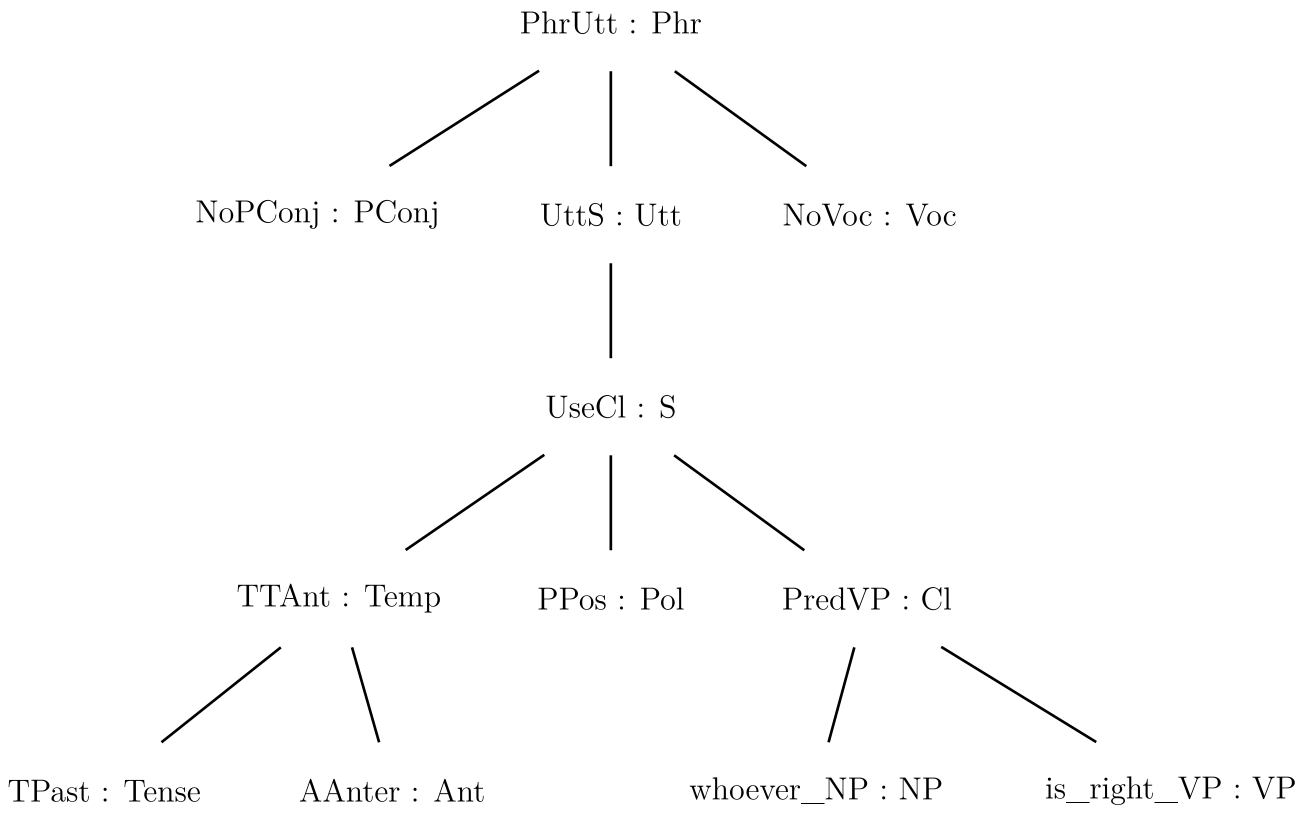
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



Unsupervised Disambiguation of Abstract Syntax

A Language Independent Unsupervised Model for Abstract Syntax Disambiguation

Master's Thesis in Engineering Mathematics and Computational Sciences

OSCAR KALLDAL, MAXIMILIAN LUDVIGSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

Unsupervised Disambiguation of Abstract Syntax

A Language Independent Unsupervised model for Abstract Syntax Disambiguation

Master's Thesis in Engineering Mathematics and Computational Sciences

OSCAR KALLDAL, MAXIMILIAN LUDVIGSSON

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Unsupervised Disambiguation of Abstract Syntax
A language independent unsupervised model for abstract syntax disambiguation
Oscar Kalldal and Maximilian Ludvigsson

Supervisor: Prasanth Kolachina, Department of Computer Science
Examiner: K.V.S. Prasad, Department of Computer Science

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A disambiguated abstract syntax tree for the sentence "Whoever was right".

Typeset in LaTeX
Gothenburg, Sweden 2018

Unsupervised Disambiguation of Abstract Syntax
A language independent unsupervised model for abstract syntax disambiguation
Oscar Kalldal and Maximilian Ludvigsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Disambiguating natural text is the task of choosing the correct meaning among several possible interpretations. This thesis focus on disambiguating parse trees created by Grammatical Framework — a formal language that represent meaning of natural language sentences with abstract syntax trees in order to do machine translation. Since one tree represents a meaning, for every sentence there exists several interpretations for which the most probable one should be chosen.

In order to achieve this, a language model on trees is defined. This is then used to compare possible trees and choose the one with the highest probability. In order to estimate the parameters of the model, the probability of the different meanings behind a word needs to be estimated. This is done using the Expectation Maximization algorithm.

Experiments are done on seven different languages to show that the method is generalizable. Different smoothing techniques as well as different dictionaries are evaluated. A novel *merged Wordnet* is constructed in order to avoid sparseness.

The method is evaluated by doing word sense disambiguation (a subtask of tree disambiguation) on standard data sets. The model is shown to be comparable to other unsupervised methods in the SemEval 2015.

Keywords: natural language processing, grammatical framework, language models, expectation maximization

Acknowledgements

Thanks to our supervisor Prasanth Kolachina who have guided us through this project, Krasimir Angelov who had the original idea to apply the expectation maximisation algorithm and Aarne Ranta, who introduced us to GF.

Oscar Kalldal and Maximilian Ludvigsson, Gothenburg, January 2018

Contents

1
Introduction

Ambiguity in language has been studied extensively in the Natural Language Processing community. It is one of the defining traits of human language often used as a vehicle of poetry and humor. To illustrate, this is a quote from the American comedian Groucho Marx:

> "One morning I shot an elephant in my pajamas. How he got in my pajamas, I don't know."

The quote highlights the ambiguity present in the first sentence. Who is wearing the pajamas, I or the elephant? For a human, the absurdness of the interpretation implied by the second sentence is a source of humor, but for a computer that doesn't know what an elephant is, the elephant wearing the pajamas is a completely valid interpretation. Ambiguity in language is one of the biggest obstacles when it comes to computers understanding language and it naturally influences many downstream tasks within NLP, such as sentiment analysis, part of speech tagging, and translation.

Disambiguating language is mostly done using context. For a human reading the first sentence in Marx' quote the context provided by the reader's prior knowledge seems to make the meaning clear: elephants don't wear pajamas. Continuing reading provides a new broader context that implies another interpretation of the sentence. For a computer, drawing these conclusions is not trivial and in methods used by computers, context usually enters as a statistical model.

1.1 Background

In this thesis, we will consider Grammatical Framework (Ranta, 2004), a language formalism that aims to connect the world of formal language with the one of natural language. GF was originally designed for multilingual text generation in controlled natural language and recently there has been successful research aimed at extending the scope to wide-coverage parsing (Angelov and Ljunglöf, 2014).

Adapting current statistical parsing methods such as those described by Jurafsky and Martin (2009, Chapter 14) to the GF parser is a problem due to the unique internal representation of language in GF: abstract syntax trees. Inspired by the representation used in compilers for programming languages, abstract syntax trees are defined so that the same type of trees is shared across languages in a way such that translations of the same sentence correspond to the same abstract syntax tree (Kolachina and Ranta, 2016). Language dependent linearization rules merely define how an abstract syntax is converted to a string in a language, while syntactic information is encoded in a language independent way in the tree. The linearization rules are designed to be reversible for use in parsers in order to convert strings into abstract syntax trees. The difficulty in using statistics to enhance parsing is to do so in a way that preserves the language-independent approach unique to GF — most state-of-the-art parsers need heavy customization when dealing with a new language.

In order to be truly language independent, abstract syntax trees must do away with much of the ambiguity in language as different languages are often ambiguous in different ways. In turn, this means that disambiguation in parsing is of utmost importance. Today disambiguation is done by considering every ambiguous word and every ambiguous grammatical rule in a tree separately and independently. This context-free model, although powerful as is, can be improved by being expanded to use contextual information, information that has been shown to be very important for the task.

1.2 Problem

This thesis aims to develop a language model for GF's abstract syntax trees to be used for disambiguating possible trees, especially the ones generated by the current parser of GF. This means that we have to develop a language model capable of assigning probabilities to abstract syntax trees. The reason this is a unique challenge is twofold:

One, we have to define a probabilistic model over a tree structure as opposed to linear text, as well as find data to estimate parameters for such a model. By utilizing the correspondence between GF and dependency trees tagged in the Universal Dependencies (UD) scheme (Kolachina and Ranta, 2016) and limiting us to modeling probabilities of syntactic head-child dependency relationships between leaf nodes in the GF abstract syntax trees we are able to define a probabilistic model for abstract syntax trees that is not dependent or large amounts of data in the form of GF treebanks. Instead, we can use UD gold standard treebanks and data from automatically UD-parsed corpora. Unlike GF, there is plenty of data available with high-quality UD trees (Ginter et al., 2017).

Two, because of the language independence inherent in GF abstract syntax trees, the leaf nodes in the abstract syntax trees does not consist of words in one language, but rather of language independent word senses. As the model is defined over the leaf nodes in the tree and since there is very little sense annotated training data available, unsupervised methods must be used so when estimating parameters for the model. This is done by using non-annotated text in several languages together with a knowledge source of possible sense interpretations for each word in each language. Estimation is then done using *Expectation Maximization*, a method for estimating unobservable variables, further explained in section 2.7.

Although we are training our model with data from a handful of languages, the goal is to have a model that is also useful for languages where data doesn't exist. The only assumption we make is that we have a linguistic knowledge base, i.e. a dictionary which maps the senses of a word in the target language to the languages used for training.

1.3 Contributions

In this thesis, we present a probability model for ranking of parse trees of natural language. Although based on the trees generated using Grammatical Framework, it supports any tree structure with Universal Dependency-like parent and child relationships. We have implemented Expectation Maximization and shown that it can be scaled up to a model with over 10^8 parameters.

Other contributions is within smoothing. We created a separate Wordnet dictionary with similar synsets merged together, in this way both reducing the size of the model as well as making the signal stronger.

1.4 Outline

In chapter 2, the theory behind our methods are further described. First, a general overview of ambiguities in languages, grammars, and parsing is given. Then, we further describe the resources used: Grammatical Framework, Universal Dependencies, and Wordnet. Last, n-gram language models and the Expectation-Maximization algorithm is explained.

In chapter 3, we describe our original work and how it was carried out. It contains a description of how we define our language model, and how we estimate the parameters of the model. Our work with the Wordnet clustering is also described.

Chapter 4 presents the results of the evaluation of our models, which is further discussed in chapter 5 together with pointers to areas of further research.

2
Theory

In this chapter, the theory behind our methods is described. In order to give an overview of the context of the research, a general description of ambiguities in languages, grammars, and probabilistic language models is given. Grammatical Framework, Universal Dependencies, and Wordnet are three important resources for the methods in this thesis and are also described. Last, we give an account of the theoretical foundations of the Expectation-Maximization algorithm.

2.1 Ambiguities in Language

When interpreting a sentence, ambiguities in natural language will inevitably be a problem. We will here introduce two kinds of natural language ambiguities: lexical ambiguity and syntactic ambiguity.

A simple example of a lexical ambiguity in natural language is the word *bass*. It might refer to a type of fish in one sentence (*I am fishing bass*) and to a type of instrument in another (*I play the bass*). These ambiguities are highly language dependent and words are often ambiguous in different ways in different languages. As an example, the Swedish words for bass the fish and bass the instrument are different (*aborre* vs. *bas*).

To understand syntactic ambiguity, we look at the sentence *I eat the food in the kitchen*, where the clause *in the kitchen* could potentially refer either to the location where the eating is done or as an attribute to the food that was eaten (*I am in the kitchen and I eat the food* or *I eat the food that is in the kitchen*). This kind of ambiguity does not depend on the vocabulary of the language but on the grammar. Depending on the intended meaning, this sentence would be translated differently into Chinese as we can see in the following example:

(1)

我	在	厨房	吃	饭
wo	zai	chufang	chi	fan
I	in	kitchen	eat	food.

‘I eat the food in the kitchen’

(2)

我	吃	在	厨房	的	饭
wo	chi	zai	chufang	de	fan
I	eat	in	kitchen	[attributive]	food

‘I eat the food in the kitchen’

These two examples illustrate that there are sentences that are syntactically ambiguous in one language but not in others. This has great implications for applica-

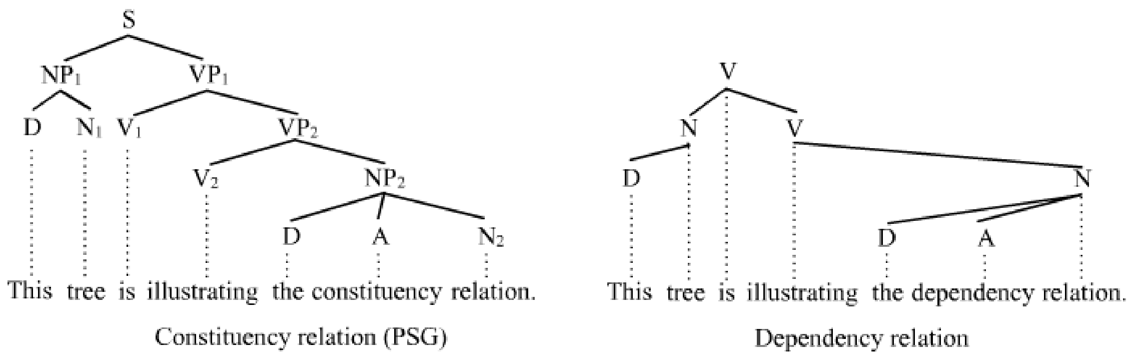

S
NP1
VP1
D
N1
V1
VP2
V2
NP2
D
A
N2
This tree is illustrating the constituency relation.
Constituency relation (PSG)
V
N
V
D
N
D
A
This tree is illustrating the dependency relation.
Dependency relation

Figure 2.1: Trees illustrating the dependency and constituency relations, adapted from Wikimedia (2011).

tions such as machine translation, as there must be a model describing how to choose the right interpretation of a sentence in order to make the correct translation.

2.2 Grammars and Syntax

In linguistics, **syntax** is a set of formal rules that govern how sentences are constructed and structured in a given language, such as in which order the subject, the verb and the object should appear (Chomsky, 1957). Together with the **morphology**, how individual words change in a language depending on the sentence they are in, these rules constitute the **grammar** of the language.

One type of grammars is phrase structure grammars, which focus on describing a sentence by identifying sub-phrases in the sentence recursively. A simple example of this is how in the sentence *Bob kicks the green ball* one can identify *kicks the green ball* as functioning as a verbal phrase in which one, in turn, can identify *the green ball* to function as a noun phrase. A different type of grammar is called **dependency grammar**. The focus of this type of grammar is on the relationship between words (Nivre, 2005), for example, in the same sentence, one can identify the words *Bob* and *ball* as depending on *kicks*, and *green* as depending on *ball.* In both these cases, the rules of the grammar can be used to draw up a hierarchical tree, called a syntax tree, that describes which rules were used to build a certain sentence. Examples of phrase structure trees and dependency trees are displayed in figure 2.1.

The process of deciding the syntax tree of a given sentence is called **parsing**. This process is non-trivial because in many cases it is non-deterministic, for one given sentence there can be many different valid syntax trees. An example is the sentence *I eat the food in the kitchen* from section 2.1 would have different syntax trees depending on the interpretation of the sentence, one of which is illustrated in figure 3.2. The fact that syntax trees can serve to formalize which one of the many syntactic interpretations a sentence is correct makes it an excellent tool for use in computers understanding and processing language.

2.3 Probabilistic Language Models

Probabilistic models for language work by computing the probability of observing a certain type of linguistic item in text or speech. Examples of such models are language models that assign probabilities to sentences or fragments of sentences in a certain language, such as n-gram models. One can also define probabilistic models over possible interpretations of the same sentence, such as the different valid syntax trees of a sentence, which can then be used as a toll for disambiguation tasks.

2.3.1 N-Gram Models

N-gram models are generative models in the sense that they assume that the probability of a certain word appearing in a sentence only depends on the history of that word, that is the words preceding it, and that the probability of a word occurring is location invariant. More specifically, n-gram models assume that the probability of a certain word appearing only depends on the previous n words for some fixed n. These models can assign a certain probability to every possible sentence by using the chain rule, can be used to predict the next word in a sentence, and can easily be used generatively to sample new sentences according to the probability distribution defined by the model. N-gram models are well researched and are considered one of the most important tools in speech and language processing (Jurafsky and Martin, 2009).

For most disambiguation tasks, probabilistic or statistical models are used. These probabilistic models are different from models such as n-grams in that they assign a probability to each sentence, computing the probability that a certain sentence should be interpreted in a certain way in the face of ambiguity. A model for syntactic disambiguation could, for example, define a probabilistic model assigning probabilities for all possible syntax trees of a sentence. With such a model disambiguation can then be done by computing the probability for each possible interpretation and choosing the one with the highest probability.

2.3.2 Probabilistic Context-Free Grammars

Among probabilistic grammar models for phrase structure grammars the most common ones are **Probabilistic Context-Free Grammars** (PCFG). According to Manning et al. (1999), PCFG:s adhere to the three assumptions of place invariance, context freeness, and ancestor freeness. Place invariance means that the probability of a sub-tree of the syntax tree occurring does not depend on where in the sentence the words that subtree spans are. As such a certain noun phrase would have the same probability regardless of whether it is first or last in the sentence. Context freeness means that the probability of a given subtree does not depend on any words outside the span of that sub-tree. For example, the probability of a given noun phrase does not depend on the main verb in the sentence. Last, ancestor freeness means that the probability of a subtree does not depend on any nodes outside that subtree, for example, the probability of a given noun phrase would not depend on it being a part of a verb phrase or a prepositional phrase.

It is obvious that these assumptions do not hold for real text and Jurafsky and Martin (2009) indeed writes that the main drawback of the PCFG is that the sweeping independence assumptions automatically discards many structural dependencies in the syntax tree. In addition, it is also insensitive to lexical information. This can be somewhat remedied by various modifications to the PCFG and Jurafsky and Martin (2009) mentions techniques such as **lexicalization** and **splitting non-terminals**.

2.3.3 Syntactic N-Grams over Dependency Trees

In order to improve the n-gram model Sidorov et al. (2014) introduced **syntactic n-grams**. These differ from ordinary n-grams in what is considered the history in the model. Instead of defining the history as the previous words in the original sentence order, a syntactic n-gram instead considers the predecessors in a **dependency tree** of the sentence as the history of a word. This allows for some long-range dependencies to be accounted for easier than in a normal linear n-gram model. For example, in *Bob kicks the green ball* the word *kicks* is the immediate predecessor of *ball* in the dependency tree and the predictive information of *kicking ball* occurring often together would be captured using only a syntactic bigram model, in the linear n-gram case, this particular case would require a 5-gram model in order to be able to capture the same relation. Due to the fact that word order is a highly language dependent property, syntactic n-grams are more suited for cross-lingual comparison (Sidorov et al., 2014).

A more generalized generative language model based on the syntactic nature of the dependency tree structure is developed by Richardson et al. (2016). Richardson et al. defines the history of each node in the tree as all nodes preceding that node given a defined traversal order. By denoting the i:th node in the tree w_i and the history of the i:th node (that is all nodes preceding w_i) as H_i, Richardson et al. also defines the 't-treelets of size l for w_i' as all connected subtrees $S' \subset H_i$ where $w_i \in S'$ and $|S| = l$. t-treelets can be divided into several types depending on the topology of the trees and a generative probabilistic model can be defined by assuming that the probability of a certain node given its history is equal to the probability of a node given all t-treelets for the node that is of certain type and size. Examples of such models are all t-treelets of size $l \leq 3$ that only consists of nodes that are ancestors to w_i or all t-treelets of size $l = 3$ that only consists of the parent node and one sibling node to w_i. The fact that one needs a well-defined traversal order of the tree to define the history of each node is important in order to create a generative model, as there would otherwise appear circular dependencies in the model.

2.4 GF and Abstract Grammars

Grammatical Framework (Ranta, 2004) is a framework for writing and implementing formal grammars mainly for natural languages. The defining feature is that grammars are divided into an **abstract syntax** and a **concrete syntax**, a concept borrowed from programming language design. According to Ranta the abstract syntax defines "the hierarchical structure of the language" and the concrete syntax “what the language looks like as it is read and written”. In GF different languages can share the same abstract syntax describing the semantics, i.e. the meaning, of statements in *all* languages, while each language has its distinct concrete syntax. The concrete syntax can be seen as a description of how the abstract syntax should be rendered in the textual form of the corresponding language, which means that by describing a statement in abstract syntax we can **linearize** that statement to the textual versions of that statement into all supported languages. Angelov (2009) then describes a method to **parse** text to get all valid representations in abstract syntax for a statement given in linearized textual form in a language that have a defined concrete grammar over the abstract syntax.

Together, the ability to parse and linearize statements for all languages with a concrete syntax over the same abstract syntax means that meaning preserving translation is possible between those languages, in effect using the abstract syntax as a **pivot language** or an **interlingua**.

The main problem in doing translation by this approach is that the parsing step in many cases is not deterministic, that is, for a given statement in textual form, there might be several valid abstract representations of that statement. This is not surprising in and is, in fact, unavoidable if one wants the abstract syntax to accurately describe the semantic meaning of each statement, and for accurate translation using the interlingua approach to work. As an example, the ambiguous sentences given in section 2.1, *I like bass* and *I eat the food in the kitchen* would have different abstract representations depending on the interpretation of each sentence, and must be **disambiguated** in order to be translated accurately to Swedish and Chinese respectively. That means that one must choose which of the possible abstract representations should be used as the immediate representation.

The algorithm for fast statistical parsing presented by Angelov and Ljunglöf (2014) introduces a statistical element to parsing by giving each possible abstract parse a probability score that aims to reflect the likelihood of a given possible abstract representation being the correct one. Angelov and Ljunglöf’s parsing method has many similarities to the PCFG method and can indeed be seen as being a context-free probabilistic model in the abstract domain. Virk et al. (2014) also use statistical parsing of abstract syntax trees together with word sense disambiguation to disambiguate lexical ambiguities to show that it is possible to parse arbitrary English text, and show improvements in translation with the interlingua approach using GF.

2.5 Universal Dependencies

Universal dependencies is an annotation scheme with the mission to create dependency-style treebanks with a common tag set for a large number of languages. It was born from Stanford dependencies (De Marneffe et al., 2006) and Google universal part-of-speech tags (Petrov et al., 2011).

Just as De Marneffe et al. (2006) has mapped the Stanford parser’s constituency trees to dependency trees, there has been done work on mapping the abstract trees in GF to UD, and the opposite (Kolachina and Ranta, 2016; Ranta and Kolachina, 2017). The problem with transforming UD trees into GF trees is twofold: Firstly,

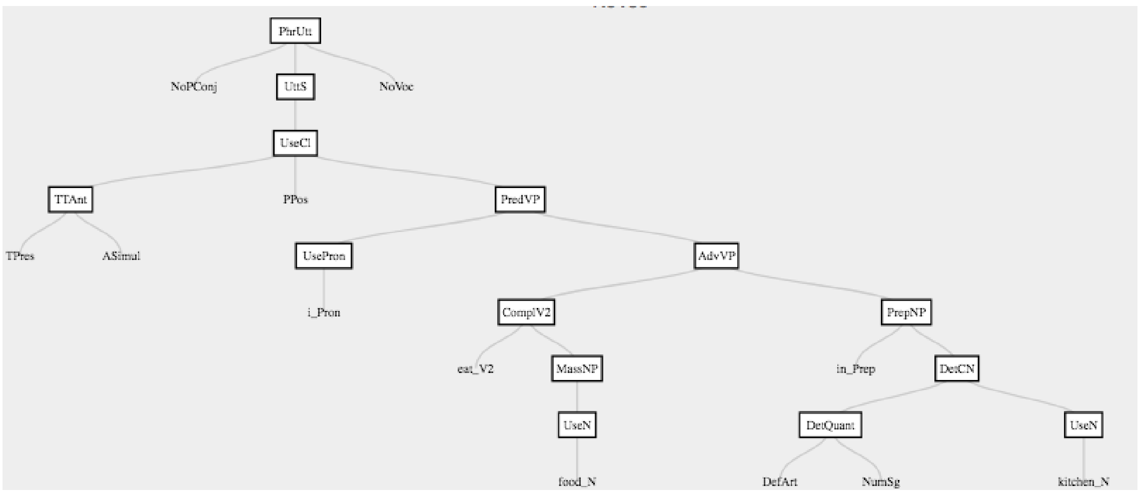


Figure 2.2: Example of a possible abstract syntax tree representing the sentence *I eat the food in the kitchen.*

the tree structure in GF is not completely determined by the UD trees; there can be several dependency trees mapping to the same GF tree. Secondly, in order to be completely language independent, GF need to know to which sense every word in the tree corresponds. This lead to the introduction of an **abstract dependency tree** by Kolachina and Ranta (2016), which is a UD tree with words annotated with their abstract meaning. The problem of annotating the word correctly, which this thesis deals with, is thus of big importance in the work of transforming UD trees into GF trees.

2.6 Wordnet

Wordnet (Miller, 1995) is a lexical and semantic resource developed at Princeton University aimed to provide traditional lexicographical information for use in computational applications. The database, originally developed for English, provides relational information over a wide array of words and lexical items such as synonymy, the notion that two words mean the same thing, and hyponymy, the notion that one thing is a sub-name for another thing (e.g. a *car* is a type of *vehicle*). The main relation in the database is synonymy and is described by assigning each word to one or several **synsets**. All words that are part of one particular synset can be said to mean the same thing in some sense. As an example, the words *pipe* and *tube* can be seen as synonyms in some contexts meaning a hollow cylindrical shape, and they are thus part of the same synset. However, because of the fact that many words are ambiguous, some words can belong to several synsets. As an example, the words *tube* and *subway* will have one synset in common, since they can both refer to a type of railway operating under ground. It is thus possible to say that each synset represents one particular semantic concept or notion, something that is utilized by (Virk et al., 2014) to use wordnet synsets as abstract representations of words in GF grammars.

The Open Multilingual Wordnet (Bond and Paik, 2012; Bond and Foster, 2013) is a project designed to standardize the format and usage of several wordnets for different languages, as well as provide links between synsets in those wordnets by using the synsets in the English wordnet as pivots. Since linking is done at the synset level lexical ambiguities otherwise encountered when linking multilingual dictionaries are avoided.

2.7 Expectation Maximization

The **EM algorithm** (Dempster et al., 1977) can be described as a method to calculate maximum likelihood estimates in the face of incomplete data. That the data is incomplete means that we have a model which includes certain variables that we can not readily observe, variables that can also be called latent variables. In our case this means assigning probabilities of occurence and co-occurence of the *meanings* of words, meanings which we can't observe. A good introduction to the algorithm can be found in Do and Batzoglou (2008) which describes EM as an iterative method to get increasingly accurate approximations of the maximum likelihood estimate for model parameters where each iteration of the algorithm consists of two steps: the expectation and the maximization step. First, in the expectation step, one calculates the expected value of the latent variables given the model parameters obtained in the previous iteration. Second, in the maximization step, these expected values are used in lieu of the actual values of the latent variables in order to compute the maximum likelihood estimate for the model parameters.

2.7.1 Problem Formulation

The example problem used by Do and Batzoglou (2008) to illustrate the algorithm is one where there are two weighted coins of different weights, and one wishes to estimate the weights of each of the coins. Five separate experiments are done where for each experiment one of the coins are chosen randomly with equal probability, and then that coin is flipped ten times. The amount of times the chosen coin showed head is recorded for each of the five experiments, but it is not known which of the two coins were chosen for any of the experiments. The problem is now to estimate the weights of the coins. this could be trivially done with maximum likelihood if we had complete information of both the counts and which coin was chosen for each experiment, but as the coin chosen for each of the experiments is unknown, we can not trivially do this.

More formally, the problem the EM algorithm sets out to solve is described by McLachlan and Krishnan (2007) as involving two data vectors x, y drawn from the two random vectors X and Y, where we know that X is completely determined by Y, that is $X = X(Y)$. The data vector X is called the incomplete information and the data vector Y is called the complete information. In the prior example of the coin flip experiment, the data vector X would correspond to the counts of the amount of heads and tails for each experiment but not what coin was used, while the data vector Y would correspond to both the head and tail counts as well as the coins used. By assuming that X and Y has the probability density functions $g(x, \theta)$ and $g_c(y, \theta)$ respectively, where θ is a set of parameters, we have defined a probability model for our data and we now want to calculate a maximum likelihood estimate of the parameter set θ, given the incomplete data vector X. The EM algorithm is often used in situations where the probability model for the complete data, g_c, takes a simpler form than the model for the incomplete data or in instances where one have a problem which can be formulated as an incomplete data problem where computation of maximum likelihood estimates is easier in the complete data model. To go back to the coin flip example, while it is theoretically possible to derive a closed form expression for the likelihood functions of the coin weights given the incomplete data, such an expression would be unwieldy and unfeasible to maximize, especially if the amount of experiments done increases. Computing the maximum likelihood estimates given the complete data however is trivial why the expectation maximization method is especially well suited for this problem.

2.7.2 The Algorithm

The maximum likelihood estimation for model parameters can be obtained by maximizing the log-likelihood function for the observed data, where the log-likelihood function looks like

$$\log(L(\theta)) = \log(g(x, \theta)).$$

However, since we want to avoid using the probability model for the incomplete data we instead chose to maximize the likelihood function for the complete data:

$$\log(L_c(\theta)) = \log(g_(y, \theta)),$$

since both models share the same parameter set. However, since this log likelihood is defined in terms of the complete data that we do not have we instead define the function

$$Q(\theta, \hat{\theta}) = E_{\hat{\theta}}(\log(L_c(\theta))|x), \tag{2.1}$$

that is the conditional expectation of the log-likelihood for the complete data given the incomplete data calculated using the parameter set $\hat{\theta}$. The EM algorithm now requires an initial value θ^0 and then proceeds in an iterative fashion by calculating $\theta^t = \arg\max_\theta Q(\theta, \theta^{t-1})$, until the convergence criterion $L(\theta^t) - L(\theta^{t-1}) < C$ for some threshold C is satisfied. It can be proved that by following the EM method, the sequence $L(\theta^t)$ will be increasing which means that convergence is assured and that it will produce better or equally good approximations of the true maximum likelihood estimates with iteration.

3
Methods

A shortcoming in the current probabilistic model used by the GF parser is that it is context-free. The current model cannot use information such as the fact that in the phrase *eat bass*, the word *bass* is much more likely to refer to the fish bass than the instrument bass, as each part of the tree is scored separately and independently of all other parts. The probabilistic model proposed is a way to use the ability to convert GF abstract syntax trees into abstract dependency trees in order to take context-based information into account. Abstract dependency trees as introduced in (Ranta and Kolachina, 2017) are basically sense tagged dependency trees using feature, part-of-speech and dependency labels according to the Universal Dependencies scheme.

The approach taken to disambiguation in this thesis is not a constructive approach in the sense that no focus is put on the parsing process and trying to *find* the correct abstract representation. Rather the approach taken is to develop a probabilistic model over **abstract dependency trees** able to score any abstract dependency tree according to the likelihood of that particular abstract dependency tree occurring in a text. This can then be used to evaluate parses done by the existing GF parser through conversion from abstract syntax to abstract dependencies and re-rank the k-best parses from the GF parser similar to what was done by Kolachina and Ranta (2015) to improve disambiguation in the GF parser. A secondary application used as an evaluation task is general purpose word sense disambiguation on dependency parsed sentences.

The probabilistic model over abstract syntax tree is a syntactic n-gram model over the abstract functions (the word senses) in the ADT, where the probabilities of a tree only depend on its subtrees of the form of one node and its $(n-1)$ immediate ancestors. Parameters for the statistical model is estimated from a large corpus of text in several different languages parsed with UD-pipe. There is no need for sense-tagged training data or parallel corpora. To circumvent the need for sense-tagged data, the sense tags are treated as latent information in the text and Expectation Maximization is used together with a knowledge source of the possible senses for each word in every language for parameter estimation. The knowledge sources used are the current GF dictionary, and synset information from the Open Multilingual Wordnet.

3.1 Tree Probabilities

In implementation and experiments focus have been limited to the exploration of syntactic bigrams only, mainly because of computational and memory complexity of the algorithm used for parameter estimation. In the bigram model used the probability for a tree T is calculated by the following product:

$$P(T) = P(N_0)\prod_{i=1}^{n} P(N_i|H(N_i)) = P(R_T)\frac{\prod_{i=1}^{n} P(N_i, H(N_i))}{\prod_{i=1}^{n} P(H(N_i))}, \tag{3.1}$$

where N_0 is the root node of the tree and $H(N_i)$ is the head node of N_i. Information contained in each node could in addition to the abstract function in the node include the dependency relation of the node and in the work both probabilities taking into account the dependency label and probabilities not taking this into account have been considered.

As a baseline to make comparisons to the current probabilistic model used in the GF parser possible a unigram model have also been used in which the probability for a tree T is calculated by the following product:

$$P(T) = \prod_{i=0}^{n} P(N_i), \tag{3.2}$$

that is a completely context free model where each node is assumed independent.

3.2 Parameter Estimation

In order for the probabilistic model to be useful one needs accurate estimations of the probabilities $P(N_i, H(N_i)), P(N_i), P(H(N_i))$. Since there is very little data labeled with abstract functions available to do such estimations directly through a method like maximum likelihood, the estimation is done through the use of a knowledge source of possible linearizations of every abstract function and corpora of untagged but dependency parsed text in several different languages. The method used is based on Expectation Maximization and is described in terms of estimating probabilities for subtrees of arbitrary size. It shall later be seen that the method does not scale very well computationally or memory wise to larger sub-trees, which is why experiments have been limited to subtrees of size two.

We assume that the set of all possible function tagged sub-trees is known and finite and we denote it

$$F = \{y_i\}_{i=1}^{N}.$$

The set of possible untagged subtrees will depend on the language the subtrees are defined over and in this work which language a subtree is defined over is always assumed to be known. The set of possible untagged subtrees for each language s is also assumed to be known and finite and is denoted

$$W_s = \{x_{si}\}_{i=1}^{M}.$$

Lastly, we also assume that we for each language have knowledge of a *possibility set*

$$D_s \subseteq W_s \times F,$$

where $(x_{si}, y_j) \in D_s$ if and only if y_j is a possible function tagged subtree given the untagged subtree x_{si}. The possibility sets used in this work are mostly derived using Wordnets and are discussed further in section **??**.

In the estimation of parameters using data from one or several corpora or treebanks, we consider a sequence of observations

$$O = (X_i, s_i)_{i=1}^{K}, X_i \in W_{s_i},$$

and assume that to every subtree X_i in the set, there is a fixed and pre-determined $Y_i \in F$ such that $(X_i, Y_i) \in D_{s_i}$. That is, we assume that there is one correct abstract representation for each observed subtree. However, these Y_i are unobservable and thus unknown. O is also called the observable information while the observable information together with the correct abstract representation is called the *complete information.*

The goal is to calculate the probability of seeing a certain abstract representation and we denote the probability of seeing the subtree $y_i \in F$ as

$$P(Y = y_i) = \pi_i.$$

Our goal will be to estimate the parameters $\pi = (\pi_1, \pi_2, ...)$. In order to facilitate the use of the Expectation-Maximization algorithm, we will also as a byproduct calculate the probability of an (untagged) sub-tree given that it has a certain abstract representation and we denote these probabilities

$$P(X = x_{sj} | Y = y_i, s) = \varphi_{sij}$$

.

3.2.1 Using a Maximum Likelihood Approach

Using a Maximum-likelihood approach to estimate the parameters (π, φ) means that we need to find the maximum of

$$P(\pi, \varphi | O) = \frac{P(O|\pi, \varphi)P(\pi, \varphi)}{P(O)},$$

where $P((\pi, \varphi))$ is our prior belief of (π, φ) and $P(O|(\pi, \varphi)) = l(O, (\pi, \varphi))$ is the likelihood function for the used data set. The problem at hand is to maximize this likelihood function with respect to the parameters .

Since we assume observations are independent the likelihood function for all observations will just be the product of the likelihood functions of each individual observation and we thus look at the likelihood function for each observation, that is $P(X|\pi, \varphi)$, separately. Now by the law of total probability, we have

$$\begin{aligned}P(X = x_j|\pi, \varphi) = &\sum_{i:(x_j,y_i)\in D_s} P(X = x_j, y_i|\pi, \varphi) = \\ &\sum_{i:(x_j,y_i)\in D_s} P(X = x_j|y_i, \pi, \varphi)P(y_i|\pi, \varphi) = \\ &\sum_{i:(x_j,y_i)\in D_s} \pi_i\varphi_{ij},\end{aligned}$$

which can be calculated without knowledge of the value of Y, since that variable was mariginalized out. However, as we want to calculate the total likelihood of all the observations

$$P(O|\pi, \varphi) = \prod_{n=1}^{K} P(X = x_{j(n)}|\pi, \varphi) = \prod_{n=1}^{K} \sum_{i:(X_n,y_i)\in D_s(n)} \pi_i\varphi_{ij(n)},$$

which unless the sets D_s are shaped very nicely will quickly become computationally intractable to maximize as the number of observations get large. However, this is almost the problem formulation that the Expectation Maximization algorithm is designed to find approximate solutions to.

3.2.2 Using Expectation Maximization

Using the expectation maximization method we look at maximizing the log-likelihood for the observations $L(O, \pi, \varphi)$, which becomes $L(O, \pi, \varphi) = \sum \log(\sum \varphi_{ji}^{s_i}\pi_i)$. Calling the complete parameter set $\theta = (\pi, \varphi)$ in the expectation step we get the expression

$$\begin{aligned}Q(\theta, \theta_{t-1}) &= E_{\theta_{t-1}}\left(L(C, \theta)|O\right) \\ &= \sum_{n=1}^{K}\sum_{j=1}^{N} L(X_n, y_j, \theta)P(y_j|X, \theta_{t-1}) \\ &= \sum_{s\in S}\sum_{i=1}^{N_s}\sum_{j=1}^{M} L(x_i, y_j, \pi, \varphi)c_{si}P(y_j|x_i, \theta_{t-1}) \\ &= \sum_{s\in S}\sum_{i=1}^{N_s}\sum_{j=1}^{M}(\log(\varphi_{sij}) + \log(\pi_j))c_{si}P(y_j|x_i, \theta_{t-1}),\end{aligned}$$

where c_{si} denotes the count of the observations of the form (x_i, s). By rewriting the conditional probability of y_i given x_i in terms of the parameters in our model

$$\begin{aligned}P(y_j|x_i, \theta_{t-1}) &= \frac{P(x_i|y_j, \theta_{t-1})P(y_j, \theta_{t-1})}{\sum_{k:(x_i,y_k)\in D_s} P(x_i|y_k, \theta_{t-1})P(y_k, \theta_{t-1})} \\ &= \frac{c_{si}\varphi_{sij}^{t-1}\pi_j^{t-1}}{\sum_{k:(X_n,y_k)\in D_s} \varphi_{ik}^{t-1}\pi_k^{t-1}},\end{aligned}$$

and denote the expected counts for the complete information at iteration t as

$$\hat{c}_{sij}^{t} = \frac{c_{si}\varphi_{sij}^{t-1}\pi_j^{t-1}}{\sum_{k:y_k\in F} \varphi_{sik}^{t-1}\pi_k^{t-1}} = c_{si}P(Y = y_j|X = x_i, s, \pi^{t-1}, \varphi^{t-1})$$

for all s, i, j such that $(x_i, y_j) \in D_s$ we see that we can write the expected value of the log likelihood in the expectation step as:

$$Q(\theta, \theta^{t-1}) = \sum_j \log(\pi_j) \sum_{s,i} \hat{c}^t_{sij} + \sum_{sij} \log(\varphi_{sij}) \hat{c}^t_{sij},$$

where the sums are all taken with respect to the sets D_s that is such that $\hat{c}^t_{sij}$ is defined. For the maximization step we can maximize each of the parameter subsets: π and $\varphi_{s.j}$ separately as they only occur in separate terms and have no interdependent constraints.

$$\pi^t = \arg\max_{\pi} \sum_j \log(\pi_j) \sum_{s,i} \hat{c}^t_{sij}, \sum_j \pi_j = 1$$
$$\varphi^t_{s.j} = \arg\max_{\varphi_{s.j}} \sum_i \log(\varphi_{sij}) \hat{c}^t_{sij}, \sum_i \varphi_{sij} = 1$$

Applying lagrange multipliers and setting derivative to zero gives:

$$\begin{aligned} \pi^t_j &= \frac{\sum_{s,i} \hat{c}^t_{sij}}{\sum_{s,i,j} \hat{c}^t_{sij}} = \sum_{s,i} \frac{\hat{c}^t_{sij}}{N} \\ \varphi^t_{sij} &= \frac{\hat{c}^t_{sij}}{\sum_i \hat{c}^t_{sij}}, \end{aligned} \tag{3.3}$$

again with the sums taken only over terms such that $\hat{c}^t_{sij}$ is defined. Note that the computational complexity of updating the probabilities will depend on the size of the sets of the possible function tagged sub-trees and the possible untagged sub-trees but most importantly of the set of possible combinations of these, that is the sets D_s. If this set would be choosen to be the full product sets $W_s \times F$ the computational complexity would be of the order $O(|S| * |F| * |W|)$, while if we use a knowledge based source, such as a dictionary to prune the set of possible combinations, we reduce the complexity to down to $O(k * |S| * |F|)$ where k is the average number of possible untagged subtrees that can be paired with each function tagged sub-tree.

3.3 Defining Possibility Sets for Sub-Trees

A key process for the parameter estimation to work is that there is a way to generate a proper mapping between observed and latent set for each language, that is D_s. This is done by assuming that the dependency structure will remain unchanged in the transition from observed dependency trees to abstract dependency trees. This means that the only thing that needs to be done is to translate each word in the tree to an abstract representation of the meaning of the corresponding word. Thus the only knowledge needed by the algorithm prior to parameter estimation on corpus data are dictionaries specifying possible abstract representations for each word in each language. Three different such dictionaries were used:

1. The current GF wide range translation dictionary.
2. Open Multilingual Wordnet.
3. A modified Wordnet where low-frequency noun synsets are merged with their hypernym synset.

For subtrees containing several words, the possible abstract subtrees are simply taken to be the Cartesian product of the possible representations for each word in the subtree together with the dependency structure for the tree. The particular model evaluated in this work focused on sub-trees of size two only and was limited to looking only at the lemma, part of speech tag, and the dependency relation between every two connected words in a dependency tree. The dictionary information given was dictionaries specifying a mapping from triples on lemma, part of speech and language to a list of possible abstract representation. As seen in figure 3.1, an example of a possible observed subtree would be *(bass, noun, obj, play, verb)*, which would mean that we observed a subtree where the noun *bass* was the child to the verb *play* with the dependency label *obj* — meaning the *bass* was the object of *play*. If in the possibility dictionary used there are two abstract representations each for the lemma/part of speech pairs, for example *bass.n.1, bass.n.2 and play.v.1, play.v.2* respectively, the possible abstract subtrees would be *(bass.n.1, obj, play.v.1), (bass.n.2, obj, play.v.1), (bass.n.1, obj, play.v.2)* and *(bass.n.2, obj, play.v.2)*.

Since the number of possible abstract representations of a subtree is equal to the number of possible combination of choices of abstract representation for each word in the subtree, the number of possibilities per tree will grow exponentially with the tree size, which is the main reason why experiments were limited to subtrees of size two.

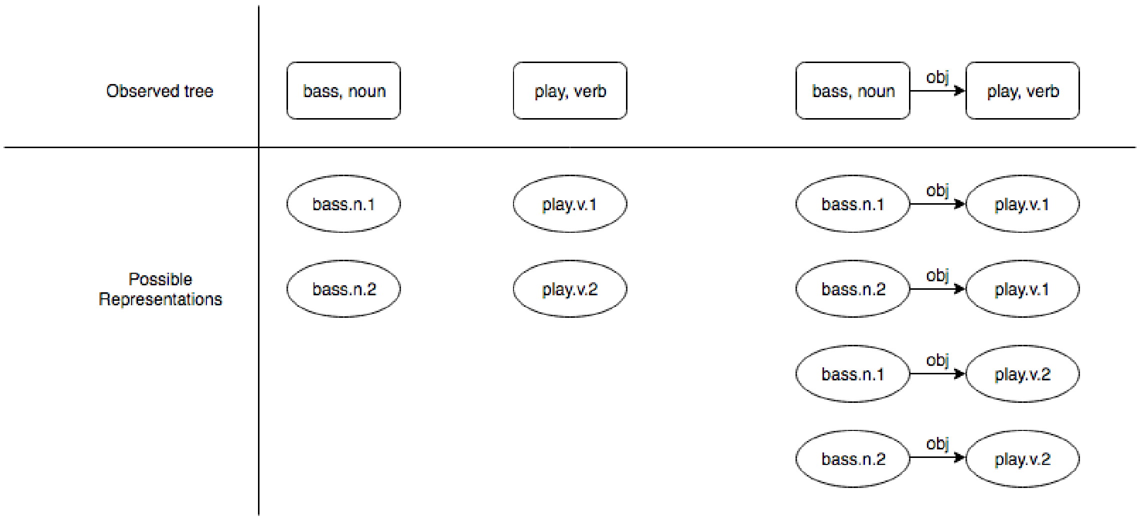


Figure 3.1: Illustration of the possible abstract representations for different subtrees of size one and two given the same dictionary of possible abstract representations..

3.4 Splitting Calculations into Sub-Problems

One way to reduce the computational complexity is to realize that in many cases the problem of estimating parameters often can be divided into several sub-problems, depending on the nature of the vocabulary sets, dictionary sets, and the abstract set. In particular we consider sets F, and for each language s W_s, D_s that can be partitioned into subsets F', W'_s, D'_s, F'', W''_s, D''_s, such that for any language, observation and abstract representation s, x, y

$$\begin{aligned}(x,y) \in D'_s, s \in S &\implies x \in W'_s, y \in F' \\ (x,y) \in D''_s, s \in S &\implies x \in W''_s, y \in F''.\end{aligned}$$

This means that if two words could possibly represent the same abstract function they must necessarily be in the same partition and if two abstract functions can linearize to the same word they must necessarily be in the same partition.

Assuming such a partition is possible we have that

$$x_{si} \in W'_s, y_j \in F'' \implies \varphi^t_{sij} = P(X = x_{si}|Y = y_j) = 0 \implies \hat{c}^t_{sij} = 0,$$

for all t. Looking at equations 3.2.2 and 3.3 it is now easy to see that for any s, i, j such that $x_i \in W'_s, y_j \in F'$ and any k, l, m such that $x_l \in W''_k, y_m \in F''$, the updated parameters corresponding to observations and abstract representations in the first set of partitions π^t_j, φ^t_{sij} will not depend on parameters corresponding to the second set of partitions $\pi^{t-1}_m, \varphi^{t-1}_{klm}$ and vice versa. This means that all calculations can be done separately for each of the two partitions. This kind of division into sub problems can of course be done recursively by applying the same procedure iteratively on the partitioned sets in order to ultimately divide the problem into a set of minimal sub problems.

In the setup used in the experiments we have used that the part of speech tag for each word in a dependency tree is known, meaning that given a bigram with a verb as the head and a noun as the dependent the abstract representation must necessarily have abstract functions of those two categories. In the setup where dependency labels are used and incorporated in the abstract representation these also further allow partitioning, as all observations of bigrams with a certain dependency label must necessarily be represented by an abstract representation containing the same dependency label.

Cursory experiments showed that for the Wordnet dictionaries, attempts of further partitioning apart from the use of the obvious partitions mentioned above likely were to get very small gains as soon as several languages were used for estimation. This because almost all of the synsets within one word-class would be in one large partition. Due to this, no investigation was done to do further partitioning.

3.5 Merged Wordnet

To reduce the size of the abstract function set a custom dictionary of possible abstract representations based on wordnet was compiled where abstract representations of each word were composed of merged clusters of wordnet synsets. These clusters were made by using the hypernym relations defined for nouns and verbs in wordnet such that rarely seen synsets were merged together with other rare synsets that share the same hypernym. For example, a possible such merge would be to take the two rarely seen synsets corresponding to the fishes bass and trout, bass.n.1 and trout.n.1, and merge them into their common hypernym synset fish.n.1. As can be seen in figure **??**, representing the two fishes with a common synset will as a result also make the two subtrees (bass, obj, eat) and (trout, obj, eat) share a common abstract representation.

The procedure for merging synsets is based on probabilities estimated using a unigram model based on the wordnet dictionary as described in the previous section, that is a model only looking at sub-trees of size one. These probabilities can be interpreted as estimations of the frequency with which each synset appears in the text. Using these probabilities, the *information content* of each synset is defined as

$$I(y) = -\log \sum_{y' \in h(y)} P(y'), \tag{3.4}$$

where $h(y_i)$ is the set of all subsumers in the hypernym hierarchy of y_i. This definition of synset information content is the same as the one used in (Resnik, 1995). An information cutoff C is made and one cluster is created for each synset with an information content lower than C. All synsets with an information content higher than C are then assigned to the cluster represented by its subsumer subset with the highest information content lower than C. That is, each synset y is assigned to the cluster represented by $f(y)$ where

$$f(y) = \operatorname*{arg\,max}_{y' \in h(y), I(y') < C} I(y'). \tag{3.5}$$

A new abstract function set and new dictionary sets can now be constructed by replacing each abstract function in the wordnet dictionary with the clustered representation of that synset.

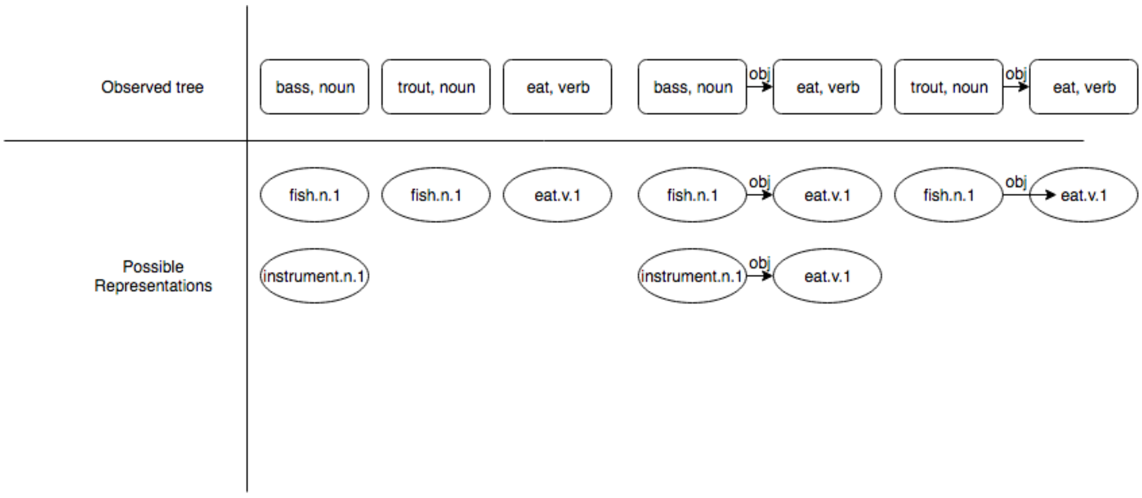


Figure 3.2: Illustration of the possible abstract representations for different subtrees using the clustered dictionary. Note that both the words trout and bass point to the cluster *fish.n.1* allowing us to capture the information that *fish* is something that often is the object of *eat* from a wider range of observations.

3.6 Data Sources and Parsing

The data used to estimate the parameters for the probabilistic model is the automatically parsed training data used for CoNLL 2017 shared task (Ginter et al., 2017). This data has been dependency parsed with the tool UDPipe and come in CoNNL-U format. From the parsed trees all subtrees of size two were counted using the lemma, part of speech tag and the dependency relation between the two nodes for each language separately. For each of the dictionaries used all subtrees containing unknown words was discarded since many trees contained nonsense words likely originating from faulty parsing by the Common Crawl, which is one of the data sources used for the CoNLL 2017 data.

3.7 Evaluation Models

For evaluation purposes, five variations of the basic model was defined. First, in order to assign probabilities for bigrams not observed in the training data, two different backoff strategies was chosen. In the first model, we utilized **stupid backoff**, introduced by Brants et al. (2007). Stupid backoff doesn't define a normalized probability, but the "score" for one bigram is defined as following:

$$S(\text{word}|\text{head}) = \begin{cases} \frac{\text{count(word, head)}}{\text{count(head)}} & \text{if count(word, head)} > 0 \\ \lambda \cdot \frac{\text{count(word)}}{N} & \text{otherwise,} \end{cases}$$

where N is the total number of counts and $\lambda = 0.4$ a standard backoff constant. This is a very simple and memory-effective smoothing method, and it has been show to approach state-of-the-art methods such as Kneser-Nay smoothing in large-scale models (Brants et al., 2007).

The second model is an interpolation model which also has been shown to yield good results (Chen and Goodman, 1996). It is a linear combination between the unigram and bigram together with the part-of-speech bigram and unigram. The score for an example bigram in this model is defined as follows:

$$\begin{aligned} S(\text{football}|\text{play}) = {} & \lambda_0 P(\text{football}|\text{play}) + \lambda_1 P(\text{football}) \\ & + \lambda_2 P(\text{noun}|\text{verb}) + \lambda_3 P(\text{noun}), \end{aligned}$$

with the interpolation constants set to $\lambda_0 = 0.4$ and $\lambda_1 = \lambda_2 = \lambda_3 = 0.2$.

Except for these two models, we also created a variant where the bigram models is not only conditioned on the abstract function of the head, but also the dependency relation label between the dependent and the head node in the tree. This means that the stupid backoff model is updated as follows:

$$S(\text{word}|\text{head, deprel}) = \begin{cases} \frac{\text{count(word, head, deprel)}}{\text{count(head, deprel)}} & \text{if count(word, head, deprel)} > 0 \\ \lambda \cdot \frac{\text{count(word)}}{N} & \text{otherwise,} \end{cases},$$

with N and $\lambda = 0.4$ as before. The interpolation model was extended in a similar way, which means that we in total have five variations on our model:

1. Unigram.
2. Bigram with stupid backoff.
3. Bigram with stupid backoff and dependency relation label.
4. Interpolation.
5. Interpolation with dependency relation label.

4 Results

The problem with unsupervised methods like ours is that there is no good standardized tests to use in order to evaluate performance. Since the main goal is to define a probability model for disambiguating abstract syntax trees, the obvious evaluation metric would be how well it ranks the trees generated by the GF parser. To do this, we defined a few example sentences with known ambiguous meanings. Another evaluation metric used is sense disambiguation. The advantage in looking at this subproblem is that we can compare ourselves to other known methods. Here, we utilize the standard measurements precision, recall and F1:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textbf{Recall} = \frac{TP}{TP + TN}$$

$$\textbf{F1} = \frac{2 \cdot \textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

For this task, two datasets are used. One is “trainomatic” created by Pasini and Navigli (2017), an automatically generated dataset with high quality sensed tagged data. Another is the Semeval 2015 disambiguation task (Moro and Navigli, 2015), a word sense disambiguation task where all words in a sentence need to be disambiguated.

4.1 Dictionaries

As our model is dependent on the underlying **knowledge graph**, or dictionary, we made experiments using three different variants:

1. Current GF dictionary
2. Wordnet
3. Merged Wordnet

Currently, it’s only the GF dictionary that is compatible with doing experiments with the GF parser, but for the words disambiguation evaluations we also did experiments on Wordnet and the merged Wordnet created by us. This merged Wordnet was created as described in section 3.5, some statistics is shown in table 4.2, and an example of the merged synset is shown in table 4.3.

Dictionary	**Language**	**Lemmas**	**Ambiguous lemmas**
GF dictionary	English	64,710	8.1%
GF dictionary	Finnish	40,907	35.5%
GF dictionary	Bulgarian	31,078	25.8%
GF dictionary	Swedish	30,196	21.2%
GF dictionary	French	23,901	23.5%
Wordnet	English	156,889	16.9%
Wordnet	Finnish	122,961	16.7%
Wordnet	French	57,252	23.6%
Wordnet	Bulgarian	6,699	22.7%
Wordnet	Swedish	5,980	16.8%

Table 4.1: Statistics about Wordnet and the GF dictionary. Ambigiuous lemmas have more than one meaning.

Average number of merged synsets per cluster:	6.3
Number of clusters:	5302
Number of synset merged:	33402

Table 4.2: Statistics about the merged Wordnet.

Cluster	spiritual_leader.n.01	investigation.n.02	sketch.n.01
Synsets	rabbi.n.01	inquest.n.01	draft.n.03
	cantor.n.02	inquiry.n.03	vignette.n.03
	patriarch.n.01	tabulation.n.02	
	catholicos.n.01	wiretap.n.01	
	evangelist.n.02	empiricism.n.02	

Cluster	seafood.n.01	bank.n.01
Synsets	octopus.n.01	riverbank.n.01
	coral.n.03	waterside.n.01
	whelk.n.01	
	roe.n.01	
	caviar.n.01	
	prawn.n.01	
	seafood.n.01	

Table 4.3: A few example of the synsets merged together in our ‘cluster’ model.

4.2 Training

We trained our models using parsed data from English, Swedish, Finnish, French, and Bulgarian, in total over 10^9 sentences. We performed the training for each model three times: First, using all five languages, second, we removed English from the training data and trained the model using the other four languages, and third, we only trained the model using English data.

4.3 GF Parser

In order to develop a small test set of sentences for qualitative evaluation, we utilized the GF gold tree database which contains a small number of trees tagged correctly. Using these, we constructed a list of simple sentences where the GF parser struggled. One such sentence is "I work at the bank" where the GF parser have troubles disambiguating between "work" in the sense of laboring and in the sense of functioning. The model was a stupid backoff model as described in section 3.7, trained on all seven languages. A summery of the results are available in table 4.5 and for the full list of sentences, see appendix A.1

The top parsing results from the sentence "he works at the bank" can be seen in table 4.4. This sentence contains two ambiguous words: *work* can mean to labor or to function, *bank* can be the river bank and the institution. We can see that our model correctly is able to determine that the probability that *he works* as in earn his living is much more probable than *he works* in the sense that a machine is functioning. The model has more problems with disambiguating between the different senses of *bank* though, which can be traced to the sparseness of these expressions in the dataset.

"He works at the bank"

GF parser	**Rerank**	**Interpretation**
26.398	16.542	he *earns his living* at the *bank institution*
29.458	16.802	he *earns his living* at the *river bank*
26.398	45.383	he *functions* at the *bank institution*
29.458	37.562	he *functions* at the *river bank*

Table 4.4: The complete phrases that the GF parser generate for the phrase "he works at the bank". The numbers are the negative log probabilities from the GF parser and from our reranking. We can see that our model successfully assigns a high probability to *earn a living* as opposed to *is functioning* at a bank.

4.4 Word Sense Disambiguation I: Trainomatic

In addition to the qualitative evaluation described in the previous section we also wanted to do a quantitative evaluation. In order to do so it is vital to define a clear testing criteria with a clear right or wrong answer. Here, the word sense

Test sentences	
GF	21
Rerank	36
Total # of sentences	47

Table 4.5: Total number of successful rankings of trees by the GF parser and our reranking model.

disambiguation is a suitable subtask. It is important to point out that this is only one part of tree disambiguation.

The first dataset we utilized was the Trainomatic dataset (Pasini and Navigli, 2017), a large automatically tagged set of sentences where one word has been tagged with its wordnet synset. This gives us a large testing dataset and the results can be seen in table 4.6 and 4.7. The evaluation is run on English language data and our model is trained with the tree different datasets described in section 4.2. As described in section 3.7, we defined five different variations of our model:

1. Unigram.
2. Bigram with dependency relationship label information.
3. Bigram without dependency relationship label information.
4. Interpolation with dependency relationship label information.
5. Interpolation without dependency relationship label information.

each of which was trained on the tree different language constellations defined in section 4.2. As a baseline, one random synset in our dictionary was selected. The results for the models using the GF dictionery is shown in table 4.6. We see that the precision is very high across the board, but the improvement from the random baseline is very low. This can be explained by that the ambiguity of the English dictionary is very low (see table **??**). This means that the sentences we are able to assign a abstract function to will generally be correct.

In table 4.7, the results from using the Wordnet dictionary is displayed. Here we can see a better improvement over the random baseline. We see that we get a performance boost for using the merged Wordnet. Additionally, we can see that the performance is consistent over the language variants, which was one of the goals. This also means that much of the signal comes from synonyms in the Wordnet graph. This is also supported by the fact that the models only trained on English generally performs the same or better than the ones trained on all languages, but on the unigram model (where we are unable to get information from synonyms) the model only trained on English are significantly worse than the random baseline.

4.5 Word Sense Disambiguation II: Semeval

The second word sense disambiguation dataset is from the Semeval 2015 task 13 (Moro and Navigli, 2015). Titled *Multilingual All-Words Sense Disambiguation and Entity Linking*, the task focus both on word sense disambiguation and entity linking, while our method only performs word sense disambiguation. We used the five different models from section 3.7 and annotated the given sentences with our predicted

Model	**Training Data**	**Precision**	**Recall**	**F1**
GF Unigram	5 languages	95.3	47.0	63.0
	No English	95.3	47.0	63.0
	Only English	95.4	47.1	63.1
GF Interpolation Deprel	5 languages	94.9	46.8	62.7
	No English	89.1	44.0	58.9
	Only English	94.3	46.5	62.3
GF Interpolation	5 languages	95.6	47.2	63.2
	No English	95.1	47.0	62.9
	Only English	95.1	46.9	62.8
GF Bigram Deprel	5 languages	94.9	46.9	62.8
	No English	89.2	44.0	59.0
	Only English	94.1	46.5	62.2
GF Bigram	5 languages	95.7	47.2	63.2
	No English	95.3	47.0	63.0
	Only English	95.1	46.9	62.8
Random Baseline		95.4	47.1	63.1

Table 4.6: Results for models using the GF dictionary on the Trainomatic evaluation set.

Wordnet senses. The authors of the task provided a scorer which we used, and got the results given in table 4.8 for normal Wordnet and in table 4.9 for the merged Wordnet. Our results are compareble to the **TeamUFAL** system described in the Semeval task description (Moro and Navigli, 2015). TeamUFAL is a unsupervised method using Wikipedia and Wordnet senses to disambiguate senses.

Model	**Training Data**	**P**	**R**	**F1**
Wordnet Unigram	5 languages	30.2	24.7	27.2
	No English	32.8	27.1	29.7
	Only English	27.5	22.9	25.0
Wordnet Interpolation Deprel	5 languages	32.7	27.1	29.6
	No English	29.2	24.0	26.3
	Only English	34.4	28.1	30.9
Wordnet Interpolation	5 languages	31.2	25.8	28.3
	No English	32.4	26.7	29.3
	Only English	32.8	26.6	29.4
Wordnet Bigram Deprel	5 languages	32.8	27.1	29.7
	No English	30.0	24.7	27.1
	Only English	34.8	28.2	31.1
Wordnet Bigram	5 languages	33.1	27.3	29.9
	No English	31.9	26.0	28.7
	Only English	34.7	28.2	31.1
Clustered Interpolation Deprel	5 languages	48.4	43.2	45.7
	No English	46.6	41.6	44.0
	Only English	49.0	44.2	46.4
Clustered Interpolation	5 languages	48.2	43.0	45.5
	No English	49.1	43.9	46.3
	Only English	48.6	43.8	46.0
Clustered Bigram Deprel	5 languages	49.2	43.9	46.4
	No English	47.3	42.6	44.8
	Only English	49.9	44.6	47.1
Clustered Bigram	5 languages	49.2	43.9	46.4
	No English	48.8	43.5	46.0
	Only English	49.3	44.4	46.7
Random Baseline		33.0	27.3	29.9

Table 4.7: Results for models using the Wordnet dictionary on the Trainomatic evaluation set.

Model	Training Data	P	R	F1
Wordnet Bigram Deprel	5 languages	47.7	43.1	45.3
	No English	44.9	40.5	42.6
	Only English	44.5	40.2	42.2
Wordnet Bigram	5 languages	46.9	42.3	44.5
	No English	45.3	40.9	43.0
	Only English	44.2	39.9	42.0
Wordnet Interpolation Deprel	5 languages	46.4	41.9	44.0
	No English	42.6	38.5	40.4
	Only English	44.4	40.1	42.1
Wordnet Interpolation	5 languages	44.3	40.0	42.1
	No English	43.1	38.9	40.9
	Only English	44.2	39.9	42.0
Wordnet Unigram	5 languages	45.0	40.6	42.7
	No English	46.7	42.1	44.3
	Only English	40.3	36.3	38.2

Table 4.8: Results for models using the Wordnet dictionary on the Semeval 2015 test data set.

Model	Training Data	P	R	F1
Clustered Bigram Deprel	5 languages	44.7	40.3	42.4
	No English	43.8	39.5	41.5
	Only English	46.0	41.5	43.7
Clustered Bigram	5 languages	45.8	41.3	43.4
	No English	46.4	41.9	44.0
	Only English	47.2	42.6	44.7
Clustered Interpolation Deprel	5 languages	44.2	39.8	41.9
	No English	44.2	39.8	41.9
	Only English	46.7	42.1	44.3
Clustered Interpolation	5 languages	44.8	40.4	42.5
	No English	46.9	42.3	44.5
	Only English	45.8	41.4	43.5
Clustered Unigram	5 languages	44.7	40.3	42.4
	No English	46.1	41.6	43.8
	Only English	44.8	40.4	42.5

Table 4.9: Results for models using the Clustered Wordnet dictionary on the Semeval 2015 test data set.

5
Conclusion

In this thesis we have implemented a method for disambiguating between parse trees, especially focused on the abstract syntax trees created by the GF parser. In order to do this, we developed a language model able to assign probabilities to these trees, we decided to work with a bigram model defined on a syntactic tree. In order to estimate the parameters of this model it is necessary to assign probabilities to the underlying meaning of the words, and not the lemmas themselves. This was done by utilizing the Expectation Maximization, an unsupervised method for estimating unobserved variables. We adapted the method to our context and ran test to show that we got useful signals. The evaluation was done on both a small set of hand-crafted gold trees, but also on datasets with sense-tagged data with which we could test the word-sense-disambiguation part of our problem. Our results show that a syntactic bigram model is enough to get signal, and can be compared to other supervised methods competing in Semeval 2015. We can also note that the quality of the dictionary is important as the performance varies widely across different dictionaries.

5.1 Discussion

One big challenge of unsupervised methods like ours is develop good evaluation criteria, and the evaluation methods we used has a few problems. First, one problem of the task with re-ranking trees parsed by the GF parser, is that we can do evaluation with any other dictionary than the one built into GF. Instead, evaluation for other dictionaries was left at doing word sense disambiguation over UD dependency trees using the Trainomatic and Semeval evaluation sets. In this evaluation the GF dictionary performed relatively poorly together with the method compared to the Wordnet based dictionary. Although our experiments indicate that the method is able to produce results in doing lexical disambiguation it does not give any indication on whether the method is able to rank syntactic ambiguities in a meaningful way or not.

For the word sense disambiguation/entity linking SemEval 2015 shared task the results are comparable to those of TeamUFAL, a unsupervised method based on Wikipedia. However, it is important to remember that this kind of task is a secondary application and should only be seen as an indicator of the methods capability of disambiguating trees in the lexical sense.

A shortcoming of the method is the disk space needed to store the estimated parameter sets even for the clustered bigram models. This points to that in order to expand the method to larger models estimating probabilities for larger subtrees, a way to shrink the parameter space through some kind of dimension reduction or pruning is necessary. Possible solutions would be to encode the information more densely through some kind of vector embeddings. The fact that the clustered wordnet dictionary, which can be seen as a type of dimension reduction, performs well can be seen as a positive indicator that investigating such techniques have merit and could in fact increase the performance because due to alleviation of the problem of data sparsity.

In analyzing the results it is hard to isolate where in the stack improvements would yield a higher performance in the model due to the many different parts of the model. Possible points where improvement could be of importance include the parsing of UD trees, the syntactic n-gram probability model used, the parameter estimation done through EM and the knowledge source used as dictionary. The point most thoroughly investigated in this work is the usage of different dictionaries, which was shown to be very significant.

5.2 Conclusion

We believe that this thesis shows that it is possible to combine the GF parser with more advanced statistics than is currently used, while still preserving the unique strengths of GF: to be explainable and language independent. This thesis could then be the starting point of several interesting research direction in which the strengths of GF and formal language methods is married with recent advances in statistical language processing.

From the results from the word sense disambiguation evaluation task it can be concluded that the test data and model where the most significant results are seen is the clustered wordnet evaluated on the trainomatic data set. For the same data set the ordinary bigram model barely performs better than choosing a synset randomly and in some cases in fact performs worse. However, for the semeval task the non-clustered dictionary performs better, but the best model, which is the bigram model with dependency labels, only performs marginally better than the unigram model trained on data from all languages except English.

5.3 Future Work

One of the largest drawback with the proposed model is that it currently doesn't scale very well with respect to the size of dictionary and the size of context used. However, as the experiments with the clustered Wordnet dictionary have shown that there is potential to maintain and in some cases even improve performance by using a simplified lower dimensional parameter space to approximate the full model. Further investigation could be done on using dimensionality reduction techniques such as singular value decomposition on the estimated parameters, which could possibly address the issue of the trained parameters taking very large disk space, although it is unclear whether this can be used to address computational issues experienced during parameter estimation.

Finding a way to scale the method would allow for the use of larger and different types of context in the models, such as the types of context used by Richardson et al. (2016) adapted to be invariant to the linear word order of sentences. This will most likely require some kind of dimension reduction and smoothing of the parameter space not only for computational reasons, but also to avoid problems pertaining to sparsity in the training data, problems that can already be seen in the bigram model.

Further application of the current model that merits exploration is further use of the language dependent parameters denoted φ. These are the conditional probabilities that a certain abstract representation is linearized to a certain observable subtree and could be used to improve disambiguation for languages where they are available, but could also be used to allow for probabilistic language generation.

As it is seen that the type of dictionary used very much influences performance of the model it would be of interest to explore other sources of knowledge to build vocabularies of possible abstract representations from. Suggestions include the use of Babelnet, which incorporates knowledge from a range of sources, the main ones including the Open Multilingual Wordnet and Wikipedia.

Finally an application of interest to investigate would be the possibility of enhancing the current context free GF parser to include context. It would be interesting to investigate whether it is possible to adapt head driven lexicalized parsers like the ones developed in (Collins, 2003) to parsing abstract syntax by discarding language specific features such as linear distance and direction.

Bibliography

Angelov, K. (2009). Incremental parsing with parallel multiple context-free grammars. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 69–76. Association for Computational Linguistics.

Angelov, K. and Ljunglöf, P. (2014). Fast statistical parsing with parallel multiple Context-Free grammars. *EACL*.

Bond, F. and Foster, R. (2013). Linking and extending an open multilingual wordnet. In *ACL (1)*, pages 1352–1362.

Bond, F. and Paik, K. (2012). A survey of wordnets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue. 64–71.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2007). Large language models in machine translation. In *In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Citeseer.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Chomsky, N. (1957). *Syntactic Structures*. Mouton & Co., The Hague, Netherlands.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa Italy.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Do, C. B. and Batzoglou, S. (2008). What is the expectation maximization algorithm? *Nature biotechnology*, 26(8):897–899.

Ginter, F., Hajič, J., Luotolahti, J., Straka, M., and Zeman, D. (2017). CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LIN-

DAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing (2:nd Edition).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Kolachina, P. and Ranta, A. (2015). GF wide-coverage English-Finnish MT system for WMT 2015. *WMT@ EMNLP*.

Kolachina, P. and Ranta, A. (2016). From abstract syntax to universal dependencies. *LiLT (Linguistic Issues in Language Technology)*.

Manning, C. D., Schütze, H., et al. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

McLachlan, G. and Krishnan, T. (2007). *The EM algorithm and extensions*, volume 382. John Wiley & Sons.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Moro, A. and Navigli, R. (2015). Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *SemEval@ NAACL-HLT*, pages 288–297.

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Pasini, T. and Navigli, R. (2017). Train-o-matic: Large-scale supervised word sense disambiguation in multiple languages without manual training data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 78–88.

Petrov, S., Das, D., and McDonald, R. (2011). A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Ranta, A. (2004). Grammatical framework. *Journal of Functional Programming*, 14(2):145–189.

Ranta, A. and Kolachina, P. (2017). From universal dependencies to abstract syntax. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies, 22 May, Gothenburg Sweden*, pages 107–116. Linköping University Electronic Press.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*.

Richardson, J., Kudo, T., Kazawa, H., and Kurohashi, S. (2016). A generalized dependency tree language model for smt. *Information and Media Technologies*, 11:213–235.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., and Chanona-Hernández, L. (2014). Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860.

Virk, S. M., Prasad, K., Ranta, A., and Angelov, K. (2014). Developing an interlingual translation lexicon using wordnets and grammatical framework. *COLING 2014*, page 55.

Wikimedia (2011). Tree illustrating the dependency and constituency relations. [Online; accessed Sep 5, 2017].

A

Appendix

A.1 Example sentences

List of example sentences used for evaluation. The checkmark is present if there is no false trees that get a higher probability than the real tree and if not the other parser performs better, that is, if one of the methods has assigned the same probability to a large set of trees while the other assigns it to only a correct subset of this set, then only the latter gets a checkmark.

Sentence	**GF parser**	**Reranker**
woman that sleeps	✓	✓
woman who sleeps		✓
woman who sleeps here		✓
woman that sleeps here	✓	✓
woman who is old		✓
woman that is old	✓	✓
whom does she see	✓	✓
because she sleeps	✓	✓
between you and me	✓	✓
both here and there	✓	✓
he works at the bank		✓
he works every day		✓
she works with many people	✓	✓
my computer works perfectly		✓
my computer works		✓
my computer doesn't work		✓
I like sleep	✓	
I am sleeping		✓
I am working		✓
he is working	✓	✓
she is a little girl		
she is a girl	✓	✓
I play football		✓
he plays the trumpet	✓	✓
the child plays with a doll		✓
the children play in the park		✓
the play is over soon	✓	✓
I went to school this morning		

he is in the cooking area		
area of the circle is small	✓	✓
fire the canons		
the work is considered canon		
you are in good company	✓	✓
this is company business		✓
what is the date today?	✓	✓
to go on a date		
he ate a date		✓
the future is uncertain	✓	✓
investing in futures is risky		
you need to lie down		✓
do not tell lies	✓	✓
submit the application		
the applications of this theory are numerous	✓	✓
he uses an application on his computer	✓	
my arm hurts		✓
the right to bear arms is important		✓
visiting relatives can be boring	✓	
total out of 47 sentences:	21	36