of a tree is the sum of the scores of its edges, so we can extend the scoring function to work on trees as well.

$$score_\theta(x, y) = \sum_{e_{ij} \in y} \boldsymbol{\phi}(e_{ij}) \cdot \boldsymbol{\theta}.$$

Because of the linearity of the dot product, we get that the feature vector of a tree is the sum of the feature vectors of its edges. We note it $\mathbf{\Phi}$:

$$\mathbf{\Phi}(y) = \sum_{e_{ij} \in y} \boldsymbol{\phi}(e_{ij}).$$

Then the Perceptron algorithm works as follow. At each learning step, we receive a new sentence x. Then we compute a score for each possible edge e over x with current model $\boldsymbol{\theta}_t$. Those scores are passed to the inference which returns a tree $\hat{y}$. Then we receive the actual tree y associated with x and compare it to $\hat{y}$. If they are different, we suffer an instantaneous loss $loss(x, y, \hat{y})$ and update the model under the Perceptron rule.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y}).$$

The pseudo code for the structured Perceptron is given in Algorithm 4.

Data: a set of sentence-tree pairs $\mathcal{D} = (x, y)$
Result: a model vector $\boldsymbol{\theta}$
begin
  $n = |\mathcal{D}|$
  Instantiate $\boldsymbol{\theta}_0 = \mathbf{0}$
  for $i \in [0...n]$ **do**
    Get $x, y = \mathcal{D}_i$
    $\hat{y} =$ predict-tree$(x, \boldsymbol{\theta}_i)$
    if $y \neq \hat{y}$ **then**
      $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y})$
    else
      $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$
  return $\boldsymbol{\theta}_{n+1}$

Algorithm 4: The Perceptron algorithm for structured prediction. The function predict-tree is a function that returns a tree given a sentence and a model.

The Perceptron is guaranteed to converge to a separator in the case of linear separability. Furthermore, it enjoys good cumulative loss bounds both in the case of linear separability and in the case of non separability [MR13].

However, despite its qualities, the Perceptron algorithms suffers a major problem. The Perceptron update rule is not corrective. This means, that if one encounters the same instance twice in a row, and the first time the algorithm made an erroneous prediction, then we have no guarantee that after the model has been updated the next prediction will be correct. This is because the error margin (the measure of how off a prediction is) is not taken into account in the update. Thus if the prediction was really away from the truth, a single update might not be sufficient to fix the model.

More formally, if the squared norm of the update is smaller than the error margin, then the update is not corrective:

$$\|\mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y}))\|^2 < \boldsymbol{\theta}_t \cdot (\mathbf{\Phi}(\hat{y}) - \mathbf{\Phi}(y)),$$

$$(\mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y})) \cdot (\mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y})) < \boldsymbol{\theta}_t \cdot (\mathbf{\Phi}(\hat{y}) - \mathbf{\Phi}(y)),$$

$$(\boldsymbol{\theta}_t + \mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y})) \cdot \mathbf{\Phi}(y) < (\boldsymbol{\theta}_t + \mathbf{\Phi}(y) - \mathbf{\Phi}(\hat{y})) \cdot \mathbf{\Phi}(\hat{y}).$$

To solve this problem Crammer et al. [CDK+06] came up with an alternative update rule that takes the error margin into account. It is basically a Perceptron with an adaptive scaling factor that ensures correctiveness. Their algorithm is called the Passive-Aggressive (or PA for short), for like the Perceptron it does not update if the model makes a correct prediction, but contrary to the Perceptron, upon error the aggressive update rule guarantees to be corrective. The aggressive update takes the form:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \tau(\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y})).$$

Where τ is the scaling factor and is equal to:

$$\tau = \frac{loss(x, y, \hat{y})}{\|\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y}))\|^2},$$

$$loss(x, y, \hat{y}) = \boldsymbol{\theta}_t \cdot (\boldsymbol{\Phi}(\hat{y}) - \boldsymbol{\Phi}(y)) + \sqrt{\Delta(y, \hat{y})},$$

where $\Delta(y, \hat{y})$ is a measure of how acceptable is to predict $\hat{y}$ instead of y. Pseudo code for the structured Passive-Aggressive is given in Algorithm 5.

Data: a set of sentence-tree pairs $\mathcal{D} = (x, y)$
Result: a model vector $\boldsymbol{\theta}$
begin
  $n = |\mathcal{D}|$
  Instantiate $\theta_0 = \mathbf{0}$
  for $i \in [0...n]$ **do**
    Get $x, y = \mathcal{D}_i$
    $\hat{y} =$ predict-tree$(x, \boldsymbol{\theta}_i)$
    if $y \neq \hat{y}$ **then**
      $\tau = \frac{[\boldsymbol{\Phi}(\hat{y}) - \boldsymbol{\Phi}(y)] \cdot \boldsymbol{\theta}_i + \Delta(y, \hat{y})}{\|\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y})\|^2}$
      $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \tau(\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y}))$
    else
      $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$
  return $\boldsymbol{\theta}_{n+1}$

Algorithm 5: The Passive-Aggressive algorithm for structured prediction.

Because the Passive-Aggressive algorithm take the error loss into account, it is more sensitive to outliers than the Perceptron. To avoid extreme corrective updates when outliers are met that would be detrimental for the model, Crammer et al. proposed two relaxed extensions to their Passive-Aggressive algorithms. The PA-I algorithm sets an upper bound C to the value of τ :

$$\tau = \min(C, \frac{loss(x, y, \hat{y})}{\|\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y}))\|^2}).$$

The PA-II algorithm dampens the value of τ by setting a lower bound to the denominator:

$$\tau = \frac{loss(x, y, \hat{y})}{\|\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y}))\|^2 + \frac{1}{2C}}.$$

In both cases, C plays a similar role in controlling the strength of the update being thus called the aggressiveness parameter.

In fact, the PA is corrective in the binary classification case. In the structured case that interests us, the correctiveness is weaker. If structure $\hat{y}$ is predicted instead of y for input x at time t, we are guaranteed that at time $t+1$, if we see x again, y will score higher than $\hat{y}$. But because there are exponentially many possible structure for x, and the update rule only considers y and $\hat{y}$, at time $t+1$ the algorithm could predict yet another structure $\hat{y}'$ as well.

Crammer et al. [MCP05a] proposed another algorithm that can take several structures into account when updating the model called the Margin Infused Relaxed Algorithm. Still, one could never take every possible structure into account this way because there are exponentially many of them, which is the basis of structured prediction. Furthermore, Perceptron and Passive-Aggressive algorithms work well in practice, so we do not need to venture away from them.

We shall also mention model voting and averaging. Perceptron and Passive Aggressive are online learning algorithms, meaning they consider one sample at a time and update their model when necessary. They have really been designed for online tasks where one only has access to one sample at each training step and we have no control over the order of presentation of the samples. However, we use them for convenience here but we have access to the whole training data. Because past information tends to fade away as more updates are performed and because we could in principle play with the sample presentation order, there is no reason to believe in the optimality of the last model. Thus people have proposed voting schemes to use not only the last model but all the intermediary ones at prediction time. The idea being to let each model vote on the final prediction with a weight equal to the number of training round it did not update [FS99]. But because it is not always practical to keep thousands of intermediary models, people have proposed to do averaging [FS99]. The idea here is similar, but instead of having models voting on the final prediction, they each participate in the final scoring function, which ends up being a weighted average of all the intermediary models but can be computed efficiently during the training process.

2.3.5 Neural Parsers

As we said earlier, the inference algorithms need to have access to substructures (edges) scores, but they are agnostic to the form the scoring function takes. Thus, with the advances in neural network architectures, recent years have seen the emergence of more and more parsers using neural networks components.

Neural networks are a machine learning formalism that originated from an attempt at mathematically replicating the way biological neurons work in the brain. Early instantiation of this idea lead to Rosenblatt's Perceptron algorithm [Ros58]. By arranging several inter-connected layers of simple learning units (so-called neurons), one can in principle learn any mathematical function. This allows neural networks to learn highly non-linear classification boundaries for example. However, discrete multi-layer Perceptrons are notoriously hard to train. So, researchers proposed to use continuous functions in neurons instead in order for the resulting function to be differentiable and thus to allow the network to be trained via back-propagation of the loss gradient [RHW86].

Regarding, graph-based dependency parsing, neural networks have been used in three somewhat overlapping ways. In a first line of works, shallow neural networks have been used to learn representations for words [KCC08, BGL14], edges [Ban15] or even dependency features [CZZ14] in order to complement traditional one-hot

feature vectors in parsers using linear models. We will look at some of those proposals in greater details in Chapter 5.

In a second line of works, feed-forward neural networks (typically multi-layer Perceptrons) have been used to directly learn the edge scoring function. Pei et al. [PGC15] propose a simple three layers architecture that receives an edge features and outputs the score of the various dependency labels for that edge. The first layer is a look-up table that retrieves the embedding of each incoming features. Those embeddings are concatenated in the second layer and then fed to a non-linear function in the third layer to output the various scores. They can then train the feature embeddings and the non-linear function weights via back-propagation.

More recently, recurrent neural networks such as bidirectional long-short term memory networks (Bi-LSTM) [HS97] have been used to encode incoming sentences. Recurrent neural networks (RNN) have an internal state that evolves as they are fed inputs. Contrary to multi-layer Perceptrons which, once trained, will always output the same values for a given input, the output of a RNN does not only depends on its current input but also on its internal state which evolves with each input and behaves as a form of memory. This way, RNN takes into account past history, which is valuable for encoding words in sentences. Using Bi-LSTM to encode words as potential governor and dependant and applying a bilinear function on the top of the induced representations, Dozat et al. [DQM17] won CONLL2017 shared-task on dependency parsing [ZPS+17] with their parser trained via back-propagation.

2.4 Conclusion

In this preliminary chapter, we have presented some key concepts from which this work stems. We have presented dependency parsing as a natural language processing task that retrieves syntactic structures of sentences in the shape of trees whose vertices are words from that sentence.We have seen how trees can be inferred from a set of substructures whose relevance is scored based on their vectorial representation and a scoring function (Section 2.2). We have also seen how the scoring function can be learned from annotated data via machine learning techniques such as online structured prediction learning (Section 2.3).

Most modern dependency parsers, and papers about dependency parsing, follow the same general structure represented in Algorithm 6. The system is given sentences, one by one. A dependency tree is inferred for the input sentence. In the case of graph-based parsing, all substructures (edges) receive a mathematical representation and are scored by the running scoring function, and only then the inference mechanism is applied to those scored substructures. But the inference and the scoring can also be intertwined as is the case in transition-based parsing where the score of each parsing action depends on the actions that have happened before it. Then the predicted tree is compared to the actual expected tree and the scoring function is updated to take the parsing errors into account. Scoring function update can happen after each sentence (online learning), every few sentences (mini-batch) or when the complete data have been parsed (batch learning). Eventually, when the learning process is over, when a stopping criterion has been met, the model is ready to be used for parsing unseen data, for a downstream task for example. In practice, the learned model is also used to parse a set of held-out data in order to be evaluated.

After this general presentation of the problem of dependency parsing and of

Data: a set of examples $\mathcal{D}$, a feature function $\boldsymbol{\phi}$, a loss function *loss*
Result: a weight vector $\boldsymbol{\theta}$
begin
  $d = dim(\boldsymbol{\phi})$
  Instantiate $\boldsymbol{\theta}_0 \in \mathbb{R}^d$

  repeat
    for $(x, y) \in \mathcal{D}$ **do**
      $l = len(x)$
      Instantiate $\mathcal{W} \in \mathbb{R}^{(l+1)\times l}$
      Score each possible edge,
      foreach *edge* $e_{ij},\ i \in [0..l],\ j \in [1..l]$ *over* x **do**
        $\mathcal{W}_{ij} = score_\theta(\boldsymbol{\phi}(e_{ij}))$

      Predict a structure from scored edges,
      $\hat{y} = \text{predict}(\mathcal{W})$

      Compare it with true tree y for sentence x
      if $loss(x, y, \hat{y}) \neq 0$ **then**
        Update model $\boldsymbol{\theta}_t$,
        $\boldsymbol{\theta}_{t+1} = \text{update}(\boldsymbol{\theta}_t, x, y, \hat{y})$
  until *Stopping criterion is met*

  return $\boldsymbol{\theta}_t$

Algorithm 6: The training process of a generic dependency parser. The main difference between online and (mini-)batch learning process if the frequency of model update. In the online case (as depicted here) model is updated for every sentence (inside the for loop). In the (mini-)batch case, the update would happen when several sentences have been parsed (outside the for loop).

the tools used to trained graph-based dependency parsers, we shall look at more specific problems that appear when performing syntactic analysis in a multi-lingual context.

Chapter 3

Representing Word Information

The previous chapter has presented dependency parsing as a tree prediction problem. We have presented the parsing mechanism in the shape of inferences over sets of scored edges for example. We have also seen how edge scoring functions can be learned with machine learning techniques. We mentioned that edges are mathematically represented by feature vectors in order to be scored, and we gave an example of such a feature vector showing that edges are represented by combining information from their head and dependant words and their contexts. We however remained quite elusive as to the representation of those words.

Word representation has had a long history. From a linguistic perspective, it is useful to assign classes and attributes to words in order to study linguistic mechanisms and extract patterns. Those classes such as parts-of-speech have also been widely used in order to mathematically represent linguistic information for computation purpose. However, despite their linguistic usefulness, they happen to lack flexibility and at time are deficient in terms of lexical semantic (the actual meaning of words) which is very important for higher level NLP tasks. Therefore, capitalising on the wide availability of textual data online, the increasing computational power and advances in machine learning techniques, many methods have been proposed in order to give representation to words based on their statistical properties as displayed by text. Those methods provide representations to words, also called word embeddings, that encode finer information than their part-of-speech and that have proven to be really useful for NLP tasks.

In this chapter, we will first look at linguistic categories used to analyse words such as parts-of-speech, morphological attributes and the like. Then, we will turn to the more recent advances in word representation on which we will build up in following chapters.

3.1 Lemmas, Parts-of-speech and Morphological Attributes

To abstract away from words and to reveal general grammatical rules, linguists have come up with devices to represent information about words. Parts-of-speech (POS for short) are word classes not based on words meaning, but rather on their syntactic behaviour. Usually parts-of-speech are statically assigned to words and appear in the dictionary alongside their lemmas (another grouping device).

Lemmas are the basic word forms used to speak of them in isolation, they are sometime called citation forms, and are typically used in dictionaries to sort words.

The choice of the form that will act as lemma is language specific. For example, English verbs lemma are their base form (the lemma of *began* is *begin*), likewise French verbs lemma are their infinitive (the lemma of *commencera* is *commencer*). Latin verbs lemma however, are their indicative present active first person singular form (the lemma of *vidistis* is *video* and not *videre*).

Morphological features (or morphological attributes) are yet another device. Morphological features are a way to represent (mostly) dynamic or contextual information about words, especially in languages where word forms change according to their context and usage.

For example, the English word *sleep* can be used in a few different ways. It can be used to express the action someone is doing like in the sentence *They sleep on a bench*, or to give an order like in *Sleep!* In those two examples, the word *sleep* is a verb, in non-third person indicative present for the first example and in imperative present for the second. It could also refer to the state of reduced consciousness animals experience when resting like in *I need some sleep*. In this case, the word *sleep* is a noun in singular. Those categories, verb and noun are parts-of-speech, they are statically assigned to the word *sleep*. In the dictionary, *sleep* has two entries, one for the verb and one for the noun. On the other hand, the fact that the verb *sleep* is in non-third person indicative present rather than imperative present is only given by the sentence in which is appears. Those categories are morphological features dynamically assigned to the verb *sleep*. Eventually, *sleep* is also a lemma. It is the lemma of the verb *to sleep* whose forms are *sleep, sleeps, sleeping, slept* and also of the noun *sleep* whose forms are *sleep* and *sleeps*.

In the following sections we make a brief description of parts-of-speech and morphological attributes and we discuss some issues that arise when using them for natural language processing especially multi-lingual setting. As lemmas are language specific and given in dictionaries, we do not discuss them any further.

3.1.1 Parts-of-speech

Parts-of-speech are used to categorise words according to their syntactic use. Depending on the grammatical tradition of a language and on the exact characteristics of the word class system, the exact number of parts-of-speech may vary, but some are linguistically universal.

The main characteristic of parts-of-speech is productivity. A closed class is a class that contains a fairly small and stable inventory of words. A typical example of this are French *conjonctions de coordination* (coordination conjunctions) which are a small class of seven words: *mais, ou, et, donc, or, ni, car* which are used to coordinate clauses or words together. On the other side, an open class can receive new entries all the time and has virtually infinitely many words. Prototypical open classes are Nouns and Verbs. In general, there are a few open classes containing a lot of content words like Nouns (and Proper Nouns), Verbs, Adjectives and Adverbs (some people consider Interjection an open class too), and more closed classes containing words like Pronouns, Adpositions, Conjunctions or Determiners.

Some part-of-speech are also strictly bound to written text rather than speech. The most famous one is Punctuation markers such as commas, colons and periods. Because punctuation is a writing device and that not all languages use it, it is also an example of part-of-speech that does not appear in all languages. For example, Gothic amongst many other classical languages did not use punctuation.

Given a language, most words are well assigned to one or more part-of-speech.

English *behaviour* is a Noun and French *écrire* is a Verb. In English, a lot of Nouns have a Verb that share the same base form (lemma) (e.g. *a start*, *to start*, *a cut*, *to cut*, and so on), but they are still different words each with its own POS. Similarly, in French a lot of Adjectives can also be used as Nouns. This is also true of closed classes, for which English *that* is a perfect example. It can be a determiner as in *"That dog is big,"* or a pronoun as in *"Have you seen that?"* or also a subordinating conjunction as in *"I know that he will come"*. A few words though might be just on the border between two classes, typically a closed and an open one with no clear assignment. Manning [Man11] and Maling [Mal83] gives the example of transitive Adjectives in English such as *like*, *near* and *worth*. Those words that are historically adjectives and behave both like adjectives and prepositions are often seen as prepositions nowadays since they do not inflect anymore.

From a cross-lingual perspective, we should say that the part-of-speech given to a word in its language's own grammatical tradition does not always meet with its actual syntactic use. A good example of this are French *adjectifs possessifs* and *adjectifs demonstratifs* which, despite being often called adjectives in French courses are in fact plain determiners and would better be analysed as such in a cross-lingual setting.

Another point to notice is that concepts that are tightly bound to some part-of-speech in a language may be rendered by a completely different part-of-speech or mechanism in another language. An example for this is the use of prepositions in English to express directions (*to, from, by*) where Finnish relies on extensive case marking. Another example is expressing position where English also uses prepositions (*over, under, on, behind...*) where Japanese uses nominal expressions (literally *at the top of, at the back of...*).

Finally, we should mention that despite their tendency for linguistic universality, because not all languages represent similar information in the same way, not all parts-of-speech need to exist in all the languages. For example, in Indo-European languages, verbs are well distinguished from adjectives based on the way they inflect and on their need of a copula to be predicative in languages that have it, like in English *He is tall*. In languages where words do not inflect and where copula is not used with adjectives, like in Mandarin, the distinction between adjectives and verbs can be very shallow[1]. On the other side, Japanese has two main classes of Adjectives, one that do not inflect like nouns but use a specific compounding particle (the so called *na* adjectives), and one that do inflect like verbs but that use a different set of suffixes than verbs (the so called *i* adjectives).

take home message

3.1.2 Morphological Features

We call morphemes, the smallest meaningful items of a language. Words can be composed of one or more such semantic items. Each morpheme carries some semantic information and contributes to the meaning of the word it composes. Some morphemes can appear as independent words while others have to be attached to existing words as shown in Figure 3.1. Morphology has to do with the analysis of words into morphemes, and how morphemes compose to create new words.

Generally, ones distinguishes two types of morphology. On the one hand, inflectional morphology deals with modifications that specify a word's meaning but do not change its overall sens nor its part-of-speech like declension, conjugation,

[1]So much so that some linguists see Chinese as having verbs only and no adjectives.

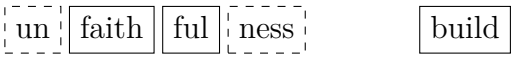


Figure 3.1: *Unfaithfulness* is made of four morphemes, two of which are bounds (*un* and *ness*), and two are free (*faith* and *full*). Notice however that when *full* is bound, it appears as *ful*. *Build* is made of a single free morpheme.

pluralisation and the like. On the other hand, derivational morphology deals with modifications that create new words from existing ones, possibly even changing their part-of-speech. Furthermore, while inflectional morphology is often productive and the sense of an inflected form is directly composed from the sense of the lemma and the sense of the morphemes involved, derivational morphology is much less predictable both in terms of meaning and productivity. Looking at the verb *pay*, an inflection of *pay* would be *paid*, both are verbs, while a derivation would be the noun *payment* for example.

In this work, we are interested in inflectional morphology because it can encode syntactic information[2] relevant to dependency parsing. Thus, from now on, whenever we speak of morphological information we will assume inflectional morphology.

The bits of information encoded via or triggering inflection are called morphological features or simply features. There are many different morphological features in the world languages, and not all languages make use of each feature. Typical examples of features are Gender (or noun class), Number, Person, Case, Tense and Mood. Less typical are Definiteness or Evidentiality. A feature when marked takes a value from a possible set. For example, in French, Number can be either Singular or Plural and Gender can be either Feminine or Masculine, while in Czech Number can be Singular, Dual or Plural, and Gender can be Feminine, Masculine or Neuter. Danish nouns can be Singular or Plural, Common (Masculine and Feminine) or Neuter and Definite or Indefinite. French or English nouns on the contrary, do not inflect for Definiteness, instead an analytical construction with a determiner is used.

In a language, morphological features are typically restricted to only certain word classes. The most common restriction being nominal, verbal and adjectival morphology. Nominal morphology touches mainly nouns and pronouns, and sometimes adjectives and participles and other verbal nouns (such as gerunds and supines). Verbal morphology on the other hand touches mainly verbs and auxiliary (and modal) verbs, and rarely adjectives. Adjectival morphology, as distinct from nominal and verbal morphology, when present, only touches adjectives and adverbs. Not all languages that use inflection have the three kinds and even when they do, they need not be as developed. Furthermore, there is an hierarchy in the application of those paradigms. Pronouns are more likely to inflect than nouns and auxiliaries more than verbs [Bla01]. English has productive paradigms for the three kinds of morphology. Nominal morphology indicates plural in nouns and also case in pronouns (*vowel, vowels*). Verbal morphology indicates third person, past tense and continuous aspect (*sing, sings, sang, singing*). Adjectival morphology indicates degrees of comparison (*easy, easier, easiest*). French has also a shallow nominal morphology, a very extensive verbal morphology and only five irregular adjectives inflecting for degrees (*bon*, *meilleur*; *bien*, *mieux*; *mauvais*, *pire*; *mal*, *pis* and *petit*, *moindre*).

[2]Derivational morphology could in principle also encode syntactic information. However, there has been much less work on the subject and data are lacking altogether.

Furthermore, as we have just pointed out, inflection can be either productive (regular) or fossilised (irregular). Productive inflections are the de-facto standard inflectional paradigms that exist in a language. They apply to any new word susceptible of being inflected. For example, any new verb entering the English language is susceptible to inflect for third person singular present indicative (*-s*), for past tense (*-ed*) and for continuous aspect (*-ing*). Likewise, any new word entering French can be inflected for plural (*-s*). Irregular inflections, on the other hand concerns mostly a few common words[3] that appeared in the language long time ago. The English verb *be* has eight forms (twice as much as a regular verb) and inflects for number[4] in the indicative past whilst no other verb does.

In the following, we call morphological attributes a couple made of a morphological feature and a value available for that feature in a given language. Examples of morphological attributes are Case=Nominative, Tense=Present or Gender=Common. From now on, we will use a sans serif font to represent attributes and morphological information as annotated in a corpus rather than the linguistic attribute itself.

Morphologically Rich Languages

Every language relies to some extent on derivational morphology to create words. It might be completely transparent like with English many noun/verb pairs that share lemmas (for example *a cut* and *to cut*). It can be the base form or the gerund or any other non finite form (compare *to start*, *a start* and *to begin*, *a beginning*). It might also be more involved, using affixation, stem modification or any other mean (for example French *couper*, *une coupure*, *une coupe*).

But when it comes to encoding syntactic and semantic information through inflectional morphology, some languages do it much more than others. For example, a regular English verb has four forms (*start*, *starts*, *starting*, *started*), the most irregular verb has height forms (*be*, *being*, *been*, *am*, *is*, *are*, *was*, *were*) while some modals only have one (*must*). A regular French verb on the contrary has about forty different forms. Similarly, while English and French nouns have two forms (some irregular nouns have two plurals *brother*, *brothers*, *brethren* and *ciel*, *ciels*, *cieux*), Finnish nouns can have thirty forms as they inflect for 15 cases and 2 numbers. French and Finnish are so called morphologically rich languages, while English is not.

There is no exact definition of morphologically rich languages (MRL). Tsarfaty et al. [TSG+10] define MRLs as *"languages in which substantial grammatical information, i.e., information concerning the arrangement of words into syntactic units or cues to syntactic relations, is expressed at word level."*

There is no clear boundary as to which language is to be considered morphologically rich, but rather general trends. Usually, languages where verbs inflect extensively for person, number and/or gender are considered morphologically rich. Because nouns are susceptible to inflect for much less categories than verbs (they do not inflect for tense, mood or aspect usually) then as soon as nouns inflect for more than one feature (Number in English) the language can be considered mildly

[3] Irregular words that are not used often tend to regularise. Those words used to occur often in the past but less now for some reasons. The French double plural of *ciel*, *ciels/cieux* is a good example. *Cieux* has a religious connotation and is a common word in prayers and scriptures. But as religion looses momentum in France, the irregular plural is less used.

[4] *You* is a second person plural in the beginning and replaced the original *thou*.

Person/Number	Spelling	Phonetic
1st Singular	chante	/ʃãt/
2nd Singular	chantes	/ʃãt/
3rd Singular	chante	/ʃãt/
1st Plural	chantons	/ʃãtɔ̃/
2nd Plural	chantez	/ʃãte/
3rd Plural	chantent	/ʃãt/

Table 3.1: The indicative present paradigm of *chanter*. There are 5 written forms but only 3 spoken ones.

rich, especially when determiners or adjectives inflect in agreement with the noun they modify.

More generally, English aside, most Indo-European languages are morphologically rich, so are Uralic (Finnish and Hungarian), Turkic, Mongolic, Semitic (Amharic and Hebrew) and various North American Languages. In the morphologically poorer side of the spectrum are some Sino-Tibetan languages (Chinese and Burmese), and many south-east Asian languages families like Tai-Kadai (Thai and Lao) and Austronesian (Vietnamese), but also some Sahel languages, English and Creoles.

Written and Spoken Morphology

Inflectional morphology, is often thought of as a written mechanism, as we tend to learn written paradigms. It is worth remembering though that spoken morphology does not necessarily equate written morphology. For example looking at the paradigmatic French first group verb *chanter* whose indicative present forms are reported in Table 3.1. It does inflect for 6 person/number pairs, but has 5 written forms, and only 3 spoken forms. Because we work here with text, we are interested with written morphology, but it is worth keeping in mind that it might not always match spoken morphology.

The reason behind form count discrepancies in Table 3.1 is a phenomenon called form syncretism. Form syncretism is the fact that several forms collapse under certain circumstances rendering them indistinguishable from each other. Form syncretism is different from not marking for certain features. For example, in Danish, verbs do not inflect for person nor number so that given a tense, a mood and a voice there is only one form for all person/number combinations. But French verbs do fully inflect for person and number, whether one considers verbs like *être* and *avoir* or other tenses like the future, there are lots of cases where each person/number combination has a different form. It turns out that through language changes, sometimes two otherwise distinct forms end up being indistinguishable from the pure spelling/pronunciation point of view, like it is the case for the first and third persons singular of French first group verbs in indicative present.

This is an important remark, because in some cases form syncretism can blur some syntactic information. For example, Latin word *templum* can be both a nominative or an accusative singular. Knowing what case it is might give precious syntactic information, as accusatives tend to be direct objects while nominatives are subjects. The form alone does not reveal this information. Using gold morphological information for parsing might look like circular, especially for those cases

where due to form syncretism only syntactic analysis can reveal the actual morphological analysis. We give a certain syntactic analysis of the sentence because of the given morphological analysis and in turn we give a certain morphological analysis because we have access to the syntactic analysis whom we used the morphological analysis to build in the first place.

However, in most cases, other clues such as word meanings and word order can help deciding the actual morphological/syntactical analysis.

Cross-Lingual Morphology

Another problem that arises from using morphological information in a cross lingual setting is cross lingual feature identification. Let us look at two examples to illustrate this.

First, French and English are two fairly similar languages, they are Indo-European languages, they share some basic typological traits like their SVO word order, and because of historical language contacts, they share a fair amount of vocabulary [FW73]. For all those reasons, it seems reasonable to train an English system and a French system together if the data are available. Assume, we learn French and English models together using morphological information as input features. French has a second person singular and a second person plural realised in pronouns, possessives and verbs, while English second person singular has completely disappeared from the language and the second person plural has taken over in all usages. If we assume a one to one correspondence between languages, the information bared by French second person singular is useless for English, and French information will be biased toward plural while English unique second person has both a plural and singular role. If we want to share in the other direction, it is not clear which weight of English unique second person should be attributed to a French singular and which to a French Plural. While this might seem anecdotal at first glance, it is in fact very common.

For example, setting Vocative aside, Latin has five cases (Nominative, Accusative, Genitive, Dative and Ablative), while Ancient Greek had four cases. There is no Ablative in Ancient Greek. But the syntactic roles taken up by Ablative in Latin also exist in Ancient Greek, so they have to be taken up by other cases. Agentive complements of passive verbs in Latin are typically Ablatives, while in Ancient Greek both Datives and Genitives are used.

This shows that there is no perfect match between morphological features in different languages. Names of features only represent their most general cross lingual uses, but each each feature has its on language specific realisations. Blake [Bla01] discusses the reasons behind the names of cases in different grammars and the role played by early descriptive linguists trying to fit their studied languages in the frame of classical and Germanic languages such as Latin, Greek and German. Thus, in spite of being a bridge between languages, morphology is still an unstable bridge.

Morphological Attributes as Parts-of-speech

As we have seen in previous section, the set of parts-of-speech used by linguists depends on many things, the most prominent being a language's grammar tradition, and others including underlying linguistic theory and downstream task specific needs.

In some cases, POS tags sets used in NLP annotations depart from traditional sets by encoding extra information directly in the part-of-speech. A great example of this is the POS tags set used for annotating the Penn treebank (a constituency trees collection) [MMS93]. Because English has a rather poor inflectional morphology, including morphological information directly in part-of-speech does not increase the set size much. Examples of those new extended part-of-speech include JJ, JJR and JJS for adjectives, comparatives and superlatives. The most interesting case is the set of verbal part-of-speech containing base verbs VB, past verbs VBD, gerunds and present participles VBG, past particles VBN, present non-third person VBP and present third person VBZ. Instead of relying on morphological features, the full English verbal morphology is accounted for with a set of 6 verbal part-of-speech. This leads to a 36 parts-of-speech set for the original Penn treebank annotation, only twice as much as the 17 parts-of-speech from the Universal Dependencies annotation scheme.

Similar representation of morphological information into POS tags has been devised for Czech and leads to a set of 3127 tags [HVH98]. Due to its highly combinatorial structure, the Czech tag set is more similar to an actual morphological analysis than to a POS tag set.

3.2 Learning Word Representation

Data representation and especially word representation is a big problem in NLP. Depending on the task at hand, one might want word representations to encode orthographic information, syntactic information, semantic information or any other relevant information. As we have seen, parts-of-speech, morphological attributes and lemmas can to some extent provide that information. However, as computers work with numbers, that information needs to be encoded with numerical values. Plunging those discrete classes into numerical space in a way that would be meaningful for NLP tasks is called embedding those classes and it has been at the heart of many recent developpements in the NLP community.

3.2.1 The Different Views of a Word

A question that needs to be addressed when dealing with word representation is that of the definition of a word. The trivial answer for speakers of European languages is what stands between two white spaces (and punctuation). However, there is much more to it than this simple answer: What about writing systems that do not use spaces such as Chinese or Japanese or even medieval European manuscripts? What about compound words and proper names like New York? And the answer one brings to that question will indeed have an impact on the design and the expressivity of its word representation methods. It should also be noted that different tasks might benefit from different definition of a word.

A word can be seen as an indivisible unit with its own form and sense. This is also called the holistic view of a word, and it leads to models where each word form has its own representation. Each vector can be packed with information, but those methods can hardly handle out of vocabulary words and thus usually need a lot of data to have a good coverage and still might not be practical for highly morphological languages.

A word can also be seen as a sequence of morphemes in which case morpheme representations are composed into word representations. This deals more

easily with out of vocabulary words, yet there might still be out of vocabulary morphemes. Furthermore, its requires some form of morpheme analysis of words and morpheme analysis and composition might not be straightforward with morphemes that have very loose meaning and borrowed words. Consider the English latinate words *construct*, *destruct* and *destroy*. While *destruct* can be analysed as *de* and *struct* because English has other similar words such as *construct*, *destroy* can hardly be analysed so, despite both words having the same etymology and analysis in Old French/Latin.

If a word is seen as merely a sequence of letters, this leads to character level models where recurrent neural network (RNN) compose character representations into words. Those recent models have shown promising results on different tasks, but much work still remain to be done especially to understand what they really encode and how powerful they can be. While they actually solve the problem of out of vocabulary words/morphemes for alphabetic/syllabic writing systems, they still face the problem of out of vocabulary characters for ideographic writing systems such as Chinese.

A word can also be seen as a lemma inflected for several morphological attributes. This requires performing some form of morphological analysis and lemmatisation, which is not trivial. But then it becomes easier to share information between those abstract categories across words, tasks and even languages. We will use this definition later in this work.

Modern systems, more and more rely on hybrid definitions for example taking both the holistic view of a word and its sequence of characters into account to compose a representation.

3.2.2 Types of Word Representations

Once a view has been chosen, there are several options for word representations. A first option is whether the representation is continuous or discrete. Discrete representations are those representation in which words can be assigned to discrete categories like parts-of-speech and morphological attributes or clusters for example, contrary to continuous representation in which words lie in a continuous vector space. In practice, grammatical categories such as parts-of-speech tend not to be continuous because it would be tedious to assign continuous representations to each word in a language by hand. However, this is not a problem for computers. Therefore, both continuous and discrete data driven representations have been used.

Another option is whether representations are assigned to tokens or to types. Tokens are words or punctuation symbols in context. Types on the other hand, are abstract archetypal entities like words in isolation. For example in the sentence *"My cat and your cat appreciate each other"*, the two occurrences of the word *cat* are different tokens, but they are both instances of the same type, the type of *cat* in which are all the *cat*s from the above parsing trees belong. But, as for most tasks contextual information is important, methods have been devised to turn type representations into token representations at run time. For example, recurrent neural networks can be used to turn a sequence of type representations into a sequence of contextualised token representations. In the example sentence above, the input representations of the two words *cat* would be the same as they are of the same type, but their contextualised output would be different as they are different tokens.

Word	Vector					
The	0	0	0	0	0	1
Cat	0	1	0	0	0	0
Sat	0	0	0	0	1	0
On	0	0	0	1	0	0
A	1	0	0	0	0	0
Mat	0	0	1	0	0	0

Table 3.2: One-hot representation of the words in the sentence *The cat sat on a mat*. The index of a word is given by its lexicographic order in the vocabulary.

3.2.3 Beyond One-Hot Encoding

The simplest approach for encoding categorical information such as word identity or part-of-speech is the so-called one-hot encoding. In one-hot encoding, each word (or any other category) is given an index in a vocabulary, like its rank in lexicographic[5] order or its frequency count rank, then each word is represented by a vector full of zeros, with a one at the corresponding index. It is a discrete representation.

More formally, let $\mathcal{V} = \{w_1, ..., w_n\}$ be a vocabulary of n words. Let there be a bijective function from the vocabulary to the n first positive integers ($\mathcal{V} \to \mathbb{N}_n$). Thus, each word has an associated index and an index is associated to one and only one word. Then, if we note $\mathbf{w}_i \in \{0, 1\}^n$ the vector representing word w_i, we have that $\mathbf{w}_i$ is the vector with a 1 at the i-th position and 0s everywhere else. Table 3.2 shows one-hot representations for a small vocabulary of six words whose index is their lexicographic order. While the requirement for bijectivity is not necessary and might even be detrimental as we shall see, for one-hot encoding of words the index mapping is indeed bijective.

The feature templates shown in Table 2.1 are indeed used for one-hot representation of edges. Given a treebank, for each template, all the instanciations of that template are gathered into a vocabulary and a unique index is assigned to each. Then the representation of an edge is just the concatenation of the one-hot vectors corresponding to each template.

The problem with one-hot encoding is that it does not encode any more information than the index of a word. A direct consequence of this, is that one-hot encoding does not carry any syntactic or semantic information at all. All the words are equally different from each other.

Another problem arising from this index encoding is the number of parameters one needs to estimate via machine learning which is further stressed by data sparsity. If in our parsing model we represent edges as pairs of head-dependant words and use one-hot encoding of those pairs, then for a vocabulary of v words, we would need to estimate v^2 parameters. For a 10 000 words vocabulary[6]. that makes 100 000 000 parameters. That is far more than what we can hope for in any "reasonably" sized corpus. For example, the Enciclopædia Universalis contains "only" 60 000 000 words. Furthermore, in practice we want to incorporate context information such as surrounding words that would increase the number of parameters by several orders of magnitude. This has two consequences. Most parameters

[5]The lexicographic order is the index in a list of word sorted alphabetically.

[6]10000 words is a small vocabulary. Modern dictionaries that only count lemmas not actual forms can have much bigger vocabularies. *Le petit Larousse illustré 2019* counts 63500 common words and the 20 tomes of *the Oxford dictionary* count more than 150000 entries.

The barked at me as I passed in front of the gate.

Figure 3.2: An English sentence with a placeholder symbol instead of a word. Even with an unknown symbol, we understand the sens of the sentence and of the missing word.

will never be estimated, either because they are linguistically irrelevant or due to their mere number. Then, they might not even fit in a computer memory all at once.

Hence it becomes clear that we need a different representation. Ideally, this new representation should be both compact and informative. By compact we mean that the length of a vector in this new representation should be much smaller than the size of the vocabulary. And by informative we mean that it should provide more information than the one-hot encoding does.

Building on the fact that language is not random and that words can be fairly well described in terms of their coocurence patterns, a solution is to directly learn word representations from raw text. With the wide availability of textual data on the Internet and with the ever increasing computing power of modern computers, it has become possible to learn word representations on billions of words.

3.2.4 Distributional Hypothesis

"You shall know a word by the company it keeps."
— John Rupert Firth

The distributional hypothesis as given by Firth's famous citation, states that one can understand a word by looking at the context it appears in. This has different implications if one is a human or a machine. For humans, this means that when one encounters a new word, one can have already a good idea of its meaning because of the surrounding words. Take the example sentence in Figure 3.2, the second word has been replaced by a symbol. The symbol is not an actual English word, but we can nonetheless have a good idea of what it stands for. It must be some sort of dog. We can infer it from the fact that it *barked* and that it was standing at a *gate*. We propagate the knowledge of the surrounding words and of the situation to the unknown word.

From a computational perspective this is slightly different. In the most basic approach, computers treat text as a string of characters. Once it has been tokenised, each token (word, number, punctuation etc.) is treated as an independent string of characters. Computers do not attach meaning to them the way we do, nor do they link similar words together. They, unlike us, do not have a sense for semantic nor for morphology. Each of those seemingly natural operations have to be performed by a program.

To exemplify this, let us look at the word *bark*. For a human who has some knowledge of English, *bark* can be essentially to different words. The noun *bark* refers to the outer layer of a tree's trunc and branches. The verb *bark* refers to the action of emitting a strong quick noise for some animals, especially dogs. For a computer, unless some preprocessing is performed so as to distinguish the two words as *bark*|*NN* and *bark*|*VB* or something alike, there is only one string of characters *bark*, and no way to distinguish them whatsoever. This shows the lack of semantic sense of computers.

	an	another	at	back	because	can	did	does	even	feed	go	got	has	make	may	me	n't	not	out	outside	professional	right	take	taking	used	with	young
horse	1	0	1	2	1	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	1	1	2	1	0	3	0
cat	0	1	0	0	1	0	0	0	0	2	0	0	3	1	1	0	0	0	0	2	0	0	0	1	0	4	1
dog	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	0	1	1	2	0	1	0	1
eat	0	0	3	0	0	7	0	1	0	0	0	0	0	0	0	0	6	7	2	1	0	0	0	0	1	0	0
buy	0	1	0	1	0	6	1	0	1	0	1	0	0	0	0	1	2	1	0	0	0	0	0	0	0	0	0

Table 3.3: A selected co-occurrence table from the Universal Dependencies EWT English treebank train set. Context words were chosen to co-occur with only two of the five words in order to get rid of too common words.

For the morphology problem, humans know that *barking* and *barked* are somehow related words and can infer the meaning of *barking* from the one of *barked*. For a computer, unless again this problem is purposely handled, *barked* and *barking* are two different strings of characters, and there are no reason they should interact in any specific way.

Those basic problems can even superpose. Take *bark* and *barks*. *Barks* can be the third person singular present of the verb *bark*, but it can also be the plural of the noun *bark*. *Barks* is two different words and relate to *bark* in two different ways, depending on its part of speech. For a computer, again, there is only one *barks* and it does not relate to *bark* in any way.

Because of those aforementioned problems, the distributional hypothesis does not really apply for computers on a single sentence. But if one has access to a big amount of text, a computer can count co-occurrences between words and contexts, and draw similarities between different words. This kind of analysis is also called distributional semantics.

3.2.5 Distributional Semantics

We have seen how a word can be defined by its context. We have also seen that because computers do not have grounded knowledge of words meaning, a single sentence might not be enough to fully capture the sense of a word. However, computers can treat massive amounts of data at a much larger speed and scale than any human could possibly do. The idea here, is that even without any sense of semantic nor morphology, just looking at co-occurrence patterns over large amounts of text already reveals word similarity. And if the input data are big enough, we expect the similarities between word co-occurrence patters to be meaningful enough so as to be usable for NLP tasks. In most languages, we expect week days to appear in similar context for example, so will months do, colours and many such words.

In order to use distributional semantics to draw similarity between words and learn representations, we need to define two sets (also called vocabularies), one for tokens and one for contexts. The vocabulary of tokens $\mathcal{V}$ is the set of objects for which we want to learn a representation. Those are usually words, but they can also be characters or morphemes or parts-of-speech for example, and it is often given by the task we want to address. The vocabulary of contexts $\mathcal{C}$ is the set

of surrounding information we will consider when looking for similarities between words. Contrary to tokens, contexts can take very different forms depending on the kind of phenomenon one wishes to study. For syntactic analysis, contexts tend to be surrounding words from a small window (2 or 3 words on each side of the current token). The idea is that the previous and next words are good indications of the syntactic use of a token, but that a word five sentences away does not carry any syntactic information any more. For topic analysis however, contexts can be paragraphs, sections or even whole documents, following the idea that documents that use the same words must be discussing similar topics and that similar topics tend to use similar words. In between, semantic relatedness is well encoded by sentence contexts. Table 3.3 gives a sample of co-occurrence counts where the contexts are surrounding words.

A good rule of thumb for designing contexts is that the narrower the context, the more syntactic information it encodes, the broader the context, the more topical information it encodes.

Before venturing any further, it is worth mentioning a few characteristics of vocabularies, contexts and co-occurrence counts. Even though the token/context co-occurrence relation is not a symmetric relation, there is something reciprocal. This means, that even though we tend to think of contexts as providing information about the tokens co-occurring with them, tokens also provide information about the contexts in which they appear. Thus we can also compare contexts with each other and learn about them.

Concerning vocabularies, languages are Zipfian in nature [Zip35], meaning that the distribution of words in a given language or a given corpus tend to follow a power law. A few words appear most of the time and most of the content words appear only a few times, many only once, if at all. For example, in the Brown Corpus of American English text, the most frequent word *the* accounts for 7% of all the words and the second one *of* for about 3.5%. Those two words already account for more than 10% of all the corpus. At the other side of the spectrum, many words appear only once and most English words are actually absent from the corpus.

In the training section of French GSD corpus available in UD version 2.2, out of 38332 forms making up 316780 words (excluding punctuation and symbols), the most common word *de* appears 24040 times, the second *le* appears 13990 times and the third *la* appears 9877 times. Altogether, the first ten words account for 97322 tokens, almost 31% of the total count. Figure 3.3 represents the first 100 words from that corpus by decreasing frequency. The last word on the plot, the hundredth most used word in the corpus is *groupe* and appears 211 times.

If not accounted for carefully, this will bias the co-occurrence counts and any subsequent analysis. If we consider whole documents as context, they will be very similar because most of their words are stop words. If we consider narrow word contexts, we will see different biases. For example, in English, nouns tend to be preceded by determiners, but while there is only one form of definite determiner (*the*), there are two such forms for indefinite determiners (*a* and *an*) and their distribution is not syntactic nor semantic but phonemic. Thus we will see three broad classes of nouns: uncountable nouns that never appear after an indefinite determiner, countable nouns beginning with a vowel sound appearing after *an* and countable nouns beginning with a consonant sound appearing after *a*. If we are aware of such issues, we can either discard determiners from our contexts, or simply replacing every instance of *an* by *a* to remove the phonemic distinction.

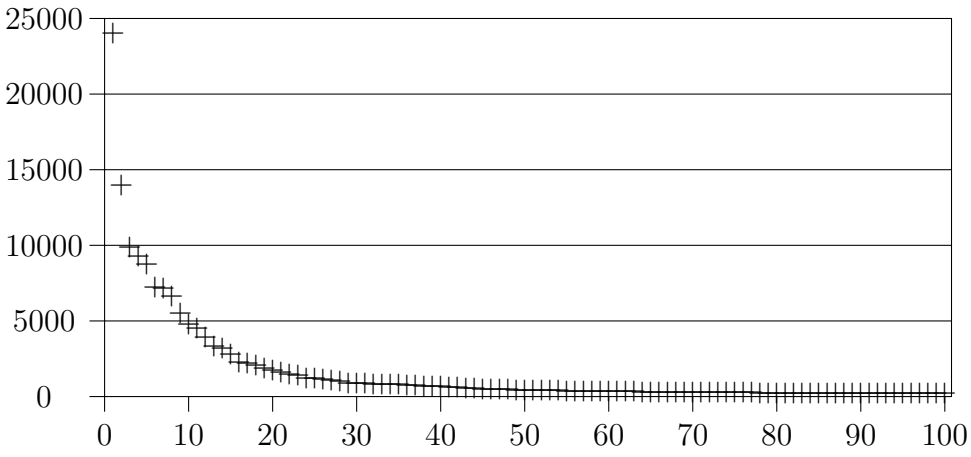


Figure 3.3: 100 most frequent words in the train set of the French GSD treebank by decreasing number of occurence. They make a typical Zipfian distribution.

Now that we have defined tokens and contexts vocabularies and that we have mentioned some of the biases that occur naturally in text and about which we should be cautious, we will look at how representations are learned in practice.

3.2.6 Continuous Representations

While co-occurrence counts could already be used to represent tokens and/or contexts, in practice they are not because they are too big. Those counts are more informative than one-hot encoding, but they are just as long if not longer, and are therefore unpractical. Instead, researchers have proposed to use much shorter vectors specially learned to retain as much of the original count information as possible. There are two main classes of approaches to learn those short vectors. The first class of approaches makes direct use of the co-occurrence counts stored in a co-occurrence matrix. The second class of approaches on the contrary works directly on the text data without ever counting explicitly co-occurrences.

The first class of approaches is also called dimensionality reduction. The co-occurrence counts are stored in a high dimension matrix whose size is then reduced using matrix factorisation methods. The idea behind factorising the co-occurrence matrix is to reveal latent representations of the input tokens and contexts that explain the co-occurrence counts using less variables. For example, if two vectors are colinear and one has twice the length of the other, this will show in the original count matrix as one having twice the co-occurrence count of the other for each context. However, it can be expressed more simply by saying that both vectors share a common support vector and have different norms. Matrix factorisation basically performs this kind of analysis for several thousands of variables and dimensions at a time. A benefit of matrix factorisation it that it allows to reduce the dimension of the input matrix while controling the information loss which is also a way to reduce eventual data noise. This is the kind of path we follow and so we leave the detailed explanation to the chapter about delexicalised word representations.

The second class of methods, also called language models, take a different path. In those methods, vectors are assigned to words and contexts, such that they maximise a given objective function. One of the most famous method in this category is Word2vec by Mikolov et al. [MSC+13]. In the Skipgram model

with negative sampling of Word2vec, word and context vectors are learned so as to maximise the dot product of words and contexts that co-occur while as the same time minimising the dot product of words and contexts that do not co-occur. In the same category are Fasttext [GMJB17] and GloVe [PSM14]. Different objective functions lead to representations with different properties. For example, Fasttext's objective is similar to the one of Word2vec but it decomposes the token vectors into sums of character n-grams vectors, effectively leaning representations for character n-grams allowing for an easy recovery of out of vocabulary words.

While their outputs may exhibit very similar properties, the difference between the two classes of approaches is that the former makes direct use of co-occurrence counts and tries to explain them, while the latter directly tries to explain the text without looking explicitly at the counts. This overall leads to very different training mechanisms.

Despite their success and popularity, those approaches suffer some inherent problems. The first being their lack of analysability and opacity. Those methods give word forms some dense continuous vectorial representation which are almost impossible to analyse for a human. It is hard to give dimensions sense and thus to make sense of values, and it is equally hard to know why a given form has received a specific vector.

The second problem is their great need for data. To be trained, they require a lot of data (on the order of billions of words) and thus a lot of time. This is a problem since not all languages have huge (even unannotated) corpora available online. Furthermore, for those approaches like Word2vec that treat each form independently, as the number of possible forms increases in a language due to morphology for example, the amount of data required increases.

Eventually, their density can also be a problem. When they are used independently the few hundred dimensions of those representations are convenient, but when it comes to learn from word interactions, it is useful to compose them via outer products and the like, which quickly leads to long dense vectors that are unpractical to work with. To have an idea of the dimensions involved, in their MSTparser, McDonald et al. [MCP05b] use sparse feature vectors with less than a hundred dimensions lit up at a time for edge representation. On the other hand, in their parser, Dozat et al. [DQM17] use dense word representations of 400 dimensions for both the governor and the dependent of each relation, giving dense edge representations of $400^2 = 160000$ dimensions. This is more than 1600 times more dimensions involved per computation for a reasonable gain.

For those reasons, people have proposed ways to induce sparse representations for words, either by directly learning discrete representation like clusters or by discretising/sparsifying dense representations.

3.2.7 Discrete Representations

Discrete representations are different from continuous ones in that they are categorical assignments, like features described in previous chapter, rather than continuous vectors. And here again, there are several possibilities.

Researchers have proposed to discretised continuous representations by binning or binarising previously learned continuous vectors [CZZ14]. Binning can be used to turn continuous valued variables into discrete categorical features by clustering values together. Binarisation is essentially the same idea, with the difference that only one or two bins are used for the most important values and negligible values

Value	A_1	A_2	A_3	A_4
0.2	0	0	1	0
0.8	0	0	0	1
-0.9	1	0	0	0
-0.1	0	1	0	0
-0.3	0	1	0	0

B_-	B_+
0	0
0	1
1	0
0	0
0	0

Table 3.4: Example of binarisation. The first table gives continuous values. The second table gives binned representation of those values. The four bins A_1 to A_4 represent the four intervals of length 0.5 in $[-1, 1]$. The third table gives a binarised representation of the values. Only the most extreme values (below -0.5 and above 0.5) are kept.

are dropped. Those two methods indeed turns dense continuous representations into discrete feature like representation (sparser in the case of binarisation). Table 3.4 shows a example of binning and binarisation.

Even if discretised vectors do not fix the size problem as they indeed increase the length of the original vector by a factor equal to the number of bins used per dimension, it has other advantages. The main advantage is that it decouples the different bins and therefore different intervals of the same dimension. It can be useful, especially when using pretrained continuous word embeddings with linear classifiers for examples, as in continuous representation, not only dimensions are not easily interpretable, but they can even have several use depending on the part of the space a word lies in, and only make sense in combination with other dimensions.

Another possibility is to directly learn categorical assignments such as word clusters. Indeed Brown et al. [FBVdLM+92] proposed to cluster words based on some measure of shared information. The idea is as follow. Given a corpus, a vocabulary $\mathcal{V}$ and a number of class c, they want to assign one of those c classes to each word in $\mathcal{V}$ such as to maximise the mutual information of adjacent classes. That is, they want to maximise $\pi(c(w_1)c(w_2)) \log(\frac{\pi(c(w_1)c(w_2))}{\pi(c(w_1))\pi(c(w_2))})$ for every sequence of two words w_1w_2 in the corpus. However, this is a NP-hard problem to solve.

So instead, they propose a greedy solution based on clustering words so as to minimise information loss. By doing it iteratively, one merges always more words effectively ending up with a single word vocabulary. But at each intermediate step, words are clustered with similar words in term of information content and those clusters form an hierarchy of nested clusters. Thus a word belongs to several clusters, and we can have clustering of different granularity. A word's representation is then the list of clusters it belongs to. This kind of method is referred to as hierarchical clustering. A bit like a cat is a cat, a feline (like lions), a mammal (like us), an animal (like crocodiles) and so on, hierarchical clustering techniques group words together in bigger and bigger groups but using information from the data instead of world knowledge. Figure 3.4 shows an example of hierarchical clustering for a few words.

Because we can choose many different criteria as basis for clustering words, researchers have proposed to apply hierarchical clustering to the output of continuous representation learning algorithms to benefit from both language modelling objectives and sparse representations. In this kind of approach, the clustering criterion is the distance between words in the continuous space for example.

Some researchers have also proposed embedding algorithms that specifically

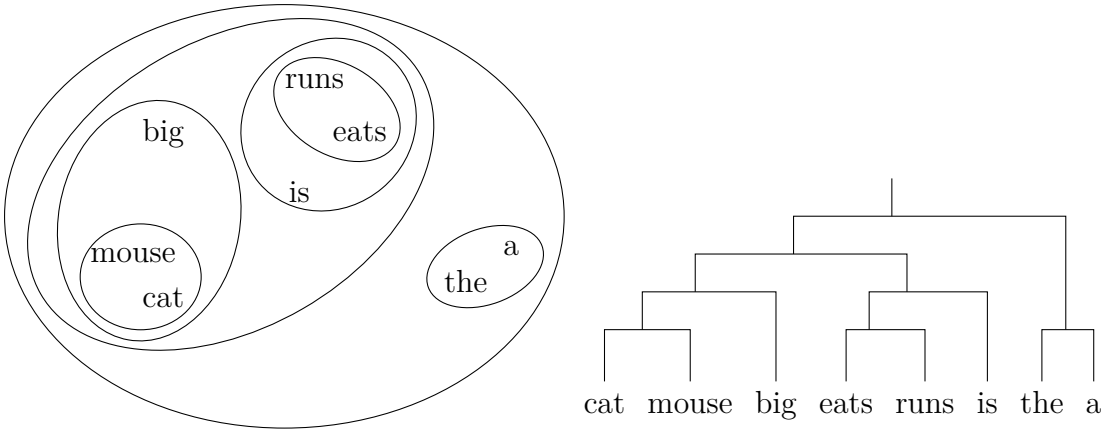


Figure 3.4: An example of hierarchical clustering. Each word is its own atomic cluster. Each ellipse represents a bigger cluster. Similar words cluster together.

require their output to be sparse but not necessarily binary. Such as the sparse over-complete representations of Faruqui et al. [FTY+15]. This is a compromise between the two broad family of approaches presented above. The idea is to have words belonging to as few classes as possible but in the same time, allowing them to adhere to those classes with more or less strength.

3.2.8 Engineering, Learning and Selection

We have just seen different ways to learn representations for words, ranging from dense vectors to nested clusters. However, as we have seen in Section 3.1 about linguistic analysis of words, the problem of extracting information from words is not a recent one. So, here we take a moment to look at how representation learning methods differ from older methods such as feature engineering and feature selection.

Approaches such as Word2vec, Fasttext or GloVe, also called language models induce word representations so as to optimise some mathematically defined criterion. For example, the objective of Word2vec is to maximise the dot product between words that co-occur while minimising the dot product of words that do not appear together in a corpus. By iteratively optimising the word representations in order to fit that objective on a given corpus, the algorithm indeed learns a language model. The fact that those methods optimise objective functions directly defined on the text give them a direct learning flavour.

For representation methods based on matrix factorisation, what the methods learn might be less clear. The goal of matrix factorisation is to find vectors with desired properties that can be used to reconstruct the original matrix, minimising the reconstruction error. In that sense, those methods learn transformations of the original input spaces that loose as little information as possible given a set of constraints, such that the transformations are linear or have a small number of basis vectors. Those methods are task agnostic and the informational content of the final representation directly depends on the input rather than on the objective function as is the case for language models.

Feature engineering corresponds to the process of hand crafting feature templates for a given task. It is often said (especially in computer vision) that as neural networks automatically extract features they have removed the problem of feature engineering. For languages, this is somewhat different. The use of repre-

sentations has not removed the problem of feature engineering, it has displaced it. When in the past decades, people had to choose whether to use information about capitalisation, prefixes and suffixes length, part-of-speech of surrounding words and so on in their feature vectors, now they can just use a dense vector that will bring all the relevant information along. However, the problem has become to learn those representations, which in the case of languages is challenging because of data sparsity and of the multiple levels of linguistic structures interacting together. The importance of the contexts used for learning representations has replaced previous feature engineering. We can indeed see the displacement from feature engineering for specific tasks to context engineering for representations in the always increasing number of embedding models, amongst which are Word2vec (SkipGram and CBOW) [MSC+13], Glove [PSM14], CoVe [MBXS17], Fasttext [GMJB17], ElMO [PNI+18] and many more.

Feature selection, is yet another related problem. Feature engineering can easily lead to relevant tokens for a task (in our case edges) to be represented by sparse vectors of millions of dimensions. While a good learning algorithm should be able to automatically tell relevant features from noisy ones, it can be hard when dealing with orders of magnitude more features than actual data points, with most of the feature appearing only rarely. Feature selection is a way to keep only those features relevant to the task at hand. As such, feature selection is a kind of dimensionality reduction. However, while feature selection works on surface features, dimensionality reduction methods select the best latent features, indeed learning a transformation of the original space, often in the shape of a projection on a latent basis. Furthermore, feature selection looks for the most relevant features for a given task, while dimensionality reduction is in general task agnostic.

We have argued that one-hot encoding is too simple to be practical and that we need more informed word representations. After having discussed the distributional hypothesis and its computational corollary distributional semantics, we have presented different ways to learn word representation automatically from data. Eventually, we have seen how those methods are indeed learning word representations and not just engineering them or selecting some salient features. We now turn to some linguistic concepts used to describe words and as features in natural language processing tasks, namely lemmas, parts-of-speech and morphological attributes.

3.3 Conclusion

In this chapter, we have seen how word information can be represented by linguistic theoretic devices such as lemmas, parts-of-speech and morphological attributes. We have also looked at how other representations such as dense embeddings or word clusters could be learned automatically from data. Those are two faces of the same problem which is abstracting information away from word forms. And in practice, linguistic theories are made to account for linguistic phenomena appearing in texts (amongst other), so data driven representations should encode similar (but not identical) information as theoretic devices.

Furthermore, as we have already said, annotation is a time and expertise consuming task, thus not all data used for NLP tasks are hand annotated for every type of linguistic information. Instead, hands annotated corpora are used to train

models that are applied to new data, sometimes even relying on data driven representations to perform lemmatisation, POS tagging and morphological analysis.

Some people have also worked on unsupervised part-of-speech tagging [CGS10]. The goal in unsupervised part-of-speech tagging is like plain supervised part-of-speech tagging, to assign a POS to each word in their sentences context. Thus categories are assigned at the token level rather than at the type level like for Brown clustering for example. But this is still a purely data driven approach to word clustering as no annotation is provided for training and as such it is very similar to representation learning.

Another point we should make is that while word embeddings can encode lexical semantics to various degree, part-of-speech and morphological attributes tend not to. Linguists have come up with other tools for working with lexical semantics such as ontologies. Ontologies are usually huge graphs in which words or sets of words are linked together via a wide range of semantic relations. One of the most famous ontologies is Wordnet [Mil95]. In Wordnet, the graph encodes that a *cat* is a *feline* which in turn is an *animal* and so on for example. We do not give further details here because while ontologies have been used jointly with parsers for semantic parsing, they have not been used for dependency parsing. Furthermore, they mostly focus on lemmas and still have a rather low coverage.

Finally, modern parsers amongst other NLP systems usually use a combination of part-of-speech and word embeddings to represent their input data, so the actual word representations fed into a system really make use of the best of both worlds.

After having described the kind of input a dependency parser receives in the shape of sequences of word representations, parts-of-speech and morphological analysis, we now turn to looking at the way machine learning models learn to parse those inputs.

Chapter 4

Related Works on Multi-Lingual Dependency Parsing

In the previous chapter, we have defined dependency parsing as the problem of automatically analysing the syntactic structure of sentences in human languages by linking words together with dependency relations. We have seen how modern dependency parsers build on graph theory, machine learning and linguistics to perform this task. However, we remained elusive regarding the kinds of data those systems process and the actual challenges presented by automated syntactical analysis of multiple languages.

In this chapter, we discuss the context of this thesis, namely multi-lingual dependency parsing. In Section 4.1, we precisely define the problem of multi-lingual dependency parsing. In Section 4.2, we present the kind of data used by parsers with a focus on the Universal Dependencies Project [NAA+18a] which is our main data source. In Section 4.3, we present some related works on multi-lingual dependency parsing.

4.1 Multi-Lingual Dependency Parsing

As we have seen, dependency parsing is a task that takes natural language sentences in and outputs trees over those sentences. However, human languages have a high degree of variability. Sentences can be made up of very simple short words or very long information packed ones and they display very different, more or less strict word orders. Therefore some grammatical or lexical information that are of prime importance for a language can lack from another altogether. Thus, it seems natural to wonder if we should design language agnostic parsing algorithms or if we should rather design algorithms targeting certain types of languages specifically. Indeed, parsing algorithms have been designed for specific languages. For example, because of its suffixed case markers and its strict right-headed word order, Japanese have received a number of efficient dedicated algorithms [TM15].

This brings us to the first sense of multi-lingual dependency parsing. In its broadest sense, it refers to the problem of devising language agnostic parsing algorithms. Multi-lingual parsers should be flexible enough to be able to learn a model and parse data from any language assuming that they are provided in a more or less uniform format. This departs from the parsing of specific languages where the peculiarities of the language are hard coded in parsing algorithms.

In a narrower sense, multi-lingual dependency parsing refers to the problem of learning to parse several languages jointly in order to build on common information

and improve parsing results [NBG12, AMB+16]. In this sense, it can be thought of as a kind of multi-task learning problem. Parsing each language is a task on its own, but they are very similar to each other and so we aim at building on those similarities to improve performances.

The first sense of language agnostic parsing is the one used for the several CONLL shared task on multi-lingual dependency parsing [BM06, NHK+07, ZPS+17, ZHP+18]. The goal of those tasks was to compare different parsers on different languages without requiring that the various parser share information. The only requirement is that the various models are trained with the same code.

In this work, we stick to the narrower sense of multi-lingual dependency parsing. We aim at learning better parsers for individual languages relying on their shared features and commonalities.

A related problem to multi-lingual dependency parsing is the task of *cross-lingual* dependency parsing. The idea behind cross-lingual dependency parsing is to use annotated data from some languages to learn models for languages that lack annotated data. We shall discuss annotation in the next section, but for now we shall remind that to train a supervised parser, one needs pairs made of sentences and their actual dependency structures.

To annotate a corpus with its dependency structure is a lengthy task that requires expert knowledge both in the grammar of one's language but also in the annotation scheme used. Hence, annotated corpora are only available for less than a hundred languages, which is very few compared to the estimated several thousands of languages spoken on Earth, and languages having corpora of more than 10000 annotated sentences are about twenty. At the same time, unsupervised dependency parsing methods are still well below supervised techniques [CJT17, LZ15, Mar16], thus researchers have tried to come up with solutions to parse languages with few annotated data if at all relying on data from resource rich languages. This problem, depending on the approach followed, has been labeled cross-lingual dependency parsing when at least some data (not annotated with dependency structure) from the target language is available or zero-shot dependency parsing when not target data is available at all.

In general, cross-lingual dependency parsing is taken as a transfer task where one tries to take information from one or more well resourced languages and transfer it to a resource poor language. There have been several proposals to achieve this goal. The simplest is maybe the use of delexicalised parsers [MPH11, AWY16, LFDT14]. Delexicalised parsers do not use lexical information as encoded by word forms, they only rely on more language independent information, typically parts-of-speech. Assuming that the source and the target corpora are labeled with the same set of POS tags, one can apply a parser trained on the source language directly to the target one without caring for the actual underlying languages.

Another possibility is to transfer annotation directly [WP12, LAWY16, HRWK01], for example via parallel text. If one has access to a parallel text in the source and the target language, one can first apply its supervised parser trained on source language data to the source language side of the word aligned parallel text, then map the structure from the source side to the target side of the parallel text and finally train a parser on the newly annotated target data. Other proposals have made use of automated translation systems to translate an annotated data set from a language to another language [RMv14]. However, this might not be practical as if a language already has a corpus big enough to train a reliable translation system, then it most likely has dependency annotated data as well. The problem

with annotation transfer is that on the top of the target language parser errors will stack up errors from the noisy transfer process, whether it is due to faulty word alignments or erroneous translations.

Eventually, with the recent success of word embeddings, people have used cross-lingual word representations in order to transfer parsing models as well [GCY+15, XG14].

This work primary focus is multi-lingual dependency parsing, where we use information from several languages in order to improve over each independent language model. Nonetheless, some of the techniques we investigated also allow zero-shot dependency parsing at a low cost thus we also present some results of zero-shot dependency parsing in chapter 6.

After having discussed the actual focus of this work, namely multi-lingual dependency parsing, we now describe the kind of data used to perform it.

4.2 Universal Dependencies

Like most NLP tasks, supervised training of dependency parsers relies extensively on annotated corpora. But since the early years of dependency parsing, resources were only available for a handful of mostly Indo-European languages. And even when resources were available, they were following different annotation schemes, using different parts-of-speech and dependency relation sets. Furthermore, some corpora used in dependency parsing were not even made with dependency parsing in mind. For example, as we mentioned earlier, the Penn Treebank [MMS93] that as been widely used to test dependency parsing techniques, was originally annotated with constituency trees. Only later, were those constituency trees converted into dependency trees using heuristics [Col97, JN07].

It was highly impractical for many reasons. Even though we wish to devise algorithms that are as language agnostic and as annotation scheme agnostic as possible, in practice they always tend to favour some structures. Because annotation schemes can have varying parsing complexity, and some part-of-speech sets contain more information than others, it was very hard to compare results across languages. Even converting annotations from one scheme to another was only partially effective as the conversion process is itself error prone and the scheme maps are never fully satisfying.

It was even worse when performing cross/multi-lingual dependency parsing. Since models needed not only to learn from different languages but also from potentially divergent annotation schemes. In order to solve those issues the Universal Dependencies project was started.

The goal of the Universal Dependencies project [NAA+18a] is indeed to address all those issues by *"developing cross-linguistically consistent treebank annotation for many languages, with the goal of facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective"*[1].

The Universal Dependencies project currently hosts more than a hundred treebanks in more than 70 languages. While a majority of the languages are still Indo-European, there are more and more languages from other families as well. All those treebanks aim at being annotated in a consistent manner so as to facilitate cross-lingual parsing study. They have therefore adopted the common CONLL-U format.

[1] http://universaldependencies.org/introduction.html

In the CONLL-U format, sentences are given one word per line followed by its annotation on 10 columns as exemplified in Table 4.1. The first column is the index of the word in the sentence starting at 1. It is useful for encoding surface words that are contractions of several syntactical words, like French *au* which is the contraction of *à le*. Those words receive a span index (such as $2-3$) and their annotation is split over as many lines as actual syntactic words they contain. Columns 2 and 3 hold lexical information as the actual word form and its lemma (or dictionary form). Columns 4 and 5 hold parts-of-speech (UPOS and XPOS). The UPOS is chosen from a fixed set of 17 universal parts-of-speech. They are reported in Table 4.2. The XPOS can be used for languages and treebanks that had previous annotation for backward compatibility. Column 6 hosts the morphological analysis of the word form in terms of morphological attributes. The analysis is based on a set of cross-linguistically frequent features such as Gender, Number and Mood, but each language is free to add specific values if necessary. There are also a few borderline attributes that are not morphological in nature such as attributes marking misspelled words and foreign words. Those morphological attributes will be primary source of information in this thesis and will be used as pivot between different languages. The cross-linguistically most frequent attributes are reported in Table 4.4 with the number of treebanks in which they are used. As not all the 122 treebanks are completely morphologically annotated yet, numbers and persons, appearing in hundred treebanks are used in virtually all the treebanks. However, even in the top 40 morphological attributes, some are only used for less than half of the treebanks (e.g. perfective aspect, future tense). Column 7 hosts the index of the head of the word in the dependency tree. The root of the tree has a head index of 0. Column 8 hosts the type of the dependency relation which is taken from a set of universal syntactic relations such as Nominal Subject (nsubj), Case marker (case) or Determiner (det). They are reported in Table 4.3. Column 9 can be used to add extra dependencies for cases where the tree formalism is considered too strong like for sentences with several verbs conjugated for a single overt subject. Eventually, column 10 can be used to add extra information like to signal that there is no space between the last word of a sentence and the following dot despite their being distinct tokens. Table 4.1 gives an example of an annotated sentence from the UD project.

Treebank sizes vary from a few hundreds sentence to more than fifty thousands. Most treebanks are split into three part, a big train set used to train parsing models, a smaller development set used for algorithm development and hyper-parameter selection and finally a test set to evaluate models on never foreseen data, but some treebanks still only have test data. The actual size (number of sentences) of each set of each treebank is given in Tables 4.5, 4.6 and 4.7. Some languages have several treebanks reflecting previous annotation efforts from several groups. The 21 languages that have at least one treebank with more than 10000 training sentences are reported in bold. Likewise, the 11 languages that only have test data are reported in italic.

The huge resource disparity across languages is a strong incentive for the development of methods that share information across languages. Obviously, languages that lack annotated data as represented by languages that only have test sets need to be handled somehow using annotated data from other languages. But even when training data are available, they may be highly insufficient, for example Kazakh, Upper Sorbian and Armenian, all have less than 50 train sentences. However, transfer methods would ignore those few sentences altogether, which is

1 Id	2 Form	3 Lemma	4 UPOS	5 XPOS	6 Feats	7 Head	8 Deprel	9 Deps	10 Misc
1	Deux	deux	NUM	_	NumType=Card	3	nummod	_	_
2	autres	autre	ADJ	_	Number=Plur	3	amod	_	_
3	photos	photo	NOUN	_	Gender=Fem Number=Plur	6	nsubj:pass	_	_
4	sont	être	AUX	_	Mood=Ind Number=Plur Person=3 Tense=Pres VerbForm=Fin	6	aux:pass	_	_
5	également	également	ADV	_	_	6	advmod	_	_
6	montrées	montrer	VERB	_	Gender=Fem Number=Plur Tense=Past VerbForm=Part Voice=Pass	0	root	_	_
7-8	du	_	_	_	_	_	_	_	_
7	de	de	ADP	_	_	9	case	_	_
8	le	le	DET	_	Definite=Def Gender=Masc Number=Sing PronType=Art	9	det	_	_
9	doigt	doigt	NOUN	_	Gender=Masc Number=Sing	6	obl:mod	_	SpaceAfter=No
10	.	.	PUNCT	_	_	6	punct	_	_

Table 4.1: An example of annotated sentence from the French Sequoia corpus of the UD project.

UPOS	Gloss	English Examples
ADJ	Adjective	Big, pink, serious
ADP	Adposition	At, in, on
ADV	Adverb	Always, better, well
AUX	Auxiliary	Might, should, will
CCONJ	Coordinating Conjunction	And, but, or
DET	Determiner	A, your, the
INTJ	Interjection	OK, thanks, bye
NOUN	Noun	Fire, noun, word
NUM	Numeral	One, 2, III
PART	Particle	Not, 's
PRON	Pronoun	Her, Somebody, Yours
PROPN	Proper Noun	Århus, Carlos, UE
PUNCT	Punctuation	, : ...
SCONJ	Subordinating Conjunction	If, since, that
SYM	Symbol	€, +, :(
VERB	Verb	Eat, sleep, rave, repeat
X	Other	

Table 4.2: The 17 universal parts-of-speech used for annotating the Universal Dependencies project. For each UPOS, we give the gloss and a few examples in English.

Relation	Gloss
acl	clausal modifier of noun (adjectival clause)
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
clf	classifier
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
dep	unspecified dependency
det	determiner
discourse	discourse element
dislocated	dislocated elements
expl	expletive
fixed	fixed multiword expression
flat	flat multiword expression
goeswith	goes with
iobj	indirect object
list	list
mark	marker
nmod	nominal modifier
nsubj	nominal subject
nummod	numeric modifier
obj	object
obl	oblique nominal
orphan	orphan
parataxis	parataxis
punct	punctuation
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

Table 4.3: The 37 universal relations used for annotating the Universal Dependencies project. For each relation, we give the gloss. Each relation can be specified with language specific information if necessary.

Attribute	English name	#Treebank
Aspect=Perf	Perfective aspect	49
Case=Acc	Accusative case	83
Case=Dat	Dative case	60
Case=Gen	Genitive case	68
Case=Nom	Nominative case	85
Definite=Def	Definite	56
Definite=Ind	Indefinite	51
Degree=Cmp	Comparative	63
Degree=Pos	Positive (Basic form)	55
Degree=Sup	Superlative	62
Foreign=Yes	Foreign Word	47
Gender=Fem	Feminine	80
Gender=Masc	Masculine	83
Gender=Neut	Neuter	51
Mood=Cnd	Conditional mood	49
Mood=Imp	Imperative mood	85
Mood=Ind	Indicative mood	91
Number=Plur	Plural	105
Number=Sing	Singular	100
NumType=Card	Cardinal Number	78
NumType=Ord	Ordinal Number	58
Person=1	First Person	100
Person=2	Second Person	99
Person=3	Third Person	101
Polarity=Neg	Negative	94
Poss=Yes	Possessive	61
PronType=Dem	Demonstrative pronoun	79
PronType=Ind	Indefinite pronoun	66
PronType=Int	Interrogative pronoun	67
PronType=Prs	Personal pronoun	89
PronType=Rel	Relative pronoun	62
Reflex=Yes	Reflexive pronoun	62
Tense=Fut	Future	59
Tense=Past	Past	94
Tense=Pres	Present	90
VerbForm=Fin	Finite Verb (Conjugated)	83
VerbForm=Inf	Infinitive	84
VerbForm=Part	Participle	87
Voice=Act	Active voice	49
Voice=Pass	Passive voice	66

Table 4.4: The forty most commonly used attributes in UD 2.2 treebanks with the number of treebanks in which they are used. There are 489 such attributes in total in UD 2.2, most of which are language specific or used by a very small number of languages. As in the 122 treebanks of UD 2.2, a few are not yet completely morphologically annotated, plural and singular numbers and first, second and third persons appearing in 100 treebanks are virtually use in all treebanks. Definiteness, perfective aspect or future tense however, are much less common.

clearly sub-optimal. This therefore calls for dedicated multi-lingual methods that use annotated data from both the target language and from other languages in order to build better parsing models.

It is worth mentioning that the UD project is an ongoing project. Treebanks and the annotation scheme evolve. Since the release of the first version of the treebanks in January 2015, there has been a new release every six months or so with a major update in 2017. While the annotation scheme tends to be stable between updates, new conventions can be adopted when necessary to improve cross-lingual consistency. Each update comes with new languages and/or new treebanks, sentences can be added to existing treebanks and train/dev/test splits can change. Likewise, in already established treebanks, annotation errors can be corrected. Altogether, this makes comparison of systems tested on different releases difficult. In the following chapters, the reported results have been obtained on either version 1.3, 2.0 or 2.2 depending on the latest available version at the time the corresponding work was done.

It should also be kept in mind that because of the ongoing nature of the project and because of the background some languages have, not all treebanks reach the same level of annotation quality and commitment to the annotation scheme. Because cross-lingual consistency is subsumed to the individual treebank committing to the annotation scheme, there remains a lot of work to increase cross treebank consistency. Furthermore, the annotation scheme itself could be improved and some conventions are disputable as discussed by Gerdes et al. [GK16].

Here we give two examples of this lack of consistency. In the Faroese treebank, the copula *vera* is tagged VERB while the annotation scheme demands copulas to be tagged AUX, which is the case in other treebanks such as the Danish one. Furthermore the Faroese treebank assumes some basic values (VERB are in finite form if not specified otherwise) whereas other treebanks make them explicit (French has VerbForm=Fin for all finite verbs). The second example comes from Hebrew. In Semitic languages (Hebrew, Arabic, Amharic...), nouns can inflect to indicate their possessor. While this is also true of other languages such as Hungarian, Hebrew possessors are treated as merely suffixed pronouns where Hungarian treats them as full inflection. This is even more surprising as it is not linguistically backed up and that to be grammatically correct, the Hebrew treebank introduces extra word that are not present in the original text. For example sentence 148 of the development set of the Hebrew treebank (UD version 2.2) contains the word *'avodati* (my job), which is analysed as *'avodah shel ani* (job of me). This is disputable since it adds extra words (absent from the original sentence) and trivial edges from *shel* to *ani* and from *ani* to *'avodah*, artificially increasing the parsing results. Those two examples are just to show the work that still remains to be done. We come back to those problems and their actual impact on dependency parsing in chapter 7 where we discuss the role of the annotation scheme.

Now that we have defined the two related problems of cross-lingual and multi-lingual dependency parsing and that we have presented the data we use throughout this work, we will look at some previous works addressing similar problems.

4.3 Related Work

A lot of methods have been proposed in order to solve cross-lingual and multi-lingual dependency parsing. Here we try to present the main lines of research that have been investigated.

Code	Language	Treebank	Train	Dev	Test
af	Afrikaans	AfriBooms	1315	194	425
am	*Amharic*	ATT	0	0	1074
ar	**Arabic**	NYUAD	15789	1986	1963
		PADT	6075	909	680
		PUD	0	0	1000
be	Belarusian	HSE	260	65	68
br	*Breton*	KEB	0	0	888
bu	Bulgarian	BTB	8907	1115	1116
bxr	Buryat	BDT	19	0	908
ca	**Catalan**	AnCora	13123	1709	1846
cop	Coptic	Scriptorium	377	77	60
cs	**Czech**	CAC	23478	603	628
		CLTT	860	129	136
		FicTree	10160	1309	1291
		PDT	68495	9270	10148
		PUD	0	0	1000
cu	Old Church Slavonic	PROIEL	4123	1073	1141
da	Danish	DDT	4383	564	565
de	**German**	GSD	13814	799	977
		PUD	0	0	1000
el	Greek	GDT	1662	403	456
en	**English**	EWT	12543	2002	2077
		GUM	2914	707	769
		LinES	2738	912	914
		PUD	0	0	1000
		ParTUT	1781	156	153
es	**Spanish**	AnCora	14305	1654	1721
		GSD	14187	1400	426
		PUD	0	0	1000
et	**Estonian**	EDT	20827	2633	2737
eu	Basque	BDT	5396	1798	1799
fa	Persian	Seraji	4798	599	600
fi	**Finnish**	FTB	14981	1875	1867
		PUD	0	0	1000
		TDT	12217	1364	1555
fo	*Faroese*	OFT	0	0	1208
fr	**French**	GSD	14554	1478	416
		PUD	0	0	1000
		ParTUT	803	107	110
		Sequoia	2231	412	456
		Spoken	1153	907	726
fro	**Old French**	SRCMF	13909	1842	1927
ga	Irish	IDT	566	0	454
gl	Galician	CTG	2272	860	861
		TreeGal	600	0	400
got	Gothic	PROIEL	3387	985	1029
gr	**Ancient Greek**	PROIEL	15015	1019	1047
		Perseus	11476	1137	1306

Table 4.5: List of UD 2.2 treebanks by alphabetical order of their language code from A to G.

Code	Language	Treebank	Train	Dev	Test
he	Hebrew	HTB	5241	484	491
hi	**Hindi**	HDTB	13304	1659	1684
		PUD	0	0	1000
hr	Croatian	SET	6983	849	1057
hsb	UpperSorbian	UFAL	23	0	623
hu	Hungarian	Szeged	910	441	449
hy	Armenian	ArmTDP	50	0	514
id	Indonesian	GSD	4477	559	557
		PUD	0	0	1000
it	**Italian**	ISDT	13121	564	482
		PUD	0	0	1000
		ParTUT	1781	156	153
		PoSTWITA	5368	671	674
ja	**Japanese**	BCCWJ	40890	8453	7913
		GSD	7164	511	557
		Modern	0	0	822
		PUD	0	0	1000
kk	Kazakh	KTB	31	0	1047
kmr	Kurmanji	MG	20	0	734
ko	**Korean**	GSD	4400	950	989
		Kaist	23010	2066	2287
		PUD	0	0	1000
kpv	*Komi-Zyrian*	IKDP	0	0	75
		Lattice	0	0	155
la	**Latin**	ITTB	15808	700	750
		PROIEL	15906	1234	1260
		Perseus	1334	0	939
lt	Lithuanian	HSE	153	55	55
lv	Latvian	LVTB	5424	1051	1228
mr	Marathi	UFAL	373	46	47
nb	**Norwegian Bokmaal**	Bokmaal	15696	2410	1939
nl	**Dutch**	Alpino	12269	718	596
		LassySmall	5789	676	876
nn	**Norwegian Nynorsk**	Nynorsk	14174	1890	1511
		NynorskLIA	339	0	1057
pcm	*Naija*	NSC	0	0	948
pl	**Polish**	LFG	13774	1745	1727
		SZ	6100	1027	1100
pt	Portuguese	Bosque	8329	560	477
		GSD	9664	1210	1204
		PUD	0	0	1000
ro	Romanian	Nonstandard	5264	1052	1052
		RRT	8043	752	729
ru	**Russian**	GSD	3850	579	601
		PUD	0	0	1000
		SynTagRus	48814	6584	6491
		Taiga	880	0	884

Table 4.6: List of UD 2.2 treebanks by alphabetical order of their language code from H to R.

Code	Language	Treebank	Train	Dev	Test
sa	*Sanskrit*	UFAL	0	0	230
sk	Slovak	SNK	8483	1060	1061
sl	Slovenian	SSJ	6478	734	788
		SST	2078	0	1110
sme	NorthSami	Giella	2257	0	865
sr	Serbian	SET	2935	465	491
sv	Swedish	LinES	2738	912	914
		PUD	0	0	1000
		Talbanken	4303	504	1219
swl	Swedish Sign Language	SSLC	87	82	34
ta	Tamil	TTB	400	80	120
te	Telugu	MTG	1051	131	146
th	*Thai*	PUD	0	0	1000
tl	*Tagalog*	TRG	0	0	55
tr	Turkish	IMST	3685	975	975
		PUD	0	0	1000
ug	Uyghur	UDT	1656	900	900
uk	Ukrainian	IU	4513	577	783
ur	Urdu	UDTB	4043	552	535
vi	Vietnamese	VTB	1400	800	800
wbp	*Warlpiri*	UFAL	0	0	55
yo	*Yoruba*	YTB	0	0	100
yue	*Cantonese*	HK	0	0	650
zh	Chinese	CFL	0	0	451
		GSD	3997	500	500
		HK	0	0	908
		PUD	0	0	1000

Table 4.7: List of UD 2.2 treebanks by alphabetical order of their language code from S to Z. For each language, we report the size of the train, dev and test sets of its different treebanks. Languages that have at least one treebank with more than 10000 training sentences are given in bold. Languages that lack train and dev sets altogether are given in italic.

4.3.1 Delexicalised Parsers

The simplest approach to cross-lingual dependency parsing is the use of delexicalised parsers [MPH11]. As they are not trained on language specific word forms, they can easily be applied to any treebank that share the same POS set. And because they do not rely on language specific information, they can be directly trained on many languages at the same time. Lynn et al. [LFDT14] experiment with delexicalised parsing for Irish from several languages and also train a parser using all those languages at once. They report a substantial degradation of the results as compared to a mono-lingual Irish baseline which is expected since they do not use any language specific information at all and just transfer the model blindly.

Indeed, unless source and target languages have very similar grammar, this might only be partially effective. In order to fix that problem, Aufrant et al. [AWY16] propose to rewrite source language data to better fit the surface statistics and typology of the target language, by for example deleting determiners or switching the word orders.

4.3.2 Annotation Projection

Because delexicalised parsers lack lexical information, people have proposed ways to transfer annotation from resource rich languages to resource poor languages in order to directly train lexicalised parsers for target languages. This is often done with the help of parallel corpora where links are projected between pairs of previously automatically aligned words. However, as words might not align perfectly between pairs of sentences from different languages, Ganchev et al. [GGT09] propose not to commit to the complete source tree and to use language specific rules to fix the target tree instead. This was especially useful before the advent of the UD project, when treebanks were following very different annotation schemes and poor results were due to clashing annotations.

McDonald et al. [MPH11] propose a transfer method that iteratively learns new parsers for the target language by using one of k best parse trees for each sentence as training sample. The training tree is chosen to mirror the source language tree as much as possible. They seed their parser with a multi-lingual delexicalised model.

Wroblenska et al. [WP12] propose another transfer method based on weighted projection. Using parallel corpora they project dependency edges from the source language to the target language, weighting projections according to the type of alignment they involve. For example, two one-to-one aligned words have a bigger chance to maintain their edge than two one-to-many aligned words. Then they make use of a spanning tree inference in order get parse tree for the target language and eventually be able to train their parser on lexicalised target data.

Lacroix et al. [LAWY16] propose yet another simpler transfer method. By showing that a parser can be trained on partially annotated trees, they propose to project dependencies only between unambiguously aligned (one-to-one) word pairs. Then, they just remove trees that are more incomplete than a given threshold. They show that using parallel data from several languages further improve parsing accuracy.

4.3.3 Cross-Lingual Representations

More recently, lexicalised cross-lingual parsers have been trained using cross-lingual word representations in order to alleviate the lack of lexical information in delexicalised parsers. Täckström et al. [TMU12] propose to use cross-lingual word clusters instead of the actual word forms in order to allow direct transfer of dependency parsing models learned from a source language to a target language. They propose two methods to induce those word clusters. The first just uses word alignment between source and target languages to project the clusters. The second methods directly induces cross-lingual clusters by using both alignment constraints and mono-lingual modeling constraints.

Xiao et al. [XG14] use the source and target parsing data to learn word representations and they enforce both spaces to be similar by tying together translation pairs from both languages. Then they just learn a parser on the source language and apply it to the target language using their word representations for lexicalisation.

In the same line of work, Guo et al. [GCY+15] propose to use word alignment from parallel corpus to align the word representation spaces of the source and target languages. They also experiment with canonical correlation analysis to align spaces. Then they can directly learn a parser for the source language and apply it to the target language using the shared word representation space. They also show that adding cross-lingual word clusters information similar to those of Täckström et al. on the top further improves the results.

4.3.4 Direct Transfer and Surface Form Rewriting

Eventually, a niche method is to use a lexicalised parser trained on a source language directly on a target language. This is indeed a possibility when the source and target languages are really close. Garcia et al. [GGRAP17] investigate the parsing of Galician, using other Romance languages as sources. They analyse parsing results of delexicalised and lexicalised parsers in light of the lexical similarity between languages. Eventually, they propose to rewrite source data not to fit the target data typology but its spelling conventions. And they show that changing Portuguese spelling to fit Spanish spelling tradition helps improve Galician[2] results.

The lines of work presented above are interesting and promising in their own rights. Annotation transfer is getting easier and easier as more parallel corpora become available online, either due to private initiatives or international institutions (European Union, United Nations...). Likewise, as word representations get more and more powerful and cross-lingual representation are proposed for tasks such as machine translation, it becomes more and more practical to learn lexicalised parsers directly in a cross-lingual representation space. Eventually, as more and more languages have annotated data with dependency structures, it becomes always easier to find a related language that have annotated data that can be adapted to a new target language. However, those techniques are all asymmetrical in that they use data from one or more source languages in order to parse a target language.

[2]Galician is close to Portuguese but is spelled with Spanish conventions.

4.3.5 Multi-Lingual Dependency Parsing

Naseem et al. [NBG12] propose a truly multi-lingual parsing method that learns to parse several languages in the same time in order to improve over independent models. They propose to learn a generative parsing model and to tie different parts of the model between languages that share typological features. They show the benefits of learning to parse several languages from different families at the same time. However, because they use diversely annotated data, they have to map each specific POS sets to a somewhat universal POS set of only 11 tags, which is very coarse.

More recently, Ammar et at. [AMB+16] propose to directly learn a parser on several languages at once. They use information such as cross-lingual word clusters, word embedding, part-of-speech and morphological information (so called fine grained part-of-speech) embedding in order to represent their parser state in a transition based parsing setting. In order to allow the parser to learn language specific information as well, they complement state representations with language information in the shape of the language identity, encoding of typological features such as word order. Furthermore, they do not enforce morphological information to be language agnostic, so it is also a way to encode language specific information. They show the benefits of their approach on parsing 7 Romance and Germanic languages.

Both previous works [NBG12] and [AMB+16] have shown the interest of performing multi-lingual dependency parsing. However, in the case of Naseem et al. for annotation consistency, they used a very coarse set of 11 parts-of-speech, which might be too coarse to fully leverage cross-lingual information. In the case of Ammar et al., they only experiment with 7 closely related Germanic and Romance languages and so it is hard to see how there model would behave with more and more diverse languages.

Chapter 5

Delexicalised Word Representation

This chapter is based on a work presented at EACL 2017 [DD17].

As we have seen in chapter 4, to share information between languages requires a common representation space or at least a way to align representation spaces. We have also pointed out the problems posed by an over simple data representation method such as one-hot encoding, and how alternative word representations can be used instead. Given those two premises, it seems legitimate to learn a common representation space to bridge the gap between languages. Furthermore, it would also seem natural to add expert knowledge about the downstream task (dependency parsing) to our representations, which we can do since we have access to annotated data.

In this chapter we are looking at a way to learn word representations infused with syntactic information using multi-lingual information and dependency information. This will allow us to investigate two hypotheses. *(i)* Using available structure when learning representations is beneficial for dependency parsing. *(ii)* Using information from multiple languages improves the quality of those learned representations. By using the structure for learning the representation, we mean that encoding that a noun depends on a verb rather than appears next to it in the representation should help, and especially for those languages in which dependency does not equate being close in the sentence, such as Classical Latin where adjectives can appear far from their head noun for stylistic reasons. For using multiple languages, we assume that using data from grammatically close[1] languages should improve the representations, at least by increasing the amount of data used, and at best by infusing relevant information into the representation space.

This chapter is organised as follows. Section 5.1 presents some related work on representation learning for dependency parsing. Section 5.2 introduces the concept of delexicalised words. Section 5.3 describes the representation learning method and explains how we use delexicalised words and dependency information to make structured delexicalised contexts. Section 5.4 describes the dependency parsing framework and how we represents edges from delexicalised word representations. Section 5.5 describes the experimental setting and discusses the results. Eventually, section 5.6 concludes the chapter.

[1]This is a rather loose idea of grammatical proximity. Two languages can be close because of genetic relationship, areal diffusion or just because they happen to share some grammatical feature.

5.1 Related Work

This section presents some related work on representation learning for dependency parsing. Beside the first work that is seminal in using word representations for dependency parsing, they all use dependency information in a way or another to learn their representations. However, they all deal with monolingual representations only, while we investigate here both monolingual and cross-lingual representations.

Koo et al. [KCC08] were the first to use word embeddings in dependency parsing. They relied on the Brown clustering algorithm to hierarchically cluster words and then they used those clusters to extend their feature set by replacing words by their clusters id. Even though, their representation were not directly tailored for dependency parsing, they paved the way for future research on representation learning for dependency parsing.

Since then, people have looked at the problem of learning good representations for words, edges and features in the context of dependency parsing. They have proposed different methods to learn representations as well as diverse ways to incorporate those representation into their parsing algorithms. The first methods though, were all based on the same principle. As we shall see, they train simple graph-based parsers [MCP05a] extending the original feature space with some extra pre-trained representations for either words, edges or features.

Bansal et al. [BGL14] propose to change the context definition of the Skipgram model of Mikolov et al. [MCCD13] in order to tailor the learned representation specifically for dependency parsing. The idea behind the Skipgram model is to predict a word w given its context c. Whilst in the original paper of Mikolov et al. the context c is a set of words surrounding w, Bansal et al. set c to be a child of w and its label as well as the parent of w and its label. Thus inducing representations for words in dependency context. They then use the new representations as surrogate for the actual forms to complement the original MST features from [MCP05a].

Bansal [Ban15] also proposes a method to embed edges directly instead of words. It is again a modification of the Skipgram model, but this time the embedded token is not a word anymore but an edge e. The context c is now defined as the label of e and its signed length[2] as well as the label of the head of e and the signed distance to its own head. The learned continuous representations for edges are then discretised with an approach similar to the Brown algorithm in order to be used as features in complement to the traditional MST features from [MCP05a].

Chen et al. [CZZ14] propose a very similar method. They also base their model on the model of Mikolov et al. but instead of embedding words or edges, they embed features. Given a feature template f of the kind presented in Table 5.5, an annotated sentence x and an edge e_{ij}, they set the context of $f(x, e_{ij})$ to be $f(x, e_{\bullet i})$ and $f(x, e_{j\bullet})$ where $e_{\bullet i}$ is the parent edge of e_{ij} and $e_{j\bullet}$ is any dependent edge of e_{ij}. Once learned, they use those feature representations in complement to MST features, just as Bansal.

Kiperwasser and Goldberg [KG15] propose another method to edge embedding.

[2]Because edges are directed, a relation going from left to right is not the same a relation going from right to left. Thus the length of an edge is the distance between its ends in number of words in between in the sentence and the sign states if the edge goes forward or backward in the sentences.

But instead of learning a dense representation for their edges, they estimate an association score between dependents and their governors. Then they learn word embeddings that have the property that the dot product between a dependent's representation and its governor's representation return their association score, thus allowing them to compute the association score for any word pair. Then they use those scores in conjunction with other information like the signed distance of the relation and the parts-of-speech of its words to make new features to complement MST features.

The four works, of Bansal et al. [BGL14], Bansal [Ban15], Chen et al. [CZZ14] and Kiperwasser and Goldberg [KG15], all share that they use auto-parsed data in order to have access to a bigger quantity of dependency information than just the one from the annotated corpus. They use their respective baseline models to parse the BLLIP corpus[3] that they then use as a source of dependency information. Our work departs from theirs as we only use the available annotated data to induce our representations. This is indeed an important difference. By sticking to the annotated data, we have a warranty on the quality of the information we use, but we restrict ourselves to a small amount of data. By using auto-parsed data, they have access to much more information (up to 3 orders of magnitude more), but this information is only as good as the parser it has been annotated with, thus making it intrinsically noisy. Furthermore, because they use their baseline parsers to parse the extra data with which they induce representations, those representations embed something of the parser already. Namely, if the parser makes a recurrent mistake, the induced representation will be biased toward that mistake and might not help solving it. The question of the trade-off between data quantity and data quality and bias is still open, and we choose here few high quality data.

Posterior to our work, following more recent trends, Dozat et al. [DQM17] use heavy neural machinery for edge scoring in their graph-based parser entry to CONLL 2017 dependency parsing shared task [ZPS+17]. They use a bidirectional LSTM[4] (Long Short Term Memory) network to represent each word in a sentence. As input to their LSTM, they use several pre-trained embeddings of the word forms (Word2vec) and of the parts-of-speech as well as character based word representations. Even though their embeddings are not pre-trained specifically for dependency parsing, the end-to-end training of their deep parser automatically tune the output of the LSTM for dependency parsing. Thus effectively providing them with contextual representation tailored for dependency parsing.

Most similar to our work, also an entry to CONLL 2017 shared task, Kanerva et al. [KLG17] propose a method to induce word embeddings using both delexicalised contexts and dependency based contexts. Their model is slightly different from ours though, as they are interested in learning representations for words and not for morphological attribute sets (delexicalised words). Furthermore, as they do transition based parsing, they also use parsing transitions as contexts.

Our work is similar to those in using dependency information (indirectly for Dozat et al. and directly for the other five) to learn representations suited for dependency parsing. However, we depart from them in that we learn representa-

[3]The BLLIP corpus is a corpus of American English news that contains 30 millions words. In comparison, the Penn Treebank, which is a subset of the BLLIP annotated with syntactic structures contains about a million words.

[4]An LSTM is a kind of recurrent neural network. They are especially suited to representing sequences of varying lengths such as sentences and words.

Øll	menniskju	eru	fødd	fræls	og	jøvn	til	virðingar	og	mannarættindi.
Alle	mennesker	er	født	frie	og	lige	i	værdighed	og	rettigheder.
Hver	maður	er	borinn	frjáls	og	jafn	öðrum að	virðingu	og	réttindum.

Table 5.1: The first sentence of the first article from the Universal Declaration of Human Rights in Faroese, Danish and Icelandic. Words are aligned with their translations, often cognates.

tions for delexicalised words, which allows us to use multi-lingual data in a quasi straightforward manner.

5.2 Delexicalised Words

An important question that arises when dealing with word representations is that of the definition of a word. Depending on the answer, the word representation might be radically different. A word can be seen as an indivisible unit with its own form and sense which leads to representations of the kind of Word2vec [MSC+13] where each form has its own representation. A word can also be seen as a sequence of morphemes in which case approaches such as the one of Botha and Blunsom [BB14] that compose morphemes representations into word representations are available. If a word is merely a sequence of letters, this leads to character level language models such as [BGJM16] where they use recurrent neural network (RNN) to compose character representations into words. A word can also be seen as a lemma inflected for several morphological attributes, this is the case in the work of Avraham and Goldberg [AG17]. Our delexicalised word embeddings fall under the same view of words as inflected lemmas.

The main barrier for working with several languages is the differences in word forms. This is even more true for closely related languages where the grammar is often very similar and the lexicon is parallel, but the actual word forms can vary a lot. This is really clear when comparing the three sentences in Table 5.1. There are the first sentence of the first article of the Universal Declaration of Human Rights in Faroese, Icelandic and Danish, all three north Germanic languages. The sentences are almost completely parallel and words can be identified to each other (most are cognates) but except for *og* (and) and *er* (are), they all have different forms. This is indeed a strong argument against form embeddings as they are performed nowadays as in a multi-lingual setting they would fail completely to encode any useful information.

The idea behind delexicalised word embeddings is that assuming we have access to the morphological analysis of word forms, we could have a language agnostic representation of our words by removing the lemmas. Moreover, as previous works on delexicalised dependency parsing have shown, lexical information is not crucial in achieving good syntactic analysis [MPH11].

This might seem strong of an assumption, but with the advances in tagging methods and with the ever growing amount of annotated data, we have access nowadays to accurate morphological taggers for the most widely spoken languages. For lesser spoken/studied languages however it is a different problem. But in general, morphological analysis or at least part-of-speech tagging, is coming before[5]

[5]Sometimes some syncretic forms need dependency information to be fully disambiguated, appealing to joint parsing and tagging. But they are rather seldom and in general tagging is still seen as a preprocessing step to parsing.

Form	Lemma	UPOS	Attributes
øll	allur	DET	Case=Nom|Definite=Ind|Gender=Neut|Number=Plur
alle	al	ADJ	Degree=Pos|Number=Plur
menniskju	menniskja	NOUN	Case=Nom|Definite=Ind|Gender=Neut|Number=Plur
mennesker	menneske	NOUN	Definite=Ind|Gender=Neut|Number=Plur
eru	vera	VERB	Mood=Ind|Number=Plur|Tense=Pres
er	være	AUX	Mood=Ind|Tense=Pres|VerbForm=Fin|Voice=Act
fødd	føða	VERB	Case=Nom|Definite=Ind|Gender=Neut|Number=Plur
født	føde	VERB	Definite=Ind|Number=Plur|Tense=Past|VerbForm=Part
fræls	frælsur	ADJ	Case=Nom|Definite=Ind|Gender=Neut|Number=Plur
frie	fri	ADJ	Definite=Ind|Degree=Pos|Gender=Neut|Number=Plur
og	og	CCONJ	_

Table 5.2: Lemma, parts-of-speech and morphological attributes for the five words of the first sentence of the first article from the Universal Declaration of Human Rights in Danish and Faroese. Most cognate words have the same UPOS and a similar morphological analysis. The two last columns make up the delexicalised words.

dependency parsing. Thus, if even tagging resources are not available, it may be unwise to try to parse a language. Indeed some people have tried to do parsing without gold POS information [SACJ11], and it should be possible given enough data and a proper word representation. But it requires a lot of data which in general is not available for those lesser spoken/studied languages. Thus we make the assumption that morphological information is available at parsing time.

We call *delexicalised word* a part-of-speech adjoined with a (possibly empty) set of morphological attributes. For example looking at Table 5.2, the delexicalised words would correspond to the combination of the two last columns. Thus the delexicalised version of *øll* would be:

DET, {Case=Nom, Definite=Ind, Gender=Neut, Number=Plur},

meaning that *øll* is a determiner. As a determiner it is indefinite and it has the form of a nominative plural neuter. Likewise, the delexicalised version of *og* would simply be CCONJ, {} as it is a coordinating conjunction and does not inflect further.

Table 5.2 shows lemmas, parts-of-speech and morphological attributes for a few Danish and Faroese words from Table 5.1. We see that, whilst most of the lemmas (and forms) are different, their parts-of-speech and morphological analysis are very similar. Furthermore, beside for cases and gender that are genuinely different between Danish and Faroese, the differences are mostly annotation artifacts, such as the auxiliary/verb treatment of the copula, as mentioned in the section 4.2 about the Universal Dependencies project. Incidentally, there is also a lot of overlap between the related forms and lemmas, which might seem a good argument in favour of character based representations as they would factor out the differences and focus on the similarities. However, there are languages grammatically close enough to help each other but that use different writing systems thus preventing a direct use of character representations. Table 5.3 gives the same first sentence of the Universal Declaration of Human Right, in Czech and Russian this time. They are once again very similar but they are written with different alphabets. This situation is less seldom that might seem, with many south Asian languages being related but using their own writing system. The most striking example being Hindi-Urdu, in which case, previous to India-Pakistan split, was a single language with two different spelling conventions. While translitteration can seem appealing,

Všichni	lidé	rodí	se	svobodní	a	sobě	rovní	co do		důstojnosti	a práv.
Все	люди	рождаются		свободными	и		равными	в	своем	достоинстве	и правах.

Table 5.3: The first sentence of the first article from the Universal Declaration of Human Rights in Czech and Russian. Even though the two versions are slightly different, we can still see the similarities.

Je	mange	du	fromage	avec	des	raisins.
(Yo)	como		queso	con		uvas.

Table 5.4: The sentence *I eat some cheese with grapes* in French and Spanish. Translations are aligned, but in this case are not cognates at all, except for the pronouns which is optional in Spanish.

it is not always an option, especially when one language is using a logographic system. There are also cases where because of borrowing or semantic shift, in very close languages translations are not cognates and character based representations do not help. Table 5.4 gives a more mundane example in French and Spanish where despite the sentences being parallel, words are completely different. Those examples are all arguments in favour of delexicalisation.

Obviously, many words will collapse to the same delexicalised version. Depending on the productivity of the inflectional system of each language, the number of such delexicalised words will range between the low hundred to several thousands. Table 5.6 in the experiment section below reports the number of delexicalised words for the datasets we used. On the lower end, English has 118 delexicalised words, while on the higher end, Finnish has 1592 delexicalised words (14 times more).

5.3 Representation Learning

In principle, we could use one-hot encoded delexicalised words in our parsers. Because there are much less delexicalised words than their lexicalised counterpart, results would most likely increase thanks to sparsity reduction. But we would still loose some important information on the way. For example, tokens only differing in gender or number like NOUN, {Gender=Fem, Number=Sing} and NOUN, {Gender=Neut, Number=Sing} would still be treated completely independently. To avoid this, we will learn dense representations for those delexicalised words. This will be done by factorising a co-occurrence matrix based on some structured delexicalised contexts that we define just below.

5.3.1 Delexicalised Contexts

Delexicalised words can be seen as yet another vocabulary alongside plain word forms, lemmas, part-of-speech and so on. Thus, they can be used both as token to be embedded and as context tokens. But there are a few things on which to be careful. First, contexts are most of the time lexicalised, thus giving matrices with tens of thousands of dimensions while there are sometimes only a few hundreds (or less) delexicalised words. The co-occurrence distributions over delexicalised contexts will thus likely be very different from the one with lexicalised contexts. One might need to use different re-weighting schemes to treat the matrix for example.

The second point has to do with cross-lingual delexicalised words. We want to learn delexicalised representation in a multi-lingual setting, hoping that by sharing information between languages, their own representations will be better. As we have mentioned earlier, the biggest barrier to multi-lingual NLP is cross-lingual word identification (knowing that *car*, *voiture* and *bil* are the same thing). Delexicalised representations avoid this problem by sticking to morphological attributes that are more easily identified across languages. Even though getting rid of lemmas is first step toward cross-lingual representations and works well for some languages with similar grammar, sometimes it is not enough. Two very close languages might have grammars different enough so that delexicalised words from one are highly unlikely to appear in the other.

Looking back at Table 5.1 and 5.2, Danish and Icelandic and Faroese have very similar grammar but Faroese and Icelandic have declensions while Danish does not. Faroese/Icelandic nouns, pronouns and adjectives fully inflect for case and gender in addition to number and definiteness also present in other north Germanic languages. This means that despite being similar, Icelandic nouns and Danish ones will hardly ever collapse to the same delexicalised word. Whilst this might not be a problem for tokens to be embedded – we want to keep the information that a nominative is different from an accusative – this is a problem for contexts because for languages to share information, they need to share contexts. It is all the worse as the contexts most likely to be shared are those least informative contexts such as conjunctions and punctuations (*og* in Table 5.2).

One way to increase the number of shared contexts and thus to share more information between languages is to truncate the delexicalised words used as contexts to make them less specific. For example, keeping only two attributes (on top of the POS) instead of four would turn the delexicalised version of Faroese *øll* from:

DET, {Case=Nom, Definite=Ind, Gender=Neut, Number=Plur},

that is never met in Danish, to the six shorter:

DET, {Case=Nom, Definite=Ind}, DET, {Case=Nom, Gender=Neut},
DET, {Case=Nom, Number=Plur}, DET, {Definite=Ind, Gender=Neut},
DET, {Definite=Ind, Number=Plur}, DET, {Gender=Neut, Number=Plur},

out of which three are also present in Danish. This way, information will flow more easily between languages.

Another possibility is to use directly single POS or single attributes as contexts. It might make sense for parts-of-speech, despite their impoverished information content with regard to full delexicalised words. Indeed, some words do not inflect and some languages do not rely on morphology at all, so bare POS is the best we can have in those cases. We investigate it in the experiment section. However, it is harder to stand for attributes as because of agreement and government, the same attributes can be borne by different words for different reasons thus encoding different information in different contexts. For example in many languages, the plural number can be marked on verbs, nouns, adjectives and determiners, for very different reasons though. In nouns, the number is a semantic feature chosen by the speaker or sometimes enforced by the language (*trousers* is always a plural in English). However, determiners and adjectives agree in number with the noun that governs them, and verbs agree in number with the noun they govern[6]. Similarly,

[6]In most European languages, verbs do agree with their subject, but there are languages where verbs agree with their object or both the subject and object.

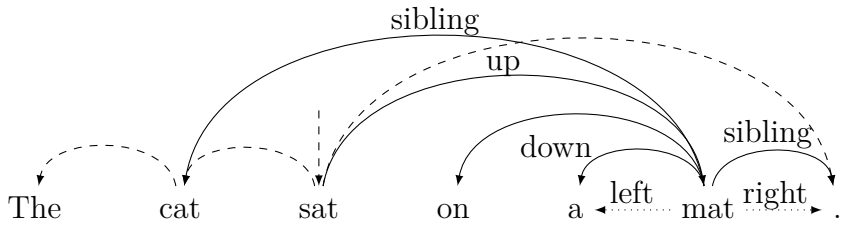


Figure 5.1: Examples of contexts on a dependency structure. Dotted lines represent sequential contexts (*right* and $left$) context. Plain lines represent dependency structured contexts (parent (*up*), dependent (*down*) and composed sibling). Dashed lines represent the rest of the dependency structure.

in languages that mark cases on both nouns and adjectives, adjectives agree in case with their governing noun, while noun cases are governed by the syntactic relation they bare with their governing verb (subject, object or any other).

However, it is important to note that context truncation might unbalance co-occurrence counts. For a word appearing in the context of *øll*, if we keep the full attribute set, *øll* will trigger once, but if we consider subsets of two attributes instead, it will trigger six times. Suddenly, a word with many attributes will have a bigger weight than a word with less attributes. Similarly a language with many attributes will have a bigger weight than a language with less attributes.

We also add special tokens such as *begin* and *end* to represent the word before the beginning and the end of a sentence. We have a *root* token that stands for the extra root node added by the graphical dependency model.

Now that we are equipped with delexicalised words that allow information to be shared between languages, we need to provide a typology for those contexts and we will use dependency information.

5.3.2 Structured Contexts

Truncated delexicalised words are only the vocabulary part of contexts. We also need to choose some topology for those context. By topology we mean the relative positions of the contexts and the tokens to be embedded. The most common contexts are windowed contexts centered around the tokens to be embedded. Small windows have been shown to give syntactic flavour to the resulting representations [BGL14], and that is exactly what we need. To give it even more syntactic knowledge, we can distinguish left context from right context, thus effectively differentiating between words that appeared before and those that appear after.

As we want to learn representations tailored for dependency parsing, it seems natural to include some dependency information as well in the contexts, like which word depends on which and the like. It turns out we can extend the concept of windowed context from sequences to arbitrary graphs. If we see a sequence of words (a sentence) as a chain graph, traditional windowed contexts can be defined as words appearing k edges remote from the center word of the window.

We now define some context functions. Let x be a sentence of length $|x| = n$ and let y be a dependency tree over x. Let $right_x(i)$ (resp. $left_x(i)$) be the word just to the right (resp. to the left) of x_i in x. We pad the sentence with extra safety words so that $left_x(0) = BEG$ and $right_x(n-1) = END$. Let $up_y(i)$ be the governor of x_i in y. We also add a special symbol here so that $up_y(r) = NIL$, where r is the root of y. Finally, let $down_y(i)$ be the (eventually empty) set of

dependents of x_i in y. Those functions can be composed and/or intersected to create more complex contexts.

Traditional window contexts are only made of $right_x$ and $left_x$. Using up_y and $down_y$, one can define context such as parent (up_y), child ($down_y$), siblings ($down_y \circ up_y$), grand-parent ($up_y \circ up_y$) and so on. We call the span of a context the number of functions that are composed to define it. The parent context is a context of span 1 because it is the result of applying once the up_y function. The grand-parent context has span 2 because it is the result of composing two up_y. Likewise, the sibling context is also of span 2 because it is the result of composing one up_y with one $down_y$. Similarly, the second word to the right is also of span 2 because it is the result of applying two $right_x$.

Let f^+ note the transitive closure of a function f, so that $right_x^+(i)$ for example is the set of words in x appearing to the right of x_i no matter how far. Then we can define the right siblings of word x_i as $rightsib_y(i) = down_y \circ up_y(i) \cap right_x^+(i)$. Similarly we can define left siblings, right children and so on.

Figure 5.1 shows some structured context for the word *mat* on our running dependency example. The parent context (one application of up_y) of *mat* is *sat*. The delexicalised form of *mat* is NOUN, {Number=Sing}. The delexicalised form of *sat* is VERB, {Mood=Ind, Tense=Past, VerbForm=Fin}. If we consider morphological attribute sets of length 2 for our delexicalised context, then NOUN, {Number=Sing} would trigger VERB, {Mood=Ind, Tense=Past}, VERB, {Mood=Ind, VerbForm=Fin} and VERB, {Tense=Past, VerbForm=Fin} for its parent context.

With truncated delexicalised words and dependency based geometry, we can make structured delexicalised contexts that will allow to share syntactic information between languages.

5.3.3 Structured Delexicalised Contexts

Before seeing how we actually induce representations for delexicalised words, we shall say a few words about the interaction between the vocabulary and the topology of the contexts (in our case, the interaction between delexicalised contexts and structured contexts). More precisely, we shall see why it might be useful to have different sizes of delexicalised words for different context types.

As Bansal et al. [BGL14] have shown, narrow windowed contexts give a syntactic flavour to the representations the are used to learn. This is expected, especially for languages with a strict word order as most of the syntactic information is analysable from neighbouring words. For example, in English or French or other languages with strict word order, if in a sentence one finds a determiner (such as *the*) and the second next word is a noun (say *car*), then one can easily deduce that the word in between is to be an adjective (including participles) (say *red*) or a (proper) noun but not a finite verb for example. On the other hand, wide windowed context are less syntactically informative. In an English sentence, what can one say of a word knowing that the fifth word to the right is a pronoun? Nothing. The word coming just after a determiner is rather well defined, but five words from it, we could be in a different clause, in a different sentence even.

This is just as true of dependency structures. Most syntactic information encoded in the syntactic structure can be retrieved from the governor and the dependents of a word. Indeed, in most languages that show some sort of morphological agreement or government, it happens between a word and its direct

governor or dependents[7]. For example, in Latin or German, where possession is expressed morphologically by the genitive case, the genitive is used for possessors irrespective of the syntactic role and case marking of the possessed word. Compare *librum fratris mei lego* (I read my brother's book) and *liber fratris mei magnus est* (my brother's book is big), the genitive *fratris mei* does not change whether *liber* appears in nominative (subject) or in accusative (object) *librum*.

Those remarks argue in favour of giving more weight to the syntactic information of directly neighbouring words and less to more remote words in the sentence or in the dependency graph. A way to do achieve this is to allow different length of delexicalised words for different context typologies. For example, in Figure 5.1, the word *a* depends on *mat* which depends itself on *sat*. The delexicalised word for *a* could trigger only the bare POS of *sat* instead of all the attribute subsets that *mat* would trigger.

Thus to fully specify a context, one needs to give a vocabulary, some context topology, and some composition rules that tell which vocabulary token is to appear in which context position. In our case, the vocabulary would be delexicalised words and their truncation, the topology is made of windows in the sequence and windows in the dependency structure, and the rules could specify that only bare POS are to be used for context of distance greater than 2.

5.3.4 Dimension Reduction

Given a vocabulary to embed, a vocabulary and a geometry for the context, and a corpus annotated for dependency structure and morphological attributes, one can gather a delexicalised words co-occurrence matrix. But as we mentioned earlier, this matrix is far too big to be usable in practice, thus we perform some dimension reduction to have lighter representations. Whilst there are many different dimension reduction techniques available, we have chosen principal component analysis (PCA) for it is both simple and has been shown to give similar results as more involved methods such as Word2vec [LG14].

Let $M \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{C}|}$ where $\mathcal{V}$ is the vocabulary of delexicalised words and $\mathcal{C}$ the set of delexicalised contexts, be the co-occurrence matrix such that M_{ij} is the number of times token $\mathcal{V}_i$ appears in context $\mathcal{C}_j$ in a corpus T. Let $M^* \in \mathbb{N}^{|\mathcal{V}^*|\times|\mathcal{C}|}$ where $\mathcal{V}^*$ is the vocabulary of part-of-speech, be another co-occurrence matrix such that M^*_{ij} is the number of times POS $\mathcal{V}^*_i$ appears in context $\mathcal{C}_j$ in corpus T. Because of the Zipfian nature of languages, many words will only appear a small number of times. Because PCA is sensitive to those words and to have a more robust projection, we discard delexicalised words that appear less often than a threshold η. We do the same for contexts. By removing words and contexts repeatedly we could end up removing more than necessary, so in practice we only remove words and contexts that are too rare once. The matrix M_η results from removing rows and columns from M that sums up to less than η. Columns removed from M are also removed from M^* giving M^*_η.

Let $M' \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{C}|}$ be the re-weighted version of M_η such that $M'_i = \frac{M_{\eta i}}{\|M_{\eta i}\|^2}$,

[7]There exist indeed a few languages where case marking stacks up. Where the possessor of an object is marked with both accusative and genitive while the possessor of a subject is marked with nominative and genitive for example. But this is a rather marginal phenomenon across languages and thus we disregard it here. For more information about stacking of case markers and how syntactic information can propagate further than dependents, see the book by Blake [Bla01].

where $M_{\eta i}$ is the i-th row of M_η. We re-weight M^* in the same way to have M'^*. Then PCA is applied to M' as follows :

$$M'^\top M' = UV^2U^\top,$$

where $M'^\top M'$ is the co-variance matrix of the contexts, $U \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is the matrix of eigenvectors of the co-variance matrix and $V \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is the diagonal matrix of eigenvalues of the co-variance matrix. The principle of PCA is to project M' onto the subspace spanned by it principal eigenvectors, also called principal components.

$$R = M'U_d^\top,$$

where U_d is the matrix of the d first eigenvectors from U sorted by decreasing order of their associated eigenvalue. The resulting matrix R has size $|\mathcal{V}| \times d$, where d is the dimension of the induced representation.

The PCA is also interesting as it can be seen as inducing context representations (the U_d matrix). One can then use this context matrix to induce representation for tokens whose count vectors where kept out of the M' matrix used to compute the PCA. We use this to compute a representation for the bare parts-of-speech:

$$R^* = M'^*U_d^\top,$$

that will be used as back-offs for delexicalised words that were either unseen or discarded previous to normalisation. The whole embedding process is summed up in Algorithm 7.

The PCA applied to co-occurrence matrix of delexicalised words and structured delexicalised contexts, leaves us with dense representations of delexicalised words that contain syntactic information. This dense representation clashes with traditional sparse feature vectors. For this reason, some people like Chen et al. [CZZ14] have tried to sparsify those representations, using either clustering algorithms or binning techniques. However, it seems surprising to turn a sparse representation (a co-occurrence matrix) into another sparse representation (e.g. a cluster id) via a dense representation, thus loosing part of the information held by the intermediary dense representation. Thus, we keep the dense representation and pay attention not to let the memory consumption of our method overblow, as we shall see in next section.

5.4 Dependency Parsing with Delexicalised Word

Following the same approach as Kiperwasser and Goldberg [KG15], Chen et al.[CZZ14] and Bansal [Ban15], we use the learned morphological representation to complement the traditional sparse features from McDonald [MCP05a]. Those features encode information about the form and part-of-speech of the two ends of the dependency relation, plus the parts-of-speech of surrounding words and in between words as well as the signed length of the relation. In the original paper, McDonald et al. use the prefix consisting of the 5 first characters of a word as a back-off for unseen words. Instead, we use the lemma as provided by the treebanks for our experiments. The detailed feature templates are repeated in Table 5.5 with the necessary corrections.

Once the structured delexicalised co-occurrence matrix M has been reduced to a new representation matrix R for delexicalised words, we need to turn those

Data: a set of examples $\mathcal{D}$, some context typology C,
three integers l, d, η
Result: embeddings E and E^* for delexicalised words and bare POS
begin
Collect all the delexicalised words, POS and delexicalised contexts,
$\mathcal{V} = \{(pos(w) : morph(w)) \mid w \in x, x \in \mathcal{D}\}$
$\mathcal{V}^* = \{pos(w) \mid w \in x, x \in \mathcal{D}\}$
$\mathcal{C} = \{(c, p : \sigma) \mid (p, \mu) \in \mathcal{V}, \sigma \in subset(\mu, l), c \in C\}$

Fill co-occurrence matrices for delexicalised words and POS,
Instantiate $M \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{C}|}$, $M^* \in \mathbb{N}^{|\mathcal{V}^*|\times|\mathcal{C}|}$
$M = 0, \quad M^* = 0,$
for $x \in \mathcal{D}$ **do**
  for $w \in x$ **do**
    $i \leftarrow \mathcal{V}.index((pos(w) : morph(w)))$
    $i^* \leftarrow \mathcal{V}^*.index(pos(w))$
    for j, *w appears in context* $\mathcal{C}_j$ **do**
      M_{ij} += 1
      $M^*_{i^*j}$ += 1

Discard rows and columns with to low counts and normalise rows,
Instantiate $S_{\mathcal{V}} \in \mathbb{N}^{|\mathcal{V}|}$, $S_{\mathcal{C}} \in \mathbb{N}^{|\mathcal{C}|}$
$S_{\mathcal{V}i} = \sum_j M_{ij}, \ \forall i$
$S_{\mathcal{C}j} = \sum_i M_{ij}, \ \forall j$
for $j \in [0..|\mathcal{C}|]$ **do**
  if $S_{\mathcal{C}j} < \eta$ **then**
    Delete column $M_{\cdot j}$
    Delete column $M^*_{\cdot j}$

for $i \in [0..|\mathcal{V}|]$ **do**
  if $S_{\mathcal{V}i} < \eta$ **then**
    Delete row $M_{i\cdot}$
  else
    $M_{i\cdot} = \frac{M_{i\cdot}}{\|M_{i\cdot}\|^2}$
for $i^* \in [0..|\mathcal{V}^*|]$ **do**
  $M^*_{i^*\cdot} = \frac{M^*_{i^*\cdot}}{\|M^*_{i^*\cdot}\|^2}$

Dimension reduction via PCA,
$U, V = PCA(M), \qquad (M^\top M = UV^2U^\top)$
$E = MU_d^\top, \ \ E^* = M^*U_d^\top$
return $E, \ E^*$

Algorithm 7: The embedding process for delexicalised words using structured contexts. The three integers are the maximum length of a delexicalised attribute set l, the counting threshold η and the desired number of embedding dimensions d. $pos(w)$ is the part-of-speech of word w and $morph(w)$ its morphological attribute set. $subset(S, l)$ is the set of subsets of S of length l or less.

Uni-gram
H-form, H-pos
H-form
H-lemma, H-pos
H-lemma
H-pos
D-form, D-pos
D-form
D-lemma, D-pos
D-lemma
D-pos

Bi-gram
H-form, H-pos, D-form, D-pos
H-form, H-pos, D-form
H-form, H-pos, D-pos
H-form, D-form, D-pos
H-pos, D-form, D-pos
H-form, D-form
H-lemma, H-pos, D-lemma, D-pos
H-lemma, H-pos, D-lemma
H-lemma, H-pos, D-pos
H-lemma, D-lemma, D-pos
H-pos, D-lemma, D-pos
H-lemma, D-lemma
H-pos, D-pos

Tri-gram
H-1-pos, H-pos, D-pos
H-pos, H+1-pos, D-pos
H-pos, D-1-pos, D-pos
H-pos, D-pos, D+1-pos

Tetra-gram
H-1-pos, H-pos, D-1-pos, D-pos
H-pos, H+1-pos, D-1-pos, D-pos
H-1-pos, H-pos, D-pos, D+1-pos
H-pos, H+1-pos, D-pos, D+1-pos

In Between
H-pos, B-pos, D-pos

Table 5.5: Feature templates for the one-hot edge feature vector for dependency parsing adaptated from [MCP05a]. H stands for the head of a relation, D for the dependent, ±1 selects the word before/after the given word. B is any word in between the Head and the Dependent of the relation. We use normal word forms and parts-of-speech but contrary to McDonald et al. citemcdonald2005spanning, we use lemmas as back-off instead instead of truncated forms. All those templates are further completed with the binned signed length of the relation.

representation into feature vectors in order to score edges. Because we are dealing with dense vectors, we have to be careful about the size of the final edge representation. Big dense vectors take a more of memory to store and more time to process than their sparse counterparts. In order to have both contextual information, and governor-dependent interaction whilst keeping the size of the feature vectors reasonable, we use the outer product of the governor centered trigram concatenated vectors with the dependent centered trigram. On the one hand, if we used only concatenation of the different words representations, the model would not learn about their interaction but only how likely is a word to be governor and how likely it is to be dependent. But on the other hand, using outer products to represent the complete edge context interaction would blow the memory up.

More formally, let $\oplus$ note vector concatenation and $\otimes$ vector outer product. Let vec($\bullet$) be the vectorisation operator that turn a matrix of size $p \times q$ into a vector of length pq. Given a sentence x and an edge e_{ij} whose governor is x_i and whose dependent is x_j. Let $\mathbf{x}_i = R_{x_i}$ be the representation of word x_i. The embedding based feature vector $\boldsymbol{\phi}_R$ is defined as:

$$\boldsymbol{\phi}_R(e_{ij}) = \text{vec}[(\mathbf{x}_{i-1} \oplus \mathbf{x}_i \oplus \mathbf{x}_{i+1}) \otimes (\mathbf{x}_{j-1} \oplus \mathbf{x}_j \oplus \mathbf{x}_{j+1})].$$

Thus $\boldsymbol{\phi}_R$ is of length $9d^2$ where d is the length the embedding vectors of R. If we had used the outer product of those 6 word representations instead, we would have a length of d^6. With word representations of 50 dimensions, we would go from 22.5×10^3 to 15.625×10^9, which in 32 bits precision makes 500Gb (against 720kb) and does not fit in most computers RAM.

The score of an edge is then computed as:

$$score_\theta(x, e_{ij}) = \boldsymbol{\theta} \cdot [\boldsymbol{\phi}(e_{ij}) \oplus \alpha\boldsymbol{\phi}_R(e_{ij})]\,. \tag{5.1}$$

Where α is a scalar parameter allowing to tune the relative importance of each part of the compound feature vector, $\boldsymbol{\phi}$ is the traditional MST feature vector and $\boldsymbol{\phi}_R$ is our augmentation defined in term of delexicalised word representations.

Edges are equipped with feature vectors based on both a traditional sparse part and a dense morphological part. The model vector $\boldsymbol{\theta}$ is then learned by the PA-II algorithm [CDK+06], and the trees are retrieved by the Eisner algorithm [Eis96]. The whole parsing process is summed up in Algorithm 8.

5.5 Experiments

5.5.1 Settings

Experiments Description

We have carried out two experiments: monolingual and cross-lingual to test our hypotheses. To assess the usefulness of using the dependency structure for learning the representations, in the first experiment, for each language we compare the results of parsers using embeddings induced on the language own training set with varying embedding typologies and sizes.

To test the relevance of cross-lingual information for learning the representations, in the second experiment, we have defined several clusters of languages based on their phylogenetic relationship and typological similarities and used them as source data for the embedding induction. For a given cluster, embeddings are

Data: a set of example $\mathcal{D}$, an integer k, two floats C and α,
a feature function $\boldsymbol{\phi}(\bullet)$, embeddings R and R^*
Result: a weight vector $\boldsymbol{\theta}$
begin
  $d = dim(\boldsymbol{\phi}(\bullet)) + 9\ dim(E)^2$
  Instantiate $\boldsymbol{\theta}_0 \in \mathbb{R}^d$

  for $it \in [0..k]$ **do**
    for $(x, y) \in \mathcal{D}$ **do**
      $l = len(x)$
      Instantiate $\mathcal{W} \in \mathbb{R}^{(l+1)\times l}$
      foreach *edge* $e_{ij},\ i \in [0..l],\ j \in [1..l]$ *over* x **do**
        $\boldsymbol{\phi}_R(e_{ij}, x) = \text{vec}[(\mathbf{x}_{i-1} \oplus \mathbf{x}_i \oplus \mathbf{x}_{i+1}) \otimes (\mathbf{x}_{j-1} \oplus \mathbf{x}_j \oplus \mathbf{x}_{j+1})]$
        $\mathcal{W}_{ij} = \boldsymbol{\theta}_t \cdot [\boldsymbol{\phi}(e_{ij}) \oplus \alpha\boldsymbol{\phi}_E(e_{ij})]$

      $\hat{y} = \text{Eisner}(\mathcal{W})$

      if $\boldsymbol{\theta}_t \cdot [\boldsymbol{\Phi}(\hat{y}) - \boldsymbol{\Phi}(y)] + 1 > 0$ **then**
        $\tau = \frac{\boldsymbol{\theta}_t \cdot [\boldsymbol{\Phi}(\hat{y}) - \boldsymbol{\Phi}(y)] + 1}{\|\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y}))\|^2 + \frac{1}{2C}}$
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \tau\,[\boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y})]$
  return $\boldsymbol{\theta}_t$

Algorithm 8: Online training process of a linear dependency parser using delexicalised words representations to complement traditional edge features. The update rule used is the one of the Passive-Aggressive II algorithm.

	Train		Test			
	sentences	words	sentences	words	delexicalised words	POS
English	12 543	204 586	2 077	25 096	118	17
Basque	5 396	72 974	1 799	24 374	845	16
Finnish	12 217	162 721	648	9 140	1 592	15
French	14 557	356 216	298	7 018	195	17
Gothic	4 360	44 722	485	5 158	662	13
Hebrew	5 142	15 558	491	12 125	480	16
Hungarian	1 433	23 020	188	4 235	651	16
Romanian	4 759	108 618	794	18 375	412	17

Table 5.6: Number of sentences and words in the training and test sets, number of delexicalised word and of POS-tags for each language. The total number or embedded tokens is |morpho-syntactic feature set| + $|POS|$ + 3 because of the POS back-offs and the special *begin*, *end* and *root* tokens.

learned on the union of the training sets of each language in that cluster, and in turn used to parse each language in that cluster. The exact clusters identity are given in Table 5.9. In principle, it is possible not to use any data from the target language when learning the embeddings, but in this study we stick to using target language data. It is worth noting here that in this setting, the actual parsers are trained on monolingual data, only the delexicalised representations are trained on multi-lingual data.

For both experiments, the baseline is a simple parser that does not use any extra representation to complement the tradition sparse feature vector.

Dataset

We have tested our parsing models based on delexicalised word embeddings on eight languages from the Universal Dependencies project (UD) v1.3 [NdMG+16]. We have chosen to work on English (en), Basque (eu), Finnish (fi), French (fr), Gothic (got), Hebrew (he), Hungarian (hu) and Romanian (ro). These languages belong to four different families, which are Indo-European (en, fr, got, ro), Finno-Ugric (fi, hu), Semitic (he), and Basque (eu) which forms a separate group. They also display various levels of morphological complexity not correlated with the families (English, French and Hebrew do not have case marking in nouns while the other five do to various degrees) as well as different grammatical typologies (Basque is an ergative[8] language, while the other seven are accusative ones). When several corpora are available for a language, we picked the canonical one. Table 5.6 provides some basic statistics on the language datasets. Our experiments follow the train/dev/test split as provided by the treebanks.

Embedding Contexts

For the embedding contexts, we consider four parameters, namely the typology and span of contexts, the maximum length for morphological attributes set truncation

[8]Ergative and accusative are two syntactic alignment types. While in accusative languages like French and English, we treat subjects of intransitive verbs (*Peter falls*) the same way as subjects of transitive verbs (*Peter eats pastas*), ergative languages treats them like objects of transitive verbs (*Peter eats pastas*). This leads to differences in case marking and conjugation when they are used.

	Contexts
Seq 1	left_0, right_0
Seq 2	left_0, right_0, second-left_0, second-right_0
Seq 3	left_*, right_*, second-left_0, second-right_0
Struct 1	parent_0, child_0
Struct 2	parent_0, child_0, grand-parent_0, grand-child_0, sibling_0
Struct 3	parent_*, child_*, grand-parent_0, grand-child_0, sibling_0
Mix 1	left_0, right_0, parent_0, child_0
Mix 2	left_0, right_0, parent_0, child_0, second-left_0, second-right_0, grand-parent_0, grand-child_0, sibling_0
Mix 3	left_*, right_*, parent_*, child_*, second-left_0, second-right_0, grand-parent_0, grand-child_0, sibling_0

Table 5.7: Details of the embedding contexts. The relations are defined as describre in section 4.3.2. The subscript represents the lengths of the attribute sets in delexicalized contexts, 0 stands for bare POS and $*$ for all the possible subsets.

used in those contexts and the dimension of the embedding space. Regarding the typology of contexts we have experiments with three settings: (*i*) strictly sequential contexts (*Seq*), (*ii*) strictly structural contexts that use governor, dependents and siblings information (*Struct*) and (*iii*) mixed contexts using both dependency-based and sequential contexts (*Mix*). Regarding the span, we have tried 1 and 2. Siblings are only used in structural and mixed contexts of span 2 because that is the length of the path between a vertex and its siblings in a tree (cf. Figure 5.1). For the length of attributes subsets in delexicalised contexts, we have tried bare POS (length of 0) and the full power set of the attributes (all possible subsets). For the embedding space dimension we have tried 50, 150 and 500 dimensions, or the maximum possible size[9] if smaller than 500. For better readability, we use shortcuts to refer to the different parameter settings: 1 = (span 1, max length 0), 2 = (span 2, max length 0) and 3 = (span 2, full lengths for contexts of span 1, max length 0 for contexts of span 2). The context composition is detailed in Table 5.7.

Experimental Settings

Besides embeddings, there are three additional hyper-parameters that need to be tuned: the C aggressiveness parameter of the PA-II algorithm, the scaling factor α that controls the relative weight of the embedding features in the edge scores as shown in equation 5.1, and the number i of training iterations of the PA-II algorithm. We have tuned these hyper-parameters through a grid search on the development sets and picked the values that gave best results on average, giving $C = 0.001$, $\alpha = 0.001$, $i = 5$.

All the scores reported below are Unlabeled Attachment Scores (UAS) measured on the test sets ignoring the punctuation marks. The result tables are already heavy and because each embedding type can exist in 3 sizes, we only report the best score of the three sizes on the test set. It is slightly unusual, but as we just want to test our hypotheses and not necessarily to compare to other methods, it is not an issue. We computed the significance of the scores using the McNemar's test.

[9]Spectral-based dimension reduction such as PCA are limited by the number of eigenvectors of the matrix to be reduced. For example a matrix of size 200×500 can only be reduced into a $200 \times n$ matrix where $n \leq 200$ via PCA. When the number of eigenvectors is smaller than 500, we use that value instead.

	Baseline	Seq 1	Seq 2	Seq 3	Struct 1	Struct 2	Struct 3	Mix 1	Mix 2	Mix 3
En	85.62	$86.07^{\star}_{40}$	85.92^{*}_{80}	$86.06^{\diamond}_{121}$	$86.01^{\diamond}_{37}$	85.95^{*}_{121}	85.94^{*}_{50}	$86.10^{\star}_{77}$	$\mathbf{86.19}^{\star}_{50}$	85.84_{121}
Eu	76.65	$\underline{76.60}_{38}$	76.70_{76}	76.73_{500}	76.67_{35}	76.85_{119}	$\mathbf{76.90}_{150}$	76.66_{50}	76.72_{150}	76.80_{500}
Fi	79.97	80.44_{36}	80.44_{72}	$80.71^{\diamond}_{150}$	80.58^{*}_{33}	80.35_{114}	80.58^{*}_{50}	$\mathbf{80.92}^{\star}_{69}$	80.40_{50}	80.60^{*}_{50}
Fr	**83.99**	$\underline{83.48}_{40}$	$\underline{83.55}_{50}$	$\underline{83.47}_{150}$	$\underline{83.42}_{37}$	$\underline{83.68}_{128}$	$\underline{83.71}_{150}$	$\underline{83.61}_{77}$	$\underline{83.89}_{150}$	$\underline{83.84}_{198}$
Got	79.16	$\underline{78.95}_{32}$	$\underline{79.10}_{50}$	79.57_{500}	79.24_{28}	79.31_{50}	$\mathbf{80.09}^{\diamond}_{500}$	79.59_{50}	79.62_{150}	79.62_{500}
He	84.05	$\underline{83.87}_{38}$	$\underline{83.86}_{50}$	84.24_{50}	84.18_{35}	84.32_{50}	84.14_{150}	84.32_{50}	84.34_{50}	$\mathbf{84.36}_{150}$
Hu	79.15	79.23_{38}	$\mathbf{80.13}^{*}_{76}$	79.83^{*}_{500}	79.67_{34}	$\mathbf{80.13}^{\diamond}_{118}$	79.69_{500}	79.94^{*}_{50}	79.64_{50}	79.80_{50}
Ro	81.35	$\underline{81.34}_{40}$	$\underline{81.29}_{80}$	$\underline{81.21}_{415}$	$\underline{81.00}_{37}$	$\mathbf{81.38}_{50}$	$\underline{81.29}_{150}$	$\underline{81.30}_{50}$	$\underline{81.26}_{150}$	$\underline{81.21}_{415}$

Table 5.8: Best UAS scores for each embedding type in monolingual setting. The best score for each language are in bold. Results below the baseline are underlined. The statistical significance (using McNemar's test) of an improvement over the baseline is indicated with a superscript mark: * stands for a significance with a p-value inferior than 0.05, $^{\diamond}$ stands for $p \leq 0.01$ and * for $p \leq 0.001$. The length of the embeddings is reported as a subscript.

5.5.2 Results

Monolingual Experiments

Table 5.8 displays UAS scores for the monolingual setting. Except for French and Romanian that do not show real improvement, the six other languages show substantial performance increases with the embeddings. These improvements are statistically significant for all languages, except for Basque and Hebrew. One of our hypotheses was that structure is important when learning an embedding for dependency parsing and indeed our results support it. The largest improvements appear with structured or mixed embeddings which rely on syntactic structures.

The results for English are significant and close to each other for all types of embeddings, this tends to show that in English, sentence structure and word order are very correlated and both contribute information. Indeed that is what one expects for English which has a rigid syntax and a poor morphology.

On the other side of the picture, Basque and Gothic display the largest improvements with structured morphological embeddings. This is also expected as those are both morphologically rich languages with more flexible word order. Even though the argument is less clear for Hungarian and Finnish, they both show that structure is important for learning informative dependency embeddings.

Cross-lingual Experiments

Table 5.10 summarises the UAS scores achieved using delexicalised embeddings learned on several languages. Parsing accuracy improve for four languages (en, eu, hu, ro) in the cross-lingual setting compared to the best monolingual setting. While the multilingual embeddings do not outperform the monolingual ones for the other four languages, they still deliver parsing performance that are better than with the baseline MST parser for all languages (but Gothic and French). That shows that indeed using data from other languages is beneficial for learning good embeddings for dependency parsing, which was the second hypothesis we wanted to evaluate. We also notice that the largest gains are achieved with structural (or mixed) embeddings, giving more evidence of the importance of using structure for learning embeddings for dependency parsing.

Let us now look more closely at which groups of source languages are most

Cluster	Languages
All	en, eu, fi, fr, got, he, hu, ro
Noun Declension	eu, fi, got, hu, ro
No Noun Declension	en, fr, he
Indo-European	en, fr, got, ro
Indo-European Declension	got, ro
Romance Friendly	en, fr, ro
Germanic	en, got
East Europe	hu, ro
Finno-Ugric	fi, hu

Table 5.9: Clusters used for training cross-lingual delexicalised representations.

Language	Baseline	Best Mono	Best All	Best Multilingual
English	85.62	86.19*	86.18$^{\star}_{seq3,50}$	**86.32**$^{\star}_{en,got,mix3,50}$
Basque	76.65	76.90	**76.97**$^{*}_{struct3,50}$	76.68$_{decl,seq2,50}$
Finnish	79.97	**80.92***	80.89$^{\star}_{struct2,50}$	80.81$^{\diamond}_{decl,seq2,50}$
French	**83.99**	83.89	83.87$_{struct1,37}$	83.89$_{en,fr,ro,mix1,77}$
Gothic	79.16	**80.09**$^{\diamond}$	79.80$_{struct2,50}$	79.99$^{*}_{got,ro,mix3,500}$
Hebrew	84.05	**84.36**	84.32$_{seq3,150}$	84.13$_{en,fr,he,mix1,77}$
Hungarian	79.15	80.13*	80.05$^{*}_{mix2,150}$	**80.30**$^{\diamond}_{decl,struct1,37}$
Romanian	81.35	81.38	81.31$_{seq3,150}$	**81.52**$_{hu,ro,mix3,50}$

Table 5.10: Best UAS scores in cross-lingual setting. Under *Best All* are the results using the embeddings learned on the set of all languages, while under *Best Multilingual* are given the best results for each language using only a subset of the languages for learning the embedding. The subscript represents the context types and the number of dimensions of the embedding. The baselines and best monolingual scores are also reported. The statistical significance (using McNemar's test) of an improvement over the baseline is indicated with a superscript mark: * stands for a significance with a p-value inferior than 0.05, $^{\diamond}$ stands for $p \leq 0.01$ and * for $p \leq 0.001$.

helpful for specific target languages. First, note in general the best performing embeddings are never those obtained by using the full set of languages (this is only the case for Basque). This is expected since we have picked languages with very different grammars thus the full embeddings can be very noisy with regard to a single language. In fact, the Basque results are rather surprising since this language differs the most from the others in terms of morphology, but also one for which we had rather small training data.

The best parsing performance for English are achieved when using additional data from Gothic. As both are Germanic languages, this tends to show that data from genetically related languages can help in learning a better representation. Even though they do not achieve the best results, similar patterns occur for French (French and Romanian are Romance languages and English has been heavily influenced by Romance languages) and for Gothic (Gothic and Romanian are both case marking Indo-European languages). Similarly, Hungarian and Romanian reach their best scores when parsed with typologically close languages that have case marking. And again, Basque, Finnish and Gothic display similar patterns. Hebrew performs reasonably well with French and English which are two languages with fairly restricted word orders and no case marking like Hebrew.

As to why some languages have better monolingual parsing results than multilingual results, we think this is at least partly due to the lack of flexibility of our model. That is, morpho-syntactic attributes sets are treated independently from one another making some of them hard to use in the cross-lingual setting. For example, Hebrew verbs display 'binyanim' (internal flection classes) that do not appear in any other language, similarly Finnish has a lot of cases that are not found in other languages. Those are indeed two languages that do not perform well with other languages. We thus believe that introducing compositionality in our embedding model should help in solving those problems and enhance the results further. While varying word order could also explain the poor parsing results when using sequential contexts, structured contexts should be immune to this problem, but they do not show much better results, therefore implying that exotic morphology is a bigger problem than word order.

5.6 Conclusion

In this first work, we have learned representations for morphological attributes sets that we have called delexicalised word embeddings. By using them in complement to more traditional features in a parser, we have shown that using syntactic structure in training the representations is beneficial for dependency parsing. This is expected and agrees with previous results from Kiperwasser et al. [KG15], Chen et al. [CZZ14] and Bansal [Ban15, BGL14].

We have also shown that in a monolingual setting, using representations learned from multiple languages can be beneficial if languages are well assorted. We left for future work the task of automatically finding good languages clusters and of learning composable morphological attributes representations for unseen delexicalised words.

Chapter 6

Phylogenetic Learning of Multi-Lingual Parsers

This chapter is based on a work accepted for presentation at NAACL 2019 [DD19].

One way to leverage data from several languages in order to improve their overall parsing performance is to use multi-task learning techniques. In multi-task learning [Car97] one has access to several data sets and/or several related objective functions and the goal is to use those related objectives to improve on learning each model.

For example, if one wants to perform translation of English sentences into French and summarisation of English sentences, one could of course train a separate model for each task. But one can also notice that those tasks are in fact very similar. Both imply understanding the input sentence to some extent, in order to rewrite it in another language in the case of translation and with fewer words and less details in the case of summarisation. Thus, one could benefit from sharing information between the two tasks, especially between the component that understands the input sentence.

There have been many different proposals for learning multiple tasks simultaneously [RBAS17, SRIV11, SG16]. Depending on the task similarities: are they the same task on different data sets or different tasks on the same data set, or even different tasks on different data sets but with similarities, and the relation between tasks: should one task be applied before another in a kind of pipeline or can they be performed in parallel, different methods can be applied.

In our case, we learn the same task (dependency parsing) with the same output space (dependency trees) on different but related inputs (different languages). Capitalising on the cross-lingually consistent morphological annotation of the Universal Dependencies data and following our hypothesis that morphology is stable enough across related languages to serve as a bridge for syntactic information, we will assume that all inputs live in the same morphological space. However, in order to faithfully represent languages specificities, we will let each language have its own parsing model and just share information across different models.

There are several ways of doing it. Assuming we have access to some similarity measure between languages (in the shape of a graph), the simplest approach is to learn a model for each language and only have them interacting once they are fully trained in a transfer learning fashion. But we would prefer to have models interacting already at training time so that each model could use more than just its mono-lingual information. A possibility arising from the fact that languages

are evolving entities that share ancestries, is to have models evolving alongside their evolutionary tree.

In this chapter, we will present a new multi-task learning framework that uses a fixed language evolutionary tree to guide the learning process of several parsing models for the languages of the tree. This multi-task learning framework has an interesting property. As it learns models for intermediary stages in the evolutionary tree, it will allow us to perform zero-shot parsing (parsing of languages for which we have no training data). If we know where a language without training data sits in the tree, we can use the model of its last ancestor for which we have one to parse it.

The remaining of this chapter is organised as follow. Section 6.1 reviews some related works on multi-task learning and on multi-lingual dependency parsing. Section 6.2 presents the phylogenetic learning framework that uses evolutionary tree to guide learning multi-lingual parsing models. Section 6.3 introduces the tree Perceptron which instantiate the phylogenetic learning framework with the Perceptron algorithm and discusses some issues that arise when using averaged Perceptron in this setting. Section 6.4 presents the neural architecture used for the neural phylogenetic multi-lingual parser. Section 6.5 gives some experimental results on multi-task methods for dependency parsing. Finally Section 6.6 closes the chapter.

6.1 Related Work

6.1.1 Multi-Task Learning

In multi-task learning literature, task relationships are often given in the shape of a graph that guides the learning process. In Cavallanti et al. [CCBG10] linear models are learned for related tasks that live in the same space. Those tasks share their Perceptron based parameter updates through a fixed similarity graph.

Bellet et al. [VBT17] extend the model propagation framework where pre-trained models are propagated through a similarity graph to share information between related tasks. They propose to add information about model confidence in order to account for unbalance in data distribution and/or quality across tasks for example.

Kumar et al. [KDI12] propose to learn the similarity structure of the tasks at the same time as the task models themselves. They assume that the surface tasks for which they want to learn a model are in fact linear combinations of a set latent tasks that is shared for all surface tasks. They then propose an algorithm to learn the parameters of the latent tasks as well as the weight of those latent tasks in the surface ones.

More recently, with the advances in neural network architectures, it has been proposed to supervise different parts of the networks with different tasks. For example, in Søgaard et al. [SG16], task hierarchy is directly encoded in the neural model allowing tasks with different output space to share parts of their parameters. Low layer neurons receive feedback from both POS tagging and CCG parsing, while higher layer neurons only receive feedback from CCG parsing.

However, all those works have in common that they assume fix task relationships. Even in Kumar et al., the relationship structure of the tasks changes because it is learned, but it converges to an optimum which is supposedly fix. In the present work, we assume the relationship between tasks can evolve over time.

Changing level in the tree can be seen as splitting the similarity graph into disjoint sub graphs. This encodes information about task evolution that lacks in other multi-task methods.

6.1.2 Multi-Lingual Dependency Parsing

In multi-lingual parsing, Ammar et al. [AMB+16] propose to train a single model to parse many languages using both typological information, cross-lingual word representations and language specific information. While their model gives good results, they only apply it to 7 Germanic and Romance languages. It would be worth doing the experiment with 50+ languages and see how the results would change. However, because of language specific information their model would probably become too big to fit in memory.

Naseem et al. [NBG12] propose to learn generative models for several languages and to tie some parameters of those models according to typological similarity between languages.

Aufrant et al. [AWY16] propose to tackle zero-shot parsing by rewriting source treebanks to better fit target language typology. Assuming that typology is homogeneous in a language family, the phylogeny should drive models to be typologically aware. However, that assumption might not always hold.

Eventually, the closest work from our in spirit is the one of Berg-Kirkpatrick et al. [BK10]. They use a phylogenetic tree to guide the training of unsupervised dependency parsing models of several languages, using ancestor models to bias descendent ones. The main difference here beside supervision, is that we do not use ancestor models as biases but rather as initialisation of descendent models.

Contrary to Waleed et al., Naseem et al. and Aufrant et al., we do not make explicit use of typological information in our model, rather we use the phylogenetic tree as a surrogate assuming that typology of related languages should be similar. Therefore, contrary to Naseem et al. that tie part of the parameters of the generative models for different languages according to typological similarity, we tie the whole discriminative models for different languages for an amount of time that reflects their shared history. Also, while Waleed et al. use language specific information alongside language agnostic one, we never explicitly use language specific information, we just let independent parsing models evolve for each language therefore capturing language specific information.

6.2 Phylogenetic Learning of Multi-Lingual Parsers

Languages change and evolve over time. A community that spoke once a single language can be split geographically or politically, and if the separation is long enough their language will diverge in direction different enough so that at some point they might not be intelligible to each other. The most striking differences between related languages are often of lexical and phonological order but grammars also change over time.

Those divergent histories are often depicted in the shape of a tree in which related languages whose common history stopped earlier branch off higher than languages that have shared a longer common path [Jäg15, Cam13]. We hypothesise that building on this shared history is beneficial when learning dependency parsing models. We thus propose to use the phylogenetic structure to guide the

learning process of multi-lingual dependency parsers that will tie parameters between languages according to their common history.

In order to do so, we introduce a new multi-task learning framework that shares information between tasks using a tree structure. The tree structure allows us to share both model parameters and training samples between related tasks. We call this new learning framework *Phylogenetic Learning*.

As our phylogenetic learning method can be used with any tree encoding tasks relationships and any learning algorithm supporting fine-tuning, we will apply it to multi-lingual dependency parsing, using a language evolutionary tree to guide the learning process and we will explore it with the Perceptron algorithm as described in Section 6.3 and with a neural parser described in Section 6.4. The Perceptron is interesting because it is both a simple linear model and a very well studied algorithm. However, linear models have limitations, amongst which not working well with dense word representations which would actually be the easiest way to lexicalise a multi-lingual parser. Thus we stick to a delexicalised model for the Perceptron. Neural networks are much easier to lexicalise in multi-lingual setting, notably by using character models, thus we also train a Multi-Lingual neural parser in lexicalised and delexicalised regimes for comparison.

As our phylogenetic learning method induces parsing models for every inner node in the phylogenetic tree, it can also perform zero-shot dependency parsing of unseen languages. Indeed, one can use the model of the lowest ancestor (in the tree) of a new language as an approximation of that language's grammar. We will also explore this possibility through experiment on data from the Universal Dependency project.

6.2.1 Model Phylogeny

Languages evolve from earlier stages and sometimes a language will change differently in different places leading to different languages with a common ancestor. This evolution process is often depicted in the shape of a tree in which leaves are actual languages and inner nodes can be either attested ancestral languages or their idealised reconstruction. Figure 6.1 gives an example of such a tree for a subset of the Slavic family of Indo-European languages [SF18].

Just as languages evolve and diverge, so do their grammars. Assuming a parsing model is a parameterised representation of a grammar, then we can expect parsing models to evolve in a similar way as their languages. Likewise parsing problems can be thought of as evolving problems. What was once a single problem (parsing sentences in Proto-West-Slavic) became a set of distinct but related problems (parsing sentences in Czech, Polish, Slovak and Upper Sorbian) as Proto-West-Slavic was evolving into its modern descendants. We thus take a multi-task approach to solving those related problems.

We assume that the grammar of the last common ancestor is a good approximation of those languages grammars. Thus it should be easier to learn a language grammar starting from its ancestor grammar than from scratch. There are however some issues with this assumption. First, a language grammar can be very different from its ancestor one from two millennia earlier. Consider the difference between modern French and early Classical Latin for example, in two millennia Latin has witnessed the loss of its case system and a complete refoundation of its verbal system. Second, a lot of languages have only started to be recorded very recently thus lacking historical data all together. And when historical records are

Figure 6.1: A possible phylogenetic tree for languages in the Slavic family.

available, much work still needs to be done to render those data usable by parsers. For example the Universal Dependencies Project [NAA+18b] only has annotated corpora for Latin, old Greek, old Church Slavonic and Sanskrit. And even for those classical languages, it is not clear to which extent their modern counterparts really descend from them. Thus we need to find another way to access the ancestor language grammar than using historical data.

We propose to use all the data from descendent languages to represent an ancestor language. In principle, one could give more weight to older languages or to languages that are known to be more conservative, but this knowledge is not available for all languages families. Thus we resort to using all the available data from descendent languages without distinction.

Another problem is that the tree view is too simple to represent the complete range of phenomena involved in language evolution, such as language contacts. Furthermore, languages do not evolve completely randomly, but follow some linguistic universals and have to keep a balance between speakability, learnability and understandability. Thus, languages can share grammatical features without necessarily being genetically related, either by contact or by mere chance. Also, phylogenetic trees are mostly based on lexical comparisons [Cam13] and not necessarily on grammatical ones. However, the tree model is still a good starting point in practice and language families align well with grammatical similarity as recent works on typological analysis of UD treebanks have shown [CG17, SA17]. We thus make the simplifying assumption that a language grammar evolves only from an older stage and can be approximated by that previous stage.

6.2.2 Phylogenetic Datasets

Let $\mathcal{L} = \{l_1, l_2, ..., l_{n_l}\}$ be a set of n_l languages and let $\mathcal{P} = \{p_1, p_2, ..., p_{n_p}\}$ be a set of n_p proto-languages (hypothesized ancestors of languages in $\mathcal{L}$). Let $\mathcal{T}$ be a tree over $\mathcal{L}^* = \mathcal{L} \cup \mathcal{P}$ such that languages of $\mathcal{L}$ are leaves and proto-languages of $\mathcal{P}$ are inner nodes. This means that we assume no two languages in $\mathcal{L}$ share

a direct parenthood relation, but they at best descend both from a hypothesised parent. Tree $\mathcal{T}$ has a single root, a proto-language from $\mathcal{P}$ from which all languages descend. This ancestor of all languages should capture linguistic universals[1] and ensures we work with a well formed tree. We use the notation $p > l$ for the fact that language/node l descends from language/node p.

For each language $l \in \mathcal{L}$, we assume access to a set of n_l annotated examples $\mathcal{D}_l$. For each proto-language $p \in \mathcal{P}$, we create an annotated set $\mathcal{D}_p = \bigcup_{p>l} \mathcal{D}_l$ as the union of its descendent sets. We could in principle have data appearing only in inner nodes. For example, if we had training data for a classical language such as Latin but no test data, then we would not need to train a specific Latin model (therefore no Latin leaf), but could still use the data inside the tree. However, we consider here that all actual languages, past or present are leaves in the tree.

For each language $l \in \mathcal{L}^*$, we will learn a parsing model $\boldsymbol{\theta}_l$.

6.2.3 Model Training

The main idea behind phylogenetic training is to initialise a new model with the best model of its parent, thus effectively sharing information between languages and letting models diverge and specialise over time. The training procedure described hereafter and is summed up in Algorithm 9.

Data: a training set $\mathcal{D}_l$ and dev set $\mathcal{D}'_l$ per language, a tree $\mathcal{T}$, two sampling sizes k, k' and a maximum number of reboot r
Result: a model $\boldsymbol{\theta}$ per node in $\mathcal{T}$
begin
  Instantiate empty queue Q
  $Q.push(\mathcal{T}.root)$
  while Q*is not empty* **do**
    $l = Q.pop()$
    $\boldsymbol{\theta}_l^0 = \boldsymbol{\theta}_{l.parent}$ ($\boldsymbol{\theta}_{root}^0$ initialised randomly)
    $reboot = 0, i = 1, s_0 = 0$

    while $reboot < r$ **do**
      $\boldsymbol{\theta}_l^i = train(\boldsymbol{\theta}_l^{i-1}, \mathcal{D}_l, k)$
      $s_i = test(\boldsymbol{\theta}_l^i, \mathcal{D}'_l, k')$
      if $s_i \leq s_{i-1}$ **then**
        $reboot = reboot + 1$
      else
        $reboot = 0, i = i + 1$
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_l^i$

    for c *in* $l.children$ **do**
      $Q.push(c)$
  return $\boldsymbol{\theta}_{\mathcal{T}}$

Algorithm 9: Phylogenetic training procedure.

At the beginning, we initialize a new blank/random model that will be the basic parsing model for all the world languages. Then, we sample sentences (we

[1] It does not mean anything about our actual belief or not in the past existence of an ancestor to all the modern natural human languages (so called monoglotto genesis hypothesis).

will discuss sampling issues in the next section) randomly from all the available languages, parse them, compute the loss and update the model accordingly. Since the training sentences are sampled from all the available languages, the model will learn to be as good as possible for all the languages at the same time.

When the model $\boldsymbol{\theta}_p$ has reached an optimum (that we define hereafter), we pass a copy of it to each of its children. Thus, for each child c of p, we initialize $\boldsymbol{\theta}_c^0 = \boldsymbol{\theta}_p$ to its parent (p) final state. Each model $\boldsymbol{\theta}_c$ is then refined on its own data set $\mathcal{D}_c$ which is a subset of $\mathcal{D}_p$, until it reaches its own optimum state and is passed down to its own children. This process is repeated until the model reaches a leaf language, where the model $\boldsymbol{\theta}_c$ is eventually refined over its mono-lingual data set $\mathcal{D}_c$.

By passing down optimal models from older/larger languages sets to newer/smaller ones, models get the chance to learn relevant information from many different languages while specialising as time goes by.

The question now is to determine when to pass down a model to its children. In other words, at which stage has a model learned enough from its data and should start to diverge to improve again?

Following the principle of cross-validation, we propose to let held-out data decide when is the right time to pass the model down. Let $\mathcal{D}'_p$ be a set of held-out sentences from the same languages as $\mathcal{D}_p$. Then, every k training examples, we freeze the model $\boldsymbol{\theta}_p^i$ at iteration i, and test it on k' sentences from $\mathcal{D}'_p$. This gives a score (UAS/LAS) to the current model. If the current score is higher than the score of the previous model $\boldsymbol{\theta}_p^{i-1}$ then training goes on from $\boldsymbol{\theta}_p^i$, otherwise we discard it and retrain from $\boldsymbol{\theta}_p^{i-1}$ for another k sentences before re-evaluating it. If after having discarded r models we have not yet found a better one, then we assume we had reached an optimal previous model $\boldsymbol{\theta}_p^{i-1}$ and pass it on to its children (unless its a leaf, in which case training is over for that language). This correspond to the innermost while loop in Algorithm 9.

For the few languages that do not have a separate dev set, we use the same sentences for training and validating the model quality, which can be problematic, therefore in the experiment section, we score those languages aside.

6.2.4 Sentence Sampling

There are a few things we should consider when drawing examples from a proto-language distribution. Beside the question of whether some languages are more conservative than others with respect to their ancestor, which we have decided to simplify saying that all languages are as representative of their ancestors, there is the problem of data imbalance and tree imbalance.

Sampling sentences uniformly is not a viable option for the size of datasets varies a lot across languages and that they do not correlate with how close a language is to its ancestor. For example, there are 260 Belarusian training sentences against 48814 Russian ones. The basic question is thus whether one should draw examples from languages or branches. Basic linguistic intuition tells us that drawing should be performed on branches. Modern languages distribution has no reason to reflect their proximity to their ancestor language. Amongst Indo-European languages, there are one or two Armenian languages as well as one or two Albanian languages (depending on the criteria for being a language), while there are tens of Slavic languages and Romance languages. However, there is no reason to believe that Slavic or Romance languages are better witnesses of proto-Indo-European

than Armenian or Albanian.

Drawing examples from languages would bias the intermediate models toward families that have more languages (or more treebanks). It might be a good bias depending on the way one compute the overall accuracy of the system. If one uses the macro-average of the individual language parsers, then biasing models toward families with many members should improve the accuracy overall.

In this work, at a given inner node, we decided to sample uniformly at random over branches spanning from this node, then uniformly at random over languages and then uniformly at random over sentences. It boils down to flattening the sub-tree below an inner node to have a maximum depth of 2. For example, at the root (World) we pick a branch at random (Indo-European), then a language at random (Slovak) then a sentence at random. Likewise, at the Germanic node, we pick a branch at random (West-Germanic), then a language at random (Afrikaans) then a sentence at random. Given that we have picked the Indo-European branch, all Indo-European languages are then as likely to be chosen which biases the probabilities toward big families. There are 12 Slavic languages but only 2 Greek ones, therefore Slavic languages will have 6 times more influence on the final model than Greek.

We could otherwise sample over branches, then over sub-branches again and again until we reach a leaf and only then pick a sentence. In this case, the Balto-Slavic branch and Greek branch would have the same probability to be picked, and then it would be as likely to pick a Slavic languages or Greek. This would give much more weight to languages high in the tree than languages low in the tree and an equal weight to all branches, however it could be detrimental to the average score. It would of course be possible to try any variation between those two, picking sub-branches according to a probability that would depend on the number of languages in that family for example, therefore mitigating the unbalance problem.

The actual phylogenetic tree used to guide the training process and to sample sentences is depicted in Figure 6.4.

6.2.5 Zero-Shot Dependency Parsing

An interesting property of the phylogenetic training procedure is that it provides a model for each inner node of the tree and thus each intermediary grammar. If one were to bring a new language with its position in the tree, then we can use the pre-trained model of its direct ancestor as an initialisation instead of learning a new model from scratch. Similarly, one can use this ancestor model directly to parse the new language, effectively performing zero-shot dependency parsing. We investigate this possibility in the experiment section.

6.3 Tree Perceptron

The Tree Perceptron algorithm is the instantiation of the phylogenetic multi-task learning framework described above with the structured Perceptron algorithm [Col02, MCP05a, MCP05b, MPH11].

As we work with edge factored graph-based dependency parsing models, each edge e is represented by a feature vector $\boldsymbol{\phi}(e)$. Given a sentence $x = x_1 x_2 \cdots x_n$, let $\mathbf{x}_i \in \{0,1\}^d$ be the vector representing word x_i. Here, $\mathbf{x}_i$ is a one-hot vector encoding the part-of-speech and morphological attributes of word x_i with ones at the actual attributes indices and at the POS index, thus the vector length d is the

number of parts-of-speech and morphological attributes. Then, let $\mathbf{b}_{ij} \in \{0,1\}^{d'}$ where $d' = |pos|^3$ be a vector encoding in between parts-of-speech. Each dimension represents a different triplet made of a begin POS, an end POS and an in between POS, and we put a 1 at each relevant position. Eventually, let $\mathbf{l}_{ij} \in \{0,1\}^{d''}$ be a binned signed distance vector representing the signed length of the edge. Here, we use the thirteen bins $\{all, \leq -10, \leq -5, -4, -3, -2, -1, 1, 2, 3, 4, 5 \leq, 10 \leq\}$. Then, the feature vector of an edge e whose governor is x_i and dependent is x_j is computed as:

$$\boldsymbol{\phi}(e) = \mathbf{l}_{ij} \otimes [(\mathbf{x}_{i-1} \oplus \mathbf{x}_i \oplus \mathbf{x}_{i+1}) \otimes (\mathbf{x}_{j-1} \oplus \mathbf{x}_j \oplus \mathbf{x}_{j+1}) \oplus \mathbf{b}_{ij}] \,.$$

Thus, the feature representation encodes information about the part-of-speech and attributes of the governor and dependent and their surrounding words, plus in between parts-of-speech and the length and direction (sign) of the relation.

Those features are reminiscent of the ones used by McDonald et al. [MCP05b] with the presence of in between parts-of-speech and signed length of the edge. However, the outer product only considers pairs of parts-of-speech or attributes, more like the biaffine model of Dozat et al. [DQM17] but with a much simpler representation as input. In wide multi-lingual setting, lexicalisation is a challenging issue and cross-lingual word representation is still a vivid area of research, therefore in the wait of a better solution, we resorted to training delexicalised models.

Based on this feature representation, the standard structured Perceptron algorithm is performed for each task until convergence and then the learned model is passed down to its descendants. More formally, let y and $\hat{y}$ be the true tree and a predicted tree over sentence x, and let $\boldsymbol{\Phi}$ be the tree feature function defined as:

$$\boldsymbol{\Phi}(y) = \sum_{e \in y} \boldsymbol{\phi}(e).$$

Then the update rule at sentence level is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\Phi}(y) - \boldsymbol{\Phi}(\hat{y}).$$

At the task level, let l be a language and p its direct ancestor in the phylogeny, then the model learned for p is used to initialise the model of l:

$$\boldsymbol{\theta}_l^0 = \boldsymbol{\theta}_p.$$

6.3.1 Averaging Policy

As we have seen in section 2.3, the Perceptron algorithm is an online learning algorithm that is used in dependency parsing for computational practicality more than for its actual learning properties. In order to make the models learned with the Perceptron more generalisable in the batch learning setting, methods such as voting and averaging have been proposed [FS99]. Averaging especially, while having a very limited time and memory overhead, has proven a very effective mean to boost parsing results.

The basic basic idea behind averaging, is that in a batch setting, each step of the online model holds relevant information (that can fade over time) and that combining all those steps should be better than just relying on the final online step. While this is sensical with the classical Perceptron algorithm, it becomes much more dubious in the multi-task and tree cases.

The idea behind the tree Perceptron is to let the model specialise more and more with the time. As earlier models are more general than later ones, they hold potentially less relevant information. In that sense, it is not clear how to perform averaging, if averaging should be performed on a model basis or a language basis and if averaging will still be beneficial altogether.

The most basic averaging policy is to average over all online steps, thus for a leaf language l, the final model would be the average of all the online models that appeared on the learning path from the tree root to l. From our own experiments, it does not improve the results when languages in the tree are very diverse. This is because we also include potentially detrimental information from higher levels.

Thus another possibility, is to average only online models that resulted from seeing an example from language l at any point in the tree. This ensure to only include at least partially relevant models. This is the policy with which we experiment in the experiment section. We should note that in that case, averaged models are not available for zero-shot languages as they do not have training data to average on.

Yet another possibility is to average on the leaf updates only. It is then similar to the traditional averaged structured Perceptron but using the tree model to initialise it instead of an empty vector. Again, in this case, zero-shot languages do not have averaged models.

In fact, instead of only averaging models resulting from updates from the single language l of interest, it should be possible to perform weighted averages, with the weights depending either on the level (higher levels having a smaller weight than lower levels) or on the similarity between l and the language of the current example or even on the number of updates since the last example from l. We have kept those policies for future work as there are a lot of possibilities to investigate.

6.4 Neural Model

Our scoring model is an edge factored graph-based neural model in the vein of recent works by Dozat et al. [DQM17]. There are two major differences here compared to the parser of Dozat et al. The first difference is in individual word representation, for which we use only the UPOS[2] tag, morphological information as provided in column 6 (cf. Section 4.2) of UD treebanks and character based word embedding, whilst Dozat et al. use also the XPOS[3] tag, holistic word vectors (from Word2Vec [MSC+13] and their own) and they do not use morphological information beside what might already be given by the XPOS. The main reason for those differences in word representation is that Dozat et al. do work in a mono-lingual setting, rather than a multi-lingual setting like us. XPOS are not available for all languages, and when they are they differ greatly from languages to languages in term of granularity. Likewise, cross-lingual word embeddings are not yet easily available for all the languages of the UD project. Despite, the fact that we use gold morphological information, our setting is closer to what is readily available. However, with the current research on cross-lingual word representation, we can expect to use holistic word vectors soon as well.

The second difference is the scoring function proper. While they use biaffine

[2]Universal part-of-speech for a set of 17 tags. Does not encode morphology.

[3]Language specific part-of-speech. Might include morphological information, but is not available for all languages.

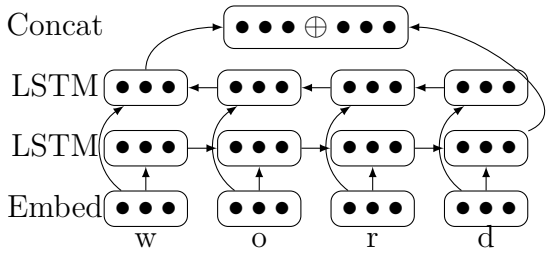


Figure 6.2: Bi-LSTM architecture for character based word representation. The final representation is the concatenation of the final cells of each layer.

scoring functions and decouple edge scoring from label scoring, we use a simple multi-layer perceptron to compute label scores and pick the max over the label as the edge score.

Let $x = (x_1 x_2 ... x_l)$ be a sentence of length l. Each word x_i is represented as the concatenation of 3 sub-vectors, one for its part-of-speech, one for its morphological attributes and one for its form:

$$\mathbf{x}_i = \mathbf{pos}_i \oplus \mathbf{morph}_i \oplus \mathbf{char}_i.$$

The part-of-speech vector ($\mathbf{pos}_i$) is from a look up table. The morphological vector ($\mathbf{morph}_i$) is the sum of the representation of each morphological attribute $\mathbf{m}$ of the word given by the treebanks:

$$\mathbf{morph}_i = \sum_{m \in morph_i} \mathbf{m}.$$

We add a special dummy attribute representing the absence of morphological attributes.

The form vector ($\mathbf{char}_i$) is computed by a character BiLSTM [HS97]. Characters are fed one by one to the recurrent neural network in each direction. The actual form vector is then the concatenation of the outputs of the forward character LSTM and of the backward character LSTM as depicted in Figure 6.2.

Once, each word has been given a representation in isolation, those representations are passed to two other BiLSTMs. Each word is then represented as the concatenation of its contextualised vector from the forward and backward layers:

$$\mathbf{c}_i = forward(\mathbf{x}_1, ..., \mathbf{x}_i) \oplus backward(\mathbf{x}_i, ..., \mathbf{x}_l).$$

We actually train two different BiLSTMs, one representing words as dependents ($\mathbf{c}$) and one words as governors ($\hat{\mathbf{c}}$). An edge score is then computed as follows. Its governor word vector $\hat{\mathbf{c}}_i$ and its dependent word vector $\mathbf{c}_j$ are concatenated and fed to a two layers perceptron (whose weights are $\mathbf{L}_1$ and $\mathbf{L}_2$) with a rectifier (noted $[...]^+$) after the first layer in order to compute the score s_{ijl} of the edge for every possible relation label l:

$$s_{ij} = \max_l s_{ijl} = \max_l (\mathbf{L}_2 \cdot [\mathbf{L}_1 \cdot (\hat{\mathbf{c}}_i \oplus \mathbf{c}_j)]^+)_l.$$

The neural model is trained end to end via back propagation of the following loss function one sentence at a time via standard stochastic gradient descent routine. Given a sentence x, we note j the index of the governor of w_i and l the

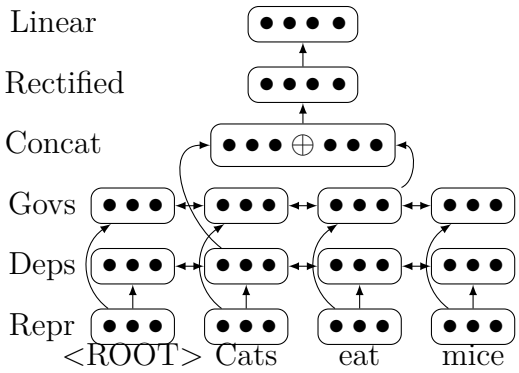


Figure 6.3: Neural network architecture for edge scoring. The contextualised representation of the governor (eat) and the dependent (Cats) are concatenated and passed through a rectified linear layer and a final plain linear layer to get a vector of label scores.

relation label of w_i, the loss function is:

$$loss(x) = \sum_{w_i} \Big[\sum_{\substack{j' \neq j \\ j' \neq i}} \max(0, s_{ij'} - s_{ij} + 1)^2 + \sum_{l' \neq l} \max(0, s_{ijl'} - s_{ijl} + 1)^2 \Big]$$

For each word, there are two terms. The first term enforces that for all potential governors that are neither the word itself or its rightful governor, their highest score (irrespective of the relation label) should be smaller than the score of the actual governor and actual label by a margin of 1. The second term is similar and enforces that for the rightful governor, any label that is not the rightful label should have a score smaller than the score of the actual label again by a margin of 1.

6.5 Experiments

To assess the potential phylogenetic training both in terms of multi-task learning and zero-shot parsing capabilities, we experimented with data from the Universal Dependencies project version 2.2 [NAA+18b]. When several corpora are available for a language, we chose one to keep a good balance between morphological annotation and number of sentences.

6.5.1 Setting

As some languages have no training data and unique writing systems rendering the character model inefficient, we resorted to use gold parts-of-speech and morphological attributes. For example, Thai has no training data, no language relative and a unique script, which altogether make it really hard to parse (from a phylogenetic perspective).

The phylogenetic tree used for the experiment is adapted from the Ethnologue [SF18]. It is reported in Figure 6.4 and 6.5. We tried to have a tree as consensual as possible, but there are still a few disputable choices, mostly about granularity and consistency. Sanskrit could have its own branch in the Indic family just as Latin in the Romance family, but because Sanskrit has no training data, that would not

actually change the results. Czechoslovak and Dutch-Afrikaans have their own branches, then what about Scandinavian languages? Then, there is the problem of Naija. As an English based Creole it could as well be put in the Germanic family, but we kept it as a separate (Creole) family.

Figure 6.4 represents the phylogenetic tree used for guiding the training process. As we only use data from the UD project 2.2, we collapse unique child so that Vietnamese is not an Austro-Asiatic language, it is just Vietnamese. We also only use well attested families, thus Buryat, a Mongolic language, is alone and not linked to Turkic languages. Maybe, the most disputable choice is to put Naija in its own Creole family instead of the Germanic family.

Regarding model training proper, for the phylogenetic models we used $k = 500$ training sentences per iteration, $k' = 500$ dev sentences to compute running accuracy and a maximum number of reboot $r = 5$. Following Dozat et al [DQM17], we use Eisner algorithm [Eis96] at test time to ensure outputs are well formed trees.

We trained both lexicalised and delexicalised neural models as cross-lingual lexicalisation is transparent with character embedding and optimised them for LAS. Linear models are only trained delexicalised and optimised for UAS. Independent models are trained in the same manner as their phylogenetic counterparts but with mono-lingual data only.

The neural model is implemented in Dynet [NDG+17] and we use Adadelta with default parameters as our trainer. We averaged the results over 5 random initialisations.

We report percentages of unlabeled and labeled edge prediction accuracy (UAS/LAS).

Model Propagation

As a mean of comparison, we experimented with the model propagation framework of Zhu et al. [ZGL03] and Bellet et al. [VBT17]. In the simplest form of model propagation, pre-trained models are propagated through a similarity graph to neighbouring tasks. In the end, each task as represented by a node in the similarity graph receives a model which is a weighted average of all the original pre-trained models and whose weights reflect directly the similarity structure of the tasks.

This is an appealing framework as it is relatively simple to implement, it works with already trained models therefore saving the whole training process when testing graph variations and so on, and it is also an easy way to compute models for tasks that do not have training data (they just average models of similar tasks that have data). However, there are a number of hyper-parameters that need to be fine tuned for the method to work well such as the weight of the original pre-trained model in the final average. Furthermore, in our setting, we do not have directly a similarity graph but a phylogenetic tree, which we need to turn into a graph, and there are plethora of possibilities here too, which makes the tuning process quite long. Also, the model averaging policy works well for linear models, but not neural networks in which case we can only average their output or have them voting. But that means we have to run every pre-trained neural model per instance instead of a single averaged model and this is also really time consuming.

Because of those issues, model propagation did not stand the comparison with phylogenetic training for languages with training data. However, it proved a viable alternative for zero-shot dependency parsing. Therefore, we decided to put the description of the methods and the parsing results for languages with training data

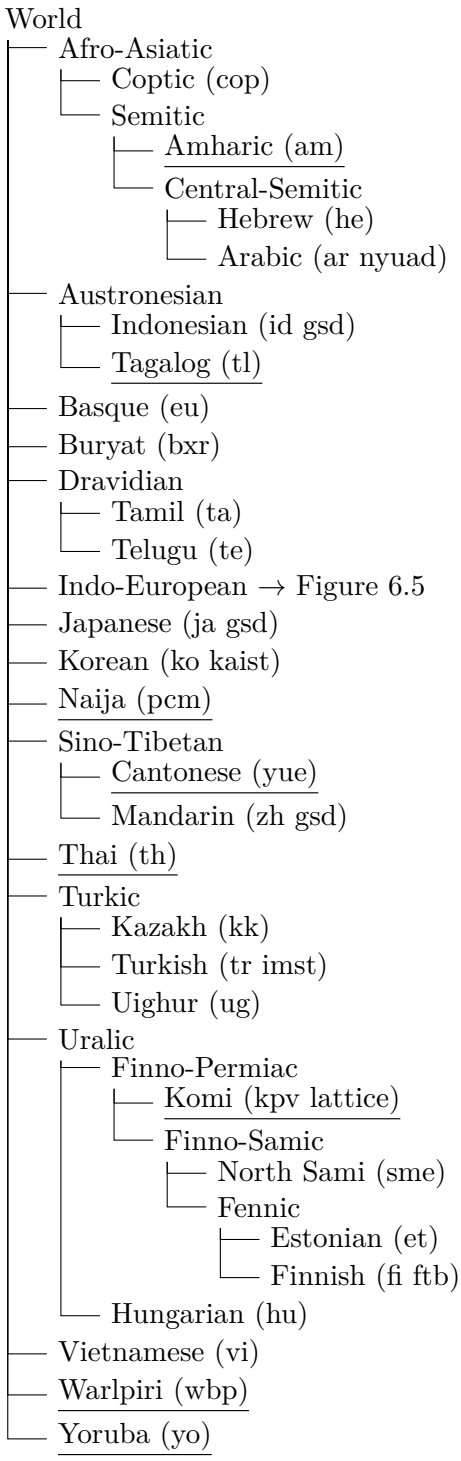


Figure 6.4: Phylogenetic tree used to guide the training process of the multi-lingual parser. Underlined languages are those that do not have a training set. The code of the language and if necessary the name of the treebank are given in parentheses. The Indo-European sub-tree is depicted in Figure 6.5.

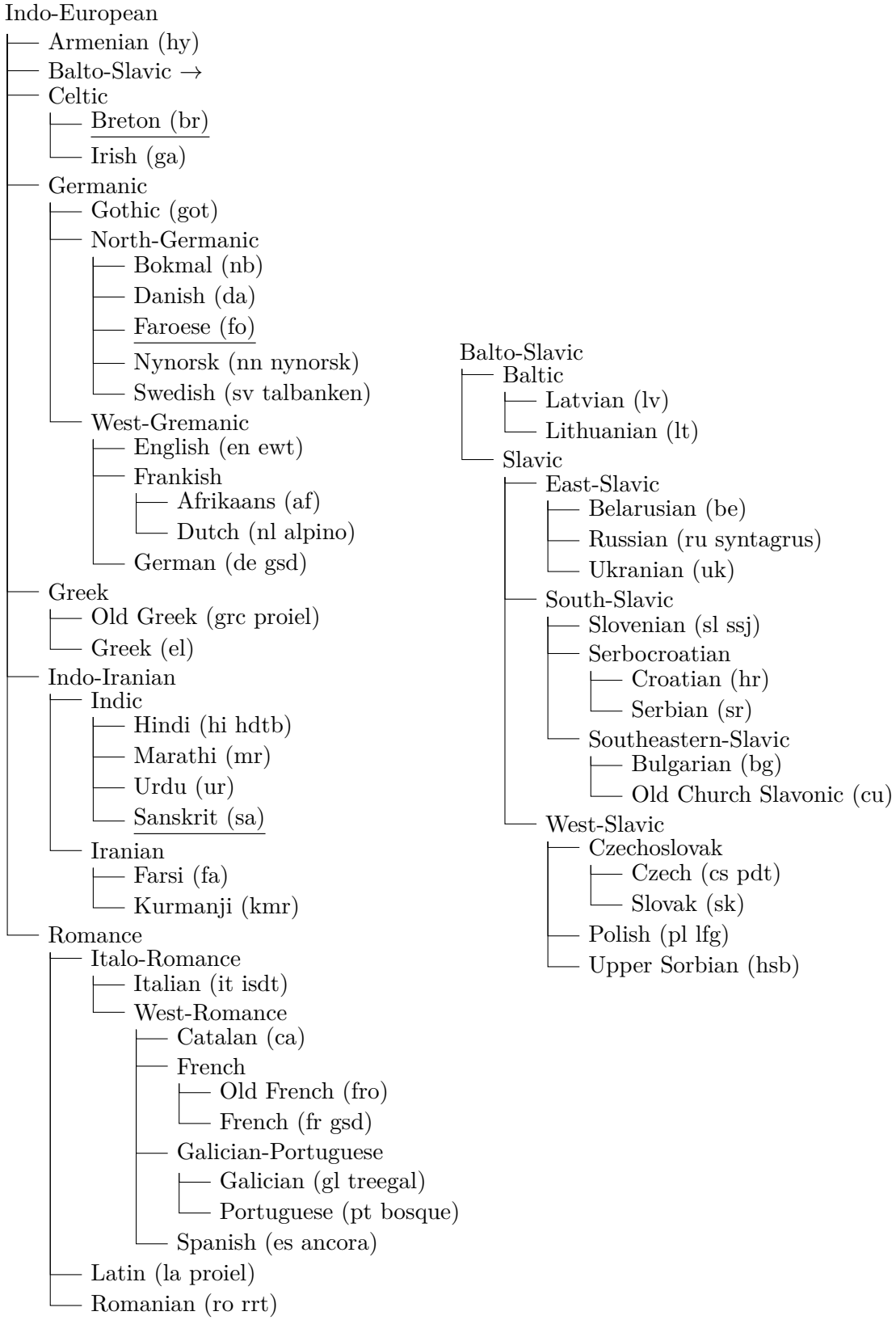


Figure 6.5: Indo-European branch of the phylogenetic tree used to guide the training process of the multi-lingual parser. Underlined languages are those that do not have a training set. The code of the language and if necessary the name of the treebank are given in parentheses.

and their analysis in the appendix (Section 9.1) at the end of this dissertation, and only keep the results of model propagation for zero-shot parsing in this section.

6.5.2 Results with Training Data

Tables 6.1, 6.2, 6.3 and 6.4 report parsing results for languages that have a training set. Table 6.1 gives unlabeled accuracy scores for phylogenetic and independent linear parsers trained with the Perceptron algorithm. Scores are computed for the last online training model (Last) and for the average of all the online models resulting from an example of the language of interest (Avg) as described in section 6.3. Tables 6.2 and 6.3 give both labeled and unlabeled accuracy scores for phylogenetic and independent delexicalised (lexicalised respectively) neural parsers. In order to make the analysis easier, Table 6.4 reports aggregated parsing results per language family from those three big tables. For each language, the best UAS/LAS is in bold, in bold italic are scores that beat their phylogenetique/independent counterpart by more than 1 point. For example, in Table 6.1, the UAS score of the last phylogenetic model for Arabic is 2.04 points higher than its independently trained counterpart, therefore it is reported in bold.

Note that a few languages do not have a separate development set. Those languages also tend to have a very small training set, therefore we used the training set as both training set and "held-out data" without splitting it. The training set size is reported in square brackets for those languages. This has low to no impact on other languages results but it could be problematic for the language itself as it could over-fit its training data especially when they are very few as is the case for Buryat (bxr) for example. To be fair, we report two different averages. Avg Dev is the average over languages that have a separate development set, and Avg No Dev the average over languages that do not have a separate development set.

On average, phylogenetic training improves parsing accuracy over independent models in every case: averaged and plain linear models and neural models, delexicalised and lexicalised and for both labeled and unlabeled scores. This is especially true for languages that have very small training sets (50 sentences or less). Those languages that also lack development set show more than 5 points improvements, up to 15 points (hsb, kmr). This shows that the ancestor's model is both a good initialisation and acts as a form of regularisation, slowing down over-fitting when very few data are available.

Phylogenetic training is clearly beneficial as one gains information from related languages as averages show in Table 6.4. Indo-European, Turkic and Uralic languages really gain from sharing information. This is especially true for Balto-Slavic (sk +5.82, lt +5.07 UAS lexicalised neural) and Indo-Iranian languages (mr +2.05 UAS lexicalised neural). While it is less true for Romance and Germanic languages in the lexicalised neural model, the improvements are more homogeneous in the delexicalised case and even more so in the linear delexicalised case. This might be due to two different factors. First, the tree might not represent well the typology for those families. Typically, English tends to group syntactically with Scandinavian languages more than with West-Germanic[4].

Then, lexicalisation might be to blame too. While lexicalisation is beneficial on a per language basis, with on average more than 1.50 points improvements

[4]So much so that Faarlund et al. [ETF14] amongst others have argued in favour of the classification of modern English as a North Germanic language rather than as a West Germanic language alongside Dutch and German.

	Phylogenetic		Independent	
	Avg	Last	Avg	Last
ar	***78,86***	***73,58***	77,32	71,54
cop	**84,06**	**80,25**	83,25	80,21
he	**81,42**	***77,99***	80,97	76,05
bxr [19]	***37,96***	***33,18***	31,00	29,54
eu	73,78	**67,69**	**74,64**	66,72
af	**81,43**	***77,04***	81,13	75,43
da	**77,86**	**72,27**	77,52	71,43
de	78,91	72,66	**79,11**	**73,54**
en	78,01	***74,90***	**78,26**	71,42
got	***77,08***	***72,77***	75,83	69,95
nb	83,61	***79,78***	**83,82**	78,41
nl	**76,91**	***69,31***	76,23	68,12
nn	**81,09**	***76,68***	80,78	74,59
sv	**79,92**	73,11	79,59	**73,58**
be	***78,14***	***74,53***	74,88	71,45
bg	**85,05**	***81,24***	84,38	79,79
cs	***78,83***	***74,13***	77,16	69,07
cu	***79,87***	***76,07***	78,38	72,91
hr	**80,95**	***75,81***	80,35	74,41
hsb [23]	***64,49***	***55,10***	48,37	44,64
lt	***59,38***	***53,70***	55,62	50,95
lv	76,22	***72,31***	**76,51**	68,46
pl	***91,00***	***89,03***	89,40	86,49
ru	**77,48**	***72,47***	76,98	70,69
sk	***82,73***	***79,60***	78,76	74,73
sl	***84,70***	***81,09***	83,18	78,32
sr	**83,67**	***79,83***	83,34	78,15
uk	***79,75***	***76,91***	78,37	73,47
ca	**84,27**	**78,96**	83,47	78,10
es	**83,86**	***79,06***	83,31	77,43
fr	**84,07**	***79,47***	83,31	78,21
fro	**79,25**	***75,45***	79,19	71,96
gl [600]	***84,42***	***80,37***	83,12	77,67
it	**86,50**	***83,11***	85,69	80,58
la	**63,58**	***58,88***	62,84	55,06
pt	***84,81***	**79,67**	83,75	78,91
ro	***80,54***	***74,20***	79,48	71,85
fa	77,27	**68,51**	**77,47**	68,43
hi	**88,39**	***83,75***	87,52	81,92
kmr [20]	***47,57***	***40,91***	40,96	38,35
mr	75,00	73,08	**75,51**	***74,49***
ur	***85,20***	***78,91***	84,08	76,03
el	**85,76**	***82,47***	84,90	79,91
grc	**70,00**	***64,58***	69,42	59,50
ga [566]	76,02	**73,18**	**76,46**	72,78
hy [50]	***60,18***	***54,22***	57,78	52,72
id	**81,52**	***74,69***	81,03	72,91
ja	86,09	81,00	**86,72**	**81,04**
ko	61,95	52,11	**62,09**	**52,36**
kk [31]	***64,15***	***58,03***	56,91	53,49
tr	**59,01**	***50,32***	58,29	47,49
ug	64,07	**59,51**	***65,16***	58,61
et	**75,83**	***70,26***	75,24	68,96
fi	***77,67***	***72,61***	75,95	69,21
hu	**77,54**	**71,41**	76,66	70,54
sme [2257]	***76,25***	***71,94***	74,05	66,78
ta	**74,13**	***69,34***	73,70	66,23
te	**86,65**	***84,86***	86,37	82,77
vi	59,67	50,46	**60,41**	***51,67***
zh	**70,93**	***60,94***	69,99	56,68
Avg Dev	78,35	73,31	77,64	71,36
Avg No Dev	63,88	58,37	58,58	54,50

Table 6.1: Unlabeled parsing results for languages with a train set for phylogenetic and independent linear models.

	Phylogenetic		Independent	
	UAS	LAS	UAS	LAS
ar	73.76	69.03	**74,60**	***70,17***
cop	83,06	73,27	**83,28**	**73,77**
he	**81,12**	73,57	80,84	**74,30**
bxr [19]	***50,66***	***32,41***	39,67	20,06
eu	74,85	66,96	**75,83**	***68,77***
af	**82,20**	**77,76**	81,99	77,71
da	**77,55**	71,40	77,42	**71,52**
de	**79,23**	***71,95***	78,33	70,94
en	78,23	73,29	**78,29**	**73,61**
got	***78,51***	***71,46***	77,19	70,29
nb	83,54	76,86	**83,83**	**77,41**
nl	***76,53***	**66,32**	75,09	65,58
nn	81,38	75,02	**81,98**	***76,05***
sv	***79,68***	***73,23***	77,77	71,13
be	***77,79***	***71,84***	76,00	70,14
bg	***86,31***	***79,54***	83,62	76,23
cs	***78,95***	***70,60***	76,16	68,43
cu	***82,47***	***75,97***	80,68	74,36
hr	***81,48***	***74,44***	79,59	72,21
hsb [23]	***73,32***	***64,88***	59,24	51,16
lt	***61,02***	***50,50***	56,78	45,62
lv	***76,93***	***68,22***	75,27	66,97
pl	***92,45***	***87,97***	91,40	86,91
ru	**77,41**	**71,94**	76,46	71,56
sk	***83,68***	***77,56***	78,71	72,76
sl	86,58	82,42	**86,89**	**83,26**
sr	***84,23***	***77,55***	82,84	76,05
uk	***76,67***	***71,82***	74,08	70,00
ca	**83,93**	**77,96**	83,29	77,52
es	82,95	76,78	**83,36**	**77,17**
fr	**84,55**	**76,70**	83,69	76,01
fro	***79,91***	***68,62***	77,38	66,32
gl [600]	**84,12**	**78,14**	83,74	77,36
it	**85,73**	**79,78**	85,47	79,31
la	***65,66***	**57,77**	64,37	56,85
pt	84,60	79,18	**84,71**	**79,57**
ro	***79,18***	***69,19***	77,96	67,93
fa	76,43	69,81	***77,53***	***71,03***
hi	87,95	78,70	**88,43**	***79,73***
kmr [20]	***69,02***	***59,49***	53,25	44,49
mr	76,86	**65,45**	**77,69**	64,94
ur	83,56	73,88	**84,16**	**74,79**
el	85,67	81,51	**86,15**	**81,69**
grc	***72,89***	***66,38***	71,39	64,85
ga [566]	**75,80**	**67,04**	75,38	66,69
hy [50]	***64,91***	***51,61***	59,49	46,73
id	***79,26***	***73,07***	77,78	71,61
ja	85,78	73,60	**86,24**	**73,97**
ko	59,45	38,79	**59,88**	**39,24**
kk [31]	***72,41***	***58,32***	63,53	44,93
tr	***57,20***	***48,12***	55,15	46,45
ug	***63,75***	***42,75***	62,15	41,53
et	***75,56***	***68,34***	73,55	66,20
fi	***78,35***	***72,21***	75,15	69,06
hu	78,04	70,84	**78,45**	**71,52**
sme [2257]	***79,28***	***75,17***	78,10	73,94
ta	**74,03**	**65,66**	73,95	65,29
te	**86,26**	**64,14**	85,50	63,17
vi	**59,36**	**55,40**	59,28	55,14
zh	**70,83**	***63,03***	70,31	61,77
Avg Dev	78,33	70,35	77,46	69,58
Avg No Dev	71,19	60,88	64,05	53,17

Table 6.2: Parsing results for languages with a train set for phylogenetic and independent delexicalised neural models.

	Phylogenetic		Independent	
	UAS	LAS	UAS	LAS
ar	74.81	70.32	**75.07**	**71.08**
cop	85.51	79.28	**86.03**	**80.15**
he	bg81.89	75.36	81.59	**75.57**
bxr [19]	***48.72***	***30.68***	37.88	18.09
eu	76.81	69.51	***78.61***	***72.76***
af	85.15	80.94	**85.44**	**81.66**
da	78.50	72.50	**79.16**	***74.13***
de	**80.37**	***73.54***	79.48	72.37
en	79.25	74.34	**79.27**	**74.66**
got	77.83	71.54	***79.91***	***74.33***
nb	**84.62**	**78.78**	83.82	78.09
nl	**77.19**	**68.55**	76.52	68.40
nn	82.39	76.44	**82.58**	**77.32**
sv	80.46	74.62	**81.17**	**75.47**
be	***80.18***	***74.11***	78.09	72.76
bg	86.01	79.16	**86.40**	**79.79**
cs	***79.78***	***71.71***	77.45	69.88
cu	82.98	77.19	**83.31**	***78.32***
hr	**81.70**	**74.73**	81.05	73.95
hsb [23]	***74.24***	***66.01***	58.59	50.37
lt	***61.42***	***50.88***	56.35	46.14
lv	***78.39***	***70.14***	76.69	68.89
pl	***92.88***	***88.53***	91.07	86.49
ru	**77.91**	72.72	77.33	**72.85**
sk	***84.91***	***79.17***	79.09	73.20
sl	87.15	83.43	***88.39***	***85.21***
sr	85.85	79.86	**86.17**	**80.47**
uk	***78.16***	***73.50***	74.96	70.91
ca	84.67	78.81	***85.69***	***80.11***
es	85.11	79.52	**85.61**	**80.18**
fr	**84.35**	77.59	84.21	**77.94**
fro	***82.32***	***74.24***	78.91	69.95
gl [600]	**83.80**	**78.06**	83.60	77.63
it	87.03	81.67	**87.10**	**82.27**
la	***66.25***	***58.88***	65.07	57.80
pt	**84.93**	79.37	84.90	**79.83**
ro	79.83	70.46	**79.93**	**70.88**
fa	78.76	72.95	***79.93***	***74.07***
hi	**89.32**	**82.89**	88.75	82.60
kmr [20]	***69.08***	***59.64***	54.77	45.07
mr	***78.65***	***68.97***	76.60	64.04
ur	84.32	77.02	**84.82**	***78.19***
el	86.44	83.30	**86.88**	**83.96**
grc	***73.82***	***67.88***	71.68	66.05
ga [566]	75.91	67.54	**76.20**	**67.72**
hy [50]	***65.03***	***51.76***	59.27	46.67
id	**81.08**	**74.97**	80.83	74.69
ja	91.22	87.31	**91.40**	**87.37**
ko	73.38	68.35	**74.23**	***69.81***
kk [31]	***70.82***	***55.42***	62.81	44.59
tr	**59.64**	**50.66**	59.00	50.54
ug	***66.33***	***48.20***	63.66	46.07
et	***75.32***	***68.13***	73.91	66.96
fi	***78.05***	***72.20***	74.66	68.22
hu	79.51	72.88	**80.15**	***74.31***
sme [2257]	***80.13***	***76.40***	78.34	74.25
ta	75.05	66.94	***76.19***	**67.93**
te	***88.88***	***74.24***	87.01	72.05
vi	65.59	61.15	**66.02**	**61.74**
zh	**80.36**	**74.79**	80.14	74.52
Avg Dev	80.05	73.35	79.47	73.02
Avg No Dev	70.97	60.69	63.93	53.05

Table 6.3: Labeled and unlabeled parsing results for languages with a train set for phylogenetic and independent lexicalised neural models.

			Afro-Asiatic	Indo-European	Germanic	Slavic	Romance	Indo-Iranian	Greek	Turkic	Uralic	Dravidian	Avg Dev	Avg No Dev
Linear	Phylo	Avg	**81.45**	***78.38***	**79.43**	***78.73***	**81.26**	***74.68***	**77.88**	***62.41***	***76.82***	**80.39**	78.35	63.88
		Last	77.27	73.64	74.28	74.42	76.57	69.03	73.53	55.95	71.55	77.10	73.31	58.37
	Inde	Avg	80.51	76.98	79.14	76.12	80.46	73.11	77.16	60.12	75.47	80.03	77.64	58.58
		Last	75.93	71.31	72.94	70.97	74.42	67.84	69.70	53.20	68.87	74.50	71.36	54.50
Delex	Phylo	UAS	79.32	***79.51***	**79.65**	***79.95***	**81.18**	***78.77***	**79.28**	***64.45***	***77.81***	**80.14**	78.33	71.19
		LAS	71.96	***72.21***	**73.03**	***73.23***	**73.79**	***69.47***	**73.95**	***49.73***	***71.64***	**64.90**	70.35	60.88
	Inde	UAS	**79.58**	77.73	79.10	76.98	80.44	76.21	78.77	60.28	76.31	79.73	77.46	64.05
		LAS	**72.75**	70.56	72.69	70.40	73.12	67.00	73.27	44.31	70.18	64.23	69.58	53.17
Lex	Phylo	UAS	80.73	***80.41***	80.64	***80.83***	**82.03**	***80.03***	**80.13**	***65.60***	***78.25***	**81.97**	80.05	70.97
		LAS	74.99	***73.73***	74.58	***74.37***	**75.40**	***72.29***	**75.59**	***51.43***	***72.40***	**70.59**	73.35	60.69
	Inde	UAS	**80.90**	78.93	**80.82**	78.21	81.67	76.97	79.28	61.82	76.76	81.60	79.47	63.93
		LAS	**75.60**	72.45	**75.16**	72.09	75.18	68.79	75.01	47.07	70.93	69.99	73.02	53.05

Table 6.4: Parsing results for phylogenetic and independent models averaged by language family. This table compiles the results of Tables 6.1 (lines 1-4), 6.2 (lines 5-8) and 6.3 (lines 9-12) for families that have at least two languages in those tables. Families are sorted in the same order as they appear in those tables. The Indo-European results are averaged over all Indo-European languages, even if they do not have any relatives in there subfamily such as Armenian (hy) and Irish (ga). Global averages are repeated for completeness.

and more than 10 UAS points for Korean (ko) or Mandarin (zh), in a multilingual setting with various writing systems and spelling conventions for any given writing system, it might just get too complicated for the model to learn meaningful character representations. It is really disputable to represent alphabetic characters (Latin, Arabic, Hebrew, Armenian...) in the same space as syllabic ones (Japanese, Korean...) and ideographic ones (Chinese, Japanese...). In fact, while Turkic and Uralic languages show the same benefits overall (ug +2.67, fi +3.39 UAS lexicalised neural), Kazakh (kk) scores are worse for the lexicalised neural parser than for the delexicalised one. This might be because Kazakh is written in Cyrillic while the only other languages written in Cyrillic in the tree are Slavic languages (Russian, Belarusian and Ukrainian).

Results for Dravidian and Afro-Asiatic languages are not as consistent. While Telugu seems to always gain from Tamil data, the reverse is not true in the lexicalised neural model case. Result variation for Arabic, Hebrew and Coptic are marginal in the case of neural models. This is likely due to the fact that we only have three quite different languages from that family and that they all have their own script. However, they all show improvements in the delexicalised linear case.

Phylogenetic training is not consistently useful for languages that do not have relatives. While Buryat (bxr) that has a very small training set benefits from universal linguistic information and gain almost 11 points UAS, Basque (eu) that has a very different grammatical structure than other languages and enough training data (5396 sentences) looses 3.25 LAS in the lexicalised neural model case. Gains and losses are more marginal for the other five "isolated" languages (id, ja, ko, vi, zh). However, despite being rather small, they are consistent for Indonesian (id), Japanese (ja), Korean (ko) and Mandarin Chinese (zh). Phylogenetic training is either always beneficial (id, zh) or detrimental (ja, ko) irrespective of the underlying learning algorithm.

Comparing the different models together, we note that neural models handle languages with very few data better than the linear one. Upper Sorbian (hsb) shows a 9 UAS points improvement, Buryat (bxr) 12 UAS points, Kazakh (kk) 18

		Linear	Delex		Lex	
	Model	UAS	UAS	LAS	UAS	LAS
am	Semitic	46,98	**57,95**	**27,22**	57,27	26,25
br	Celtic*	60,19	60,91	42,46	**61,36**	***43,89***
fo	North-Germanic	50,05	***53,58***	***47,55***	52,40	46,52
sa	Indic	64,11	***64,93***	***51,41***	56,18	40,46
kpv	Finno-Permiac*	47,32	57,62	41,15	***65,14***	***52,11***
pcm	World	51,49	55,59	40,67	***60,43***	***43,80***
th	World	30,65	***33,59***	***21,90***	29,14	17,61
tl	Austronesian*	68,09	***75,15***	***52,51***	70,89	50,38
wbp	World	65,12	82,02	62,56	***87,67***	***65,66***
yo	World	38,25	51,83	35,24	***56,16***	***37,51***
yue	Sino-Tibetan*	40,87	**41,78**	**25,46**	41,68	25,02
Avg		51,19	57,72	40,74	58,04	40,83

Table 6.5: Parsing accuracies for languages without a training set. For each language, the first column report the identity of the model used for parsing. The following columns report unlabeled and labeled parsing accuracies for the linear model, the delexicalised neural model and the lexicalised neural model respectively.

UAS points and Kurmanji (kmr) 22 UAS points improvement. It might be thanks to the learned dense representation of morphological attributes that is used in the neural networks and that is absent from the linear model. As we will see in Chapter 7, dense representations are not really necessary for morphological attributes when enough data are available. As they are much fewer and more frequent than word forms for example, a few hundreds sentences are already enough for a linear model to learn to work with one-hot encoded morphological attributes. However, some languages have less than 50 sentences, which is very few, and it might be that in those extreme conditions, dense representation of morphological attributes is necessary for good performance.

Regarding better resourced languages, the two delexicalised models (linear and neural) have comparable results overall. The linear model is better for Afro-Asiatic languages (ar, he, cop), Indonesian (id) and Korean (ko) for example while the neural model better fits other languages such as Slovenian (sl).

The lexicalised neural model is on average better than both delexicalised ones. However, is has a slightly lower UAS score on languages with few training data that the delexicalised neural model and is overall much less consistent in terms of benefits from using phylogenetic training. This is one more argument showing the problem of lexicalisation in a multi-lingual setting.

Overall results are a bit below the state of the art, but again the model is very simple and does not include a pre-trained language model that could help especially for languages with limited resources. Furthermore, we rely extensively on gold morphology which is not exactly comparable with other works.

6.5.3 Zero-Shot Dependency Parsing

Table 6.5 reports parsing results for languages that do not have a training set. Because of phylogenetic training and the tree structure that guides it, it can happen that a language ancestor's model is in fact trained on data only accounting for a narrow range of later stages. For example, while Faroese uses the North-Germanic

model refined on both Norwegians, Swedish and Danish data, Breton uses the Celtic model only refined on Irish data thus making it more an Irish model than an actual Celtic model. Those cases are marked by an asterisk in the table. Komi model is refined on Finno-Samic data, Tagalog model on Indonesian data and Cantonese model on Mandarin data.

Looking at Table 6.5, we make the following observations. As expected scores are on average lower than for languages with training data, however the UAS/LAS gap is substantially bigger from 6.71 to 17.08 points for the lexicalised neural model. It is hard to compare to other works on zero-shot parsing since they use different data and scores span a big range, but our results are comparable to those of Aufrant et al. [AWY16] and Naseem et al. [NBG12], while our zero-shot models are given for free by the phylogenetic training method.

On a language per language basis, we see that there are a few important factors, the most striking being genre. Tagalog (tl) and more surprisingly Warlpiri (wbp) have relatively high parsing accuracy despite being either completely isolated or having only one relative (Indonesian). This is likely because their data are well annotated stereotypical sentences extracted from grammars, thus making them easy to parse.

Then we see that Naija (pcm) and Yoruba (yo) are about 20 points higher than Thai (th) despite them three having very simple morphology (in the treebanks) and no relatives. As they have different genres (spoken, bible, news and wiki), without a deeper look at the trees themselves, our best guess is that this is caused by Thai having a different script. Naija and Yoruba both use the Latin alphabet, and as such they can rely to some extent on the character model to share information with other languages, to at least organise the character space.

This analysis also carries over to Cantonese (yue). It is a morphologically simple language, and despite the presence of a relative (Mandarin), its score is rather low. The genre alone (spoken) would not explain everything as Naija has also a spoken treebank and a higher score. The writing system might be at blame once again. Indeed, Chinese characters are very different from alphabetic characters and are much harder to use in character models because of sparsity. Comparing Mandarin and Cantonese test sets with Mandarin train set, the amount of out-of-vocabulary words is 32.47% of types (11.90% of tokens) for Mandarin and 54.88% of types (56.50% of tokens) for Cantonese. The results for out-of-vocabulary characters are even more striking with 3.73% of types (0.49% of tokens) for Mandarin and 12.97% of types (34.29% of tokens) for Cantonese. This shows that not only there are a lot of OOV in Cantonese test set, but that those words/characters are common ones as 12.97% of character types missing make up for more than a third of all character tokens missing, where on the contrary Mandarin OOV are seldom and account for less tokens percentage than types.

Indeed, while neural models consistently beats the linear model, half of the languages behave better delexicalised, amongst which Thai and Amharic have their own writing systems, Sanskrit shares its writing system with Hindi only, and Faroese which while using the Latin alphabet also uses some unique characters such as ð. The improvement seen with the delexicalised model for Tagalog is however surprising as it also uses a fairly standard Latin alphabet and would require a deeper analysis.

Overall, this shows both the importance of lexicalisation for dependency parsing and the complexity of performing it in a multi-lingual setting where many different scripts are involved.

	Propagated		Phylogenetic		
	Unweighted	Weighted	Linear	Delex	Lex
am	58.13	**58.38**	46,98	57,95	57,27
br	54.27	54.95	60,19	60,91	**61,36**
fo	46.65	44.65	50,05	***53,58***	52,40
sa	48.14	45.97	64,11	***64,93***	56,18
kpv	60.49	56.26	47,32	57,62	***65,14***
pcm	60.15	54.95	51,49	55,59	**60,43**
th	***41.99***	38.32	30,65	33,59	29,14
tl	***89.56***	88.78	68,09	75,15	70,89
wbp	85.12	75.67	65,12	82,02	***87,67***
yo	51.34	42.06	38,25	51,83	***56,16***
yue	***58.60***	55.91	40,87	41,78	41,68
Avg	59.49	55.99	51,19	57,72	58,04

Table 6.6: Unlabeled parsing accuracies for languages without a training set. The first two columns report the scores of the propagated models corresponding to the setting described in previous section. The following columns report unlabeled parsing accuracies for the phylogenetic linear model, delexicalised neural model and lexicalised neural model respectively for comparison.

Other important factors are typology and morphology. Amharic (am) despite its unique script has a higher score than Cantonese that actually shares its scripts (to some extent as we have seen) with Mandarin. The key point for Amharic score, is that all its relatives (Hebrew, Arabic and Coptic) have their own scripts and are morphologically rich, thus the model learns to use morphological information. The analysis is similar for Komi which on top of sharing morphology with its relatives also share the writing system which provides it an extra gain. However, this might work in the opposite direction as well, as we can see with Faroese, Breton and Sanskrit. Faroese (fo) is morphologically rich and that should help, however its North-Germanic relatives are morphologically much simpler. Thus the model does not learn to rely on morphological attributes nor on word endings for the character model as much. The same is true for Sanskrit (sa), which is morphologically richer than its modern Indic relatives, with an extra layer of specific writing systems. Eventually, Breton model (br) is refined over Irish data only and while Irish is a typological outlier amongst Indo-European languages because of its Verb-Subject-Object word order, Breton has the standard Subject-Verb-Object, thus using Irish data might actually be detrimental.

These arguments show the respective importance of the writing system, the genre of the data, the morphological analysis and the typology in phylogenetic zero-shot dependency parsing. Those factors can either work together positively (Komi) or negatively (Cantonese) or cancel each other out (Amharic, Faroese).

Model Propagation for Zero-Shot Dependency Parsing

Table 6.6 reports the results of the propagated models of the previous section for languages that do not have a training set. Their phylogenetic scores are repeated for comparison. Contrary to phylogenetically learned models, it is straightforward to have zero-shot parsing models even for languages that have a training set in the model propagation case. Because we can not compare them to other results

and they are much lower than supervised results as expect, we report them in the appendix.

It is interesting to see that some languages really benefit from model propagation with up to 17 points improvements for Cantonese (yue). Those are languages that are high in the tree. In fact, other languages high in the tree (pcm, wpb, yo) have results on par with their phylogenetic counterparts. This shows that the early models high in the phylogenetic tree are too generic and too shallow for a thorough syntactic analysis. Using a mixture of more mature models is preferable in that situation.

Baring in mind that we did not have the time to explore all the hyper-parameter possibilities for propagation, it is encouraging to see that it is a viable option for zero-shot dependency parsing of isolated languages, and surely with more work those scores would increase.

6.6 Conclusion

In this chapter, we have discussed methods to learn parsing models for several languages in a true multi-lingual manner, meaning that the model learned for a given language benefits from information learned from other languages as well and uses all the available data. A way to do it is to learn parsing models for several languages at the same time and to share update information between those different models according to the similarity of their respective languages.

We have presented a multi-task learning framework that allows one to train models for several tasks that have diverged over time. Leveraging their common evolutionary history through a phylogenetic tree, models share parameters and training samples until they need to diverge. As a by product of this phylogenetic training, we are provided with intermediary models that can be used to zero-shot a new related task, given its position in the evolutionary history.

We have applied this framework to dependency parsing using either a linear Perceptron parser or a graph-based neural parser and the phylogenetic tree of the languages from UD 2.2 to guide the training process. Our results show that phylogenetic training is beneficial for well populated families such as Indo-European and Uralic. It also helps generalisation and prevents over-fitting when very few data are available.

Comparing results from delexicalised models with results from models lexicalised via character models, we have shown that lexicalisation is important for parsing but that it becomes challenging to have good lexicalisation in a truly multi-lingual setting with several writing systems where alphabets mix with syllabaries and ideographic systems. Furthermore, for zero-shot parsing, genre and morphology are other crucial factors to have good results.

We also compared those models with models propagated in a graph derived from the tree structure (cf. Appendix 9.1). While model propagation takes time to tune its hyper-parameters, it has proven a viable alternative to phylogenetic training for languages with very few to no data and data isolates (languages without training relative).

Some works have been done on automatically learning task relationship in multi-task setting [KGS11, KDI12]. It would be interesting to see how the algorithm could figure out when and how to cluster languages automatically as phylogenetic trees do not directly depict grammar evolution. Likewise, our model does not know that Latin came before Old French and before modern French, or

that despite being Germanic, English underwent a heavy Romance influence. It would be worth investigating softening the tree constraints and instigating more evolutionary information in the structure.

Another direction for future research is lexicalisation. As we have seen, direct lexicalisation with character level representation is complicated in a heavily multi-lingual setting with various writing systems. A possible way to go would be to start lexicalising models only lower in the tree. While this would solve part of the problems, it would also loose the benefits of phylogenetic training at least regarding lexical information. Other possibilities could come from cross-lingual word representation, but there is still a lot of work to be done there before to have good results, or plain transliteration, turning Thai or Amharic characters into Latin ones would solve part of the problems as well.

Concerning model propagation, the first direction to go would be sparsifying the graph to make hyper-parameters tuning faster. Then, follow Naseem et al. and Aufrant et al. [NBG12, AWY16], it would be interesting to have different propagation graphs for different part of the models. Maybe language l_i has a similar verb system as language l_j, but its noun phrases look more like those of language l_k.

Finally, in principle, nothing prevents us from propagating models that have been trained via the phylogenetic procedure. This should further improve the parsing results of languages with few data and languages that are too high in the tree.

In chapter 5, we looked at ways to improve representation of morphological information in a multi-lingual context. In this chapter, we have seen that delexicalised parsers using only morphological information are able to learn to parse several languages jointly. In the following chapter, we will investigate the actual role of the morphological information used in this chapter and the previous for dependency parsing.

Chapter 7

Measuring the Role of Morphology

This chapter is based on a work presented at EMNLP 2018 [DD18].

Throughout our work we have relied heavily on morphology. We have used morphology as means to reduce data sparsity (morphological attributes are fewer and denser than forms) both in morphologically rich languages and in morphologically poorer languages. We have used morphology to bridge the gap between languages seeing it as a common representation space. We have also used morphology as our primary source of features for learning parsing models.

One question that we have not addressed yet, and that has not received much attention from researchers, is that of the intrinsic worthiness of morphology for dependency parsing (though some work has been done on the line of morphological attributes selection for syntactic parsing [DTvG11]). Indeed morphological information is an efficient tool to reduce data sparsity but it also encodes some syntactic information of prime relevance for dependency parsing. Thus, we hypothesise that using morphological information is beneficial for dependency parsing, especially for morphologically rich languages.

However, not all morphological attributes necessarily encode syntax. Syntax encoding attributes such as cases like nominative and accusative or number and gender following agreement are called morpho-syntactic. Attributes that encodes more semantic information such as tenses or moods are called morpho-semantic[1]. Amongst morphologically rich languages, some languages make more use of morpho-syntactic attributes while some use rather morpho-semantic ones. Thus, we also hypothesise that as not all morphologically rich languages encode as much syntactic information in their morphology, not all will benefit as much from using morphological information for parsing.

A third hypothesis is that we can indeed tell the two benefits of using morphological information (sparsity reduction and syntactic encoding) apart and that we can use some measures independent of the parsing algorithms to distinguish languages that use morphology to encode syntax from languages that do not.

In order to test those hypothesis, we will compare results of parsing models that make use of different word representations to reduce sparsity and/or encode

[1]Tense can be triggered in the case of sequence of tenses, but as changing the principal clause tense would change the subordinated clauses tense without affecting the syntactic structure of the whole sentence, it is still considered morpho-semantic. For a more detailed explanation of the difference between morpho-syntax and morpho-semantics see Kibort [Kib10].

morphological information with measures of morphological complexity.

In the following, we first discuss morphological richness (Section 7.1) and present some ways of measuring it (Section 7.2). Then we turn to morphological richness in dependency parsing proper and morpho-syntactic richness (Section 7.3) and we introduce a new measure based on head preferential attachment that tries to capture morpho-syntactic complexity (Section 7.4). We also discuss the weight of the annotation scheme when considering dependency parsing and morphology (Section 7.5). Eventually, we present some empirical results showing the relevance of the investigated distinction between morphological and morpho-syntactical information (Section 7.6), as well as the relevance of the newly proposed measure of morpho-syntactic complexity in studying annotation schemes.

7.1 Morphological Richness

As we have seen in section 3.1, there is no clear boundary between morphological richness and morphological simplicity, and thus no simple criterion of morphological richness. But it refers to the productivity of a language's inflectional morphology.

Despite its fuzzy definition, morphological richness is of prime importance for two different reasons when it comes to natural language processing problems and especially dependency parsing. So much so that workshops focusing on syntactic analysis of morphologically rich languages specifically have been organised [TSG+10].

The first reason, which is rather pragmatic has to do with machine learning proper. As we have seen, words are entities that need to be mathematically represented in some way in order to be worked with. Whether we use one-hot vectors or denser representations for the words, the more words the more parameters need to be estimated and the more data are needed for a good estimation of those parameters. With one-hot vectors, the actual model will tend to be huge and require a lot of data to be trained, while with embeddings the model can be much smaller and need less data, because a lot of parameters are pre-trained and hidden in the embedding itself which requires a lot of data to be trained.

The second reason is more linguistic in nature. If in a language a word can take on different forms, those differences must be meaningful otherwise they disappear (this is also called the principle of economy [Vic03] by which languages avoid unnecessary redundancies). The meaning of those different forms however can be of many different kinds. Words can change form for phonological reasons (English *a* becomes *an* before a word starting with a vowel sound), syntactic reasons (Latin *rosa* becomes *rosam* when it is used as direct object), semantic reasons (Spanish *como* becomes *comeré* when the action happens in the future) and pragmatic reasons (Japanese *tabenai* becomes *tabemasen* when speaking to an older person for example.). Being able to extract this information, and especially the syntactic information from those forms should improve the results of parsers. For example, in a language like Latin that inflects verbs for their subject number and nouns for cases, it is highly unlikely that a noun in accusative plural be subject of a verb in singular, but it could well be its object.

Another point to be made before we start digging into morphological complexity measurement proper, is the inherent difference between theoretical complexity and measured complexity. As we also mentioned in section 7.1, because of syncretism and irregularities, there might not be a different written/spoken form for

every realisable set of morphological features. Furthermore, in languages with heavy inflectional paradigms, not all forms are as likely to appear and most will never appear even in a huge corpus. This is typical of French verbs. A regular French verb from the first group can have up to 40 different forms, but in practice only a handful of those are ever used and their use highly depends on the genre of the corpus.

Now that we have seen how important inflectional morphology is in encoding syntactic and semantic information, we shall see how one can measure its richness.

7.2 Measuring Morphological Richness

As we have just seen, morphological richness has direct implications on NLP tasks and on how models and representations used to solved them are learned. Quantifying morphological richness is thus of prime importance as well. And while the definition of morphological richness is fuzzy, it is nonetheless measurable to compare the relative complexity of different languages. Then, measures of morphological richness can be used to inform the choice of a certain model or certain representations in order to solve a problem for specific languages.

Before presenting some of those measures, we shall note that while they are used as measures of a language's morphological richness, they are often based on data and as such only measure the morphological richness of the data. Because those measures are used to draw conclusions about languages, this has three important implications. First, when using those measures to compare languages, one should try to have corpora as similar as possible in those different languages, otherwise the differences might not be due to actual language differences but to domain difference. Wall Street Journal English is only so close to Social Media English. Then, one should use a corpus as large and diverse as possible if he were to draw conclusion about the language in general. Eventually, even with a huge corpus, spoken languages are only so close to their written forms. This means that those measures would at best inform us about the morphological richness of written languages and not so much of spoken languages.

7.2.1 Related Work on Measures of Morphological Richness

Several measures have been proposed to account for morphological richness. They can be either derived from theoretical considerations like the number of cases a noun can inflect for in a language, or derived from data instead like the number of words are necessary to express a certain idea. They are based on different intuitions about morphological richness, but have been shown to be correlated and to converge to the same conclusions [BRKS16] as to which language is more or less morphologically rich. We review some of them in this section.

A very intuitive measure of the morphological richness of a language is the **type/token ratio** (**TTR**). In simple words, if a language has a rich morphology, then it will have more forms and each form will appear less often than in a poorer language. For example the English word *are* corresponds at least to the four French forms *es*, *sommes*, *êtes* and *sont*[2]. Thus we expect to see it more often than any

[2]The English form *are* can also replace other French words, most notably *a* (has) in the multiple word expression *il y a* (*there is/are*) that literally translates as *he there has*.

of the four forms it can stand for.

Given a text T drawn from a vocabulary $\mathcal{V}$, let $freq_T(\mathcal{V}_i)$ be the number of occurrences of $\mathcal{V}_i$ in T. The token/type ratio is:

$$TTR(T) = \frac{|\mathcal{V}|}{\sum_{i=0}^{|\mathcal{V}|} freq_T(\mathcal{V}_i)} = \frac{|\mathcal{V}|}{|T|}.$$

It is equal to the number of words in the vocabulary divided by the total length of the text.

Another possible measure is the **word entropy** as defined by Shannon [Sha48]. The word entropy is a measure of how much information is needed to disambiguate a word. The intuition here being that the more forms a language has, the more information is needed to tell forms apart. Given a text T draw from a vocabulary $\mathcal{V}$, let $\pi(\mathcal{V}_i)$ be the probability of seeing words $\mathcal{V}_i$. Then the total entropy of the text is:

$$H(T) = \sum_{i=1}^{|\mathcal{V}|} \pi(\mathcal{V}_i) \log_2(\pi(\mathcal{V}_i)).$$

On top of those two common measures, Bentz et al. [BRKS16] also investigate less common measures. A **word alignment based measure** uses two parallel corpora aligned for words in different languages to estimate a relative complexity measure. The idea here being that what is expressed via morphology in a language might be expressed via syntax in another and thus use more words, thus the language with more morphology should use less words than the other. Compare French and English futures *mangera* and *will eat*, and comparative adjectives *plus grand* and *taller*. Assuming we go from French to English, the former is a so called one-to-many alignment and the latter a many-to-one. By comparing the number of one-to-many with the number of many-to-one, one can measure the relative complexity of two languages. This measure is relative and asymmetric by nature because of its use of bilingual alignments, but if one has access to parallel corpora in several languages, by selecting some pivot languages then one can use it to effectively compare languages.

The **relative entropy of word structure** measures the information content of words inner structure. Given a text T and an alphabet $\mathcal{A}$, it is measured as the difference of the character entropy of the original text T and a scrambled version T'. The scrambled text T' is made by replacing every token by a sequence of random characters of equal probability of the same length as the original token. This measures the information stored in words via morphological regularities.

They also consider a measure based on theoretical typological information. The idea being that one can estimate the complexity of a language morphology by looking at high level features like the complexity of the case system or the number tenses exhibited by verbs. For example, regarding noun inflection, Finnish has some 15 cases, Latin has 6, German and Icelandic have 4 and French has none. This right-away shows that Finnish will have a richer morphology than say French. By aggregating this kind of information we can have a feeling of how complex is a language morphology. They use data from the World Atlas of Language Structure (**WALS**) [DH13] to estimate a morphological complexity measure. Different features are associated to morphological complexity scores. Ordered features are directly assigned increasing scores (binned number of cases directly maps to integers, for example having no case has a score of 0 whilst having 2 has a score of 1 and having more than 10 a score of 7) while unordered features

receive a hand crafted score reflecting their inherent contribution to morphological complexity.

Bentz et al. show that with enough data, those different measures tend to converge to the same conclusion regarding languages morphological complexity at a large scale. However, they do not measure the same phenomena. **TTR** and **word entropy** by focusing on word types, are directly impacted by irregular paradigms and spelling variations. However, they miss both form syncretism and homographs[3]. **Relative entropy of word structure** is impacted in different directions by irregularities and by derivational morphology because of its being based on character level information. On the one hand, irregularities increase character entropy with structures that are otherwise seldom in a language. On the other hand, derivational morphology creates new words that are as (ir)regular as their stem word thus reducing entropy. Compare English *know* and *recognise* with French *connaître* and *reconnaître*. The **alignment based measure** is interested in the relative analytic construction frequency between languages. And while less morphology often correlates with more analytic construction, this measure does not look at morphology at all. Finally, the typological measure (**WALS**) does not consider actual forms neither but takes a higher level stand point. Indeed, all those measures consider different aspect of human languages which are indeed all linked to some degree.

7.2.2 Form per Lemma Ratio

Another measure of morphological complexity very related to the type/token ratio not discussed by Bentz et al. is the **form/lemma ratio**[4]. The form per lemma ratio (hereafter noted **F/L**) measures the averaged number of form a lemma can take in a corpus. Let T be a text, $\mathcal{V}$ the vocabulary of forms, $\mathcal{L}$ the vocabulary of lemma used in T and $form_T : \mathcal{L} \to \mathcal{V}^*$ the function that returns the set of forms a lemma appears in text T.

$$F/L(T) = \frac{\sum_{l \in \mathcal{L}} |form_T(l)|}{|\mathcal{L}|}.$$

This measure requires corpora annotated with lemma or the use of a reliable lemmatiser. However, there are more and more such corpora available for an increasing number languages.

The type/token ratio is the inverse of the average number of time a type (a form) appears in a text. The form/lemma ratio measures the average number of forms a lemma takes in a text. Those are slightly different because, as we have seen, inflectional morphology creates new forms, thus a language with a high number of form per lemma is likely to have a high number of forms (types) in the first place and thus high TTR. However, a very analytic language with very poor morphology if at all, could use lots of different words (types) like adpositions and adverbs in its syntactic constructions, thus giving a high TTR with a low F/L. In fact, this is a difference we expect to see between a long lived analytic language

[3]Homographs are different words that happen to have the same spelling like French *son* (noise) and *son* (his). Syncretisms are different forms of the same word that happen to have converged over time and to have lost part of their information such as French *je chante* (I sing) and *il chante* (he sings).

[4]This measure is not very popular amongst computational linguists, maybe because it needs lemmatisation of the data, but is used in corpus linguistic [GP09]. However, it has been used in linguistic analysis of Russian [SUW13] or Yiddish [AR13] amongst other.

like Mandarin Chinese and new languages like creoles and pidgins. Creoles tend to be morphologically poorer than the languages they take their vocabulary from and in the same time have a smaller vocabulary than long lived languages [Pat04]. Thus with both a low form per lemma ratio, Chinese should have a higher TTR than Tok Pisin for example.

We have also mentioned earlier that a language need not use the complete spectrum of morphological inflection at one. An example of this is the lack of dedicated comparative/superlative forms for adjectives and adverbs in Romance languages whilst English (and the other Germanic languages) have them. Which contrasts with the rather impoverished verbal morphology of modern English (and other Germanic languages) as compared to the Verbal systems of Romance languages. This is important since a language could in principle have inflection for only one class of word but a very productive for that class indeed. We propose thus an extension to the form/lemma ratio in the **form per inflected/inflectable lemma ratio**, which indeed measures the average number of form a lemma takes in a text, only for those lemmas that appear in at least 2 forms. With the same notation as above, let $\bar{\mathcal{L}} = \{l \in \mathcal{L} \mid |form_T(l)| > 1\}$ be the vocabulary of lemmas that appear in more than one form in T.

$$F/iL(T) = \frac{\sum_{l \in \bar{\mathcal{L}}} |form_T(l)|}{|\bar{\mathcal{L}}|}.$$

This is an approximation since an inflectable lemma could still appear in only one form and thus be ignored but with a large enough text the bias should be smoothed out. Furthermore, this might even be a good bias as a lot of words that appear rarely (the long tail of the Zipfian distribution) are indeed inflectable in theory (for plural say) but never are in practice.

Those measures (F/L and F/iL) are interesting in accounting for the morphological complexity of languages. However, they only consider morphology per se and not its interaction with other linguistic phenomena and they are blind to syncretism. Thus, they do not tell much about the role of morphology in encoding syntax or semantics for example. We will show through experimental results that indeed those measures are of no help when discussing parsing performance.

Because we use morphological information as input to our parsing models, it would be useful to know exactly what is the role of morphology in encoding syntactic information, especially for those languages that rely a lot on morphology in general. The following section will discuss the possible implications of using morphology in dependency parsing and see how we can measure its role in different languages.

7.3 Morphological Richness in Dependency Parsing

As we have seen earlier, inflectional morphology encodes linguistic information in word forms. Thus languages with richer morphology will have more word forms than languages with poorer morphology, and more forms means more parameters to estimate and less occurrences on average of each form.

The UD project has released a set of parallel corpora available in different languages. The same text in English contains 5165 word types[5] out of which 3276

[5]In section 3.2, we defined a type as a word in isolation, while a token is a word in context.

(63.4%) appear only once, in French has 5767 types with 3790 (65.7%) appearing only once and in Finnish has 7399 types out of which 5988 (80.9%) appear only once.

In the same time, more information in words correlates with freer word order. Those two factors have shown to be detrimental to parsing results in earlier multilingual parsing campaigns, such as CONLL shared tasks on dependency parsing [BM06, NHK+07]. And those were also the main concerns and what researchers addressed in early works on syntactic analysis of morphologically rich languages [TSG+10], by proposing methods for encoding morphological information, reducing the impact of data sparsity and unseen words and coping with free word order.

Morphological analysis can be used as a mean to reduce data sparsity. And we expect it to show real improvements for those languages that have a lot of forms with few occurrences like Finnish. Most parsers (as most NLP systems) already use part-of-speech information to reduce sparsity in an attempt to generalise beyond word forms. Using morphological attributes is a way to extend beyond sole POS by providing more and finer information.

In the same time, if morphology is just a matter of increasing data sparsity and nothing more, maybe one can find other ways to reduce sparsity without relying on morphology at all like by using lemmas instead of forms for example. We investigate this in the experiment later on. This would mean one of two things. Either, lexical semantics as encoded by the lemma and morphological information are redundant (which is unlikely as we said earlier), or morphology encodes information that is not relevant to syntactic analysis, such as phonological information or semantic information. And indeed, morphology encodes information in forms and that information might as well be useful for syntactic analysis. If that is the case, then using morphological information as input for our parsers should improve their performance, beyond what merely reducing sparsity (with lemma for example) would do.

However, while we can expect to improve parsing results by reducing sparsity for languages with rich morphology, there is no guarantee that their morphology encodes syntactic information to the same degree if at all. For example, in the two Latin sentences *Petrus equum vidit* and *Petrii equum vidit*, the difference in form from *Petrus* to *Petrii* not only changes the meaning of the sentences from *Peter saw a horse* to *He saw Peter's horse* but also changes the syntactic analysis in that in the first sentences *Petrus* is subject and depends on the verb *vidit*, whilst in the second, it is a genitive depending on *equum*. However, in *Petrus equum videt* the change from *vidit* to *videt* only changes the time reference of the verb (*Peter is seeing a horse*) but not the syntactic structure of the sentence, *Petrus* is still subject of *videt*. In fact, while in the second example we have a pure semantic change, in the first example the change is primarily of syntactic order, the semantic change is merely a consequence of the change in the syntactic structure. This shows that not all forms or rather form changes encode useful information for dependency parsing.

Before going further, we should also note something important about morpho-syntax and grammatical cases. Grammatical cases might seem the most if not the only morpho-syntactic grammatical feature. While this is mostly true for our well known dependent-marking Indo-European languages, this is not true for all languages of the world. First of all, number, gender are also morpho-syntactic

In the previous sentence, there are two tokens *word* (in context) both of which share the same type *word* (in isolation).

when adjectives and determiners agree with their governing nouns. However, they are less morpho-syntactic than cases as they are usually not enough to trigger dislocation [Bre98]. For example, we do not say in French *"j'ai mangé des hier fraises délicieuses" (I ate some yesterday strawberries delicious)*, even though the determiner *des* is agreeing (in number) with its noun *fraises*. Likewise, *"j'ai mangé des fraises hier délicieuses" (I ate some strawberries yesterday delicious)* has only the interpretation that the strawberries were delicious yesterday whenever I might have happened to eat them, with *hier* attaching to *délicieuses* and thus not being an actual dislocation. The interpretation where *hier* attaches to *mangé* is ungrammatical. In fact, we can even add a conflicting adverbials or change the tense of the main clause without touching the object and have a perfectly grammatical sentence, showing that *hier* really attaches to *délicieuses*. For example, *"j'ai mangé ce matin des fraises hier délicieuses" (I ate this morning some strawberries yesterday delicious)* is perfectly acceptable. In the contrary, it would be completely grammatical in Latin which turns out to use cases to move word around while still keeping the several interpretations.

Furthermore, there are a lot of non Indo-European head-marking languages where a verb conjugates not only for its subject but also for its direct object and sometimes its indirect object as well. In languages that practice what is called polypersonal agreement, the verb can agree in gender, number and person with its various complements, the role of which being usually given by the position of each marker. For example, in Amharic (a Semitic language), *alläñ* can be translated as *I have him*, the *ä* denoting something masculine singular, while *alluñ* would be *I have them* where *u* denotes a plural irrespective of the gender. *I have her* with the possessed being feminine singular would be *alläčč̈iñ*, where in the three cases *all* is the root and *ñ* denote a first person possessor [App13]. It is also the case in Basque, in Georgian as well as in some Bantu languages of Africa and some North American languages. In those languages, gender, number and person are much more morpho-syntactic than in Indo-European languages.

While the various morphological complexity measures presented above take good account of the morphological richness of languages, they do not measure the syntactic content of that morphology. Thus they can at best tell us what to expect from reducing data sparsity and not from using morphology for its informational content.

We then present a new measure of morpho-syntactic complexity that indeed focuses on the syntactic content of morphology rather than the morphology itself.

7.4 Measuring Morphology Syntactical Information

We propose a measure of morpho-syntactic complexity that aims at accounting for the syntactic role of a language morphology rather than its mere sparsity increasing property. Looking back, at the Latin example from previous section, we want that the distinction between *Petrus* and *Petrii* be taken into account by the measure while that between *videt* and *vidit* be ignored. The measure is based on preferential head attachment.

The **Head Part-of-speech Entropy** (**HPE** for short) is a measure of how much information a token has about the part-of-speech of its governor. Let T be a text annotated for dependency relations. Let $\mathcal{V}_{dw}$ be the vocabulary of delexi-

Head	ADJ	NOUN	VERB	Total
# of occurrence	1	7	6	14
Probability π_w	0.071	0.500	0.429	1
$-\log_2(\pi_w)$	3.807	1.000	1.222	-
Entropy	0.272	0.500	0.524	1.296
# of occurrence	5	400	3	408
Probability π_w	0.012	0.980	0.007	1
$-\log_2(\pi_w)$	6.350	0.0286	7.087	-
Entropy	0.078	0.0280	0.052	0.158

Table 7.1: Computation of the HPE for the delexicalised words PRON:Case=Acc|Gender=Neut|Number=Sing|Person=3|PronType=Prs|Reflex=Yes corresponding to *itself* (top) and PRON:Number=Plur|Person=3|Poss=Yes|PronType=Prs corresponding to *theirs* (bottom) on data from the English EWT treebank.

calised words[6] and $\mathcal{V}_{pos}$ the vocabulary of parts-of-speech. In our case, we compute the average HPE for delexicalised words as we are interested in how much inflectional morphology encodes syntactic information. Let $POS_h : \mathcal{V}_{dw} \rightarrow 2^{\mathcal{V}_{pos}}$ be the function that maps a delexicalised word to the set of POS that its governor can take. Finally, let $\pi_w(pos)$ be the probability that the governor of delexicalised word w be *pos* in text T. Then, the Head Part-of-speech Entropy (HPE) of a delexicalised word w is defined as follow :

$$HPE_w = -\sum_{p \in POS_h(w)} \pi_w(p) \log_2(\pi_w(p)).$$

The HPE of a text is just the average of its attributes sets HPE :

$$HPE(T) = \frac{\sum_{w \in \mathcal{V}_{dw}} HPE_w}{|\mathcal{V}_{dw}|}.$$

To illustrate the computation of the HPE, let us look at two examples from the English EWT treebank from UD 2.2. Let us consider the delexicalised word PRON:Case=Acc|Gender=Neut|Number=Sing|Person=3|PronType=Prs|Reflex=Yes. It is a pronoun, more precisely the reflexive personal pronoun of the third person singular in the accusative case, thus *itself*. It appears 14 times in the train set, out of which it depends 6 times on a VERB, 7 times on a NOUN and once on an ADJective. The computation of its HPE is broken down in Table 7.1 alongside the HPE of the delexicalised word for *theirs*. We can see that while both can depend on the same three POS, the distribution of *theirs* is much more skewed toward nouns resulting in a much smaller entropy.

The intuition behind only looking at the part-of-speech of the governor is that it is both a salient and stable word feature for syntactic analysis. In most languages, except those that allow case stacking, a word's syntactic properties depend mostly on its direct governor. English is a good example, despite its poor morphology, pronouns mark cases. In the three sentences *She knows him*, *She has known him* and *She was scared of knowing him*, *him* is always in the accusative case and always the direct object of the verb *know*, irrespective of its appearing as the finite form

[6] A delexicalised word is a set of morphological attributes with a part-of-speech as defined in section 5.2.

knows, the past participle *known* or the gerund *knowing*. And this is the primary use of the accusative/dative in English to show attachment of a pronoun to a verb as its object (whether direct or indirect) and this irrespective of the verb form[7].

As a measure of how much syntactic information is encoded in the morphology, HPE should allow us to distinguish languages that will benefit from using morphological clues for parsing from those that just benefit from sparsity reduction.

It is worth mentioning that as a measure of preferential head attachment, HPE misses some syntactic information that can be encoded by head marking phenomena. Declension is a typical example of dependent marking phenomenon. The word *Petrus* changed form depending on its relation to its governor and the relation are marked on the dependent. Conjugation however, is a head marking phenomenon. The governor changes form according to the relation its dependents (not necessarily all of them) have with it. Compare, *you eat* and *he eats*. The verb carries the mark of its subject. Similarly, in French and Italian, there are cases where a direct object appearing before a verb also changes its form. Compare, *Le mail (m.) que je lui ai écrit* and *La lettre (f.) qu'il m'a écrite*. The auxiliary carries the mark of its subject and the participle the mark of its direct object.

In European languages, head marking is reduced to verbs and often also associated with either dependent marking (cases) or word order, thus it does not carry a lot of information. However, there are languages mostly in the Americas and in Papua-New Guinea and Australia where it plays a more important role. In Tzutujil, a Mayan language, verbs carry marks of both the subject and the object whilst neither the subject nor the object has any specific marker of its status [DH13].

The main difference between head marking and dependent marking from dependency parsing point of view and morpho-syntactic complexity is that dependent marking is simpler to analyse than head marking as words have only one governor but can have several dependents. It is not straightforward to extend HPE to take head marking into account.

An interesting feature of HPE is that it does not rely on any parsing algorithm or architecture, thus making it possible to estimate the morpho-syntactic richness of a language independently of any parsing framework[8]. However, despite its being independent of any parsing algorithm, HPE is still based on some annotated data and is thus dependent on the annotation scheme. Thus it seems normal to wonder how much of an impact the annotation scheme has on the measure. We investigate this further in the next section.

7.5 Annotation Scheme Design

When crafting an annotation scheme for dependency parsing there are a number of factors to consider like the type of information that should be included beside of plain word forms (lemmas, parts-of-speech...) or ways to treat ties (what is

[7]The subject is treated a bit differently though. In main clauses and in subordinate with finite verbs, it is expressed with the nominative case as in *I am here* and *You know (that) I were there*. However, for non finite forms, both the accusative case and the genitive case can be used, with arguable a slight difference in emphasis. *I am tired of him being silly*, *I am tired of his being silly*.

[8]If one had access to two parallel corpora consistently annotated in two different languages, then the score of a parser using only morphological information would be a measure of their relative morpho-syntactic complexity. However, it would depend highly on the parsing framework (projectivity/non projectivity, edge representation, parsing algorithm...) and a different setting could give different results on the same data.

the head of a conjunction). Here we will discuss three orthogonal choices (some of which are not specific to dependency parsing) that impact parsers as well as measures of morpho-syntactic richness that depend on the annotation scheme. In the experiment section, we will quantify their impact with empirical measurements.

7.5.1 Part-of-speech Tags

An important design choice is the type of side information available on top of the forms and the dependency structure. And how this information will be encoded. Usually, words in treebanks are annotated with their part-of-speech (which set to use is also a design choice) and to some extent with morphological analysis. In the UD project, words are also annotated with lemmas. Those questions are not restricted to dependency parsing but are of prime importance for it. For example in the Penn Treebank, words were morphologically analysed just as in the UD treebanks, however the analysis was conflated into the POS tags set. The equivalent of UD Verb,{Mood=Ind, Number=Sing, Person=3, Tense=Pres, VerbForm=Fin} is VBZ in the Penn treebank. Of course, some researchers have proposed methods that share information between various related POS (like adding coarse-grained POS such as VB for all verbs), but those strong choices suggest to treat different verb forms independently from each other.

This first choice is important since parsers will need to learn not the use of verbs, but the use of verbs in third person, the use of gerunds, of participles and so on, for as many extra POS added to represent some morphological variation. Regarding morpho-syntactic richness measure, this would also be misleading, for as we have mentioned earlier, accusative (direct objects) attach to verbs irrespective of their morphological analysis. If we were to split verbs into several classes like in the Penn Treebank, we would artificially increase the HPE. Likewise, if we were to merge lexical verbs with auxiliaries and modal verbs, we could also miss some important distinctions.

In order to test this, we will look at the parsability of different parts-of-speech sets and compare it to their respective HPE on the same corpus.

7.5.2 Word Tokenisation

An orthogonal design choice to the previous one is what is to be considered a word. Again, this is not a problem specific to dependency parsing, but important in NLP and in linguistic in general. This actually encompasses several related problems. We are used to consider a word a sequence of characters separated by spaces before and after. To that we add a few exceptions for the treatment of punctuation tokens. However, this only works for scripts that use white space to mark word boundaries. This sounds like a circular argument, but let us consider Asian scripts. On the one hand, in Chinese, words are not separated by spaces whatsoever. On the other hand, in Tibetan and Vietnamese, the separation is at the level of syllables (with spaces in Vietnamese and dots in Tibetan).

Usually, tokenisation (word separation) is considered solved before applying dependency parsing. However, this does not address the problem of contractions that are also common in European languages. Shall the English *don't* and *cannot* be treated a individual words or shall they be analysed as *do not* and *can not*? The fact that the merger is only optional in English points toward the second solution. In French, *au* and *du* result from the compulsory merger of a preposition and the

masculine definite article (*à le* and *de le*), but their feminine counterparts *à la* and *de la* are not merged. This also points toward treating them as separate words. That is indeed the convention taken in the UD project. In those cases, there are some formal reasons to choose one design over another.

A certain number of languages use affixes to mark possession on nouns amongst them are Hungarian, Burusashki and the Semitic languages. Despite the frequency of this phenomenon, the UD project has not yet chosen a unified treatment of those markers. In Hungarian those endings are treated as morphological markers and are thus expressed under the feature category as Number[psor] and Person[psor]. However, in Hebrew those endings are treated as merged words. As we mentioned earlier, the form *'avodati* is treated as *'avodah shel ani* (job of me). This is disputable for at least three reasons. From a synchronic point of view, Hebrew has another way of expressing possession indeed using the (inflected) preposition *shel* after a definite noun, while in the inflected nouns there is no clue of neither *shel* not the definite article *ha*. From a diachronic point of view, possession inflection is an old feature of Semitic languages that is also found in Arabic and Amharic for example, thus one would expect the original suffixed pronouns to be fully grammaticalised as bound morphemes by modern times. Eventually, from a multi-lingual parsing point of view, treating those suffixes as separate words both artificially increases the number of edge in a structure with and conflicts with the idea of treating similar features in different languages as similarly as possible in the annotation scheme.

By artificially merging and splitting words, one changes the number of vertices and edges in the dependency structure which can impact parsing results. Furthermore, one redistributes syntactic information in a way that may both impact morpho-syntacticity measures and parsing algorithms.

To test this, we will look at the parsability of a corpus with varying tokenisations and compare it to their respective HPE.

7.5.3 Dependency Scheme

Eventually, the last design choice orthogonal to the two precedent is the syntactic/semantic nature of dependency relations and what words are considered important in the structure. For example, in a compound verb phrase that contains an auxiliary and a participle, which word shall be the governor? Let us take the sentence *He has eaten a hamburger*. The syntactic view is to choose the syntactic head of the phrase, thus *has*, as it is the word that carries the inflection and the syntactic information. The semantic view on the contrary is to choose the semantic head, *eaten*, as it is the word that carries the meaning of the phrase. Similar questions arise for noun phrases, where some have argued in favor of the determiner being the syntactic head. For example, the first version of the Turin University Treebank was annotated with determiners governing noun phrases [BLVL00].

The dependency version of the Penn treebank was annotated with syntactic dependency rules. The UD project has chosen semantic rules for it is more relevant when dealing with several languages at once. For example, where a language uses a compound tense, another may only have a simple tense (where semantic and syntactic heads are the same), likewise where a language uses an adposition, another can use a declension thus only retaining the semantic head. Semantic dependencies tend to make much more stable structures across languages (as well as through paraphrasing and summary). However as their name suggests, semantic

dependencies tell less about syntax and morpho-syntactic properties of a language. For example, because some languages lack of an overt copula (*to be*) in predicative use, adjectives are semantic heads, their syntactic status depending on the presence of the copula. In the English sentence *Peter is tall*, *tall* is the semantic head while *is* is the syntactic head. Thus, in English, despite sentences always having overt finite verbs, a subject is not consistently governed by one. It can as well depend on an adjective as we have just seen, a noun (also in predicative use), a participle or even an adposition amongst other. In syntactic dependency style, a subject would always depend on a finite verb, making for a more coherent syntactic analysis inside the language.

Different dependency schemes are likely to have an impact on both parsing accuracy and morpho-syntactic measurements. However, whilst some treebanks already have annotation with different tag sets and we can thus use them right away, and whilst some treebanks are also annotated with merged syntactic words making their analysis quite easy, it is not straightforward to change of dependency scheme.

For example, considering the change from semantic head to syntactic head in verb phrases, one needs to find the appropriate auxiliary to become head of the verb phrase (there might be several candidates), then one needs to change the head of some but not all from semantic to syntactic head. Typically, subject, adverbials and other verbs will attach to the syntactic head, however, object like complement might still attach to their semantic head. Take the sentence *I have eaten strawberries*, assuming syntactic head parsing, *I* and *eaten* would both be governed by *have*, however, *strawberries* could still depend on *eaten* or on *have*. Furthermore, to keep the annotation coherent, a whole lot of other changes need be applied to the treebank. For example, copulas (the *be* verb) should also become head of attributive noun phrases and adjectives. In *You are tall*, *are* would become head, no longer *tall*.

For all these reasons, we keep this investigation for future work.

We shall instead points toward some previous works discussing theoretical motivations and practical parsability of different dependency scheme choices. Bosco et al. [BMM+10] compare different dependency formalisms (one that considers noun as head and one that considers determiner as head for noun phrases) used to annotate Italian treebanks using different parsing algorithms. Gerdes and Kahane [GK16] discuss more specific choices in Universal Dependencies such as the choice of head in verb phrases where the object is the actual predicate (*to give a slap*).

We have discussed three annotation choices that must be considered when annotating a new treebank. By changing the structure of the trees, the parts-of-speech and label distributions and the mere word count, it will have an impact on both dependency parsing results and morpho-syntactic measurements. In the next section, we will quantify this impact via experiments on dependency parsing with varying tag sets and tokenisation rules.

7.6 Experiments

We ran a series of experiments using data from the UD project in order to test the hypothesis that using morphological information plays a double role for parsing in both reducing sparsity and encoding syntactic information. We also tested the idea that not all languages are equal with regard to their encoding of syntax with morphology. To further qualify the impact of the annotation scheme on the

morpho-syntactic complexity measure and its relation to dependency parsing results, we conducted parsing experiments with modified annotations and compared the results to their morpho-syntactic complexity measures.

In this section, we first describe the parsing model we use and then the experiments proper and discuss their results.

7.6.1 Parsing Model

We work with a graph-based dependency parsing architecture very similar to those of the previous chapters. The main difference is that we use a very simple feature vector to represent edges so that we can investigate the actual importance of the information encoded by different word representations.

For a given sentence $x = (x_1, x_2, ..., x_n)$, the vector representation of an edge e_{ij} whose governor is the x_i and dependent is x_j, is defined by the outer product of their respective representations in context. Let $\oplus$ note vector concatenation, $\otimes$ the outer product and $x_{k\pm1}$ be the word just before/after x_k, then:

$$\boldsymbol{\phi}(e_{ij}) = \text{vec}[(\mathbf{x}_{i-1} \oplus \mathbf{x}_i \oplus \mathbf{x}_{i+1}) \otimes (\mathbf{x}_{j-1} \oplus \mathbf{x}_j \oplus \mathbf{x}_{j+1})] \ \in \mathbb{R}^{9d^2}.$$

The vector $\boldsymbol{w}_i$ is a dense representation of length $d \ll \mathcal{V}$, its learning is detailed in next section.

We use the averaged Passive-Aggressive online algorithm for structured prediction [CDK+06] for learning the model $\boldsymbol{\theta}$. Given a score for each edge, we use Eisner algorithm [Eis96] to retrieve the best projective spanning tree. Even though some languages display a fair amount of non-projective edges, on average Eisner algorithm scores higher than Chu-Liu-Edmonds algorithm [CL65] in our setting.

7.6.2 Word Representation

We construct separate vectorial representations for lemmas, forms and morphological attributes, either learned via dimensionality reduction of their own co-occurrence count matrices or represented as raw one-hot vectors. We also create two extra types by concatenating a word's POS tag and its form (pform) or lemma (plemma). This is a simple way to disambiguate homographs, which in turn should make the dimension reduction more robust.

Let $\mathcal{V}$ be a vocabulary (it can be lemmas or forms or morphological attributes (incl. values for POS, number, case, tense, mood...)) for a given language. Correspondingly, let $\mathcal{C}$ be the set of contexts defined over elements of $\mathcal{V}$. That is, lemmas appear in the context of other lemmas, forms in the context of forms, and attributes in the context of attributes. Then, given a corpus annotated with lemmas and morphological information, we can gather the co-occurrence counts in the matrix $\boldsymbol{M} \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{C}|}$, such that $\boldsymbol{M}_{ij}$ is the frequency of lemma (form or morphological attributes) $\mathcal{V}_i$ appearing in context $\mathcal{C}_j$ in the corpus. Here, we consider plain sequential contexts (i.e. surrounding bag of "words") of length 1, although we could extend them to more structured contexts [BGL14]. Those co-occurrence matrices are then reweighted by unshifted Positive Point-wise Mutual Information (PPMI) and reduced via PCA such as explained in section 5.3.4. For more information on word embedding via matrix factorisation, please refer to [LGD15].

Despite its apparent simplicity, this model is as expressive as more popular state of the art embedding techniques. Indeed, Goldberg and Levy [LG14] have shown that the SkipGram objective with negative sampling of Mikolov's

	da	en	et	eu	fi	fr	got	he	hu	ro	sv
Train	4383	12543	2263	5396	12217	14553	3387	5241	910	8043	4303
POS	17	17	16	16	15	17	14	16	16	17	16
Feats	44	35	58	69	88	36	40	48	73	59	39

Table 7.2: Basic datasets statistics. The first line gives the number of train sentences for each language. The second and third give the number of part-of-speech tags and of morphological attributes for each language.

Word2vec [MSC+13] can be framed as the factorisation of a shifted PMI weighted co-occurrence matrix.

This matrix reduction procedure gives us vectors for lemmas, forms and morphological attributes, noted $\boldsymbol{R}$. Note that while a word has only one lemma and one form, it will often realise several morphological attributes. We tackle this issue by simply summing over all the attributes of a word (noted $Morph(w)$). If we note $\boldsymbol{r}_w$ the vectorial representation of word w we have:

$$\boldsymbol{r}_w = \sum_{a \in Morph(w)} \boldsymbol{R}_a.$$

Simple additive models have been shown to be very efficient for compositionally derived embeddings [ALM17].

7.6.3 Experimental Setting

In the first experiment we train dependency parsers for 11 languages from the UD project, using either words forms, lemmas or morphological attributes as input. Those input can be either used raw, or embedded in a low dimension space before hand (we explain the embedding process below). Basic statistics about those languages are given in Table 7.2.

For each language, ten parsers are trained (two for each type in lemma, plemma, form, pform and gold morphological attributes, one using one-hot input and one using embedded input). Each parser is trained for 10 iterations of the whole training set via the Passive-Aggressive algorithm and the best iteration is kept based on its accuracy on the development set.

We then trained another four parsers per languages using predicted morphological attributes instead of the provided gold. This is to test the viability of morphological parsers when only forms are given and measure the impact of prediction noise.

Each word is represented as a vector encoding prefix and suffix information for the word itself and its left and right neighbours, information about the word length and its capitalisation are also included. Then a multi-class SVM model is trained for each morphological feature (gender, case, mood, tense...) and for parts-of-speech. For each morphological feature, an empty value is added meaning the irrelevance of that feature for that word. We used the SVM implementation of Scikit-learn [PVG+11] with out-of-the-box parameters.

We experimented with two prediction regimes, either using the argmax value for each feature, or a softmax over its possible values. Those two outputs can be used as such (we call it the one-hot regime) or passed down to an embedding matrix. This gives us four parsers: two prediction regimes (argmax or softmax) times two representation regimes (one-hot or embedding). Attributes prediction results are

		Lem	pLem	Form	pForm	Morph
da	one-hot	58.47	62.36	56.92	60.48	**73.76**
	embed	68.33	72.00	70.64	70.30	73.27
en	one-hot	67.05	68.87	65.83	67.49	**76.27**
	embed	75.21	74.04	75.13	75.36	76.17
et	one-hot	44.96	48.80	41.36	43.42	**71.21**
	embed	59.23	57.10	57.36	56.09	70.20
eu	one-hot	60.27	62.32	57.67	60.45	**73.81**
	embed	69.59	70.10	65.64	67.09	73.21
fi	one-hot	56.06	57.04	51.50	52.69	**75.58**
	embed	66.90	68.27	60.38	63.87	73.33
fr	one-hot	70.93	72.02	70.35	71.67	78.67
	embed	71.90	72.90	76.05	77.68	**78.79**
got	one-hot	60.63	61.09	58.68	59.31	**76.88**
	embed	71.14	71.18	68.72	68.63	76.37
he	one-hot	65.70	67.17	67.08	67.50	**77.65**
	embed	72.52	74.44	72.68	72.51	76.95
hu	one-hot	47.32	48.47	44.35	46.04	**69.67**
	embed	51.04	54.01	55.06	50.63	69.38
ro	one-hot	68.69	69.47	67.20	68.04	**76.57**
	embed	72.83	74.17	73.21	73.05	76.32
sv	one-hot	61.30	64.37	58.54	61.86	**76.42**
	embed	73.27	74.13	72.72	71.60	76.02

Table 7.3: UAS scores for parsers using lemmas (Lem), word forms (Form) or morphosyntactic attributes (Morph) representations as features. Lemmas and forms are also disambiguated with their POS giving pLem and pForm. For each language, the top line holds results using one-hot representation and the bottom line holds results using embeddings instead.

given in Table 7.4. It should be noted that the low results in POS tagging are due to the model being very local and not using any sentencial structure. Likewise, for morphological attributes, the empty feature is taken into account in measuring the accuracy and each feature predictor is independent from the other thus in theory allowing nouns to have moods or punctuation to have tenses. Despite its weaknesses, this model is very simple to implement and train.

7.6.4 Results

Assessing the Role of Sparsity Reduction and Morphology

Word forms are the simplest input one can use as they are directly available from raw text. The one-hot encoding of word forms is the simplest representation available as it requires no further processing than indexing words but suffers the issues we mentioned in section 3.2. It will be our baseline (Form and pForm).

In order to assess the role sparsity reduction, we trained parsers using form embeddings (Form and pForm). Embeddings are a good way to reduce data sparsity since as they represent words in dense low dimensional spaces, information can flow between words directly.

Because form encode morphological information, to test the sparsity reduction

effect without the interaction of forms syntactic information, we trained parsers using lemmas (Lem and pLem) instead of forms as another sparsity reduction device. We used the lemmas as provided in the UD data.

Then, to assess the role of morphology as a syntax encoding device, we trained parsers using morphological attributes alone (Morph). There again, we have a parser using one-hot representation and one using embedded morphological attributes.

Sparsity Reduction Unlabeled parsing results are reported in Table 7.3. For completeness but because the analysis carries on to labeled parsing results, they are reported in the appendix. There are a few things to notice here. For lemmas and forms (the first four columns), parsers working on embedded input (bottom row) score consistently higher than those working with one-hot encoding (top row). Similarly, one-hot parsers (top row) using lemmas (first two columns) score consistently higher than their forms using counterparts (next two columns) except for Hebrew (we analyse this result later). Those two facts show that indeed reducing data sparsity in a way (embedding) or another (using lemma rather than forms) is beneficial for parsing for all languages.

Then, we see that except for French where the difference is not significant, parsers using one-hot morphological attributes outperform parsers using embedded morphological attributes. This is likely explained by two factors. First, the number of morphological attributes is very small (a few tens) compared to the number of forms and lemmas. In that sense, one-hot encoding of morphological attributes does suffer from data sparsity, and thus embedding them does not bring the gain seen for very sparse tokens such as lemmas and forms. Second, it might be because of our too simple embedding scheme. Again, for very sparse input, it already does the job of reducing data sparsity, but for already somewhat dense inputs such as morphological attributes we may need a smarter embedding technique to further gain from information sharing.

However, both parsers using morphological attributes consistently outperform the other eight parsers using lexical input (forms or lemmas). Nonetheless, those gains are hardly consistent between languages. Because forms are the easiest tokens to work with as they are given directly by the text, we will focus on them for rest of the analysis. We see that while French and English have gains of about 2% between embedded form parsers and morphological attribute parsers, Hungarian and Estonian have 14% gains and Finnish has up to 15% improvement.

This is important for two reasons. From a typological perspective, French is recognised as a morphologically rich language for its rich verbal morphology. However, it behaves more like English which is a morphologically poor language. Likewise, Romanian and Danish only witness gains of about 3%. Hungarian and Finnish (amongst other) are also morphologically rich languages, but have much bigger benefits from using morphological attributes. This calls for a better explanation than the typical morphologically rich versus morphologically poor language classification.

Then, from an application perspective, those 2-3% improvements of French or Romanian are obtained with gold morphological attributes. In practice, predicted attributes will contain errors that might wipe those 2-3% out, thus rendering morphological parsers less useful than parsers using plain forms. In order to answer this issue, Table 7.4 gives the scores of four parsers using morphological attributes predicted with a simple SVM instead of the provided gold ones.

		Form Emb	Morph One-hot	Morph Emb	Pos	Morph
da	argmax	**70.64**	64.69	64.19	87.37	98.02
	softmax		65.43	65.33		
en	argmax	**75.13**	69.32	69.51	87.24	97.74
	softmax		71.36	70.75		
et	argmax	57.36	57.16	55.53	84.49	96.20
	softmax		**58.76**	57.24		
eu	argmax	65.64	64.51	64.10	86.09	97.56
	softmax		**67.02**	66.18		
fi	argmax	60.38	64.33	62.58	86.14	97.09
	softmax		**66.84**	65.04		
fr	argmax	**76.05**	72.82	72.28	90.60	97.60
	softmax		73.29	73.18		
got	argmax	68.72	69.94	69.29	90.44	96.00
	softmax		**70.86**	70.46		
he	argmax	72.68	71.60	71.51	90.93	97.84
	softmax		**72.75**	71.85		
hu	argmax	55.06	61.80	59.62	88.43	92.76
	softmax		**61.91**	60.64		
ro	argmax	**73.21**	71.36	70.86	90.64	97.76
	softmax		72.03	71.34		
sv	argmax	**72.72**	67.99	67.82	89.23	96.88
	softmax		69.48	68.60		

Table 7.4: UAS scores for parsers using predicted morphosyntactic attributes and morphological attributes prediction accuracy. For each language, the top line holds results corresponding to argmax prediction of attributes, while the bottom line holds results using probability distributions. The second column shows results using one-hot representation (or probability distributions), and the third shows results when the embeddings are used. The scores of parsers using embedded forms is given for comparison. The last two columns report the part-of-speech and attributes prediction accuracy.

	da	en	et	eu	fi	fr	got	he	hu	ro	sv
F/L	1.44	1.39	1.60	2.32	2.19	1.38	2.44	1.83	1.46	2.03	1.59
F/iL	2.80	2.76	3.35	4.29	4.68	3.15	4.20	3.39	3.03	3.76	2.91
HPE	1.07	1.12	0.55	0.51	0.57	0.87	0.60	1.01	0.60	0.71	0.84

Table 7.5: Morphological complexity measures (F/L, F/iL) and morpho-syntactical complexity measure (HPE) for each language as computed on their respective train sets.

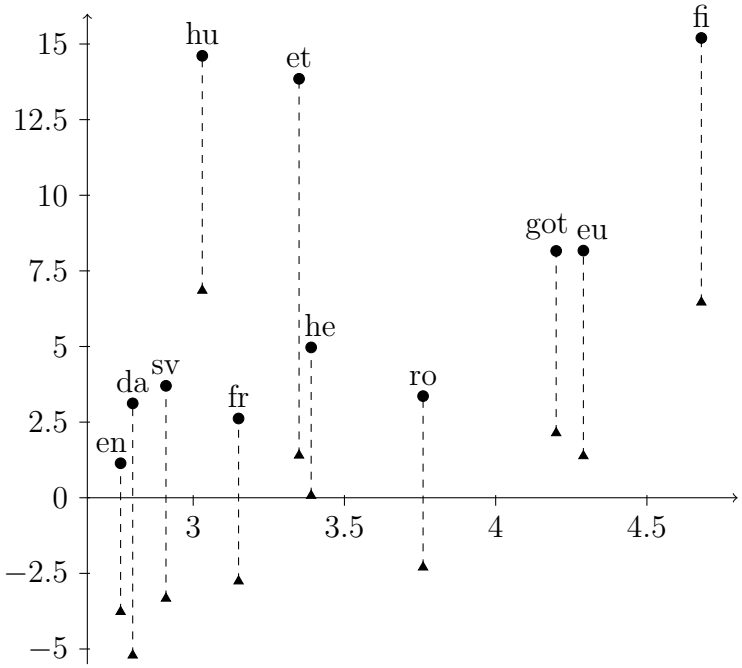
Morphological Information Just looking at the last two columns of Table 7.4[9], we see that one-hot morphological attributes consistently outperforms embedded morphological attributes, which is in agreement with results from Table 7.3. We also remark that soft decisions (bottom row) beats hard decisions (top row) which is in agreement with previous work on multitask learning [SB17].

Now looking at the whole table, we see that as expected, using poorly predicted morphological attributes decreases the parsing scores to the point that for half of the languages, it is no more beneficial to use morphological attributes over plain forms. However for the other half that had already high gains with gold attributes, using predicted morphology is still better than resorting to plain forms. But again, French and Romanian, despite their being morphologically rich, stick with English which is morphologically poor, thus the typical morphological richness dichotomy does not solve the matter.

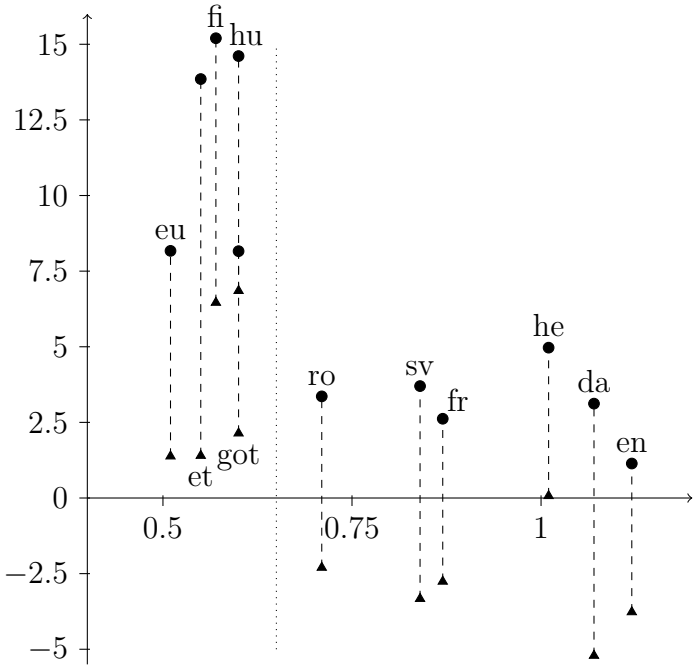
We also notice that there is not clear link between the part-of-speech and morphological attributes prediction scores and the parsing accuracy. There are a few reasons for this. The most trivial and less interesting one is that there are plethora of other conflicting factors such as the size of the training set that we do not take into account. The other reasons are more interesting. Our prediction model is both very simple and local. As the predictors for each feature (and the part-of-speech) are independent, the model has no reason and no way to enforce consistency amongst predicted attributes. Nouns can have moods and pronouns tenses, nothing in the model prevents it beside of the training data. And in fact, this might not even be a problem. As will become clearer in next paragraph, not all morphological attributes are as useful in encoding syntactic information, and as such, they could be very badly predicted and still not impact the actual parsing results as the parser would learn to ignore them in the first place. Finally, there are no constraints about surrounding words attributes. As words attributes are predicted in isolation from other words attributes, the model is unable to capture syntactic information such as agreement between nearby words and so predicting attributes is just about predicting classes and not about learning syntax.

Parsing Accuracy and Morphological Complexity Our hypothesis was that morphology plays a double role in dependency parsing in both reducing data sparsity and encoding syntactic information. We have already assessed the sparsity reduction role. In Figure 7.1, we plot the accuracy differences between parsers using either predicted or gold morphological attributes and parsers using embedded forms with respect to their respective language form per inflected lemma ratio (F/iL) and head POS entropy (HPE). The corresponding morphological complexity measures are reported in Table 7.5.

[9]For this experiment too, labeled parsing results are reported in the appendix.



(a) F/iL



(b) HPE

Figure 7.1: Accuracy differences (y-axis) between parsers using form embeddings and parsers using one-hot attributes, with respect to morphological complexity (x-axis). Dots represent the gold attributes scores and triangles the predicted attributes scores.

Looking at Figure 7.1a, whilst there is a clear trend in the direction for increasing gains with higher F/iL, Estonian, Hungarian and Romanian are somewhat off. Romanian is clearly a morphologically rich languages with regard to its F/iL, but it has surprisingly low gains. On the contrary, Hungarian and Estonian are in the low end of F/iL with morphologically poorer languages and still have surprisingly high gains.

Figure 7.1b on the other hand, shows two clear clusters of languages. Below 0.65 HPE, we have languages displaying rich morphology and that consistently benefit from encoding their morphology (predicted or gold) over using plain forms. Those are the languages that encode a lot of syntax in their morphology and that we could call morpho-syntactic languages, borrowing on the terminology of Kibort [Kib10]. Above 0.65 HPE, we have languages that do benefits from morphology mostly as a data sparsity reducing device, and their benefits are nullified as soon as we had noise to their morphological analysis, and this irrespective of their being morphologically rich or poor. Those languages tend to encode mostly semantic information in their morphology thus we call them morpho-semantic languages. Hebrew might seem an outlier here, as was in Table 7.3, but this is likely due to annotation choices specific to its data that we will discuss in Section 7.6.5.

So far, we have shown that indeed morphology plays a double role in dependency parsing. For all the languages, it behaves as a data sparsity reduction device. Then, for a subset of morphologically rich languages that we call morpho-syntactic languages, it also provides syntactic information directly used by parsers. Furthermore, whilst traditional morphological complexity measures do only consider morphological productivity and not its actual uses and thus do not explain parsing results well, a measure of morpho-syntactic information such as the HPE predicts better parsing performances. Moreover, the HPE is independent of the parsing algorithm one might use and only depends on the annotation scheme underlying the data, which is also an interesting bias in the sense that not all annotation schemes are equally easy to parse.

The fact that our results are quite low compared to that of state-of-the-art parsers is due to the model over simplicity. To let us investigate the role of different word representations, we have ripped most of the information away from the edge representation. Good models encode at least the length and direction of the relation and maybe information about words appearing in between the ends of the relation. Here, we just use information about the triplets of words at each end. Indeed, what we are interested in here is the relative differences between different parsers scores.

7.6.5 Assessing the Impact of the Annotation Scheme

In this section, we consider targeted experiments trying to uncover the role of the annotation scheme in dependency parsing. We investigate the role of the POS-tag set, the impact of treating bound morphemes as syntactic words, and the weight of choosing syntactic heads versus semantic heads in the structure.

Choice of POS-Tags Set

To measure the role of the POS-tag set, we used the same model as above to parse the English treebank of the UD project using different tag sets. The CONLL-U format provides a slot for corpus specific parts-of-speech to keep backward compatibility for corpora that have been adapted to Universal Dependencies guidelines

from an earlier format. The English treebank uses that slot to keep its original Penn Treebank style part-of-speech that include extra morphological information directly into the tag set. We can use it to compare the accuracy of a parser using those more specific parts-of-speech to a parser using less specific more universal parts-of-speech and compare their respective HPE as well.

Several parsers were trained, one using the Universal part-of-speech (UPOS), one using the PTB part-of-speech (PPOS), one using morphological attributes as a part-of-speech (MPOS) and one using one-hot encoding of morphological attributes (Morph). The difference between MPOS and Morph is that while in Morph each single feature/value pair is encoded in a one-hot vector, in MPOS the whole attributes set is encoded as one value. Results are reported in Table 7.6[10]. We also report the respective HPE of each type considering that its head is of each other type. For example the HPE (PPOS) of UPOS is the average entropy of a universal part-of-speech assuming its head were tagged with PTB parts-of-speech. This might not make sense for all combinations, but it gives a sense of the information content of each tags set.

The actual entropy of the one-hot encoding of morphological attributes is actually hard to estimate as it can select only the relevant information in the one-hot vector. However, we can say that it is smaller than its MPOS counterpart where morphological attributes are seen as a single entity, because of the more flexible information.

In Table 7.6, numbers in bold are the HPE relevant to the results in the first row. We see that as HPE decreases, accuracy increases, which is expected as the more information a token has about its governor on average, the easier it is to do parsing. Also as expected, we see that UPOS being more coherent (only one POS for verbs) than PTB POS (several POS for tenses, persons and participles), it leads to lower entropy and is easier to parse.

The only quirk here is the HPE (MPOS) of morphological attributes used as part-of-speech. It corresponds to a higher accuracy than that of the UPOS parser and does not fit the trend drawn by the three other values. In fact, this shows a limitation of the HPE measure as a tool to compare tag sets. HPE is measured for types at the level of corpora, while dependency parsing is done one tokens at sentence level. As the number of possible head types increases, so does the HPE as dependent types have more and more potential heads at the corpora level. However, it gets less and less likely that two types will co-occur in a sentence and thus a dependent token might have enough information at sentence level to get its head right. This is even truer here, given that morphological attributes are actually sentence level token information while parts-of-speech are high level type information.

This shows that HPE is a good tool to compare corpora and languages that use similar high level tag sets. It can both tell something about languages relative complexity and tagging relative "parsability". However, as it is a measure of corpus level type information, it breaks down when considering token level information in the head.

[10]Results discrepancies with Tables 7.2 and 7.3 are due to slight deviations in model implementation, code running on a different machine and annotation corrections between UD 2.0 and UD 2.2.

	PPOS	UPOS	MPOS	Morph
UAS	65.31	68.99	70.86	72.55
HPE (PPOS)	**2.74**	2.98	1.81	>1.81
HPE (UPOS)	1.69	**1.87**	1.04	**>1.04**
HPE (MPOS)	2.93	3.22	**1.96**	>1.96

Table 7.6: UAS for parsers using Universal POS, PTB POS, morphological POS and one-hot morphological attributes (Morph) on the English (EWT) corpus of UD 2.2. Below are reported head entropies when using different dependent and head representations.

Language	#tokens	#contracted words	#surface forms	% of tokens
Basque	72974	0	0	0.00
Danish	80378	0	0	0.00
English	204607	0	0	0.00
Estonian	85567	0	0	0.00
Finnish	127845	486	243	0.19
French	366371	19516	9758	2.66
Gothic	35024	0	0	0.00
Hebrew	169360	71012	31680	23.22
Hungarian	20166	0	0	0.00
Romanian	185113	0	0	0.00
Swedish	66645	0	0	0.00

Table 7.7: Contracted words statistics for 11 languages from the UD project. First column gives the number of syntactic tokens in the train set. Second column gives the number of syntactic words that are merged (annotated as such) in the train set. Third column gives the number of surface forms corresponding to those merged words. Eventually, fourth column gives the fraction of syntactic tokens that are hidden in the surface forms computed as $\frac{\#merged - \#surface}{\#tokens} \times 100$.

Syntactic Words: The Case of Hebrew

Now that we have considered the impact of the tag set on the measure of morpho-syntactic complexity in relation with parsing accuracy, we turn to the second consideration namely the choice of what is to be considered a word and what a morpheme.

As we have seen earlier, the Hebrew treebank from the Universal Dependencies project makes some strong choices about what is to be considered a syntactic word. Some of those choices are conflicting with cross-lingual consistency aimed at by the UD project like the treatment of possessive inflections and some are linguistically disputable like the treatment of the prefix *ha* as an actual determiner. However, even disregarding linguistic soundness of such choices, raw numbers already give us some insights. Table 7.7 reports some statistics on word merges for the train sets used in Section 7.6.

The number of merged words is the span sum of spanning indices as given in the UD train sets. For example French *au* is annotated as spanning over indices i-$i+1$ because it actually represents syntactic words *à* at index i and *le* at index $i+1$, and thus counts for 2 words. The number of surface forms is the number of such spanning indices. Here, *au* counts for 1. The number of tokens is simply the raw

count of syntactic tokens in the train set. Because unmerged words also make up for syntactic tokens and edges, we measure the percentage of edges/words that appear in the process of splitting merged words. Thus each surface form that results from merging n words actually adds $n-1$ edges/words to the original surface token counts. The percentage of thus created edges/words is then $\frac{\#merged-\#surface}{\#tokens} \times 100$.

We see that whilst in French, unmerging words makes up for 2.66% of the total syntactic tokens, due to preposition merging with determiners for euphonic reasons. German, which we did not use in our experiment is similar to French with 1.72% of syntactic words coming from unmerging preposition/determiner pairs.

Languages can have 0 merged words because they genuinely do not merge words or because merged words are not annotated as such yet. We made a simple script to measure this, assuming two syntactic words were actually merged if there were no space between them in the original surface sentence, none of them is a number, a symbol or a punctuation and if their boundary is not marked with an hyphen or an apostrophe. Thus English *don't* or *cannot* are considered merges of *do n't* and *can not* but not *I'm*. With this method, we see that English has actually 0.57% of merged words, which is still quite low and mostly accounts for negations.

In Hebrew, unmerged words account for almost a quarter of all the syntactic tokens. Which is about 9 time as much as in French. Furthermore, it turns out upon further investigation that merged words have only two dependency patterns. Either they all attach to the same head, as is the case for French *du (de le)* and *au (à le)* or they but one attach to syntactic words inside the surface token such as in the second sentence of the Hebrew dev set *veharevakha* where *ve* and *ha* both attach to *revakha*. This thus makes for somewhat easier dependencies than usual.

That might explain the odd results we witness for Hebrew in the previous experiments as what is considered a morpheme and thus need to be learned by the morphological tagger in other languages is treated as a word and provided as such in Hebrew thus making those words/morphemes less noisy in Hebrew than in other languages. Furthermore, because those words/morphemes attach in a very simple and regular pattern to their head, it artificially increases the parsing accuracy by adding many easy dependencies.

In order to test this hypothesis, we made an experiment on a modified version of the Hebrew treebank. The Hebrew definite article (*ha*) is always prefixed to every noun and adjective in a noun phrase, so instead of considering *ha* as an independent syntactic word as is done in the original Hebrew treebank, we considered it a marker of definiteness. We remove each *ha* from the treebank and added the attribute Article=ha to its following word morphological attributes. This makes for 13925 *ha* in the train set, so 10.11% of the total token count.

Likewise, most adpositions are also prefixed to their head noun, so we also treated them as cases markers instead of independent words. Out of the 14316 prefixed preposition in the train set, 227 are head of at least one other token thus we did not change them. The remaining 14089 preposition (10.23% of the tokens) were removed and added as an attribute Case=*adp* to their head Morphological attributes with *adp* being the form of the adposition. Then we corrected sentences and trees indices to account for the deleted words. We applied the same transformation to the dev and test sets.

We then parsed the newly created Hebrew treebank and also measured its HPE and compared it to the original treebank. Results are given in Table 7.8. We also report the accuracy on the original treebank, ignoring articles and adpositions.

Table 7.8 shows that the UAS on original treebank ignoring adpositions and

	UPOS	Morph
UAS	68.20	73.56
UAS -ADP -DET	59.70	66.19
UAS Modified	62.11	70.88
HPE	1.67	>1.01
HPE -ADP -DET	1.72	>1.01
HPE Modified	1.71	>0.75

Table 7.8: UAS for parsers using Universal POS and one-hot morphological attributes (Morph) on the Hebrew corpus of UD 2.2 and a modified version for merged adpositions and determiners. Below are reported head entropies.

articles are much lower than the complete UAS. This validates the idea that articles and adpositions are easy to parse in Hebrew and artificially increase the parsing score. We also notice that including those as attributes rather than as independent syntactic words is indeed beneficial as the accuracy increases back when doing so. This is consistent with the evolution of the treebank HPE that decreases as the score increases. A lower HPE meaning that dependents have more information about their governor and thus being easier to parse.

Those results indeed give some indication as to why Hebrew results are off in Figure 7.1, especially with predicted morphological attributes as articles and adpositions are small words with no further morphological analysis and are straightforward to parse. Furthermore, they also assess the usability of HPE as a measure of morpho-syntactical annotation consistency and parsability.

7.7 Conclusion

In this section, we have addressed the question of the intrinsic worthiness of morphology from a syntactic parsing perspective. Through experiments on parsing using various inputs (forms, lemmas, morphological attributes, one-hot and embedded) and with the help of a new measure of morpho-syntactic complexity, we have shown that morphological analysis plays a double role in syntactic parsing. For all languages, morphologically rich or not, it acts as a sparsity reduction device and improves results over plain forms when high quality (gold) analysis is available. Then, for a subset of morphologically rich languages, that we called morpho-syntactic languages, morphology also encodes syntactic information crucial for parsing. For those languages, even poorly predicted morphological analysis still helps over plain forms.

We have also shown that the newly proposed HPE (Head Part-of-speech Entropy), a measure of morpho-syntactical information, is indeed predictive of those behaviours and as such succeeds where more traditional measures of morphological complexity such as the forms per lemma ratio fail, in telling apart morpho-syntactic (Finnish, Gothic, Basque) languages from morpho-semantic (French, Hebrew) and morphologically poor (English) ones. This is interesting from both a computational point of view, as it can be used to inform representation choices, as well as from a linguistic point of view, as it is a new tool to be used in typological studies.

As the HPE does not depend on any parsing algorithm but only on annotated data, we have finally investigated the impact of the annotation scheme (choice of POS tag set and choice of syntactic words) on the measure in relation to pars-

ing results. We have shown that not only was the HPE able to tell languages apart from a typological point of view, it is also predictive of the relative consistency/parsability of similar tag sets and of annotation choices in general. With the general trend that assuming reasonable annotation, parsability increases as HPE decreases. By reasonable, we mean that the annotation should provide some reliable information, as if a corpus was annotated with only one part-of-speech, its HPE would be 0 but it would in the same time be unparsable.

Chapter 8

Conclusion

8.1 Contribution

This thesis has investigated multi-lingual dependency parsing with a special focus on using morphological information as a bridge between languages. The main hypothesis was that when automatically learning models of natural languages syntax, using data from multiple languages at the same time should be beneficial. The basic underlying intuition is twofold: First, languages are not isolated and they share a lot of structure amongst which syntactic structures, whether its due to their common history or to areal convergence or by mere chance. Second, even without sharing a common ancestor (this is yet to be shown), languages display many commonalities and general trends making it interesting to learn from several languages at the same time.

In Chapter 5, following similar works on graph-based dependency parsing [BGL14, Ban15, CZZ14] we added morphological information to the original one-hot feature vector in the shape of delexicalised word embeddings. We showed that it indeed helps improving parsing results when delexicalised word embeddings are trained on mono-lingual data. We then used multi-lingual data to train these delexicalised word representations and showed that it can also slightly improve parsing accuracy of mono-lingual parsers. This showed that even indirect multi-lingual supervision can be beneficial.

In Chapter 6, we turned to more direct multi-lingual supervision, by directly sharing information between parsing models based on their respective languages similarities. We proposed the phylogenetic multi-lingual training method that makes use of a language similarity tree to guide the training of several parsing models at the same time. We showed that phylogenetic training is really beneficial and especially for languages with very few training data, but that lexicalisation is complicated in a multi-lingual setting with different spelling conventions and writing systems. We also looked at the more conventional model propagation method. Despite its being hard to tune because of the many hyper-parameters and the time complexity, it is actually quite useful for languages that have few to no training data, especially when they are high in the tree as there the phylogenetic models are still very generic and shallow.

After using morphology as a bridge between languages and as intermediary information between parts-of-speech and words, in Chapter 7, we investigated the actual roles played by morphology in dependency parsing. Via a series of experiments, we have shown that beside of its being a language bridge for multi-lingual dependency parsing, morphology has two main roles in dependency parsing. First

of all, it is a way to reduce data sparsity. It is both finer-grained than raw parts-of-speech and much more dense than actual words, and it fills the gap left by words in delexicalised parsers. Then, it also directly encodes syntactic information relevant to dependency parsing, typically via cases in European languages, but not only. However, not all languages make the same use of morphology when it comes to encoding syntax. While traditional morphological complexity measures can not predict which languages use morphology in a syntactic way, we proposed a new morpho-syntactic complexity measure called Head POS Entropy, that measures head preferential attachment of morphological items and is able to distinguish morpho-syntactic languages that use morphology for encoding syntax from morpho-semantic ones that do not. Eventually, we discussed the impact of annotation scheme on both parsing results and morpho-syntactic measurements and their correlation.

8.2 Future Works

Multi-lingual natural language processing is still a restricted research field, with most effort going to annotation transfer and learning cross-lingual word representations. While this was mostly due to a lack of consistently annotated data, things are changing, but there remain a certain number of challenges. We review some here.

A lot of work have been done in the field of multi-task learning to automatically learn task relationships, but they are often considered frozen in time. In computational historical linguistic and computational biology, a lot of research is targeting the automatic retrieval of evolutionary history of languages/organisms. It would be interesting to see to which extent those works can be applied and extended to our phylogenetic learning problem in order to automatically learn the similarity structure of languages. In the same time, the phylogenetic tree might not be the best at encoding languages grammatical evolution, but languages and grammars take clearly place in an evolutionary history and considering them frozen in time might be detrimental, especially when we have access to those different time periods (in UD, we now have data in Latin, Old French and Modern French). This is also true in the case of model propagation, where as models are pre-trained, they do not benefit from encoding history but would possibly gain from a sparser and better encoding of language similarity.

Another real problem that we mentioned in the preliminary chapter and that we did not address, is that of cross-lingual morphological feature alignment. Indeed, we used morphology as a bridge between languages, and this requires that languages be annotated consistently with the same morphological features. However, not all morphological features/values need be present in two different languages and even when they are, they need not cover the exact same role and use. Thus, our methods are only as good as the cross-lingual morphological information that underlies them. It would thus be worth investigating means to better align morphological features cross-lingually. Linguists have looked at that problem, regarding notably the definition of grammatical cases [Bla01]. We can imagine using computational methods to attack this problem as well.

Eventually, our analysis of the morpho-syntactic/morpho-semantic dichotomy in chapter 7, was mainly based on dependent marking that use cases. As such, our morpho-syntactic complexity measure is based on head selection. However, not all morpho-syntactic languages rely exclusively on head selection. Actually,

even French verb phrases can be seen as head marked as they agree in number and person with their subject and to some extent in number and gender with their object. But some Caucasian and American languages do it much more widely and consistently. It will be interesting to attack that problem too, once data are available.

Chapter 9

Appendix

9.1 Model Propagation for Dependency Parsing

Model propagation is a simple way to leverage model from several related tasks. Assume we have access to several similar tasks, by similar we mean that their inputs and outputs lives in the same spaces respectively. For example we might need to perform part-of-speech tagging for two or more related languages. Those tasks share both input space (words or character sequences) and output space (POS tag sequences). However, as they use data from different languages, their models will eventually be different. Nonetheless, if languages are close enough, parts of their respective models can be shared to some extent. The extent to which their are close and can share information is encoded in a similarity graph whose vertices are tasks and weighted edges correspond to task similarity.

In the context of federated learning, Bellet et al. [VBT17] propose to let each task (in their case, each user) train its own independent model and then to share those models with similar tasks as encoded by a similarity graph. The final model of each task is then a weighted average of its neighbours models and its own one such that models evolve smoothly along the graph. This means that models of neighbouring tasks should not diverge to much. This shows particularly useful when tasks have only access to a limited amount of data.

More formally, let $\mathcal{L}$ be a set of languages and let $\mathcal{S} \in \mathcal{L} \times \mathcal{L} \to \mathbb{R}_+$ be a measure of similarity between languages, such that $\mathcal{S}_{ij} > \mathcal{S}_{ik}$ if and only if l_i and l_j are closer than l_i and l_k for $l_i, l_j, l_k \in \mathcal{L}$. Assume we already have learned a model $\hat{\boldsymbol{\theta}}_i \in \mathbb{R}^d$ for each language l_i. Then, we want to enforce that similar languages have similar models. This is in fact the basic idea behind Zhu et al. [ZG02] label propagation, where we would see languages models as a form of labeling. However, Bellet et al. propose to add an extra confidence constraint. Whilst, in the original work of Zhu et al. individual model $\boldsymbol{\theta}_i$ departing from its original value $\hat{\boldsymbol{\theta}}_i$ is only determined by its neighbours similarities $\sum_{j \in n(i)} \mathcal{S}_{ij}$ and an hyper-parameter μ, Bellet et al. add an extra self confidence parameter that take into account the intrinsic quality of the model, which the propose to estimate using the number of training samples.

The objective function that they minimise is:

$$\frac{1}{2}\Big(\sum_{i \neq j} \mathcal{S}_{ij} \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^2 + \mu \sum_{i=1}^{n} D_{ii} c_i \|\boldsymbol{\theta}_i - \hat{\boldsymbol{\theta}}_i\|^2\Big),$$

where $D_{ii} = \sum_j \mathcal{S}_{ij}$ is the matrix of degree as defined in section 2.1, μ is the same hyper-parameter as in the original label propagation and c_i is the confidence value

of model $\hat{\boldsymbol{\theta}}_i$. If we note $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_l]$, then solving the above function gives the closed form solution:

$$\boldsymbol{\Theta} = \bar{\alpha}(I - \bar{\alpha}(I - C) - \alpha P)^{-1} C \hat{\boldsymbol{\Theta}},$$

where C is the diagonal matrix where $C_{ii} = c_i$, $\alpha = \frac{1}{\mu+1}$, $\bar{\alpha} = 1 - \alpha$, $P = D^{-1}S$ and I is the identity matrix.

Voting

Instead of using linear combinations of the original models, one can also apply ensemble techniques such as voting to those models. The idea behind voting is that when one has access to several models, instead of merging them into a single model, one can also apply all the models and have them voting for the best possible output. This can be very useful especially when there is no easy way to merge the various models, either because they use different architectures or because they do not combine linearly (like neural networks). Imagine you have a number of graph-based parsers and transition based parsers using different feature functions. Then given a new sentence, each model can predict a structure and all the structures can then be used for voting for the final structure.

9.1.1 Experiments

Setting

There are a lot of hyper-parameters that can be tuned for model propagation. First of all, the graph underlying the propagation is a parameter in itself. We decided to turn the phylogenetic tree into a similarity graph. To try to counter the huge tree imbalance, instead of the more classical path length between two leaves in the tree, we used the depth of their last common ancestor. The effect of this measure is that even languages that are very close in terms of path length will be considered far from each other as soon as their path goes through nodes high up in the tree. For example, with the tree from Figure 6.4, Thai is only 2 edges away from Naija but 6 edges away from Bulgarian, still because both path go through the node World, they will have the same distance. Then we turned that distance into a similarity with a Gaussian kernel of variance 2 and keep the complete graph (those are also hyper-parameters on their own).

We have used the complete graph with a similarity derived from the phylogenetic tree. For two languages l_i and l_j, we note δ_{ij} the maximum depth of the tree rooted at their last common ancestor. Then their similarity is computed as:

$$\mathcal{S}_{ij} = \exp\left(-\frac{\delta_{ij}^2}{2\sigma^2}\right),$$

with $\sigma = 2$. We also tried a version where all the languages pairs have the same similarity, thus discarding any family information.

Other hyper-parameters include the μ parameter controlling for the strength of the original model, the way the confidence is computed for each individual language and the actual averaging/voting mechanism. We set $\mu = 2$. For the model confidence, we tried a simple confidence based on the number of training sentences. Languages with more than 4000 training samples have a confidence of 1. Others have a confidence equal to their number of train samples divided by

4000. Finally, for languages without training samples, we set their confidence to a small value of 0.0001 to avoid having a non invertible propagation matrix. We also tried the plain propagation without confidence of Zhu et al [ZG02].

The main problem we faced was the huge amount of hyper-parameters to consider and the time it takes to evaluate them. Indeed, while linear models can be summed up and thus once their sum as been computed as a new model, only this model needs to score the edges at parsing time, neural models can not be summed up in the same way because of their being highly non linear. Thus, only the final scores can be averaged over a set of models, which in turn imply running all those models. In our case, we have 60 languages with a training set, thus 60 models and as we use a complete similarity graph, each 60 neural models have to be evaluated for each edge, which is very slow.

Then there are a lot more options to investigate such as whether to average the models or have them voting for the final structure. In the case of voting, we can either decide to use the actual tree inference or the surrogate used to train the models. Voting is slow because it requires that each model is evaluated for each edge, but as it is already the case for neural networks it can be another option.

Because of all of those possible options, it is long and difficult to tune the model propagation system. Thus, we have chosen simple sensical parameters rather than looking for optimal ones for the linear models alone and we did not use the neural models.

Eventually, we compared using the last or the averaged independent model of each language.

9.1.2 Results

Table 9.1 reports unlabeled parsing results for the propagated linear models using either the unweighted averaged model or the one weighted with and without confidence parameters with the tree derived similarity using either last or averaged mono-lingual models for propagation. Independent model results are also reported for comparison. The typography is a bit different from other tables here. For independent models (last two columns), bolding takes the whole row into account. The score is in bold if it is higher than all the other UAS scores for that language (and italic if by more than 1 point). For propagated models (six first columns), we only consider the propagated scores ignoring the last two columns. Therefore, Polish (pl) unweighted averaged model using averaged mono-lingual models (79.89) is in bold italic because it is more than 1 point higher than the other propagated scores, but it is still far lower than the independent model score (89.40) also in bold italic.

While the results of the propagated models are quite low, there are a few interesting patterns to notice. First of all, as expected the tree based propagation is slightly better on average than the plain averaged model (without confidence parameters though). However, this in not true for all languages, most notably languages with very few training samples and most Romance and Slavic languages. For Romance and Slavic languages, this might be because of the huge language family imbalance. As most languages in the data are Indo-European, the average model is a good Indo-European model. In fact, Germanic, Slavic, Romance, Greek and Irish and non Indo-European Uralic languages have scores in the 50-70 range, while Indic, Turkic and other Asian languages have scores in the 30-50 range and achieve their best propagated results with phylogenetic information. This is somewhat reminiscent of the Standard Average European hypothesis of Whorf

	Unweighted		Weighted Without Confidence		Weighted With Confidence		Independent	
	Avg	Last	Avg	Last	Avg	Last	Avg	Last
ar	44.6	38.09	**50.07**	46.32	49.8	46.56	***77.32***	71.54
cop	66.43	60.5	***74.61***	72.68	69.06	68.39	***83.25***	80.21
he	51.23	44.23	***62.74***	60.46	61.67	60.16	***80.97***	76.05
bxr [19]	***45.55***	42.91	35.23	33.01	41.62	42.44	31	29.54
eu	47.69	44.61	***60.27***	56.53	59.93	56.66	***74.64***	66.72
af	57.64	51.77	67.1	**67.99**	62.07	63.8	***81.13***	75.43
da	***64.55***	59	58.04	59.2	58.13	59.53	***77.52***	71.43
de	65.91	59.58	**67.15**	64.93	67	65.15	***79.11***	73.54
en	58.6	53.37	**61.83**	60.58	61.25	60.37	***78.26***	71.42
got	54.32	50.66	54.67	51.66	**54.68**	52.3	***75.83***	69.95
nb	***67.19***	61.02	63.95	64.41	63.92	64.51	***83.82***	78.41
nl	60.56	54.23	**61.25**	59	61.04	59.07	***76.23***	68.12
nn	**63.76**	58.36	62.85	62.29	62.73	62.5	***80.78***	74.59
sv	***65.67***	60.04	53.75	57.09	54.2	57.48	***79.59***	73.58
be	***61.29***	57.11	56.04	55.97	56.86	57.29	***74.88***	71.45
bg	***69.08***	62.92	63.2	62.34	63.26	62.35	***84.38***	79.79
cs	***61.48***	54.96	47.08	46.28	47.42	46.6	***77.16***	69.07
cu	53.62	50.78	**55.40**	54.02	55.3	54.06	***78.38***	72.91
hr	***60.64***	53.89	55.39	54.98	55.38	55.04	***80.35***	74.41
hsb [23]	***59.85***	53.52	46.04	40.38	52.47	52.57	48.37	44.64
lt	***44.91***	39.45	41.59	40.38	40.86	40.96	***55.62***	50.95
lv	***55.36***	49.33	51.11	51.5	51.01	51.45	***76.51***	68.46
pl	***79.89***	75.87	74.91	73.73	75.05	74.04	***89.40***	86.49
ru	***59.60***	52.99	54.74	52.8	54.67	52.83	***76.98***	70.69
sk	***65.97***	60.34	57.13	55.08	57.25	55.44	***78.76***	74.73
sl	***69.00***	61.98	58.38	55.24	58.61	55.5	***83.18***	78.32
sr	***60.93***	53.77	56.67	54.94	56.4	55.31	***83.34***	78.15
uk	***60.54***	52.61	51.69	48.65	51.68	48.79	***78.37***	73.47
ca	***63.54***	56.71	53.37	50.06	53.6	50.56	***83.47***	78.1
es	***63.63***	56.83	50.12	43.81	50.08	44.08	***83.31***	77.43
fr	64.65	58.42	66.39	65.41	**66.40**	65.49	***83.31***	78.21
fro	63	59.71	66.49	66.69	66.4	**66.72**	***79.19***	71.96
gl [600]	**63.50**	56.35	59.71	56.42	62.61	63.27	***83.12***	77.67
it	68.68	62.74	**70.06**	67.64	69.92	67.76	***85.69***	80.58
la	45.49	42.56	**46.38**	44.73	46.3	44.82	***62.84***	55.06
pt	**65.14**	58.56	64.24	64.02	63.95	63.92	***83.75***	78.91
ro	***57.67***	51.62	52.84	51.55	52.88	51.85	***79.48***	71.85
fa	41.16	32.63	***50.61***	45.93	48.77	45.87	***77.47***	68.43
hi	37.61	38.74	41.55	43.55	42.04	**44.31**	***87.52***	81.92
kmr [20]	***50.09***	46.25	38.13	35.07	41.61	41.1	40.96	38.35
mr	54.34	52.97	**64.44**	63.94	58.78	60.14	***75.51***	74.49
ur	35.98	35.8	**50.16**	49.55	49.66	49.23	***84.08***	76.03
el	***64.96***	59.38	62.87	59.22	62.05	60.62	***84.90***	79.91
grc	**50.62**	47.53	50.6	49.37	50.6	49.48	***69.42***	59.5
ga [566]	53.77	48.53	***58.49***	56.85	54.3	53.45	***76.46***	72.78
hy [50]	**48.26**	42.35	38.93	36.99	41.89	43.28	***57.78***	52.72
id	55.68	48.53	***70.31***	64.58	68.85	65.09	***81.03***	72.91
ja	28.65	29.54	***69.50***	68.15	66.45	66.73	***86.72***	81.04
ko	31.94	30.86	37.86	**38.82**	36.86	37.96	***62.09***	52.36
kk [31]	49.62	48.45	54.98	53.15	54.31	***56.79***	**56.91**	53.49
tr	33.96	32.35	40.86	42.13	40.81	**42.41**	***58.29***	47.49
ug	37.71	36.05	***47.58***	45.76	45.22	45.92	***65.16***	58.61
et	***54.22***	47.92	42.6	43.76	42.83	44.11	***75.24***	68.96
fi	54.07	50.71	**55.13**	54.72	55.09	54.8	***75.95***	69.21
hu	***59.69***	53.78	49.88	49.64	50.7	51.63	***76.66***	70.54
sme [2257]	49	47.68	**52.11**	50.1	51.34	50.22	***74.05***	66.78
ta	39.52	38.64	59.14	***60.25***	47.42	50.39	***73.70***	66.23
te	66.1	69.66	**81.22**	80.78	74.39	75.07	***86.37***	82.77
vi	44.14	38.13	***51.51***	48.09	45.78	46.39	***60.41***	51.67
zh	33.28	30.35	***49.69***	43.32	46	42.3	***69.99***	56.68
Dev	55.5	50.57	57.06	55.66	55.98	55.26	77.64	71.36
No Dev	52.45	52.35	47.95	45.25	50.02	50.39	58.58	54.5

Table 9.1: Comparison of parsing results for several model propagation settings. The two first columns correspond to an complete unweighted graph. The two second columns correspond to the model of Zhu et al. without task confidence parameters. The two third columns correspond to the model of Bellet et al. with task confidence parameters. The two last columns correspond to the independently trained mono-lingual models. For each setting, we report the UAS results using either last model of averaged model as original model for each language.

[LW41] stating that languages of Europe share some common characteristics that are absent from most languages around the world making European languages peculiar rather than typical.

Regarding low resourced languages, it is interesting to remark that the plain averaged model has a higher score than both the weighted model and the independent ones. This shows that with very few resources, it is crucial not to rely too much on the over-fitted mono-lingual model. This is in agreement with results from the phylogenetic model.

The poor results of the models with confidence shows us that our confidence measure is clearly too weak and/or arbitrary. Maybe a better option would be to use held out data to estimate the parsing score and use that as a confidence score.

The available results are quite low, but it should be borne in mind that there are a lot of hyper-parameters to be tuned in this method and with no doubt, we could have better results with more time. At this stage stage the only certainty, is that model propagation is not an easy out-of-the-box solution for multi-lingual natural language processing, but that it has some promising results for languages with few to no data as we will see again in the following section.

Tables 9.2 and 9.3 report zero-shot parsing results for the same propagation setting as above for languages with and without a training set respectively. For languages that have a training set, zero-shot results correspond to using the same averaging over the mono-lingual models as for Table 9.1 except that we now set the influence of the original model on its own language to 0, thus effectively only using models from all the other languages.

In Table 9.2, results are slightly lower than in the supervised case (Table 9.1) as expected, and they are even more skewed toward the unweighted model without confidence. This is again because of the huge data imbalance. It is really striking to notice that whilst there are really score drops for Semitic and many Asian languages, for Indo-European (mostly Slavic) and Uralic languages, scores barely decrease at all.

Table 9.3 is just a more complete table for zero-shot parsing results of languages without a training set. The analysis of previous tables carries on to this one as well.

Figure 9.1 represents the weight matrix resulting from propagating on the similarity graph described above without confidence parameters. Without confidence, the objective function to minimise is:

$$\frac{1}{2}(\sum_{i\neq j}\mathcal{S}_{ij}\|\boldsymbol{\theta}_i-\boldsymbol{\theta}_j\|^2+\mu\sum_{i=1}^{n}D_{ii}\|\boldsymbol{\theta}_i-\hat{\boldsymbol{\theta}}_i\|^2).$$

It has the closed form solution:

$$\boldsymbol{\Theta}=W\hat{\boldsymbol{\Theta}}=\bar{\alpha}(I-\alpha P)^{-1}\hat{\boldsymbol{\Theta}},$$

where $\alpha=\frac{1}{\mu+1}$, $\bar{\alpha}=1-\alpha$, $P=D^{-1}S$, I is the identity matrix and $\hat{\boldsymbol{\Theta}}$ is the matrix resulting from stacking all the original models. With our setting $\mu=2$, this gives : $W=\frac{2}{3}(I-\frac{1}{3}P)^{-1}$. The similarity matrix S and thus the degree matrix D are defined in terms of maximum depth under the last common ancestor in the phylogeny passed through a Gaussian.

	Unweighted		Weighted Without Confidence		Weighted With Confidence	
	Avg	Last	Avg	Last	Avg	Last
ar	43.67	37.44	45.54	**45.79**	44.92	45.67
cop	**65.86**	59.94	64.92	64.09	65.01	64.46
he	**50.70**	43.58	50.00	49.69	48.58	48.78
bxr [19]	***45.62***	43.17	42.02	42.90	41.73	42.57
eu	***47.07***	44.19	44.17	44.14	44.59	45.09
af	**57.30**	51.24	55.88	56.54	55.82	56.65
da	***64.29***	58.65	61.66	62.64	61.45	62.55
de	**65.44**	58.94	63.35	64.57	62.64	64.31
en	***58.36***	53.14	55.58	56.79	55.33	56.58
got	***54.11***	50.30	50.21	50.65	49.70	50.65
nb	***66.92***	60.76	63.23	64.62	63.08	64.56
nl	**60.35**	53.92	59.11	60.20	58.46	59.97
nn	***63.46***	58.11	60.17	61.35	60.02	61.21
sv	***65.54***	59.92	62.18	63.56	61.94	63.47
be	***61.42***	57.07	56.60	56.86	56.62	56.88
bg	***68.86***	62.89	62.65	62.56	62.65	62.94
cs	***61.51***	55.00	54.38	53.80	54.65	54.72
cu	***53.30***	50.46	49.89	49.39	49.94	49.93
hr	***60.33***	53.62	52.68	52.34	52.60	52.47
hsb [23]	***59.82***	53.60	52.18	52.24	52.44	52.56
lt	***44.71***	39.12	40.33	40.48	40.40	40.13
lv	***55.23***	49.10	48.93	48.91	49.15	49.68
pl	***79.85***	75.80	75.46	75.36	75.69	76.17
ru	***59.45***	52.85	52.34	52.43	52.13	52.41
sk	***65.93***	60.35	60.65	60.18	60.73	60.81
sl	***68.90***	62.04	61.47	61.73	61.59	62.28
sr	***60.58***	53.51	52.70	52.35	52.60	52.53
uk	***60.51***	52.59	52.95	53.06	52.80	53.03
ca	**63.15**	56.28	62.34	62.77	62.03	62.86
es	63.28	56.41	63.00	63.57	62.53	**63.61**
fr	64.18	57.93	63.37	64.61	63.02	**64.69**
fro	***62.82***	59.47	59.06	59.09	58.94	59.30
gl [600]	62.99	55.90	62.15	63.85	61.81	**63.89**
it	***68.33***	62.51	66.15	67.01	65.80	67.32
la	***45.21***	42.24	42.39	42.66	42.27	42.96
pt	64.73	58.22	64.18	64.55	63.75	**64.90**
ro	***57.40***	51.39	55.03	55.32	54.68	55.57
fa	***40.53***	32.21	25.33	23.20	26.69	25.90
hi	36.73	38.24	43.27	43.81	42.57	**44.16**
kmr [20]	***49.89***	46.35	41.30	40.41	41.55	41.11
mr	53.91	52.90	57.56	55.91	**58.35**	57.28
ur	35.33	35.32	36.11	36.21	36.38	***37.55***
el	***64.66***	59.11	59.47	59.03	59.13	59.28
grc	***50.21***	46.90	46.53	46.14	46.27	46.52
ga [566]	***53.09***	48.08	49.02	49.03	49.01	49.59
hy [50]	***48.20***	42.41	41.55	42.48	41.80	43.37
id	***55.22***	47.99	49.01	48.34	49.32	50.49
ja	28.13	29.14	28.02	28.98	28.06	**29.41**
ko	**31.74**	30.72	30.12	30.68	30.14	30.74
kk [31]	49.17	47.80	56.19	***57.95***	54.13	56.64
tr	33.77	31.94	***37.50***	36.47	34.34	35.64
ug	37.42	35.81	39.95	40.53	38.84	**40.68**
et	***54.20***	47.91	45.00	45.14	45.89	46.71
fi	***53.92***	50.56	49.69	50.09	49.49	50.37
hu	***59.59***	53.73	47.88	49.38	48.99	51.78
sme [2257]	**48.86**	47.63	47.43	48.36	47.24	48.40
ta	38.93	38.00	41.13	***42.69***	38.49	39.82
te	66.10	69.37	75.85	***77.61***	69.41	70.20
vi	***43.91***	37.85	39.47	38.17	39.86	39.78
zh	***33.03***	30.13	29.63	30.15	29.55	30.61
dev	55.19	50.29	52.19	52.43	51.88	52.62
no dev	52.20	52.10	48.98	49.65	48.71	49.77

Table 9.2: Comparison of zero-shot parsing results for several model propagation settings for languages that have a training set. The two first columns correspond to an complete unweighted graph. The two second columns correspond to the model of Zhu et al. without task confidence parameters. The two last correspond to the model of Bellet et al. with task confidence parameters. For each setting, we report the UAS results using either last model of averaged model as original model for each language.

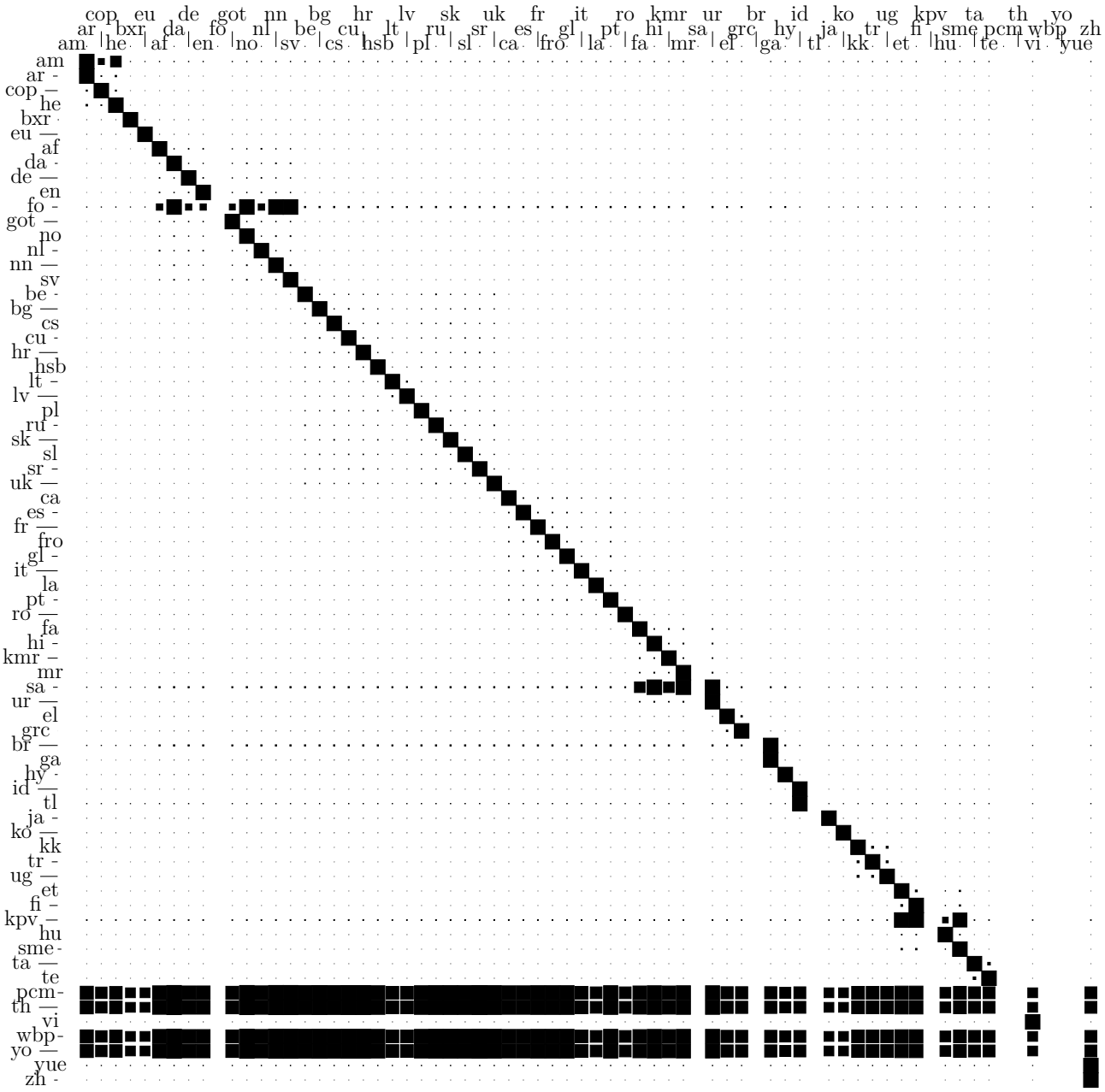


Figure 9.1: Propagation weights W used for model propagation derived from the phylogenetic tree. Language without training data have no impact on final models thus we set their weights to zero. Rows have been normalised so that full black cells correspond to the maximum weight.

	Unweighted		Weighted Without Confidence		Weighted With Confidence	
	Avg	Last	Avg	Last	Avg	Last
am	58.13	59.04	58.38	57.21	**59.06**	57.75
br	54.27	51.77	**54.95**	54.45	53.39	53.77
fo	***46.65***	43.54	44.65	45.63	44.39	45.33
sa	48.14	**48.67**	45.97	44.98	45.99	46.05
kpv	***60.49***	57.28	56.26	56.17	56.89	57.43
pcm	***60.15***	54.07	54.95	54.31	55.21	55.68
th	***41.99***	37.16	38.32	37.38	38.17	38.22
tl	89.56	90.67	88.78	85.67	**90.89**	89.22
wbp	***85.12***	75.07	75.67	75.17	76.45	79.51
yo	***51.34***	40.30	42.06	40.60	42.03	42.38
yue	***58.60***	54.74	55.91	54.61	56.41	55.50
zero shot	59.49	55.66	55.99	55.11	56.26	56.44

Table 9.3: Comparison of zero-shot parsing results for several model propagation settings. The two first columns correspond to an complete unweighted graph. The two second columns correspond to the model of Zhu et al. without task confidence parameters. The two last correspond to the model of Bellet et al. with task confidence parameters. For each setting, we report the UAS results using either last model of averaged model as original model for each language.

On the plot of W, we have normalised rows so that full black cells represent the models of maximum influence and white cells have less than a hundredth time the maximum influence.

We see faint blocks standing in the place of language sub/families. The two languages in the middle of the Slavic languages (be to uk) are Baltic languages (lt and lv). Then we see that for languages without training data, if it belongs to a family (am, fo, sa, br, tl, kpv, yue), then its family bares most of the influence, but if it is isolated (pcm, th, wbp, yo), then all models are used more or less equally.

		Lem	pLem	Form	pForm	Morph
da	one-hot	48.09	50.85	45.12	47.66	**69.19**
	embed	62.47	66.03	65.09	64.81	68.71
en	one-hot	57.09	59.23	54.97	56.96	**72.32**
	embed	70.95	69.16	71.20	71.19	72.22
et	one-hot	25.30	27.43	21.29	22.61	**64.06**
	embed	48.17	46.63	45.79	43.77	62.81
eu	one-hot	45.96	47.66	40.53	42.50	**68.19**
	embed	62.52	62.44	57.42	58.56	67.30
fi	one-hot	40.78	41.45	34.59	35.60	**71.00**
	embed	59.34	60.89	52.67	55.78	68.70
fr	one-hot	64.88	65.88	61.95	63.60	73.92
	embed	65.62	67.56	70.81	72.35	**73.96**
got	one-hot	46.85	47.71	45.19	46.53	**71.04**
	embed	61.37	62.01	59.35	59.34	70.41
he	one-hot	54.91	57.55	55.82	58.02	**72.66**
	embed	64.41	68.61	66.92	66.68	71.77
hu	one-hot	27.80	29.35	25.60	26.30	**64.31**
	embed	41.59	43.56	44.3	39.42	63.45
ro	one-hot	56.89	57.5	53.83	54.96	**68.94**
	embed	64.76	65.78	65.13	64.9	68.76
sv	one-hot	48.61	52.06	45.0	47.82	**69.97**
	embed	65.70	66.53	64.93	63.73	69.69

Table 9.4: LAS scores for parsers using lemmas (Lem, pLem), forms (Form, pForm) or morphosyntactic attributes (Morph) representations as features. For each language, the top line holds results using one-hot representation and the bottom line holds results using embeddings instead.

9.2 Measuring the Role of Morphology

Tables 9.4 and 9.5 report the labeled accuracy scores corresponding to the unlabeled accuracy scores reported in Tables 7.3 and 7.4 respectively.

The labeled scores are much lower than their unlabeled counterparts, but for such a simple model, this is completely expected. Otherwise, the analysis carries on to these tables as well. For sparse tokens (forms and lemmas), embedding clearly helps improving results. For morphological attributes, on the contrary, whilst one-hot is beats all other representations, embedding is slightly detrimental. Same goes for results of parsers using predicted attributes. The only noticeable point is that the Hebrew score using predicted morphology is lower than the one using form representation, which puts back Hebrew where it belongs amongst morpho-semantic languages.

		Form Emb	Morph One-hot	Morph Emb
da	argmax	**65.09**	58.33	57.72
	softmax		59.68	59.13
en	argmax	**71.20**	62.64	62.73
	softmax		65.59	64.97
et	argmax	45.79	43.80	42.22
	softmax		**47.05**	45.64
eu	argmax	57.42	55.81	55.06
	softmax		**59.43**	58.25
fi	argmax	52.67	54.42	52.79
	softmax		**58.74**	56.51
fr	argmax	**70.81**	66.66	66.25
	softmax		67.39	67.0
got	argmax	59.35	59.73	59.14
	softmax		**62.36**	62.02
he	argmax	**66.92**	63.74	63.57
	softmax		66.25	65.33
hu	argmax	44.30	52.41	49.99
	softmax		**53.63**	52.61
ro	argmax	**64.13**	62.10	61.67
	softmax		63.26	62.65
sv	argmax	**65.93**	60.29	60.03
	softmax		62.44	61.47

Table 9.5: LAS scores for parsers using predicted morphosyntactic attributes. For each language, the top line holds results corresponding to argmax prediction of attributes, while the bottom line holds results using probability distributions. The second column shows results using one-hot representation (or probability distributions), and the third shows results when the embeddings are used. The scores of parsers using embedded forms is given for comparison.

Acknowledgement

It is now the place to thank people that have helped me doing this Ph.D. in a way or another, some who were supportive, some who made life easier or even enjoyable at time and some without whom I would not have done what I have done. I hope it will not get too autobiographical, but those people deserve that I take time to write those words for them. The reader who is not interested in these words or think my writing style has eventually become to informal can always ignore this chapter.

I shall first and foremost thank Dr. Pascal Denis and Pr. Marc Tommasi for having supervised my work (one officially, the other almost daily) for the last four years (it all started with an internship).

Since our second encounter during the my second year of Master, Pascal Denis has proven an important part of my life. After a Master project and an internship, he has provided me with scientific guidance, support, feedback and interesting discussions during those four years, at the occasion our quasi weekly meetings. Surely enough, some deep personal research work could be acknowledged by the obtention of the title of Doctor, but I believe that a Ph.D. is more than just some research work and it is not a lonely task either. A Ph.D. is an occasion to learn to be a researcher, to learn rigour and at time to challenge its lack, to learn scientific communication, to learn to organise one's thoughts and ideas. And whilst this could be learned alone, supervision is an important part of the Ph.D. and the supervisor has a great role in the becoming of its Ph.D. student. I therefore want to thank Pascal Denis for the amount of time and energy he invested in me. I hope it is/will be paid back tenfold in pride, joy or any other kind of fulfilment.

I also want to thank Marc Tommasi for having been a more distantiated supervisor of mine, both in terms of scientific area of expertise and of number of offices between ours. He has been of great help all along these years, always caring for the people around him, organising just the right amount of socialisation events and distilling kindness around. Also, he was the first to prepare gluten free pastries for me (others quickly followed). Just as sentences are more than the mere sum of their words, Magnet is much more than the mere sum of its members and Marc Tommasi does a lot for it.

Speaking of Magnet, I want to thank all of its past and present members (officially or at heart). Rémi Gilleron is a great person, a smiling light in the mist. He was a great teaching colleague at Lille University and is a great research colleague at Inria. His advice are invaluable and I am honoured to have had the opportunity to work next to him.

Mikaela Keller is also an amazing, kind and interesting person. She always have interesting ideas and discussions. She is a really entertaining person, in the noblest sense of the word.

I want to thank Nathalie Vauquier. I am not quite sure of where is the border

of colleague-hood, but I can clearly say she is far beyond. During those years of shared work and car pooling, she has been a great teammate and more. She had allowed me not to pollute alone in my car. And while other Magnet members listen to me more or less politely at lunch, she also listened to me in the car and in the jams. Thank you very much.

I want to thank my office-mates that have changed through time : David, Pauline, Carlos, Mahsa, Thanh, William, Igor (if you don't know why, mango does), Juhi, Pierre and Quentin. I also want to thank all my other teammates : Fabio, Geraud, François, Arijus, Thibault, Onkhar, Brij, Marianna, Cesar, Bo and Julie. Then, again, I am not quite sure of the definition of friendship and I am not sure to which extent keeping in touch weights in and if it matters at all, but those people are more than just colleagues to me.

Inria is a very modern institute in terms of communication. Project teams are tagged with keywords and clustered into interest groups. Magnet is often associated with machine learning in graphs, privacy preserving machine learning, distributed learning and natural language processing. Maybe it is not professional enough or it is out of scope or maybe poetry and humanistic values got lost along the way. But I would add one keyword to that list and that is Kindness.

Some might have noticed that someone is missing and for sure he is, but with this thesis I learned to write better transitions and to structure my ideas, and he is a perfect transition. I want to thank Aurelien Bellet, if not for his central accent, his legendary positivity and his knowledge about decentralised learning, for he was the bridge between Magnet and Fei Sha's team at Los Angeles and without him I would never have had the chance to go there.

I want to thank Pr. Fei Sha from the University of Southern California in Los Angeles who have hosted me in his team twice. I learned a lot from him and his team about research in other countries and about different approaches to team work. I also want to thank members of his team that are great (future) researchers and were genuinely kind to me : Weilun, Ke, Zhiyun, Chao-Kai, Yury, Melissa, Jeremy, Jiyun and the others I sadly can not remember the actual name.

I want to deeply and warmly thank Dr. Soravit Beer Changpinyo, one of Pr. Sha's former Ph.D. student and the one with whom I worked the most. He is an incredible person, really kind and super hard working (so much so that I have a complex of inferiority). He took great care of me and helped me many times while I was in the United States. He had me discovering Thai cuisine, Californian fast-food, Ethiopian cuisine, Chipotle, cold brew, Thai tea and so much more. I was and still is very critical toward the United States of America. Without him, I would not have enjoyed Los Angeles as much as I did. Thank you.

I shall also thank my teaching colleagues Rémi Gilleron and Thibault Lietard (already mentioned) and Louis Bigo. I really enjoyed teaching those two years by their side. And, if my students have not learned much with me, I did learned a lot from teaching and from them.

I also want to thank linguists from the STL with whom we shared a reading group on distributed semantics, with whom I worked and who invited me to speak at their seminar : Bert, Ilse, Giuditta, Christopher, Rafael and Fayssal.

I also want to thank Liva Ravailona for his ideas and his enthusiasm. We had a few but very interesting working sessions with him and Pascal.

To close the academic section of these acknowledgement, I shall thank a teacher of mine. Surely, several of them if not all have shape me to some extent and have brought me where I am, but one amongst them has done a lot for me. Celine

Kuttler was my first true encounter with natural language processing and computational linguistics. At the end of the Bachelor, and at a time I was wondering about my future, she was the one who saw potential in me. She was the one who told me about summer school like ESSLLI to which I went twice and where I met a lot of great people. She was also the one that sent me to Århus for one year Erasmus and that has been one of the best experience in my short life so far. Thank you.

Speaking of Denmark, I will thank my friends from Århus in Danish. If you want, give it a try, it is not hard nor long.

Jeg skal give mange tak til mine danske vener. Mine kære Anita (fra Tjekkiet), Flavio (fra Italien), Luise og Til (fra Tyskland) og Thomas (fra Danmark). Jeg har elsket at lave mad og spise og snake med jer. Det var rigtlig hyggelig tide. Jeg skal også giv tak til min kære Egå folket : Charlotte, Nathan, Corentin, Emeline and Adrien (fra Frankrig), Rui (fra Kina), Junna (fra Japan) og Dima (fra Russland). Uden jer, Danmark har ikke været Danmark.

Amongst the many great people I met at conferences and summer schools, I shall thank Laura who I first met in Düsseldorf at ESSLLI and then at Tübingen at ESSLLI again, and then again at EMNLP in Brussel. She understand me and the world and that is very precious. I also want to thank Anne-Lau, Hector, Valera and Emanuela who are great people and researchers. May we work together one day.

Then, I want to thank Maxime and Erwan, my only two true friends from university. I do not exactly know why but those two guys are important to me.

Amongst the people have met in preparatory school and with whom I spent some nice time, I shall thank Eve, Florian and Thibault (the same as above). Life goes on, we change, but they are still there. Likewise, I shall thank Florence and Romain, the only people that I still see regularly from middle and high school. They are great friends.

Then, amongst the people I met along the way, here and there, I want to thank Wassila, Kaoutar and Monica. For the music, for the conversations, for the tea, for being part of this thing people call life, thank you.

Eventually, we reach family time. I obviously could not do without thanking my parents Bénédicte and Fréderic. They have invested much more than the expected blood, sweat and tears in me. Without their education, dedication and constant care I would not be where I am. I also want to thank my sister Margot and my brother Victor. We are very different and at times I wish we shared more but I nonetheless really love them and would not trade them for something else. As Margot is already doctor, even though we mock her (she is dentist), it now on me to pass the baton to Victor to hopefully give a full doctor batch to our parents.

I shall also thank my many aunts and uncles who are supportive at diverse degrees and who understand what I do at diverse degrees but are nonetheless great people and make for an amazing family. I want to thanks especially Bénédicte and Dominique who have been chosen by my parents one day of 1992 for some reasons to be special for me and it has worked well so far. I also want to thank my grandparents Marguerite and Jacques. Finally, I want to thanks my cousins. They also make up for great family meetings. Amongst them, I want to give a shout out to la Cousinade : Romain, Margot (my sister), Thibault, Victor (my brother) and Nathan. They are of great company when it comes to go to concert, festivals and Flemish restaurant.

I should also thank a few youtubers, a few dozens of bands and musicians, a

GP, a neurologist and a psycologist. Clearly, I would not have done it through this thesis without their support. But it was getting too autobiographical as I feared and too far from the scope of this thesis so I will wrap up.

If you are one of the persons I mentioned above, Thank you very much. If you are a person I forgot, but know me enough to forgive me, Sorry and Thank you very much. If you are a person I did not forget but know that you have counted/count and that I could not be exhaustive, Thank you very much. If you are nothing of the above but have read those words, from the bottom of my heart, Thank you very much.

I will finish with a pop culture quote as this is a twenty first century computer science and language thesis after all.

> *"I don't know half of you half as well as I should like;*
> *and I like less than half of you half as well as you deserve."*
>
> Bilbo Baggins

Bibliography

[AG17] Oded Avraham and Yoav Goldberg. The interplay of semantics and morphology in word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 422–426. Association for Computational Linguistics, 2017.

[ALM17] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *International Conference on Learning Representations 2017*, April 2017.

[AMB+16] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444, 2016.

[App13] David Appleyard. *Colloquial Amharic, The Complete Course for Beginners*. Routledge, 2013.

[AR13] Netta Abugov and Dorit Ravid. *Assessing yiddish plurals in acquisition: Impacts of bilingualism*, pages 90–110. 12 2013.

[AWY16] Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge. In *COLING 2016, the 26th International Conference on Computational Linguistics*, pages 119–130, Osaka, Japan, 2016. The COLING 2016 Organizing Committee.

[Ban15] Mohit Bansal. Proceedings of the 1st workshop on vector space modeling for natural language processing. pages 102–108. Association for Computational Linguistics, 2015.

[BB14] Jan A. Botha and Phil Blunsom. Compositional morphology for word representations and language modelling. *CoRR*, abs/1405.4273, 2014.

[BGJM16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[BGL14] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics, 2014.

[BK10] Taylor Berg-Kirkpatrick and Dan Klein. Phylogenetic grammar induction. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 1288–1297, 2010.

[Bla01] Barry J. Blake. *Case*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2 edition, 2001.

[BLVL00] Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. Building a treebank for italian: a data-driven annotation schema. In *In Proceedings of the Second International Conference on Language Resources and Evaluation LREC-2000 (pp. 99*, pages 99–105, 2000.

[BM06] Sabine Buchholz and Erwin Marsi. Proceedings of the tenth conference on computational natural language learning (conll-x). pages 149–164. Association for Computational Linguistics, 2006.

[BMM+10] Cristina Bosco, Simonetta Montemagni, Alessandro Mazzei, Vincenzo Lombardo, Felice Dell'Orletta, Alessandro Lenci, Leonardo Lesmo, Giuseppe Attardi, Maria Simi, Alberto Lavelli, Johan Hall, Jens Nilsson, and Joakim Nivre. Comparing the influence of different treebank annotations on dependency parsing. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*, 2010.

[Bre98] Joan Bresnan. Morphology competes with syntax: Explaining typological variation in weak crossover effects. *Is the best good enough*, pages 59–92, 1998.

[BRKS16] Christian Bentz, Tatyana Ruzsics, Alexander Koplenig, and Tanja Samardzic. A comparison between morphological complexity measures: typological data vs. language corpora. In *Proceedings of the workshop on computational linguistics for linguistic complexity (cl4lc)*, pages 142–153, 2016.

[Cam13] Lyle Campbell. *Historical linguistics: An introduction: Third edition.* 01 2013.

[Car97] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[CCBG10] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *J. Mach. Learn. Res.*, 11:2901–2934, December 2010.

[CDK+06] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December 2006.

[CG17] Xinying Chen and Kim Gerdes. Classifying languages by dependency structure. typologies of delexicalized universal dependency treebanks. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017), September 18-20, 2017,*

Università di Pisa, Italy, number 139, pages 54–63. Linköping University Electronic Press, Linköpings universitet, 2017.

[CGS10] Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[CJT17] Jiong Cai, Yong Jiang, and Kewei Tu. Crf autoencoder for unsupervised dependency parsing. *arXiv preprint arXiv:1708.01018*, 2017.

[CL65] Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14, 1965.

[Col97] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 16–23, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[Col02] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[CZZ14] Wenliang Chen, Yue Zhang, and Min Zhang. Feature embedding for dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 816–826. Dublin City University and Association for Computational Linguistics, 2014.

[DD17] Mathieu Dehouck and Pascal Denis. Delexicalized Word Embeddings for Cross-lingual Dependency Parsing. In *EACL*, volume 1 of *EACL 2017*, pages 241 – 250, Valencia, Spain, April 2017.

[DD18] Mathieu Dehouck and Pascal Denis. A Framework for Understanding the Role of Morphology in Universal Dependency Parsing. In *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing*, Proceedings of EMNLP 2018, Brussels, Belgium, October 2018.

[DD19] Mathieu Dehouck and Pascal Denis. Phylogenetic Multi-Lingual Dependency Parsing. In *NAACL 2019 - North American Chapter of the Association for Computational Linguistics*, Proceedings of NAACL 2019, Minneapolis, USA, 2019.

[DH13] Matthew S. Dryer and Martin Haspelmath, editors. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[DQM17] Timothy Dozat, 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 2017. Association for Computational Linguistics.

[DTvG11] Jon Dehdari, Lamia Tounsi, and Josef van Genabith. Morphological features for parsing morphologically-rich languages: A case of arabic. In *SPMRL@IWPT*, 2011.

[EB07] Jason Eisner and John Blatz. Program transformations for optimization of parsing algorithms and other weighted logic programs. In Shuly Wintner, editor, *Proceedings of FG 2006: The 11th Conference on Formal Grammar*, pages 45–85. CSLI Publications, 2007.

[Edm67] Jack Edmonds. Optimum branchings. *Journal Research of the National Bureau of Standards*, 1967.

[Eis96] Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[ETF14] Joseph Emonds and Jan Terje Faarlund. *English: The Language of the Vikings*. 12 2014.

[FBVdLM+92] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, T. J. Watson, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics, Volume 18, Number 4, December 1992*, 1992.

[Fot06] Kilian Foth. *Hybrid methods of natural language analysis.* PhD thesis, 01 2006.

[FS99] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, December 1999.

[FTY+15] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and A. Noah Smith. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500. Association for Computational Linguistics, 2015.

[FW73] T. Finkenstaedt and D. Wolff. *Ordered profusion; studies in dictionaries and the English lexicon.* Number vol. 13 à 15 in Annales Universitatis Saraviensis: Reihe Philosophische Fakultät. C. Winter, 1973.

[GCY+15] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244. Association for Computational Linguistics, 2015.

[GGRAP17] Marcos Garcia, Carlos Gómez-Rodríguez, and Miguel Alonso Pardo. New treebank or repurposed? on the feasibility of cross-lingual parsing of romance languages with universal dependencies. *Natural Language Engineering*, 24:1–32, 10 2017.

[GGT09] Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377. Association for Computational Linguistics, 2009.

[GK16] Kim Gerdes and Sylvain Kahane. Dependency annotation choices: Assessing theoretical and practical issues of universal dependencies. In *LAW@ACL*. The Association for Computer Linguistics, 2016.

[GMJB17] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431, 2017.

[GP09] Sylviane Granger and Magali Paquot. *Lexical verbs in academic discourse: A corpus-driven study of learner use*, pages 193–214. 01 2009.

[Hal07] Keith Hall. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 392–399, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[HRWK01] Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. Evaluating translational correspondence using annotation projection. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 392–399, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[HVH98] Jan Hajič and Barbora Vidová-Hladká. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of the COLING - ACL Conference*, pages 483–490, Montreal, Canada, 1998.

[Jäg15] Gerhard Jäger. Support for linguistic macrofamilies from weighted sequence alignment. *Proceedings of the National Academy of Sciences*, 112(41):12752–12757, 2015.

[JN07] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for english. In *NODALIDA*, 2007.

[KCC08] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603. Association for Computational Linguistics, 2008.

[KDI12] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

[KG15] Eliyahu Kiperwasser and Yoav Goldberg. Semi-supervised dependency parsing using bilexical contextual features from auto-parsed data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1348–1353. Association for Computational Linguistics, 2015.

[KGS11] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 521–528, 2011.

[Kib10] Anna Kibort. A typology of grammatical features., 2010.

[KLG17] Jenna Kanerva, Juhani Luotolahti, and Filip Ginter. Turkunlp: Delexicalized pre-training of word embeddings for dependency parsing, 01 2017.

[KMN09] Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009.

[LAWY16] Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *HLT-NAACL*, 2016.

[LCHZ18] Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214. Association for Computational Linguistics, 2018.

[LFDT14] Teresa Lynn, Jennifer Foster, Mark Dras, and Lamia Tounsi. Cross-lingual transfer parsing for lowresourced languages: An irish case study. In *In Proceedings of Celtic Language Technology Workshop 2014*, 2014.

[LG14] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[LGD15] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225, 2015.

[LW41] Benjamin Lee Whorf. *The Relation of Habitual Thought and Behavior to Language*, volume 1, pages 75–93. 01 1941.

[LZ15] Phong Le and Willem H. Zuidema. Unsupervised dependency parsing: Let's use supervised parsers. *CoRR*, abs/1504.04666, 2015.

[Mal83] Joan Maling. *Transitive Adjectives: A Case of Categorial Reanalysis*, pages 253–289. Springer Netherlands, Dordrecht, 1983.

[Man11] Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer, 2011.

[Mar16] David Marecek. Twelve years of unsupervised dependency parsing. In *ITAT*, pages 56–62, 2016.

[MBXS17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *CoRR*, abs/1708.00107, 2017.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[MCP05a] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98. Association for Computational Linguistics, 2005.

[MCP05b] Ryan McDonald, Koby Crammer, and Fernando Pereira. Spanning Tree Methods for Discriminative Training of Dependency Parsers, 2005.

[Mil95] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[MMS93] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[MPH11] Ryan McDonald, 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. Association for Computational Linguistics, 2011.

[MR13] Mehryar Mohri and Afshin Rostamizadeh. Perceptron mistake bounds. *CoRR*, abs/1305.0208, 2013.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[NAA+18a] Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni,

Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. Universal dependencies 2.3, 2018. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[NAA+18b] Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam

Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos,

Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. Universal dependencies 2.2, 2018. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[NBG12] Tahira Naseem, Regina Barzilay, and Amir Globerson. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 629–637, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[NDG+17] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.

[NdMG+16] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[NH05] Joakim Nivre and Johan Hall. Maltparser: A language-independent system for data-driven dependency parsing. In *In Proc. of the Fourth Workshop on Treebanks and Linguistic Theories*, pages 13–95, 2005.

[NHK+07] Joakim Nivre, 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)*, 2007.

[Pat04] Peter L Patrick. Jamaican creole: morphology and syntax. 2004.

[PGC15] Wenzhe Pei, Tao Ge, and Baobao Chang. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 313–322. Association for Computational Linguistics, 2015.

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018. cite arxiv:1802.05365Comment: NAACL 2018. Originally posted to openreview 27 Oct 2017. v2 updated for NAACL camera ready.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RBAS17] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard. Sluice networks: Learning what to share between loosely related tasks. *ArXiv e-prints*, May 2017.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

[RMv14] Loganathan Ramasamy, David Mareček, and Zdeněk Žabokrtský. Multilingual dependency parsing: Using machine translated texts instead of parallel corpora. *The Prague Bulletin of Mathematical Linguistics*, 102:93–104, 2014.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[SA17] Natalie Schluter and Željko Agić. Empirically sampling universal dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 117–122, 2017.

[SACJ11] Valentin I Spitkovsky, Hiyan Alshawi, Angel X Chang, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1281–1290. Association for Computational Linguistics, 2011.

[SB17] Anders Søgaard and Joachim Bingel. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 164–169, 2017.

[SF18] Gary F. Simons and Charles D. Fennig, editors. *Ethnologue: Languages of the World, Twenty-first edition.* SIL International, Dallas, TX, USA, 2018.

[SG16] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*, 2016.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[Smi11] Noah A. Smith. *Linguistic Structure Prediction*. Morgan & Claypool Publishers, 1st edition, 2011.

[SRIV11] Avishek Saha, Piyush Rai, Hal DaumÃ© III, and Suresh Venkatasubramanian. Online learning of multiple tasks and their relationships. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 643–651, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[SS10] Drahomíra Spoustová and Miroslav Spousta. Dependency parsing as a sequence labeling task. *The Prague Bulletin of Mathematical Linguistics*, 94(1):7 – 14, 2010.

[SSO+14] Per Erik Solberg, Arne Skjærholt, Lilja Ovrelid, Kristin Hagen, and Janne Bondi Johannessen. The norwegian dependency treebank. 05 2014.

[SUW13] S. Sharoff, E. Umanskaya, and J. Wilson. *A Frequency Dictionary of Russian*. Routledge, 2013.

[TJHA05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, December 2005.

[TM15] Takaaki Tanaka and N Masaaki. Word-based japanese typed dependency parsing with grammatical function analysis. 2:237–242, 01 2015.

[TMU12] Oscar Täckström, Ryan T. McDonald, and Jakob Uszkoreit. Cross-lingual word clusters for direct transfer of linguistic structure. In *HLT-NAACL*, 2012.

[TSG+10] Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kubler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. Statistical Parsing of Morphologically Rich Languages (SPMRL) What, How and Whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, United States, 2010. Association for Computational Linguistics.

[VBT17] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 509–517. PMLR, 2017.

[Vic03] A Vicentini. The economy principle in language. *Mots, Palabras, Words*, 3:37–57, 01 2003.

[WP12] Alina Wróblewska and Adam Przepiórkowski. Induction of dependency structures based on weighted projection. In Ngoc-Thanh Nguyen, Kiem Hoang, and Piotr Jdrzejowicz, editors, *Computational Collective Intelligence. Technologies and Applications*, pages 364–374, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[XG14] Min Xiao and Yuhong Guo. Proceedings of the eighteenth conference on computational natural language learning. pages 119–129. Association for Computational Linguistics, 2014.

[YM03] H. Yamada and Y. Matsumoto. Statistical Dependency Analysis with Support Vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, 2003.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, 2002.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, pages 912–919, 2003.

[ZHP+18] Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[Zip35] George Kingsley Zipf. *The Psychobiology of Language*. Houghton-Mifflin, New York, NY, USA, 1935.

[ZPS+17] Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav

Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada, August 2017. Association for Computational Linguistics.