Feature Engineering for Data-Driven Dependency Parsing

Christian Rishøj Jensen

August 8, 2009

Forsiden Følgende oplysninger skal fremgå af specialets forside eller siden umiddelbart efter:

Titel og evt. undertitel Københavns Universitet Institut/Fag Dit navn Din vejleders navn År og måned for aflevering.

Alle specialer skal ved afleveringen have anført, hvor mange tegn inklusive mellemrum det færdige speciale udgør.

Contents

Chapter 1

Introduction

Language competence might well be among the most impressive of cognitive abilities, both with respect to the conceptual complexity conveyable in the linguistic vehicle, and the effectiveness with which we are able to grasp utterances and make sense of them. Thanks to the generative quality of language, by which phrases and sentences may be combined and embedded within each other recursively, the extension of language is potentially limitless. On the comprehension side, structureally ambigous and semantically underspecified utterances are swiftly decoded and assigned with a likely interpretation. A task that from a formal point of view has immense complexity is continually carried out by the language user, seemingly without any effort. During this process the comprehender presumably registers a variety of phenomena in the uttered speech or written text in order to contruct the proper interpretation. As examplars of this, apart from the sequence and identity of the uttered words themselves, the comprehender is likely to take of the *kind* of word being used, i.e. the *part of speech* to which the word belongs, *morphological* structure of the words themselves, as well as grammatical pauses in between words, and employ this information in interpretation. I might not know the word *flocculation*, but from its suffix of *-tion* I can tell that it is a noun, probably for an action or a condition, which can help me get an idea of how it fits in a structural analysis of the sentence. Clarifying remarks in the middle of an utterence might be indicated by a brief pause. And when my mother reports on a heated discussion with a salesperson, a clear change in tone of voice provides me with an indication of when she is *directly reporting* the other person's speech.

EXPAND: quotations in Danish?

In the field of human language technology, a computer system that performs a syntactic, semantic, or other level of structural analysis of a sentence is referred to as a *parser*. Modern day parsing systems also benefit from information other than the surface form of processed words. A common step prior to parsing is to identify the part of speech for each word. Tense, case, number, gender and other inflectional information is often also extracted and made explicit, in order to make the information carried herein available for the parser to take advantage

of it when making decisions in the parsing process.

One approach to automatic syntactic analysis, referred to as *dependency parsing*, has gained increasing interest in the last decade. While neither dependency parsing nor the underlying formalism of *dependency grammer* are new inventions, the intense interest likely stems from the success researches have had in constructing dependency parsers that do not rely on hand-crafted grammatical rules and lexica, but rather *learn* from vast amounts of example analyses. This training material, referred to as a *treebank,* consists of collections of sentences and corresponding structural analyses. Such *data-driven dependency parsers* have quickly risen to become competitive with traditional grammar-based systems in parsing accuracy, without the immensely time-consuming task of manually crafting a grammar. There is optimism in the research community that a hybrid of both worlds, in which a combination of different parsers allows one to benefit from the analysis of the other, will lead to even better results.

While data-driven parsers are able to learn their trade from examples, they are not cognitive models of human language ability, that is to say, approximations of human cognitive processes for the purpose of comprehension and prediction. Parsing systems are constructed with other goals in mind, predominantly automated language processing for such tasks as information retrieval, document classification and summarization, machine translation and automated dialog systems.

However, as parsing systems already benefit from cues such as inflectional features and part of speech, it does not seem far-fetched to assume that parsing could benefit from other cues that are of use in human apprehension.

The aim of this thesis shall be to investigate the following question: Is it possible to augment treebanks with additional or modified features, such that existing data-driven parsing systems generate better dependency parsers?

In parsing system research, there seems to be an emphasis on constructing parsers for written text. While noone would argue that a spoken utterence from a language user turns into another language if she chooses to write it on a piece of paper, there are however substantial differences. Interpretational cues found in spoken language do not always have direct equivalents in writing. Variations in tone of voice for example is not obvious from a written accout. Some cues are present in writing, though, and while the cues in speech are often subtle, the cues that are present in writing stand out more clearly. In particular, directly reported speech is conventionally marked with quotation marks, and inserted supplementary remarks are typically surrounded by parentheses, hyphens or – more subtly – commas. Thus, there are plenty of potential features in written texts to choose from as well.

So, the focus area of this project is squarely investigating feature engineering in data-driven parsing systems. But conversely, such experiments with feature engineering in data-driven parsing systems may also offer an interesting opportunity to investigate which features of spoken or written utterances might be of value for comprehension in genereal – including human. Given a sufficiently capable and trainable parsing model, if we are interested in whether for example capitalization of written text could play any role in comprehension, we could

train two models with the same training material, but make word capitalization explicit to only one of them. Once trained, let both models attempt an interpretation of the same text, and see if the extra feature provided to the latter model had any significant effect on the analytical performance.

Of couse, an observed positive effect of a feature in a parsing system would not entail the existence of a cognitive counterpart in use by language users. But it would provide evidence of the informational value of the cue with respect to structural comprehension.

Chapter 2

An Overview of Data-Driven Dependency Parsing

2.1 Dependency Grammar

By some estimates the tradition of dependency grammar can be traced as far back as some of the earliest works of descriptive and generative linguistics, namely the Sanskrit grammar of *Pāṇini* a few centuries before the Common Era. The tradition includes a large and diverse family of grammatical theories and formalisms, all sharing the basic assumption that syntactic structures essentially consists of *words* that are related to each other by binary, assymmetrical relations called *dependencies* [Kübler et al., 2009, Nivre, 2005], with one word in a dependency relation being the *head*, and the other being the *dependent*.

With respect to this basic property, dependency structures can be contrasted to *constituency structures*, another dominant syntactic representation, in which words are not directly related to each, but only indirectly through *non-terminal* nodes, each of which may in turn be related to another non-terminal node. A non-terminal node in a constituency structure is not manifest in the surface form of the sentence, but forms a group of words, or a *phrase*, which stand together as a conceptual unit.

The same conceptual units can be recognized in a dependency structure by their heads: All words that transitively depend on a head are part of the

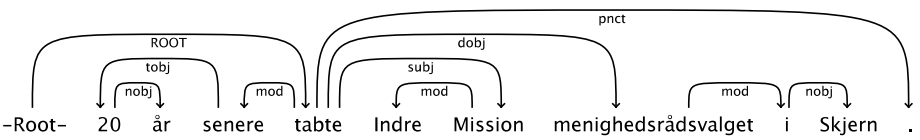


Figure 2.1: Dependency structure for a Danish sentence from the Danish Dependency Treebank [Kromann, 2003].

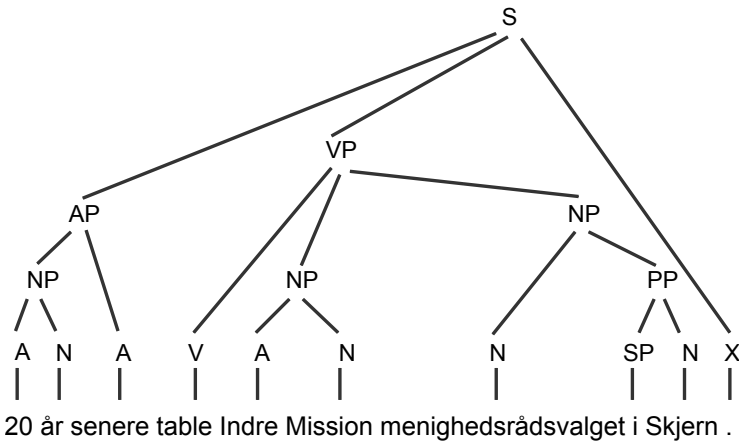


Figure 2.2: Possible constituent structure for the sentence in figure 2.1 on the previous page.

conceptual unit *governed* by the head. As a bit of additional nomenclature, dependents are said to *modify* their heads.

Various criteria for identifying dependency relations have been proposed. Some common criteria are [Nivre, 2005]:

1. A head determines the syntactic category of the dependent and can often replace the dependent.
2. A head determines the semantic category of the dependent.
3. Dependents provides semantic specification.
4. A head is obligatory, while dependents may be optional.
5. The head selects its dependent and determines whether the dependent is obligatory or optional.
6. The form of the dependent is determined by the head (agreement or government).
7. The linear position of a dependent is specified with reference to the head.

These criteria refer to a mix of phenomena on different levels of linguictics analysis, from morphosyntax to semantics. There seems to be no universally authoritative set of criteria for all dependency relations Kübler et al. [2009]. Rather, several levels of dependency analysis can be made on a sentence, and the criteria for identifying dependencies are chosen accordingly. Levels of dependency analysis include:

Morphosyntactic in which inflectional affixes are represented as separate tokens (useful for highly inflected languages).

Syntactic where dependencies identify syntactic functions, including PREDICATE, SUBJECT and OBJECT.

Semantic where semantic roles (including AGENT, PATIENT and GOAL) are designated by dependencies.

It is worth noting that dependency representation is not inherently limited to these levels of analysis. In principle, there is nothing that hinders the use of dependency analysis for identifying other relations between tokens for which a consistent set of criteria can be formulated. Additionally, theoretical frameworks for *multi-stratal* representations exist, in which several levels of analysis is represented in the same structure. For this project, the focus is exclusively on mono-stratal syntactic analysis.

In comparison to constituency structures, dependency structures are a more constrained representation, as the number of nodes to connect in the structure is fixed by the number of words in the sentence, whereas the constituency structure contains additional non-terminal nodes. To many researchers in computational linguistics, working with dependency representations has seemed more tractable and more apt for further semantic processing, thanks to the relatively transparent encoding of the predicate-argument structure of a sentence through word dependencies Nivre [2005].

2.1.1 Formal definitions

Following Kübler et al. [2009], let a sentence S be defined in formal terms as a sequence of tokens:

$$S = w_0 w_1 ... w_n$$

Dependency structure is defined on these tokens, so tokenization of the sentence must have taken place prior to the dependency analysis. Further, let R denote a finite set of dependency *relation types* – or *arc labels* – that can hold between tokens:

$$R = \{r_1, r_2, ..., r_m\}$$

A *dependency graph* G is then defined as a *labeled, directed* graph of nodes V and arcs A:

$$G = (V, A)$$

The nodes V in the graph are comprised of sentence tokens, and the labeled arcs A connect these:

$$\begin{aligned} V &\subseteq \{w_0, w_1, ..., w_n\} \\ A &\subseteq V \times R \times V \end{aligned}$$

In these terms, a dependency graph G represents a particular dependency analysis of the sentence S, with the arcs A being the dependencies posited by the analysis. For mono-stratal analysis, we only allow a single relation to hold from one token to another:

$$(w_i, r, w_J) \in A \Longrightarrow (w_i, r', w_J) \in A \text{ for all } r' \neq r$$

Note that the single arc restriction in this formulation is only refers to one direction, namely from w_i to w_j (denoted by $w_i \rightarrow w_j$). It does not prohibit another relation from going the other way, $w_j \rightarrow w_i$. Additionally, there is nothing that prohibits *cycles* from occurring in the the graph. Thus, let a *well-formed dependency graph* be any dependency graph G of nodes V and arcs A that is a *directed tree* rooted in node w_0 and has node set V_S which spans all nodes in the graph: $V = V_S$. It follows from the *spanning* property that the well-formed dependency graph is also *connected*. Well-formed dependency graphs are also called *dependency trees*. The constraints of well-formedness are assumed by most mono-stratal dependency theories, and in most cases also holds within each layer of a multi-stratal theory[Kübler et al., 2009].

A property of dependency trees that follow from the tree constraint itself is the *single-head property*, which states that if a dependency relation $w_i \rightarrow w_j$ holds between two tokens $w_i, w_j \in V$, then there can be no other $w_{i'} \in V$ such that $i' \neq i$ and $w_{i'} \rightarrow w_j$. There is some controversy to this property, as there are cases where it seems natural to have multiple heads on a dependent. In particular, in sentences with coordinated verbs – e.g. *guests enter and exit through the lobby* – it would appear reasonable to let the subject depend on both verbs. Most formalisms deal with dependents of coordinated phrases by simply letting the dependent modify the head of the coordinated phrase – whether it is chosen to be the coordination itself or one of the coordinated tokens.

2.1.2 Projectivity

A further restriction on dependency trees is often assumed for reasons of computational tractability, namely *projectiveness*. A dependency tree if projective if and only if it satisfies the *planar property*, namely if it is possible to graphically configure all the dependency arcs in a place above the sentence – without any crossing arcs. For such a depiction to be possible, the head w_i of all dependencies $w_i \rightarrow w_j$ in the tree must dominate every node w_m which occurs between

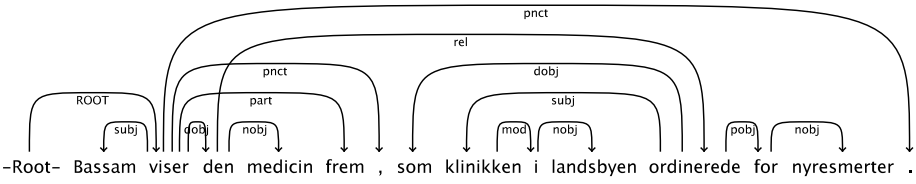


Figure 2.3: Non-projective dependency tree. The dependency *den* → *ordinerede* is crossing the dependency *viser* → *frem*.

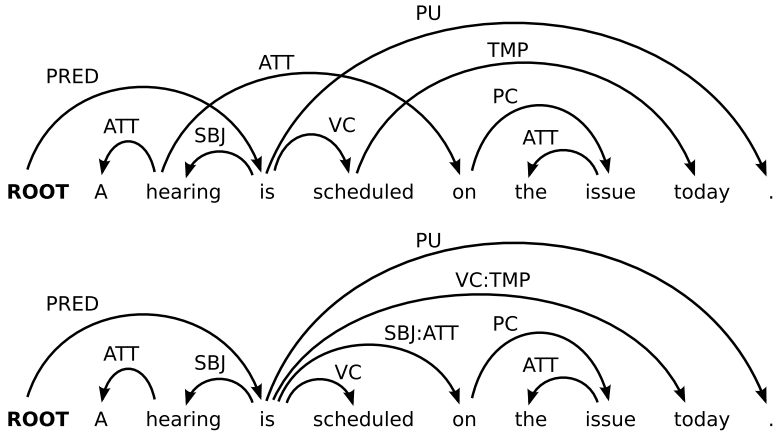


Figure 2.4: Projectivization of a non-projective dependency tree [Kübler et al., 2009].

the endpoints w_i, w_j in the linear order of the sentence. *Dominate* here is the transitive and reflexive closure of the dependency relation, so that a node is dominated by its head, the head of its head, and so forth up until the root node of the tree.

In formal terms, this amounts the following definitions [Kübler et al., 2009]:

1. Let $w_i \twoheadrightarrow w_j$ denote the reflexive and transitive closure of the dependency relation in a tree $G = (V, A)$: $w_i \twoheadrightarrow w_j$ if and only if $i = j$ (reflexive) or both $w_i \twoheadrightarrow w_{i'}$ and $w_i \rightarrow w_j$ for some $w_{i'} \in V$.

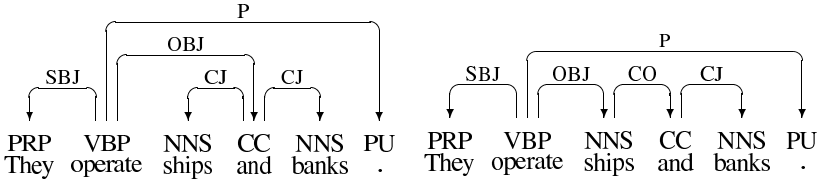
2. An arc $(w_i, r, w_j) \in A$ in a dependency tree $G = (V, A)$ is projective if and only if $w_i \twoheadrightarrow w_k$ for all $i < k < j$ when $i < j$, or $j < k < i$ when $j < i$.

3. A dependency tree $G = (V, A)$ is a projective dependency tree if all $(w_i, r, w_j) \in A$ are projective.

For notational convenience, let $\mathcal{G}_S^P$ denote the set of prejective dependency trees for a sentence S.

2.1.2.1 Projectivization

Non-projective dependencies can be caused by wh-movement, which occurs naturally in many languages with SVO as dominant order. As such, projectivity is not a linguistically plausible constraint. In cases where it nonetheless is assumed, non-projective cases are normally handled by having procedures for converting non-projective trees to projective trees (*projectivization*) and back (*de-projectivization*) as steps of pre- and post-processing (see figure 2.4).



(a) Coordinating conjunction as head (b) Coordinated conjunct as head

Figure 2.5: Different conventions for annotating coordinated noun phrases [Nivre, 2005].

2.2 Dependency Parsing

In this section I aim to outline of the problem dependency parsing, with the emphasis on data-driven parsing and the learning process entailed.

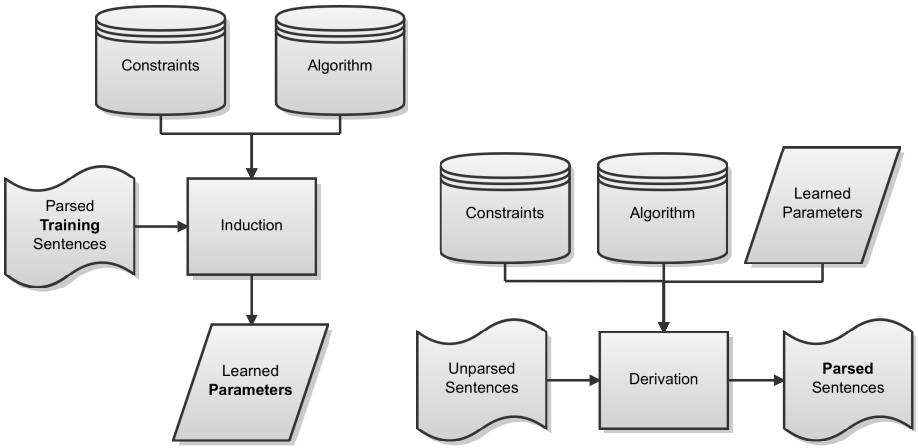
Simply put, dependency parsing is the task of constructing a dependency tree for a sentence. Approaches to the parsing problem can be characterized by being *grammar-based* or *data-driven*. An approach is grammar-based if a formally specified and often hand-crafted grammar is employed in parsing, such that it makes sense to ask whether a given sentence is producable by the grammar or not. In most data-driven approaches, a method from *machine learning* is used to train a parsing model on a collection of example analyzed sentences, resulting in some *learned parameters*, which in turn is used by the parsing model to parse new sentences. It is worth noting that these categories are not mutually exclusive, as some approaches induce grammars from collections of analyzed sentences, and this are indeed both data-driven and grammar-based.

In formal terms, a dependency parsing model M consists of:

1. a set of *constraints* Γ which define the space of possible dependency structures producable for a sentence,
2. a fixed *algorithm* h, and
3. a – possibly empty – set of *parameters* λ that guide the parsing algorithm.

For a grammar-based system, the constraints would include the rules of grammar in use.

The parsing problem in formal terms is then to find the most likely dependency tree $G = h(S, \Gamma, \lambda)$ for a sentence S. Which tree is most likely depends, apart from the general criteria for dependencies outlined on page 8, on the specific dependency formalism in use. In between formalisms there are various annotational conventions, not only with respect to the set of dependency types used, but also differences in treatmeant of such phenomena as coordination [Nivre, 2005], head-status of functional categories [Øvrelid , 2009] and locutions [Bosco et al., 2008], which are particularly open to interpretation from a dependency viewpoint.



(a) Learning phase (b) Parsing phase

Figure 2.6: Phases of data-driven parsing.

A data-driven parser must learn these annotational conventions when presented with the training material. In formal terms, the training material $\mathcal{D}$ is comprised of pairs of sentences and dependency trees [Kübler et al., 2009]:

$$\mathcal{D} = \{(S_d, G_d)\}_{d=0}^{|\mathcal{D}|}$$

The core of the learning phase typically involves adjusting the parameters λ in order to optimize some function over the training set $\mathcal{D}$. The nature of the parameters depend on the specific parsing model in use. In one model, a parameter might represent the likelihood of a dependency arc occurring in a sentence.

TODO: root token

2.2.1 Feature function

Also involved in the learning phase is a *feature function* $f(x) : \mathcal{X} \rightarrow \mathcal{Y}$, that maps some input x into the feature space $\mathcal{Y}$. The specific input for the feature function depends on the parsing model, but almost always includes a token from the sentence, and in some cases the state of the parser. In broad terms, the feature function takes information about a sentence that is deemed useful for the purpose learning and parsing, and makes it explicit to the machine learning component, typically in the form of a high-dimensional real-valued feature vector.

Example features for a given token include:

- Word form.
- Lemma.

- Part of speech.
- Morphological and inflectional properties.
- Similar information about adjacent tokens.

Features like these are mostly categorical, so it may not be obvious how to represent them in a real-valued vector space. The method of choice is to represent a m-valued categorical feature as an m-dimensional vector of zeros and a single one, indicating the categorical feature value.

2.2.2 Transition-based models

Transition-based models process sentence tokens sequentially in a stepwise manner, and construct a dependency tree for the sentence as a side-effect of the *transitions* made in an *abstract machine* as the parsing proceeds. At each step, the machine is in a certain *state* determined by the position in the sentencen and the *parsing history*, ie. the transitions performed prior to the current state.

2.2.2.1 Shift-reduce models

Several transition-based models have been proposed [Nivre et al., 2007], many of which are inspired by traditional shift-reduce parsers[Nivre, 2003, Hall et al., 2007]. In these, the state – or *configuration* – of the transition-based parser for a sentence S is a triple $c = (\sigma, \beta, A)$ of:

1. A stack α of tokens $w_i \in V_S$,
2. a buffer β of tokens $w_i \in V_S$ and
3. a set A of dependency arcs $(w_i, r, w_j) \in V_S \times R \times V_S$.

As the transition-based parser proceeds, the state contains a partial analysis of the sentence, where A contains arcs of the partially completed dependency tree, the buffer β contains tokens remaining to be processed and the stack σ contains partially processed tokens.

The transitions of one shift-reduce parser [Nivre, 2003] are:

Shift removes the first word w_i in the buffer and pushes it on top of the stack.

Left-Arc(r) for any dependency label r. This transition has the effect of adding a dependency arc with label r from the first token on the buffer w_i to the token on the top of the stack w_j, and removing the top token from the stack (*popping* the stack).

Right-Arc(r) again for any dependency label r. This transition works as Left-Arc(r), except that in this case the introduced dependency arc goes the other way, from the token on the top of the stack to the first token on the buffer. In addition, the first token on the buffer is replaced by the token from the top of the stack.

The transitions have certain inevitable preconditions, e.g. SHIFT is only permissible when the buffer is non-empty, and both ARC transitions only only makes sense when both the stack and the buffer are non-empty.

With an *initial state* $c_0 = ([w_0], [w_1, w_2, ..., w_n], [])$ with the first (artificial root) token on the stack, the rest of the sentence in the buffer and no arcs, the shift-reduce parser proceeds through a sequence of states $C_{0,m} = (c_o, c_1, ..., c_m)$ until it reaches the *terminal state* where the buffer is empty. At this point the dependency tree produced by the shift-reduce parser is contained in A.

The resulting dependency arcs in A will be *acyclic*, *spanning* and *single-headed*, but for some transition sequences, A may fail to satisfy the *connectedness* criteria of well-formed dependency graphs. In Nivre's [2003] parser, possibly disconnected trees in A are simply connected to the root node at the completion of the transition sequence.

As with other transition-based approaches relying on a *stack* for partially processed tokens, the model outlined here is restricted to produce projective dependency trees [Nivre, 2008]. The constraints Γ inherent in this parsing model is therefore the set of constraints characterizing projective dependency trees $\mathcal{G}_S^P$. Non-projective parses can be achieved either by modication of the parsing model, or by steps of pre- and post-processing.

TODO: runtime

Other transition-based approaches include list-based models [Covington, 2001] and generalized LR-parsers [Watson and Briscoe, 2007, Sagae and Tsujii, 2007]. List-based models, in which a lists are used instead of stacks, allow operations on any of the partially processed tokens – rather than just the most recent, as is the case with a stack – enabling them to produce non-projective graphs.

For all transition-based models, the training phase essentially consists of learning a model of what good transitions are given the parser state, which – to reiterate – both involves features of the tokens currently being processed and the previous transitions made.

2.2.3 Graph-based models

Turning to another successful approach, the class of graph-based models, the learning task is concerned with what a good dependency graph is like, rather than which transitions are good.

Graph-based models define a scoring function over possible dependency graphs for a sentence, which is used at parse time in conjunction with a search algorithm to find for the highest-scoring graph. This score function should provide a measure score(G) of the likelihood that a certain dependency graph G is the correct analysis for the sentence S.

2.2.3.1 Arc-factored models

In the training phase, graph based models learn the parameters of the scoring function. Different scoring functions and search algorithms have been proposed, all sharing the common property that the scoring function is decomposed

into smaller functions that score local properties of the graph being evaluated, namely single attachments (for *first-order* or *arc-factored* models) or pairs of attachments (for *second-order* models). Typically, the score of a local property, be it a single arc or a pair, is calculated as a the dot product of a weight vector and the feature vector for the arc – or in other words, a weighted sum of the arc features.

The score for the whole graph is typically found by summing the local property scores. The particular way in which subgraph scores is combined to achieve the score of the whole graph may differ between different graph-based models, but they share the fundamental assumption that the score of a graph G factors through the scores of the subgraphs of G [Kübler et al., 2009]. In the case of arc-factored models, where each arc is scored individually by the function $\lambda(w_i, r, w_j)$, we can write[1]:

$$\text{score}(G) = \sum \lambda(w_i, r, w_j)$$

In formal terms, the parsing algorithm h in a graph-based model produces a dependency graph for a sentence S by finding the graph the maximizes the scoring function:

$$h(S, \Gamma, \lambda) = \arg \max_{G \in \mathcal{G}_S} \text{score}(G)$$

The graph-theoretic term *spanning tree* denotes a subgraph $G' = (V', A')$ of a graph $G = (V, A)$ which satisfies the following criteria:

1. G' is a directed tree.
2. G' spans all the nodes of G, that is $V = V'$.

With a scoring function for subgraphs in the graph, the *maximum spanning tree* (MST) for a graph G is simply the highest scoring spanning tree for G. Graph theory provides several algorithms for finding maximum spanning trees, which can be used if the parsing problem is cast as a graph search problem: Given the sentence S and dependency types R, let $G_S = (A_S, V_S)$ denote the graph where the nodes V_S are the sentence tokens $\{w_0, w_1, ..., w_n\}$ and there is an arc of all possible dependency types between all pairs of tokens, except for dependencies from the root token:

$$A_S = \{(w_i, r, w_j) | \text{ for all } w_i, w_j \in S \text{ and } r \in R \text{ and } j \neq 0\}$$

As G_S has multiple arcs – one for each dependency type $r \in R$ – between each pair of nodes, and in both directions, it is a *multi digraph*. However, in the case of arc-factored models, which score each arc individually, we can safely remove all but the highest scoring arc between any two nodes, thus reducing the multi digraph G_S to a digraph $G'_S = (V'_S, A'_S)$:

[1] Additionally, some models [Nakagawa, 2007] include global properties of the graph in the scoring function.

$$\begin{aligned} V'_S &= V_S \\ A'_S &= \{(w_i, w_j) | w_i, w_j \in V_S, j \neq 0\} \\ \lambda(w_i, w_j) &= \max_r \lambda(w_i, r, w_j) \end{aligned}$$

This reduction is risk-free with arc-factored models, for the maximum arc scores we reduced to must include those of the maximum spanning tree. Otherwise, it would be possible to get a higher scoring tree by substituting arcs in it from A'_S. In order to recover the arc labels, it is necassary to keep track of the label r that yielded the highest arc score $\lambda(w_i, r, w_j)$ for each pair of nodes.

The set of possible dependency trees for the sentence S is identical to the set of spanning trees for G_S, and with the scoring function score(G) we can get the best parse by finding the maximum spanning tree of G_S. A maximum spanning tree finding algorithm is Chu-Liu-Edmons, which proceeds in part with a greedy strategy by assigning the hightest scoring arcs to each node, and in part with a recursive procedure for untying possible loops produced in the greedy step. Given a sentence of length n, the Chu-Liu-Edmonds algorithm takes $O(n^3)$ steps to find the maximum spanning tree, but with some trickery [Tarjan, 1977] it can be made to run in $O(n^2)$ steps. The algorithm works on the digraph G'_S which must be constructed beforehand, yielding a total runtime of $O(|R|n^2 + n^2) = O(|R|n^2)$.

As mentioned on the preceding page, arcs scores are a weighted sum of arc features $f(w_i, r, w_j)$:

$$\lambda(w_i, r, w_j) = \mathbf{w} \cdot f(w_i, r, w_j)$$

For a whole dependency graph $G = (V, A)$ the score amounts to the sum of the arc scores:

$$\text{score}(G) = \sum_{(w_i, r, w_j) \in A} \lambda(w_i, r, w_j) = \sum_{(w_i, r, w_j) \in A} \mathbf{w} \cdot f(w_i, r, w_j)$$

And the dependency tree for a sentence S yielded by the graph-based model is the spanning tree $G \in \mathcal{G}_S$ yielding the highest score:

$$h(S, \Gamma, \lambda) = \arg\max_{G \in \mathcal{G}_S} \text{score}(G) = \arg\max_{G \in \mathcal{G}_S} \sum_{(w_i, r, w_j) \in A} \mathbf{w} \cdot f(w_i, r, w_j)$$

At training time, the graph-based model must learn the weight vector $\mathbf{w}$ from training data. In the simplest case, a perceptron [Rosenblatt, 1958] can be used to construct the weights. With the percepron, learning proceeds by iteratively looking at the sentences in the training data, and for each sentence:

1. Find the hightest-scoring tree $h(S, \Gamma, \lambda) = \arg\max_{G \in \mathcal{G}_S} \sum_{(w_i, r, w_j) \in A} \mathbf{w} \cdot f(w_i, r, w_j)$ using the current weight vector.
2. Compare this to the correct tree given in the training data.

	# of tokens (*1000)	# of sents (*1000)	tokens per sentence
Arabic	54	1.5	37.2
Bulgarian	190	12.8	14.8
Chinese	337	57	5.9
Czech	1249	72.7	17.2
Danish	94	5.2	18.2
Dutch	195	13.3	14.6
German	700	39.2	17.8
Japanese	151	17	8.9
Portuguese	207	9.1	22.8
Slovene	29	1.5	18.7
Spanish	89	3.3	27
Swedish	191	11	17.3
Turkish	58	5	11.5

Table 2.1: Treebanks at the CoNLL shared task [Hall and Nilsson, 2006]

3. Adjust the weight vector by increasing the weights for the features present in the correct tree, while decreasing weights for other features.

The perceptron is an inference-based algorithm and achieves perfect classification when the data is linearly seperable [Witten and Frank, 2005]. Many other learning techniques apply, often involving computations that outweigh the maximum spanning tree algorithm itself in terms of complexity.

2.3 State of the Art

In this decade, data-driven dependency parsing has received a large amount of interest from researchers in human language technology. For two successive years at the Conference on Computational Language Learning [Buchholz and Marsi, 2006, Nivre et al., 2007], teams of researchers gathered in a friendly competition in multi-lingual data-driven dependency parsing. The parsing systems submitted includes several transtition- and graph-based models, including the graph-based MSTParser [McDonald et al., 2006] and the transition-based MaltParser [Nivre, 2003], jointly ranking as the best performing systems.

Corpora with dependency analyses (dependency *treebanks*) for thirteen languages were compiled for the conference, either from existing dependency treebanks, or by converting treebanks with constituency structure into to dependency format through a process of recursively defining a head for each constituent [Buchholz and Marsi, 2006]. Treebanks used differ not only in language, but also type of text (news reports, academic texts, dialogue transcripts, etc.), annotational conventions and richness as well as in size. All were converted into the same flat file format format (referred to as *CoNLL format*), allowing comparison of different systems across the whole spectrum of corpora.

	LEMMA	# CPOSTAG	# POSTAG	Labeled root	Add. morph. info	# DEPRELS
Arabic	Yes	14	19	Yes	Yes	27
Bulgarian	No	11	53	No	Yes	19
Chinese	No	22	303	No	No	134
Czech	Yes	12	63	Yes	Yes	84
Danish	No	10	24	No	Yes	53
Dutch	Yes	13	302	No	Yes	26
German	No	52	52	No	No	46
Japanese	No	20	77	No	Yes	8
Portuguese	Yes	15	21	Yes	Yes	55
Slovene	Yes	11	28	Yes	Yes	26
Spanish	Yes	15	38	No	Yes	21
Swedish	No	37	37	No	No	63
Turkish	Yes	14	30	No	Yes	26

Table 2.2: Treebank details [Hall and Nilsson, 2006]

2.3.1 CoNLL Format

The CoNLL format has one token per line, with a blank line separating sentences. Lines are divided into a number of fields [Buchholz et al., 2006]:

ID Token counter, starting at 1 for each new sentence.

FORM Word form or punctuation symbol.

LEMMA Lemma or stem (depending on particular data set) of word form, or an underscore if not available.

CPOSTAG Coarse-grained part-of-speech tag, where tagset depends on the language.

POSTAG Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.

FEATS Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.

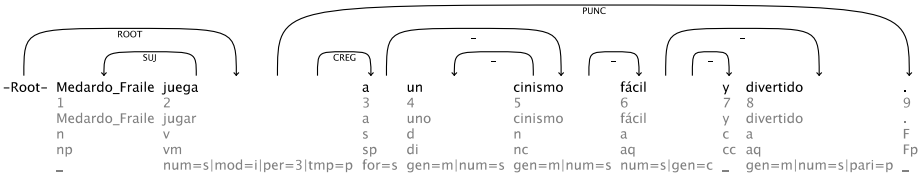
HEAD Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.

DEPREL Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.

PHEAD Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID

1	Medardo_Fraile	Medardo_Fraile	n	np	_	2	SUJ	2	SUJ			
2	juega	jugar	v	vm	num=s|mod=i|per=3|tmp=p	0	ROOT	0	ROOT			
3	a	a	s	sp	for=s	2	CREG	2	CREG			
4	un	uno	d	di	gen=m|num=s	5	_	5	_			
5	cinismo	cinismo	n	nc	gen=m|num=s	3	_	3	_			
6	fácil	fácil	a	aq	num=s|gen=c	5	_	5	_			
7	y	y	c	cc	_	6	_	6	_			
8	divertido	divertido	a	aq	gen=m|num=s|pari=p	6	_	6	_			
9	.	.	F	Fp	_	2	PUNC	2	PUNC			

(a) Annotated sentence as represented in the corpus file



(b) Visualization of the implicit dependency structure

Figure 2.7: Example sentence from the Spanish corpus Cast3LB [Civit and Martí, 2004, Civit et al., 2006]

of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).

PDEPREL Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.

DATA SPLITS?

2.3.2 Evaluation Metrics

Systems were evaluated by the *labeled accuracy score* (LAS) of the predicted parse trees, i.e. the percentage of dependencies where both the dependendy and its type is consistent with those of the gold standard parse trees. Additionally, *unlabeled attachment score* (UAS), in which the dependency labels are disregarded, was reported for each system, as well as *label accuracy* (LA), where only the dependency label is considered[2].

	Arabic	**Bulgarian**	**Chinese**	**Czech**	**Danish**	**Dutch**	**German**
Canisius	57.64	78.74	78.37	60.92	77.90	74.59	77.56
Attardi	53.81	72.89	54.89	59.76	66.35	58.24	69.77
Wu	63.81	79.73	74.81	59.36	78.38	68.45	76.52
Carreras	60.94	83.30	83.68	68.82	79.74	67.25	82.41
Yuret	52.42	73.49	72.72	51.86	71.56	62.75	63.82
Bick	55.37	79.21	76.18	63.02	74.61	69.51	74.74
Nivre	66.71	87.41	86.92	78.42	84.77	78.59	85.82
Schiehlen	44.39	0.00	66.20	53.34	76.05	72.11	68.73
Ma	50.74	67.64	75.29	58.52	77.70	59.36	68.11
Dreyer	53.37	74.81	71.63	60.54	66.61	61.56	70.97
O'Neil	66.71	85.24	86.70	76.60	82.83	77.51	85.36
Do	60.92	0.00	85.05	72.88	80.60	72.91	84.17
Johansson	64.29	0.00	72.49	71.46	81.54	72.67	80.43
McDonald	66.91	87.57	85.90	80.18	84.79	79.19	87.34
Riedel	66.65	0.00	89.96	67.44	83.63	78.59	86.24
Sagae	62.71	0.00	84.73	75.24	81.56	76.61	84.92
Shimizu	62.83	0.00	0.00	0.00	75.81	0.00	0.00
Corston-Oliver	63.53	83.36	79.92	74.48	81.74	71.43	83.47
Cheng	65.19	86.34	84.27	76.24	81.72	71.77	84.11
Average	*59.94*	*79.98*	*78.32*	*67.17*	*78.31*	*70.73*	*78.58*
Std. dev.	*6.53*	*6.30*	*8.82*	*8.93*	*5.45*	*6.66*	*7.51*

	Japanese	**Portuguese**	**Slovene**	**Spanish**	**Swedish**	**Turkish**	*Average*
Canisius	87.41	77.42	59.19	68.32	79.15	51.07	*70.80*
Attardi	65.38	75.36	57.19	67.44	68.77	37.80	*61.23*
Wu	90.11	81.47	67.83	72.99	71.72	55.09	*71.71*
Carreras	88.13	83.37	68.43	77.16	78.65	58.06	*74.72*
Yuret	84.35	70.35	55.06	69.63	65.23	60.31	*65.01*
Bick	84.75	78.18	64.31	71.37	74.09	53.87	*70.00*
Nivre	91.65	87.60	70.30	81.29	84.58	65.68	*80.19*
Schiehlen	83.35	71.01	50.72	46.96	71.10	49.81	*62.81*
Ma	70.84	71.13	57.21	65.08	63.83	41.72	*63.29*
Dreyer	82.87	75.28	58.73	67.62	67.58	46.05	*65.23*
O'Neil	90.57	84.69	71.08	79.82	81.78	57.52	*78.43*
Do	89.07	83.99	69.52	79.72	82.31	60.51	*76.80*
Johansson	85.63	84.57	66.43	78.16	78.13	63.39	*74.93*
McDonald	90.71	86.82	73.44	82.25	82.55	63.19	*80.27*
Riedel	90.51	84.43	71.20	77.38	80.66	58.61	*77.94*
Sagae	90.37	86.01	69.06	77.68	82.00	63.21	*77.84*
Shimizu	0.00	0.00	64.57	73.17	79.49	54.23	*34.18*
Corston-Oliver	89.95	84.59	72.42	80.36	79.69	61.74	*76.94*
Cheng	89.91	85.07	71.42	80.46	81.08	61.22	*77.70*
Average	*85.86*	*80.63*	*65.16*	*73.52*	*76.44*	*55.95*	
Std. dev.	*7.09*	*5.83*	*6.78*	*8.41*	*6.46*	*7.71*	

Table 2.3: Scores (LAS) achieved by each system at CoNLL-X [Buchholz et al., 2006].

	Arabic	**Bulgarian**	**Chinese**	**Czech**	**Danish**	**Dutch**	**German**
McDonald	66.91	87.57	85.90	80.18	84.79	79.19	87.34
Nivre	66.71	87.41	86.92	78.42	84.77	78.59	85.82
	Japanese	**Portuguese**	**Slovene**	**Spanish**	**Swedish**	**Turkish**	*Average*
	90.71	86.82	73.44	82.25	82.55	63.19	*80.83*
	91.65	87.60	70.30	81.29	84.58	65.68	*80.75*

Table 2.4: Scores for the top two systems: The graph-based MSTParser [McDonald et al., 2006] and the transition-based MaltParser [Nivre, 2003].

2.3.3 CoNLL Results

Scores achieved by the participating systems are shown in table 2.3 on the previous page. It is worth noting that while there is some variation in scores in between the different systems, there is no clear winner. Indeed, the top two systems – the graph-based MSTParser by McDonald et al. [2006] and the transition-based MaltParser by Nivre [2003] – are more or less on par, as shown in table 2.4.

These two top scoring systems represent both of the general approaches to data-driven parsing outlined in section 2.2. While it is tempting to conclude that there does not seem to be much empirical difference between them, McDonald and Nivre [2007] find several substantial differences in a detailed error analysis, with respect to the types of error made by each system:

> [...] MaltParser tends to perform better on shorter sentences, which require the greedy inference algorithm to make less parsing decisions. [...]
>
> MSTParser is far more precise for longer dependency arcs, whereas MaltParser does better for shorter dependency arcs. This behaviour can be explained using the same reasoning as above: shorter arcs are created before longer arcs in the greedy parsing procedure of MaltParser and are less prone to error propagation. Theoretically, MSTParser should not perform better or worse for edges of any length, which appears to be the case. [...]
>
> MSTParser's precision degrades as the distance to the root increases whereas MaltParser's precision increases. [...] Dependency arcs further away from the root are usually constructed early in the parsing algorithm of MaltParser. Again a reduced likelihood of error propagation coupled with a rich feature representation benefits that parser substantially. Furthermore, MaltParser tends to overpredict root modifiers, because all words that the parser fails to attach as modifiers are automatically connected to the root [...]

[2]Note that these are all measures of *precision* only. However, as the number of dependencies predicted for a sentence is fixed by the number of tokens, *recall* becomes redundant, at least scoring across all dependency types together. When evaluating dependency arcs with different labels (or other property), it makes sense to report recall as well.

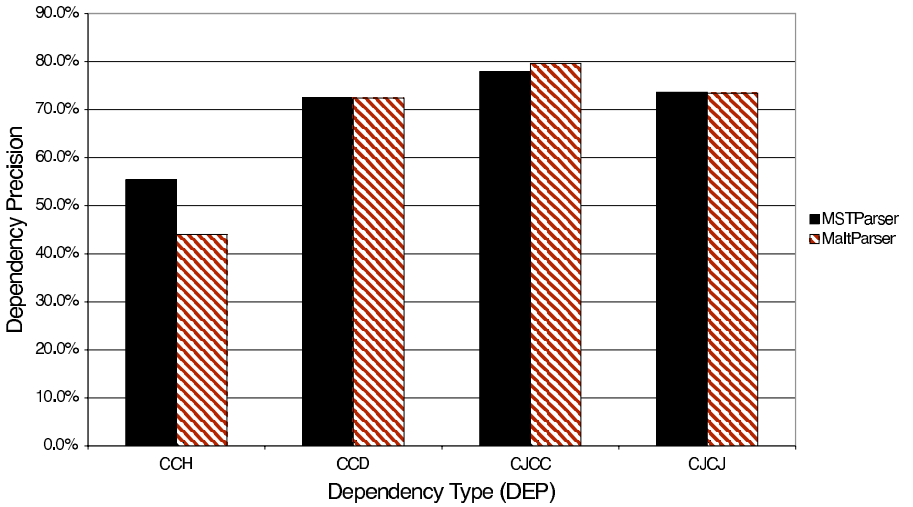


Figure 2.8: Precision across different annotational conventions for coordination [McDonald and Nivre, 2007].

The authors also find differences in accuracy in relation to linguistic categories of tokens (part of speech) and dependency types, but explain most in terms of the inherent strengths and weaknesses of the two models pointed out above.

They do find a marked difference in how well a specific annotational convention for coordinations is handled, namely *coordinating conjunction as head* (CCH), which MSTParser predicts nearly 20 percentage points more accurately (figure 2.8). As briefly described in section 2.2, this is the convention where the conjunction itself it head of the coordination, thus somewhat occluding the function of the coordinate structure in the larger syntactic structure. The CCH convention is contrasted to CCD, CJCJ and CJCC, in which one of the conjuncts act as head of the coordination.

Interestingly, the CCH convention seems to have a substantially lower precision *regardless* of parsing model. If such a difference is not entirely attributable to "prior" difference in average treebank precision, it would support a central hypothesis of this project: that explicitness in treebank annotation can cause a significant difference in accuracy of data-driven parsers.

Chapter 3

Feature Engineering in Dependency Parsing

3.1 Research Question

Data-driven dependency parsers presented at CoNLL generally perform well, but with an average precision in labeled attachment across languages for the top systems of about 81%, there is plenty of room for improvement. Each parsing model has its strengths and weaknesses, and the averaged score is of course also an average of the weaknesses. What if we focus on the strengths of each model? To get a rough[1] upper bound on how much improvement can be achieved by the strengths all models in unison, McDonald and Nivre [2007] simulated an *oracle parser* that chose the most accurate of all parses produced by the CoNLL systems. Labeled attachment score with the oracle parser rose to 86.9%.

In this project, instead of trying to improve parsing models themselves, I will turn attention to the input data for the data-driven systems: treebanks. Is it possilbe to augment treebanks with additional or modified features such that existing data-driven parsing systems generate better dependency parsers? In particular, can some linguistically motivated features reduce errors of tricky constructions such as parentheticals?

3.2 Review of literature and research

Feature engineering has already received a fair amount of interest. Partly from parser creators, as they must choose which features of which tokens – and in which combinations – to let their parser consider, and partly from people re-

[1] The oracle parser suffers from error propagation in the underlying parse trees, so it is not a fair upper bound for hybrid systems employing an ensemble of systems in its parsing decisions.

searchers wanting to improve performance of existing systems. I will not attempt a systematic and exhaustive overview – instead, I'll treat the field a bit like a zoo and merely point out some of the interesting animals.

3.2.1 Phrase identifying features

In the original paper for the MSTParser, McDonald et al. [2006] note that the accuracy of verb and conjunction attachment for multiclause sentences in Spanish is well below the average:

> Although overall unlabeled accuracy is 86%, most verbs and some conjunctions attach to their head words with much lower accuracy: 69% for main verbs, 75% for the verb *ser*, and 65% for coordinating conjunctions. [...] These weaknesses are not surprising, since these decisions encode the more global aspects of sentence structure: arrangement of clauses and adverbial dependents in multi-clause sentences, and prepositional phrase attachment.

They describe prelimenary experiments with engineering a feature to improve the accuracy:

> [...] we added features to count the number of commas and conjunctions between a dependent verb and its candidate head. Unlabeled accuracy for all verbs increases from 71% to 73% and for all conjunctions from 71% to 74%. Unfortunately, accuracy for other word types decreases somewhat, resulting in no significant net accuracy change. Nevertheless, this very preliminary experiment suggests that wider-range features may be useful in improv ing the recognition of overall sentence structure.

In a chunking-like preprocessing step, Zhou [2000] identifies "blocks" in unrestricted Chinese text and reports successful usage in dependency parsing, although no direct comparison to a baseline system without block identification is reported.

Attardi and Dell'Orletta [2008], working with the English treebank, report of benefits from chunking text and determining the head of each chunk (not unlike shallow parsing):

> Knowledge about chunks can be helpful to a Shift/Reduce dependency parser, since it typically operates with a limited lookahead and hence has only a narrower vision of the phrase being analyzed than for instance a Maxi- mum Spanning Tree parser [...].

However, the chunking is done on the basis of gold standard (manual) annotation. When they use a statistical chunker instead, the effect is reduced, likely due to propagation of errors in the statistical chunking. Thinking that the parser might as well perform the chunking task, they then resort to only chunking *simple noun phrases*, which can be done deterministically from patterns of POS tags:

> Chunking exploits POS tags produced by previous process- ing steps and hence using a chunker leads to a more com- plex layered parser architecture. But since the parser it- self may have access to the same information that the chun- ker uses to infer the chunks, one may wonder whether the parser might itself subsume the task of the chunker. We will show that indeed the addition of simple chunker-like features allows the parser to achieve an accuracy close to that from using gold chunk data.

Using the Thai part of a parallel Thai-Japanese corpus, Tongchim et al. [2008] also examine the use of a *base noun phrase* chunker as a preprocessing step to dependency parsing. They report of relatively low chunking accuracy (due to ambiguities of noun phrases in Thai), but nonetheless find that the chunk information benefits the parser, yielding a gain in LAS of 3.34 percentage points.

Hamabe et al. [2006] report of trouble parsing spontaneous Japanese speech:

> [...] automatically detected dependencies often cross clause boundaries erroneously because sentences including quotations or inserted clauses can have complicated clause structures.

They then employ a support vector machine (SVM) based chunker that considers information on morphemes, pauses and fillers in the transcribed spontaneous speech to identify QUOTATIONs and INSERTED CLAUSEs, and find that the clause boundaries improve accurary of dependency structure analysis.

Ciaramita and Attardi [2007] look at named, nominal and numerical entities, often occurring as segments of multiple words, and note:

> When we consider segments composed of several words there is exactly one dependency connecting a token outside the segment with a token inside the segment; e.g., "CBS Inc." is connected outside only through the token "Inc.", the subject of the main verb. With respect to the rest of the tree, segments tend to form units, with their own internal structure. Intuitively, this information seems relevant for parsing. This locally-structured patterns could help particularly simple algorithms like ours, which have limited knowledge of the global structure being built.

As may be guessed from this, they are working with a shift-reduce model. Introducing an EOS feature on segment tokens, indicating the distance to the end of segments, they achieve an improvement in LAS of 0.88 on the Wall Street Journal part of the English CoNLL treebank. They also find that a semantic feature indicating the broad ontological category of a token (*semantic super tagging*) increase accuracy, and that both features in combination gain even more when adopting *second order feature maps*, which explicitly represent pairs of all features, essentially allowing the model to learn correlations of feature co-occurrences.

3.2.2 Semantic features

Øvrelid and Nivre [2007] note that parsing of Swedish poses special problems for the identification of subjects and objects due to limited case marking and ambiguous word order patterns. Referring to the survey work of de Swart et al. [2008], Øvrelid [2008b] remarks that the referential property *animacy* of nominal elements has been shown to play a role in argument disambiguation. Introducing a feature of ANIMATENESS in the Swedish treebank, she achieves modest but significant ($p < 0.01$) improvement in overall accuracy, and notes:

> We find that animacy influences the disambiguation of subjects from objects, objects from indirect objects as well as the general distinction of arguments from non-arguments.

Øvrelid [2008a, 2008b] also finds that DEFINITENESS help disambiguate subjects and subject predicates, and that verbal features (FINITENESS and VOICE) improve accuracy of dependency relations involving verbal elements.

Bharati et al. [2008] augment an Indian treebank with features to distinguish HUMAN from NONHUMAN and ANIMATE from INANIMATE nominals (yielding three classes, as the combination INANIMATE and HUMAN is impossible). They are motivated by the fact that certain Indian verbs only take human subjects, and find that the feature helps capture argument structure in dependency parsing.

Working with the Czech treebank, Novák and Žabokrtský [2007] introduce a variety of features , indicating amount other things if tokens are CAPITALIZED and COORDINABLE, as well as some technical suffices given by the lemmatizer in use. With all features in combination, they achieve an improvement in LAS of 0.45. Due to the relatively large number of feature introduced, the feature space dimension in the model grows a lot, impairing the runtime of the system. To counter this, they conduct some experiments in reducing dimensionaly, resulting in a compromise with a much smaller feature space and fair improvement in accuracy.

TODO: Soft and hard constraints on dependency length [Eisner and Smith, 2005]?

3.3 Experiment Design

In the following I will describe how I augment treebanks with new features, and how I evaluate the effect of adding a feature. I shall refer to parsers trained on original, unmodified treebanks as *baseline*, and use the term *system* for a parser trained on an augmented treebank.

TODO: choice of parser?

3.3.1 Experiment setup

As a broad outline, an experiment proceeds in the following steps:

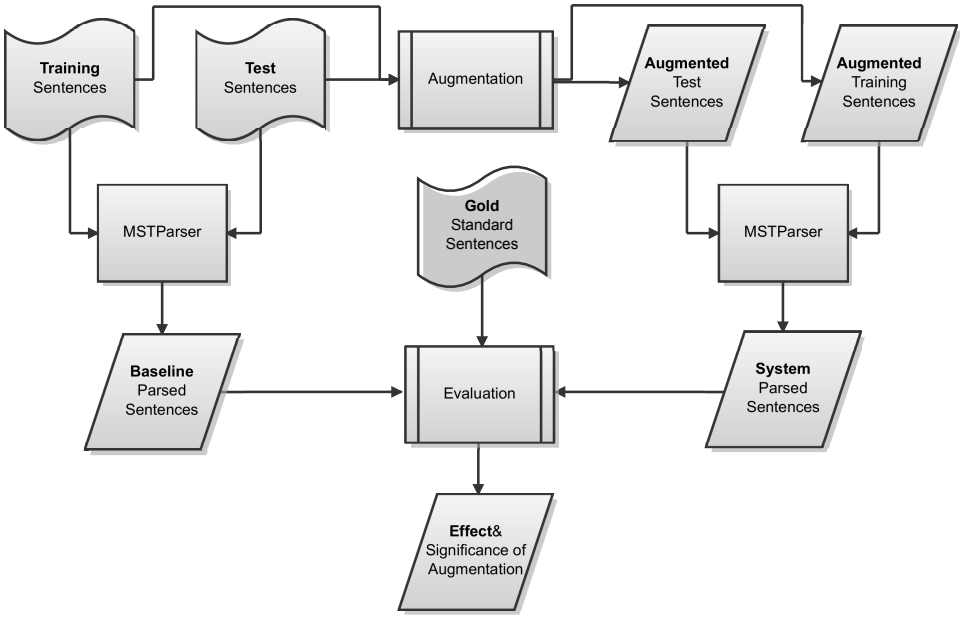


Figure 3.1: Overview of experiment setup.

1. Obtain *baseline* parses:
   (a) Train a parser on the training part of the original treebank.
   (b) Parse the test part of the original treebank using the trained parser.
2. Introduce feature:
   (a) Augment a copy of the original training part by adding the desired feature.
   (b) Do the same for the original test part.
3. Obtain *system* parses:

(a) Train a parser on the training part of the augmented treebank.

(b) Parse the test part of the augmented treebank using the newly trained parser.

4. Evalutate effect and significance:

   (a) Obtain accuracy scores for both baseline and system parses.

   (b) Determine possible effect of the augmentation, and estimate statistical significance of the observed effect[2].

3.3.2 Treebank augmentation

In augmenting a treebank, new features may be added to sentence tokens, or existing properties (such as POS tags) modified. While some features can be added on a per-token basis, e.g. *animacy* classification, addition of other features require an overview of the whole sentence, as would be the case for determining open-ended quotations. The procedure for augmenting a treebank allows for both token-level and sentence-level processing, and proceeds with these steps:

1. Initialization (e.g. of a classifier to be used).
2. Read treebank.
3. For each sentence:

   (a) Allow pre-processing of sentence.

   (b) For each token: Allow processing of token.

   (c) Allow post-processing of sentence.

4. Output augmented treebank.

3.3.3 Parser evaluation

Parsed sentences are obtained from both the baseline and the system parser (with the latter working on the augmented treebank), and predictive performance in terms of parsing accuracy of each is evaluated.

To this end, the standard CoNLL evaluation tool[3] `eval.pl` is used for each parser in turn to compare its predicted parses to the gold standard (correct) parses given in the treebank. The obtained labeled accuracy score (LAS) for the system is compared to the baseline score, providing a measure of the overall effect on parsing accuracy of adding the feature in question.

[2] As outlined here, the parser is trained on one treebank partition and predictive performance is estimated on the other partition. In other words, only a single round of cross-validation is performed. In general, and especially for small treebanks, it is considered good practice to perform several rounds of cross-validation using different partitions of the data set (K-fold cross-validation, with 10 folds commonly used). However, as the treebanks I use come pre-partitioned, and are evalutated with these splits in other research, I chose to report performance with the existing train/test splits for reasons of comparability.

[3] Available from the CoNLL Shared Task website Buchholz et al. [2006].

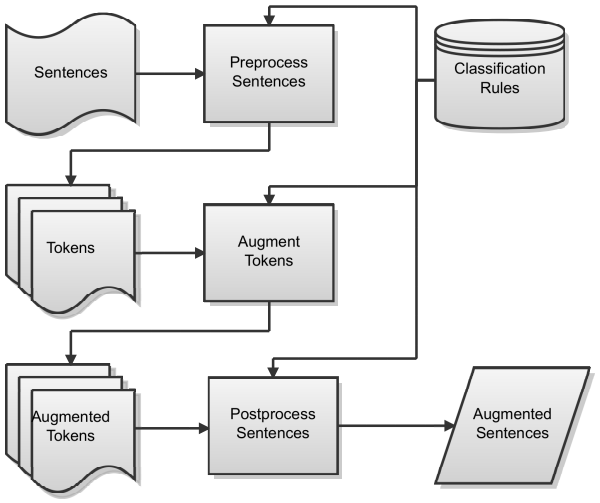


Figure 3.2: Augmentation process for a treebank.

3.3.3.1 Focused evaluation

Some features only directly affects part of a treebank. A feature related to quotations for example is only added to tokens involved in quoted phrases. This begs the question: What effect does the added feature have on the affected tokens? And also interesting: What effect does it have on the remaining, unaffected tokens in the treebank?

With only a part of the tokens augmented, one option is to count dependency errors for the affected tokens separately. If for example one were to add a feature clearly marking tokens that occur between parentheses, this would give a ratio of correct dependencies for parenthesized tokens, as well as a ratio for the remainder. However, this approach fails to clearly capture a potentially interesting class of dependencies, namely those erroneoulsy *crossing* the boundary of e.g. a parenthesized clause (as the crossing dependency might originate from a token outside the clause).

It is not impossible conceive a heuristic for counting these problematic dependencies specifically, but it would be of little use when the objective is to compare a baseline parse to another by the same standard. The reason for this is simply that dependencies predicted for the baseline and the system may differ, so one would likely be counting different things in the baseline and the system, yielding incommensurable ratios.

Given the apparent difficulties of token- and dependency-level categorization, I have chosen to categorize on the *sentence-level* instead when estimating the effects of a feature on targeted versus untargeted parts of the text. I thus categorize *sentences* by whether the added feature applied to *any* token in the sentence, and evaluate each part separately, yielding two ratios, namely one for *affected sentences* in which augmented tokens occur, and one for *unaffected*

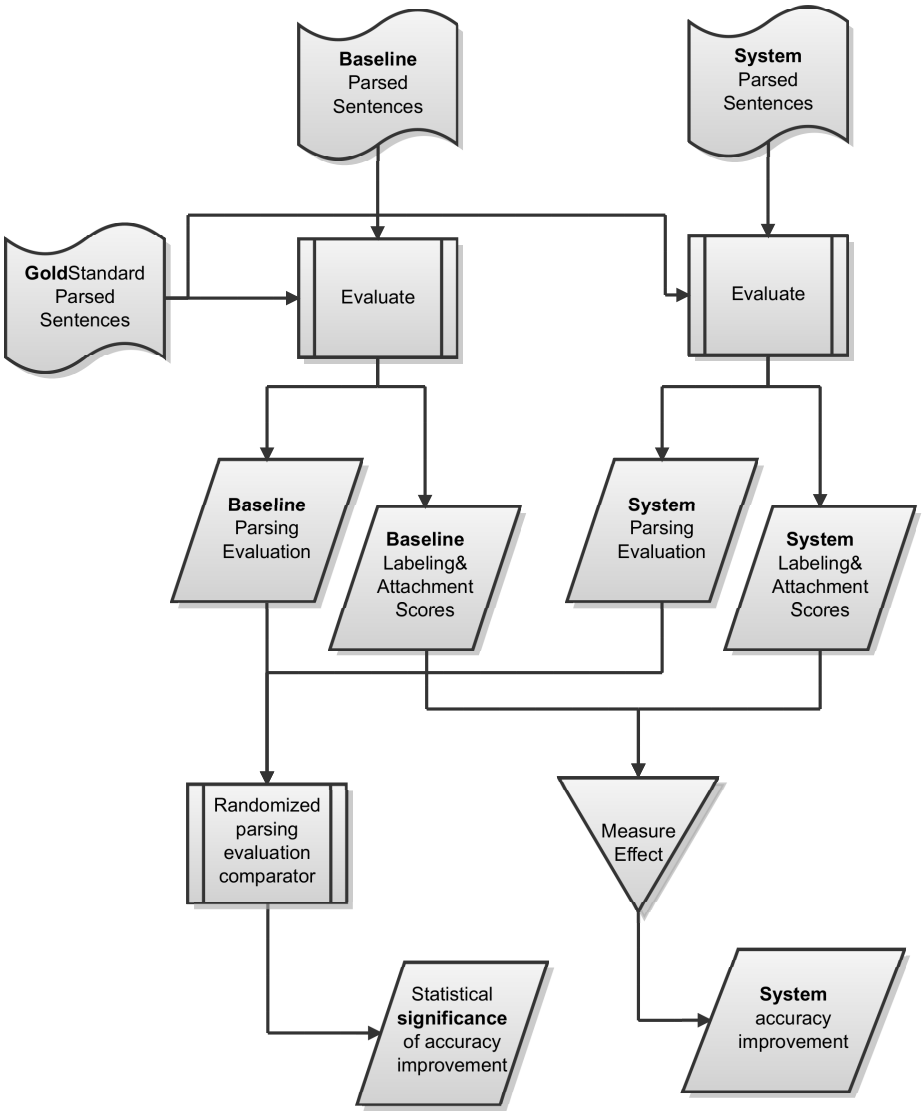


Figure 3.3: Evaluation and comparison of parser accuracy. The system parse is first scored and compared to the baseline. Then effect and statistical significance is measured.

sentences which were left untouched by the augmentation.

It is the hope that these ratios can cast light on the question of how valuable a feature was in parsing the linguictics phenomena it was intended to cover, and which effect the alteration in the parsing model had on the ability to parse the rest of the sentences. Did it enable the parser to clearly distinguish tokens that require different treatment and thus help unclutter the parsing model, or did it have an adverse effect by perhaps preventing a useful generalization in the parsing model?

3.3.3.2 Punctuation

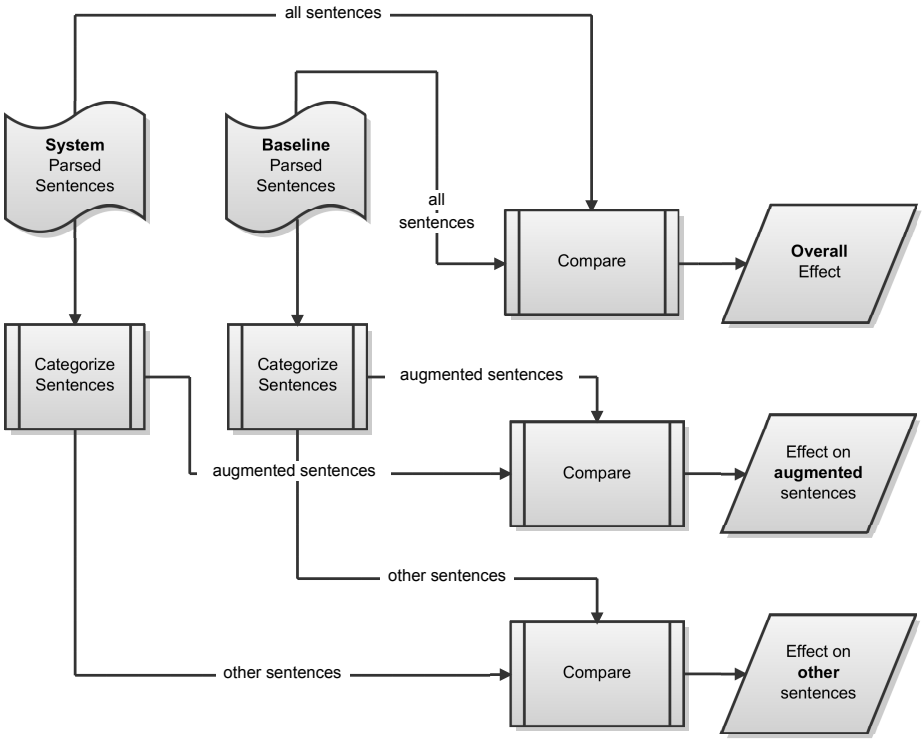
Contrary to the convention of the CoNLL shared task [Buchholz and Marsi, 2006], I am including tokens considered punctuation when evaluating parser accuracy. As several of the phenomena in my experiments involve constructions marked by punctuation (e.g. quotation marks and parentheses), I expect the inclusion of the punctuation tokens themselves to contribute a valuable indication of a parsers ability to correctly predict dependency structure.

As long as I let the trained parsers work within the same treebanks, annotational conventions for punctuation should be consistent and possible to predict. If one were to conduct experiments in domain adaptation using different treebanks with inconsistent conventions, it would make sense to exclude punctuation.

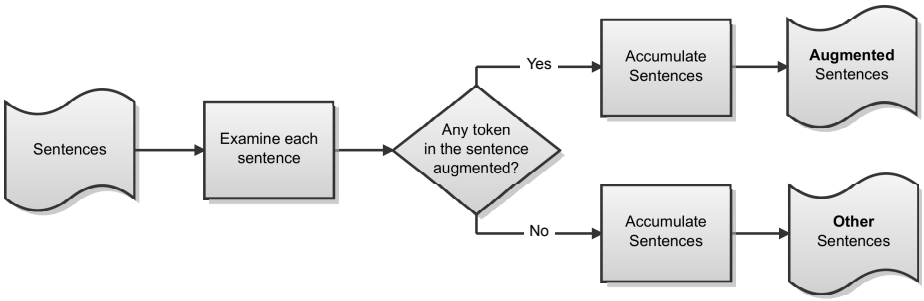
3.3.4 Statistical significance

To quantify the likelihod of observed effects occurring merely by chance, Dan Bikel's "Randomized Parsing Evaluation Comparator" [Bikel, 2004] is used to compute p-values. In the words of the author, the statistical significance test employed

> [...] is a type of "stratified shuffling" (which in turn is a type of "compute-intensive randomized test"). In this testing method, the null hypothesis is that the two models that produced the observed results are the same, such that for each test instance (sentence that was parsed), the two observed scores are equally likely. This null hypothesis is tested by randomly shuffling individual sentences' scores between the two models and then re-computing the evaluation metrics (precision and recall, in this case). If the difference in a particular metric after a shuffling is equal to or greater than the original observed difference in that metric, then a counter for that metric is incremented. Ideally, one would perform all 2^n shuffles, where n is the number of test cases (sentences), but given that this is often prohibitively expensive, the default number of iterations is 10,000. After all iterations, the likelihood of incorrectly rejecting the null is simply $\frac{nc+1}{nt+1}$, where nc is the number of random differences greater than the original observed difference, and nt is the total number of iterations.



(a) Three separate evaluations occur: One for the overall effect of adding the feature, one for the effect on affected sentences, and one for the effect on sentences unaffected by the augmentation.



(b) Procedure for categorizing sentences for focused evaluation.

Figure 3.4: Procedures for focused evaluation of the effect of an augmentation.

Bikel's tool estimates two p-values for the observed differences: One for precision and one for recall. The p-values reported in this project relate to the observed difference in *precision*.

The author also remarks:

> This type of testing method assumes independence between test instances (sentences). This is not a bad assumption for parsing results, but is not correct, either.

3.3.5 Treebanks

Several of the treebanks used at CoNLL are subject to restrictive licensing terms [Buchholz and Marsi, 2006]. The treebanks I was able to obtain for this project are listed below. Most[4] are identical to those used at CoNLL, and some are converted phrase structure treebanks made specifically for CoNLL.

Arabic Prague Arabic Dependency Treebank (PADT) [Hajič et al., 2004, Smrž et al., 2002]

Bulgarian Derivative of the Bulgarian HPSG Treebank [Simov et al., 2005, Simov and Osenova, 2003, Simov et al., 2002]

Czech The Prague Dependency Treebank (PDT) [Böhmová et al., 2001]

Danish The Danish Dependency Treebank (DDT) [Kromann, 2003]

Dutch The Alpino Dependency Treebank [van der Beek et al., 2002b,a]

English ???

German The TIGER Treebank [Brants et al., 2002]

Italian Turin University Treebank (TUT) [Bosco et al., 2008]

Japanese Verbmobil [Kawata and Bartels, 2000]

Portuguese [Afonso et al., 2002]

Slovene [Džeroski et al., 2006]

Spanish Cast3LB [Civit et al., 2006, Civit and Martí, 2004, Civit et al., 2003]

Swedish Talbanken05 [Nilsson et al., 2005, Einarsson, 1976, Teleman, 1974]

Turkish Metu Sabanci [Oflazer et al., 2003, Atalay et al., 2003]

[4] In particular, the Italian treebank was not used at CoNLL.

		Training set		Test set			
Corpus	Language	Sentences	Tokens	Sentences	Tokens	UAS	LAS
DDT	Danish	5190	94386	322	5852	88.12%	82.89%
Talbanken	Swedish	11042	191467	389	5656	87.27%	81.08%
BIO	English	7482	197724	1450	38120	91.71%	91.47%
Law	Italian	1001	25423	100	2607	91.45%	77.71%
Newspaper	Italian	1001	28233	100	2357	85.53%	75.14%
Law+Newspaper	Italian	2002	53656	200	4964	88.86%	76.59%
Cast3lb	Spanish	3306	89334	206	5694	82.19%	78.42%
Tiger	German	39216	699610	357	5694		
Tiger (6K)	German	6000	109035	357	5694	85.34%	81.03%
Bul	Bulgarian	12823	190217	398	5934		
Bul (6K)	Bulgarian	6000	80905	398	5934	89.91%	85.24%
SDT	Slovene	1534	28750	402	6390	78.56%	69.59%
PDT	Czech	72703	1249408	365	5853		
Metu	Turkish	4997	57510	623	7547	79.89%	69.05%
Alpino	Dutch	13349	195069	386	5585		
Alpino (6K)	Dutch	6000	65839	386	5585	76.97%	72.53%

Table 3.1: Treebank sizes and baseline parsing accuracies achieved.

Several of the treebanks have a size which exceed the capacity of the computer system I had available for experiments[5] (treebank sizes are shown in table 3.1). In particular, I was not able to train a baseline parser and obtain baseline scores for the German Tiger treebank, the Bulgarian treebank, the Czech PDT and the Dutch Alpino treebank. In order to be able to experiment with these languages regardless, I produced versions of the treebanks with a managable size by truncating the training part to 6000 sentences. When referring to these truncated treebanks, I will use a "(6K)" suffix.

3.4 Feature Designs

EXPAND: What I will do. A lot of it will have to do with punctuation. Prelimenary survey. Quantify suspicion.

3.4.1 Quotation

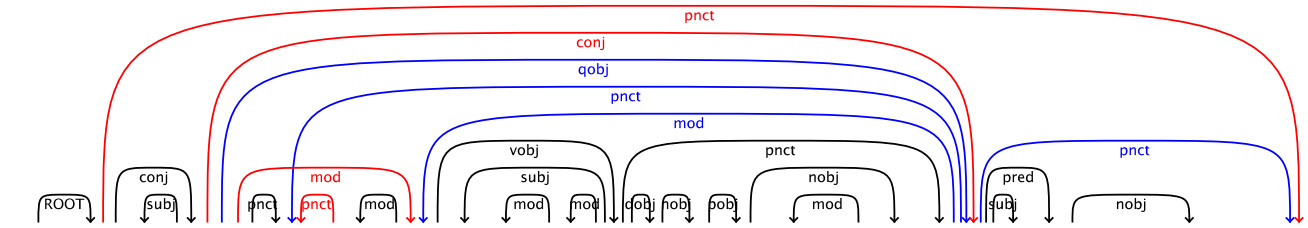
The prelimenary survey (3.2 on the next page) indicates that quotations occur in nearly a third of the Danish test. Although the average LAS for these sentences is not far behind the overall average for the treebank, manual inspection of baeline parses (3.5 on page 37) indicate that errors often involve dependencies erroneously crossing the boundary of a quoted phrase, as well as incorrect attachment of quotation marks themselves .

Intuitively, it seems reasonable that quotations require special treatment. Quoted phrases are often complete in themselves, often containing a subject and predicate, conveying a statement, question or exclamation, sometimes comprising a main clause and one or more subordinate clauses.

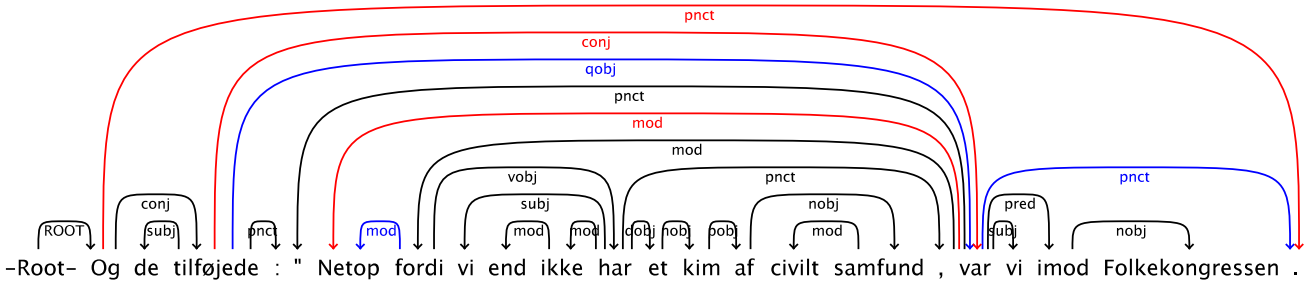
[5]The system used was a 64 bit Intel® architecture server with 8GB of system memory. 4GB heap space was allocated to the Java virtual machine.

	Presence				Difficulties								Indicator of potential			
	% of sentences with...				LAS for sentences with...				Δ LAS				Δ LAS x presence			
Corpus	“	( )	:	-	“	( )	:	-	“	( )	:	-	“	( )	:	-
DDT	*28.9%*	0.6%	*8.7%*	4.3%	82.62%	84.75%	76.84%	77.42%	*-0.27*	1.85	*-6.06*	*-5.48*	*5*	0	*2*	1
Talbanken	0.0%	4.6%	2.8%	*6.7%*		72.67%	56.47%	68.67%	0.00	*-8.41*	*-24.61*	*-12.41*	0	1	1	*2*
BIO	0.2%	*26.0%*	3.1%	0.1%	78.23%	88.77%	85.45%	92.05%	*-13.24*	*-2.70*	*-6.02*	0.58	0	*3*	0	0
Law	0.0%	*21.0%*	2.0%	0.0%		77.49%	87.76%		0.00	*-0.22*	10.04	0.00	0	*5*	0	0
Law2	0.0%	*21.0%*	2.0%	0.0%		86.33%	93.88%		0.00	1.21	8.76	0.00	0	*3*	0	0
Newspaper	*14.0%*	*7.0%*	*11.0%*	*9.0%*	70.50%	69.78%	81.59%	73.40%	*-4.64*	*-5.36*	6.45	*-1.74*	*4*	*2*	*2*	*2*
Law+News	*14.0%*	*28.0%*	*13.0%*	*9.0%*	71.25%	76.39%	82.40%	77.56%	*-6.46*	*-1.33*	4.69	*-0.15*	*4*	*7*	*2*	*2*
Cast3lb	*10.7%*	0.0%	*5.3%*	*19.9%*	77.63%		72.91%	79.73%	*-0.79*	0.00	*-5.51*	1.31	*2*	0	1	*4*
Tiger (6K)	*16.2%*	0.0%	*6.2%*	2.2%	78.78%		80.68%	82.91%	*-2.25*	0.00	*-0.35*	1.88	*3*	0	1	0
Bul (6K)	*9.5%*	*5.3%*	4.8%	*10.8%*	83.50%	84.09%	90.69%	83.96%	*-1.74*	*-1.15*	5.45	*-1.28*	2	1	0	2
SDT	*48.0%*	0.2%	2.5%	*8.7%*	62.52%	56.52%	50.69%	54.36%	*-7.07*	*-13.07*	*-18.90*	*-15.23*	*18*	0	1	*4*
PDT	*5.5%*	*6.8%*	4.4%	*11.5%*					0.00	0.00	0.00	0.00	*5*	*7*	*4*	*12*
Metu	*6.4%*	0.6%	*6.1%*	2.4%	59.84%	61.00%	65.32%	59.26%	*-9.21*	*-8.05*	*-3.73*	*-9.79*	*3*	0	*2*	1
Alpino (6K)	0.5%	*9.3%*	*8.3%*	0.0%	74.07%	72.80%	76.15%		1.54	0.26	3.62	0.00	0	*3*	2	0

Table 3.2: Prelimenary survey of the occurences of various punctuation and its potentially adverse effect on parsing accuracy. Green color indicates widespread usage. ΔLAS shows the difference in LAS for the sentences in focus compared to the baseline. An apparent impact on parsing precision is indicated by red color. Finally, a coarse indication of potentiality (for remedy) was calculated using the heuristic: impact × presence (rounded and scaled). In order to be suitable for experiments, a particular phenomenon should be both well represented in the treebank, and relatively poorly handled by the baseline parser.



(a) Baseline parse



(b) Parse when augmented with a feature marking tokens in the quoted phrase (the QuotedPhrase scheme)

Figure 3.5: Danish sentence with a quotation. Black arcs were correctly predicted, red arcs erroneously predicted, and blue arcs should have been predicted, but were not. These and subsequent joint visualizations of baseline/system and gold standard dependency structure were made using the excellent corpus visualization tool *What's Wrong with my NLP?* [Riedel and Vladimir, 2009].

From a linguistics viewpoint, quoted phrases exhibit special behavior in some languages. Apart from having an internal structure on their own right,Haberland [1986] notes that unquoted or *indirect reported speech* in Danish exhibits slighty different word order than quoted or *direct reported speech*. Contrast these (condensed) sentences:

(1) Direct reported speech:

> *Og de tilføjede: "Vi har end ikke et kim af civilt samfund".*
> [And they added: "We haven't even got a germ of civil society".]

(2) Indirect reported speech:

> *Og de tilføjede, at vi end ikke har et kim af civilt samfund.*
> [And they added that we haven't even got a germ of civil society.]

The complementizer *at* is dropped in the first sentence (direct report), and in the second sentence (indirect report), the negation *ikke* is moved to a pre-verbal position.

As quotation marks in the Danish treebank are categorized as the same part of speech as all other punctuation, they are only sinlged out at the word form level, even though quotation marks (and other symmetrical indicators of parentheticals) do not share much behavior with other forms of punctuation, such as point indicators (commas, colons, semicolons, dashes) [Nunberg, 1990]. Noting that quotation marks require special treatment, Søgaard (personal communication) has experimented with "amplifying" the uniqueness of quotation marks by introducing a separate part of speech tag for these, as well as for tokens in their vicinity, and saw an improvement of 0.50 in LAS ($p < 0.1$) for a truncated part of the Danish treebank. This was achieved with a 0-1-2 window when modifying the POS tags, meaning that the POS tag was changed for the quotation mark itself as well the two tokens following it.

A 0-1-2 window will inevitably mark some tokens immediately following a quoted phrase, as well as tokes occuring in the beginning of these, so this augmentation does not clearly indicate which tokens are actually quoted. Intuitively, this would be valuable information. As an attempt to clearly mark quoted phrases, I am using an augmentation scheme which adds a QUOTE feature not only to quotation marks themselves, but also to the quoted tokens. This augmentation is not entirely trivial, as quotation marks do not always occur in pairs, for several reasons:

1. When beginning a multi-sentence quote begins, only the initial (*opening*) quotation mark is seen. The *closing* quotation mark[6] occurs in a *following* sentence (see 3.6 on the next page for an example).

2. *Graphic absorption rules* of written accounts [Nunberg, 1990] mean that a closing quotation mark may be left out in certain cases.

[6] Contrary to the convention of certain other languages (e.g. French), there is not a strong tradition for graphemic distinction between opening and closing quotation marks in Danish.

Here is what [George Kennan] had to say, and it's revealing: "*[...] we have about 50% of the world's wealth but only 6.3% of its population.* ► [...] In this situation, we cannot fail to be the object of envy and resentment. ► Our real task in the coming period is to devise a pattern of relationships which will permit us to maintain this position of disparity [...] ► We need not deceive ourselves that we can afford today the luxury of altruism and world-benefaction. ► We should cease to talk about vague and [...] unreal objectives such as human rights, the raising of the living standards, and democratization. ► The day is not far off when we are going to have to deal in straight power concepts. ► The less we are then hampered by idealistic slogans, the better."

Figure 3.6: Multi-sentence quotation [Chomsky, 1984], which in the treebank would occur as separate sentences (indicated here by ►).

3. Typos.

Instead of constructing a complex system to make sense of the unruly realization of quotations, I focused on a few clear and unambigous cases, which cover a good part (46%) of the cases encountered in the test set:

- First part of multi-sentence quotations introduced with a colon preceeding a quotation mark, which unambigously begins a quotation. E.g. (with ► denoting the treebank sentence delimiter):

  After some big slave revolts in Jamaica, the British basically said: "*It's not paying anymore.* ► Let's reconsider."[7]

- Quotation marks which occur in pairs, anywhere in a sentence, e.g.:

  So what everyone was asking in Parliament in London was, "*How can we force them to keep working for us, even when they're no longer enslaved into it?*"

For the Danish treebank, the scheme for augmentation outlined above, which I shall call QuotedPhrase, yields a significant ($p < 0.002$) improvement in LAS of 0.75, equivalent to a 4.40% reduction in error rate. The improvement is greatest (1.21) for sentences matched by the scheme, yet substantial (0.64) and significant ($p < 0.02$) for other sentences, indicating that the augmentation helped unclutter the learned parsing model in general.

For comparison, I also tried another scheme, QuotationMark, which alters the POS tag of quotation marks, but leaves the quoted tokens untouched. This

[7] With the proposed augmentation scheme, a QUOTE feature is added to the tokens marked by *italics*, i.e. only the first of the quoted sentences, which in the treebank would occur as part of the outer (quoting) sentence.

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
DDT	QuotedPhrase	overall			88.12%	82.89%	88.88%	83.65%	0.75	0.75	*4.40%*	*0.0014*
		affected	43	13%	86.10%	81.17%	87.31%	82.38%	1.21	1.21	*6.43%*	*0.0123*
		unaffected	279	87%	88.62%	83.32%	89.26%	83.96%	0.64	0.64	*3.84%*	*0.0106*
DDT	QuotationMark	overall			88.12%	82.89%	88.55%	83.56%	0.43	0.67	*3.90%*	*0.0066*
		affected	93	29%	87.41%	82.62%	88.01%	83.32%	0.60	0.70	*4.03%*	*0.0681*
		unaffected	229	71%	88.54%	83.05%	88.86%	83.70%	0.32	0.65	*3.83%*	*0.0260*
DDT	QuotedPhrase + QuotationMark	overall			88.12%	82.89%	88.74%	83.56%	0.62	0.67	*3.90%*	*0.0125*
		affected	93	29%	87.41%	82.62%	88.01%	83.04%	0.60	0.42	*2.42%*	0.2230
		unaffected	229	71%	88.54%	83.05%	89.16%	83.86%	0.62	0.81	*4.78%*	*0.0133*
SDT	QuotedPhrase	overall			78.56%	69.59%	79.39%	70.49%	0.83	0.89	*2.93%*	*0.0070*
		affected	44	11%	75.63%	63.37%	78.13%	66.27%	2.50	2.90	*7.92%*	*0.0260*
		unaffected	358	89%	78.96%	70.43%	79.56%	71.05%	0.60	0.62	*2.10%*	*0.0474*
SDT	QuotationMark	overall			78.56%	69.59%	79.11%	70.11%	0.55	0.52	*1.70%*	*0.0909*
		affected	193	48%	72.41%	62.52%	73.28%	63.36%	0.87	0.84	*2.24%*	*0.1231*
		unaffected	209	52%	83.21%	74.94%	83.51%	75.21%	0.30	0.27	*1.08%*	0.2488
Newspaper	QuotedPhrase	overall			85.53%	75.14%	85.87%	75.31%	0.34	0.17	0.69%	0.3701
		affected	14	14%	83.00%	70.50%	84.00%	70.00%	1.00	*-0.50*	0.00%	0.3098
		unaffected	86	86%	86.05%	76.09%	86.25%	76.39%	0.20	0.30	*1.25%*	0.2732
Newspaper	QuotationMark	overall			85.53%	75.14%	86.00%	75.48%	0.47	0.34	*1.38%*	0.2359
		affected	14	14%	83.00%	70.50%	83.00%	71.25%	0.00	0.75	*2.54%*	0.2289
		unaffected	86	86%	86.05%	76.09%	86.61%	76.34%	0.56	0.25	*1.05%*	0.3193
Bul (6K)	QuotedPhrase	overall			89.91%	85.24%	89.96%	85.07%	0.05	*-0.17*	0.00%	0.3201
		affected	38	10%	88.62%	83.50%	88.49%	83.25%	*-0.13*	*-0.25*	0.00%	0.3982
		unaffected	360	90%	90.10%	85.50%	90.18%	85.35%	0.08	*-0.15*	0.00%	0.3447
Bul (6K)	QuotationMark	overall			89.91%	85.24%	90.12%	85.59%	0.22	0.35	*2.38%*	0.1629
		affected	38	10%	88.62%	83.50%	89.00%	84.40%	0.38	0.90	*5.45%*	0.1891
		unaffected	360	90%	90.10%	85.50%	90.30%	85.77%	0.20	0.27	*1.86%*	0.2463

Table 3.3: Results for the two schemes for quotation augmenting features, QUOTEDPHRASE (which adds a feature to quoted tokens) and QUOTATIONMARK (which alters the POS tag of the quotation mark only). For each augmentation scheme and language, the difference in labeled and unlabeled attachment score is shown for sentences overall, as well as for sentences affected by the augmentation scheme and for sentences to which the augmentation scheme did not apply. Finally, the effect is expressed in terms of *relative error reduction* of labeled attachment, and the statistical significance for the observed difference in LAS is given.

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
Talbanken	Parenthesis	overall			87.27%	81.08%	86.99%	81.15%	*-0.28*	0.07	0.37%	0.4202
		affected	18	5%	78.60%	72.67%	82.20%	75.42%	3.60	2.75	*10.06%*	*0.1335*
		unaffected	371	95%	88.06%	81.85%	87.42%	81.67%	*-0.64*	*-0.18*	0.00%	0.3614
Newspaper	Parenthesis	overall			85.53%	75.14%	85.87%	75.27%	0.34	0.13	0.51%	0.3873
		affected	7	7%	76.44%	69.78%	79.56%	72.00%	3.12	2.22	*7.35%*	0.2546
		unaffected	93	93%	86.49%	75.70%	86.54%	75.61%	0.05	*-0.09*	0.00%	0.4135
Law	Parenthesis	overall			91.45%	77.71%	91.98%	78.21%	0.54	0.50	*2.24%*	*0.1441*
		affected	21	21%	90.03%	77.49%	90.19%	77.49%	0.16	0.00	0.00%	0.4402
		unaffected	79	79%	91.89%	77.78%	92.54%	78.44%	0.65	0.66	*2.97%*	*0.1210*
Both	Parenthesis	overall			88.86%	76.59%	89.24%	76.89%	0.38	0.30	*1.29%*	0.1758
		affected	28	14%	86.66%	76.39%	86.89%	76.27%	0.23	*-0.12*	0.00%	0.4922
		unaffected	172	86%	89.31%	76.63%	89.73%	77.02%	0.42	0.39	*1.67%*	*0.1391*

Table 3.4: Results for the PARENTHESIS augmentation scheme, which marks parenthesized tokens with the feature PAREN. *Newspaper* and *Law* are the Italian treebanks, and *Both* are the two concatenated.

scheme yields a smaller but still significant ($p < 0.007$) overall improvement of 0.67 on the Danish corpus. QUOTATIONMARK is closer to that of Søgaard, although not identical[8].

A combination of both schemes (QUOTEDPHRASE+QUOTATIONMARK) yields smaller improvement than that of QUOTEDPHRASE alone. This could mean that there is some overlap in the coverage of the two schemes, or it could mean that the model starts overfitting.

Significant improvements are also seen for the Slovene treebank, in which quotations are also fairly common. Less convincing or insignificant effects are seen in the Italian newspaper corpus and the bulgarian corpus. In the latter, QUOTEDPHRASE actually has an *adverse* effect on precision.

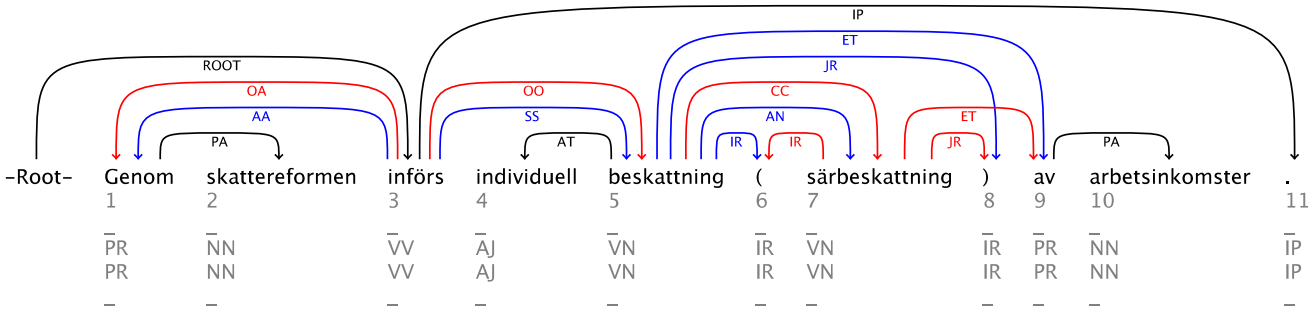
Results the remaining treebanks are not reported. Conducting experiments is a computationally intensive task[9], and it was deemed futile to conduct experiments on treebanks containing with very few quotations. This principle also applies to the following experiments.

3.4.2 Parenthesis

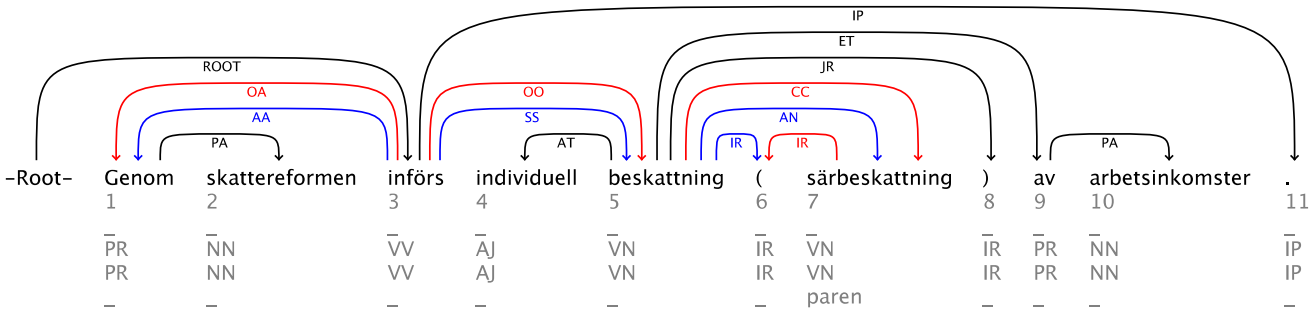
The prelimenary survey of precision for sentences with various punctuation (3.2 on page 36) reveals that the average labeled attachment score for sentences in the Swedish treebank in which parentheses occur is 8.41 – or roughly 10% – lower than the average for the treebank. Parsing of sentences with parenthesis in the Italian *Newspaper* treebank also seem to suffer (5.36 worse), and the same

[8] Søgaard also used a truncated treebank for training, so the results are not comparable.

[9] On the relatively modern hardware used, training often took many hours, and the parameters learnt by a parsing model easily takes up several GB on disk.



(a) Baseline parse



(b) Parse when augmented with a feature marking parenthesized tokens.

Figure 3.7: Parses of a Swedish sentence with parentheses. Black arcs were correctly predicted, red arcs erroneously predicted, and blue arcs should have been predicted, but were not. Comparing the system parse to the baseline, two erroneous dependencies, both crossing the boundary of the parenthesized phrase, have been corrected.

goes for the Slovene and Turkish treebanks (13.07 and 8.05 worse, respectively). For the latter two treebanks, parenthesized tokens only occur in 1 and 4 test sentences respectively – i.e. too sparsely for testing. Relative to the treebank average, parentheses in the Italian *Law* treebank do not seem to impair parsing. Nonetheless, this treebank was included in experiments, partly because parentheses in this treebank are widespread (21% of sentence), and partly for increasing the size of the critically small[10] Italian treebank in a trial where *Newspaper* and *Law* are concatenated to form a single treebank (*Both*).

Augmenting parenthesized tokens with a PAREN feature yields a modest, yet statistically insignificant[11], improvement in overall labeled attachment score for both Italian treebanks, and for the combined treebank. For the *Newspaper* treebank, the augmentation has a positive effect on the sentences with parentheses, and close to no effect on the rest. The opposite is evident for the other Italian treebank (*Law*), where there is a modest positive effect on other sentences *without* parentheses, but the effect on augmented sentences is next to none. This absence is perhaps due to the fact that on average, sentences with parentheses in *Law* were already on par with the treebank overall. Still, it seems the augmentation helped improve the learned model slightly overall, also evident for the combined treebank (*Both*), but again not significantly so.

For the Swedish treebank, there no or a slight adverse effect of the augmentation overall, but a marked increase in LAS of the sentences with parenthesis, amounting to a reduction in error rate of 10%. Yet none of the observed differences are statistically significant, with $p = 0.13$ for the affected sentences being closest.

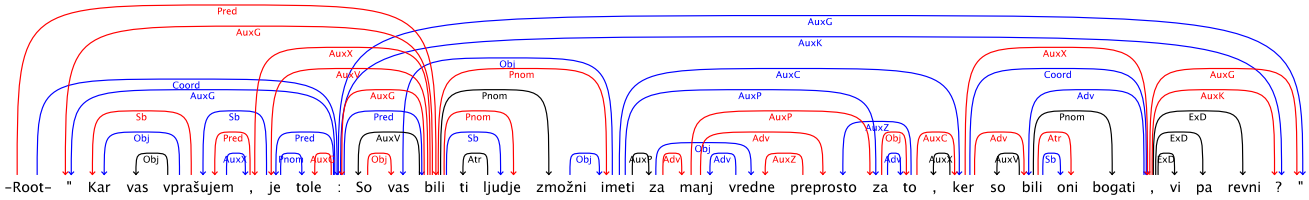
3.4.3 Colon

For several languages, colons seem to be correlated with poor parsing precision. As with quotation marks and parenthesis, colons are often used to introduce a parenthetical phrase within another sentence. These parentheticals may be complete sentences on their own right, and as such, should often not have its dependency structure mixed up with the outer sentence. Manual inspection of the baseline parses of sentences with colons indicate that dependency errors often involve arcs erroneously crossing the boundary to the outer sentence (see figure 3.8 for an example). In particular, in case a token in the nested sentence is mistakenly taken to be head of the sentence as a whole, many derived errors will likely follow from this.

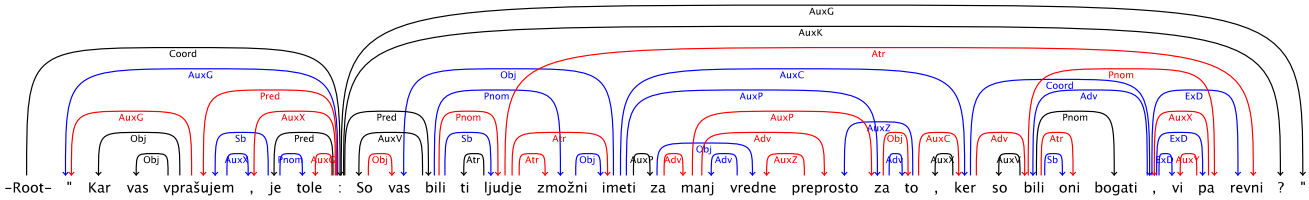
To help distinguish nested sentences introduced with a colon, the feature scheme COLON simply adds the feature COLON to tokens occurring *after* a colon.

[10] As noted on page 29, it is not really methodically sound to base experiments on a small treebank in a fixed train/test split. The smaller the treebank, the greater the variance of results on different treebank partitions. Normally, in estimating the predictive accuracy, one would employ 10-fold cross-validation or another technique to compensate somewhat for the large variance. However, for comparability, I have chosen to use the same setup as on the EVALITA 2007 conference [Magnini and Cappelli, 2007] and keep the given train/test split.

[11] With a 95% confidency interval. Refer to table 3.4 for the p-values of the observed effects.



(a) Baseline parse



(b) Parse when augmented with a feature marking tokens which occur after a colon

Figure 3.8: Parses of a Slovene sentence where the latter half, which is a complete sentence on its own part, is introduced with colon. Black arcs were correctly predicted, red arcs erroneously predicted, and blue arcs should have been predicted, but were not. Comparing the system parse to the baseline, the seperation of the two sentences is much better: No dependencies erroneously cross the boundary between the two. The system also correctly identifies the root token of the sentence, which seems to be the source of much confusion for the baseline.

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
Talbanken	Colon	overall			87.27%	81.08%	87.41%	81.29%	0.14	0.21	*1.11%*	0.2620
		affected	5	1%	63.77%	53.58%	65.28%	55.85%	1.51	2.27	*4.89%*	0.3777
		unaffected	384	99%	88.43%	82.42%	88.50%	82.54%	0.07	0.12	0.68%	0.3612
Both	Colon	overall			88.86%	76.59%	89.00%	76.69%	0.14	0.10	0.43%	0.3503
		affected	0	0%	0.00%	0.00%	0.00%	0.00%	0.00	0.00	0.00%	
		unaffected	200	100%	88.86%	76.59%	89.00%	76.69%	0.14	0.10	0.44%	0.3503
SDT	Colon	overall			78.56%	69.59%	78.75%	69.97%	0.19	0.38	*1.24%*	0.1531
		affected	10	2%	63.19%	50.69%	63.19%	52.08%	0.00	1.39	*2.82%*	0.2552
		unaffected	392	98%	79.29%	70.49%	79.48%	70.81%	0.19	0.32	*1.08%*	0.1964

Table 3.5: Results for the COLON augmentation scheme, which marks tokens occurring after a colon with the feature COLON. Again, *Both* refers to a concatenation of the two Italian treebanks *Newspaper* and *Law*.

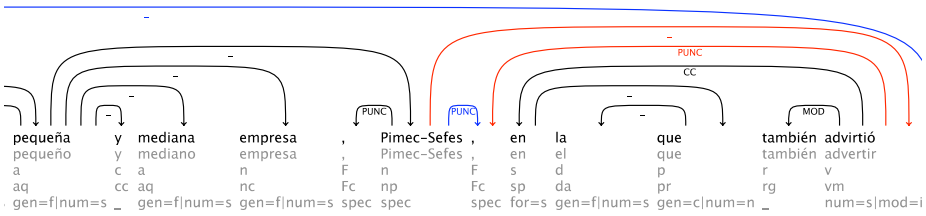


Figure 3.9: Zooming in on some trouble with parsing a nominal apposition.

The assumption is that singling out tokens after the colon is consistent with the way colons are used (the nested sentence is never the one before the colon).

Some heuristics are employed to handle colons appearing within a pair of parentheses. Clearly, such a case should not result in tokens outside the parenthesis being marked.

Experiments with Swedish, Italian and Slovene show modest, although not statistically significant, improvements in LAS, with Slovene topping by an overall error reduction of 1.24% ($p = 0.15$). Focusing in on sentences with colons, we see the greatest improvement for the Swedish treebank: 2.27, or a 4.89% reduction of the error rate, but again not significant ($p = 0.37$).

Colons are not exactly widely used in these treebanks, and in the cast of the test part of the Italian treebank, they occur exclusively sentence final. No tokens occur after the colon in the sentence final case, and thus no sentences in the test corpus are directly affected by the augmentation. The small difference observed on the Italian treebank is solely caused by augmentation of the training corpus[12].

I am assuming that the sparsity of colons is partly to blame for the relatively poor statistical significance of the observed effect, and I believe that a significant effect might be shown with a treebank where colons occur more frequently.

3.4.4 Apposition

Appositions are parallel constructions, in which two or more words or phrases are syntactically parallel without an explicit conjunction, but often surrounded by a pair of hyphens or commas. They are commonly used to specify an alternate form or synonym for an entity or a term which might be unfamiliar to the audience, as exampled by the following (tokenized) sentence from the Spanish corpus:

```
Esta puntualización fue considerada como un " avance " por los sindicatos
Confederación_Francesa_del_Trabajo - CFDT - , Fuerza Obrera - FO - ,
Confederación_Francesa_de_Trabajadores_Cristianos - CFTC -
Confeder_ación_Francesa_de_Directivos_de_Empresas - CFE-CGC - .
```

[12]The observation of this detail is the reason for including the outcome of the Italian Colon experiment at all – if not for this, it would have been unnecessary.

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
Cast3lb	CommaApposition	overall			82.19%	78.42%	82.28%	78.42%	0.09	0.00	0.02%	0.4937
		affected	8	4%	80.29%	77.78%	82.08%	79.93%	1.79	2.15	*9.68%*	*0.0657*
		unaffected	198	96%	82.29%	78.45%	82.29%	78.34%	0.00	*-0.11*	0.00%	0.3836
Cast3lb	DashApposition	overall			82.19%	78.42%	82.56%	78.70%	0.37	0.28	*1.32%*	0.2228
		affected	10	5%	86.79%	84.08%	85.29%	82.28%	*-1.50*	*-1.80*	0.00%	0.1897
		unaffected	196	95%	81.91%	78.06%	82.39%	78.47%	0.48	0.41	*1.87%*	*0.1354*
Cast3lb	CommaApposition + DashApposition	overall			82.19%	78.42%	82.40%	78.52%	0.21	0.10	0.48%	0.3782
		affected	17	8%	83.45%	81.03%	83.79%	81.21%	0.34	0.18	0.95%	0.4895
		unaffected	189	92%	82.05%	78.12%	82.24%	78.22%	0.19	0.10	0.46%	0.3920

Table 3.6: Results for the augmentation schemes for appositions.

Being syntactically parallel constructions, appositions involve some of the same difficulties as other coordinated and syntactically parallel constructions, which are notoriously troubled [Kübler and Prokić, 2006]. Being present in 19.9% of the sentences, hypens are higly frequent in the Spanish corpus, which makes it a good candidate for trying to single out appositions with an improvement of accuracy in mind.

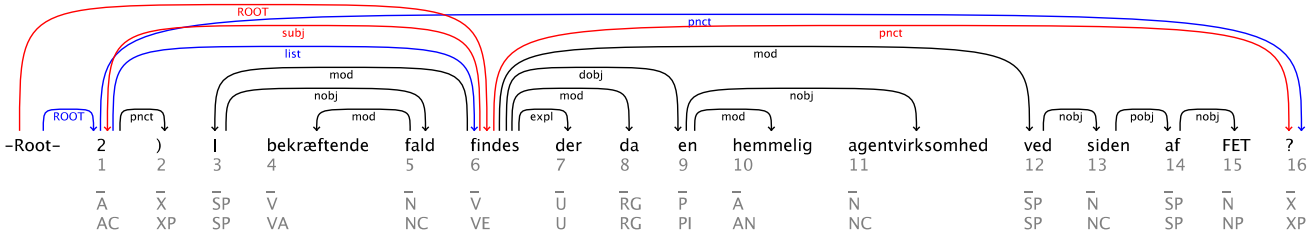
Hyphens and especially commas are used for other purposes than appositions, so it is not entirely trivial to determine which cases an augmentation should apply to. Taking a simple but relatively safe approach, I use the heuristic of only augmenting single, nominal[13] tokens which are surrounded by a pair of hyphens or commas. The chosen heuristic is likely too restrictive, thus sacrificing coverage. Perhaps consequently, the observed effect is somewhat dubious.

The augmentation scheme with commas, CommaApposition, yields a large (2.15) and nearly significant ($p \approx 0.066$) improvement in LAS for sentences which were affected by the augmentation, but a has a slight adverse effect on the remaining 96% of the sentences, therefore yielding next to no effect on the overall average.

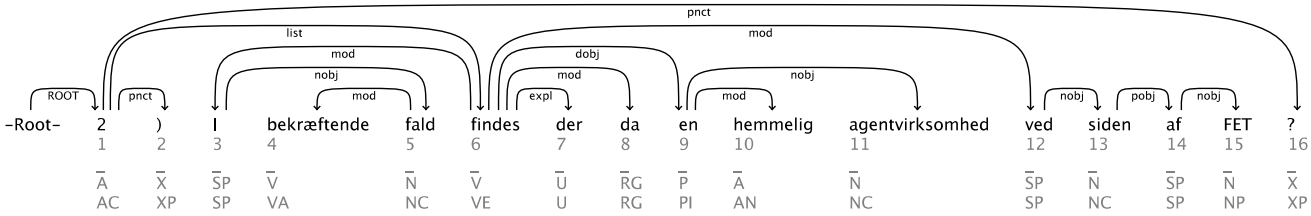
Turning to the scheme for appositions marked by hypens, DashApposition, an adverse effect (-1.80) is observed for the affected sentences, and a small postive effect (0.41) for the rest. Table 3.6 also shows the outcome of a trial with both schemes applied, the result of which is slightly more encouraging, though still not statistically significant.

The chosen heuristic for identifying appositions is almost certainly bad, and likely the cause of the unclear results. It would be interesting to train a probabilistic model for labeling appositions and conduct the same experiment with this driving the augmentation instead of the heuristic. Several treebanks (e.g. Arabic and Czech) clearly indicate appositions in the dependency structure. Deriving training data from these existing treebanks and training an HMM- or CRF-based tagger [Jurafsky and Martin, 2008] to more reliably identify appo-

[13] In the case of the Spanish corpus, tokens with a prefix of "N" in the part of speech tag are taken to be nominal.



(a) Baseline parse



(b) Parse when augmented with a BULLET feature marking the list counter

Figure 3.10: Parses of an itemized sentence from the Danish Dependency Treebank. In this case, the augmented system correctly identifies the sentence head and achieves 100% accuracy.

-Root-	3	)	Har	ministeren
	A	X	V	N
	AC	XP	VA	NC
	case=unmarked|bullet	bullet	mood=indic|tense=present|voice=active	gender=common|number=sing|case=unmarked|def=d

Figure 3.11: Zooming in on the feature BULLET as added to the tokens comprising the list counter.

sitions in unparsed text seems feasible.

3.4.5 List items

Another phenomenon of written text posing a challenge to the baseline parser is items in of a numbered or bulleted list. In particular, 3% of the test part of the Danish Dependency Treebank are sentences of the following kind:

> 3) Har ministeren løjet om sit og regeringens kendskab til virksomheden?

Two-thirds of the itemized sentences in the DDT test set are numbered list items like this, while the rest have the form of description lists, e.g.:

> Foto : Palle Hedemann

In these itemized sentences, the list counter – e.g. *3* – acts as head of the sentence, with the token that would otherwise be the head (commonly a finite

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
DDT	NumericalBullet	overall			88.12%	82.89%	88.53%	83.19%	0.41	0.30	*1.73%*	*0.1426*
		affected	6	2%	83.78%	76.58%	90.99%	85.59%	7.21	9.01	*38.47%*	*0.0287*
		unaffected	316	98%	88.21%	83.02%	88.49%	83.14%	0.28	0.12	0.71%	0.3276
DDT	Bullet	overall			88.12%	82.89%	88.55%	83.58%	0.43	0.69	*4.01%*	*0.0094*
		affected	9	3%	81.82%	76.22%	87.41%	80.42%	5.59	4.20	*17.66%*	0.1579
		unaffected	313	97%	88.28%	83.06%	88.58%	83.66%	0.30	0.60	*3.54%*	*0.0203*

Table 3.7: Results for augmenting the Danish Dependency Treebank with a feature marking list counters (NumericalBullet) and list bullets in general (Bullet).

verb) depending on the list counter. The baseline parser rarely gets this right (an example of baseline error is shown in figure 3.10 on the previous page).

A simple augmentation scheme (NumericalBullet) provides a remedy for this: Add a BULLET feature to tokens forming list counters, e.g. *3* and closing parenthesis in the example above (see figure 3.11 on the preceding page). A significant ($p < 0.03$) improvement in LAS is seen for sentences with numerical list counters, specifically a gain of 9.01, equivalent to a 38.5% reduction in error rate. A more modest improvement is seen for unaffected sentences (not significant).

With an extended augmentation scheme (Bullet in table 3.7) that includes list markers in general, e.g. also *Foto* and the colon from the example on the preceding page, there is a significant ($p < 0.01$) overall improvement in LAS of 0.69, equivalent to a reduction in overall error rate of 4.0%. Interestingly, while itemized sentences only account for 3% of the sentences overall, there is a marked effect on other, non-itemized sentences, specifically a 0.60 LAS gain ($p < 0.03$). Non-symmetrical usage of parentheses for itemization clearly requires another treatment than the general symmetrical use of parentheses[14], and this result could be construed as helping the parser discriminate these.

3.4.6 Coordinated enumerations

Results. Discussion.

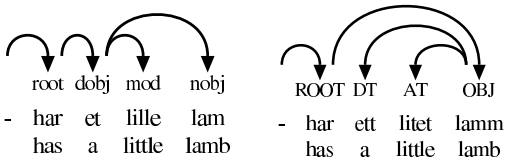
3.4.7 Lemmatization

[Bart and Haltrup, 2008]

Results: Danish.

Discussion: Allows parser to generalize.

[14]The same can be said of the numerals used as list counters.



(a) DDT (b) Talbanken05

Figure 3.12: Different treatment of functional categories in DDT and in Talbanken05. In DDT, determiners – such as *et* on the left hand side – act as heads with nominal dependents, whereas Talbanken05 treats *nouns* as heads with functional dependents, as illustrated on the right hand side. The dependency structure examples are from Lilja Øvrelid's report on porting the animacy feature to Danish [Øvrelid , 2009].

3.4.8 Animacy

Results: Danish.

Discussion: Coverage and quality of animacy data. Head status of functional categories in DDT versus Talbanken. [Øvrelid , 2009]

3.4.9 Feature combinations

Effects of combining features.

The precise effect of combining is unclear.

Does not seem to be additive.

FUTURE WORK

3.5 Results

Summarize results.

Can only do so much with this approach: Can't explicitly introduce features for pairs of tokens.

Chapter 4

Conclusion

4.1 Future research

Optimized parser configurations to bring baseline scores on par with CoNLL results. Scoring as in CoNLL. For comparison.

Expand on pronominalization, givenness and word order [Weber and Müller, 2004].

Convert CCH annotation of conjunctions to (one of the other conventions).

Mark italics (*mention*).

FUTURE: discriminate appositions and parentheticals.

FUTURE: Many of the feature schemes depend on the high-level segmentation implied by the punctuation of a sentence. Symmetrical markers of parentheticals (colon, quotation mark) introduce a scope which often limit the extend of the augmentation desired. As an example, if a colon occurs within a pair of parentheses, and we trying to discriminate nested sentences, our augmentation should *only* apply as far as the closing parenthesis. In augmentation, it would be extremely useful with a *(*possibly syntactically agnostic) system for *high-level segmentation* of a sentence. Also, if one were to design a feature which in general indicates whether a token could potentially be head of the sentence as a whole, a high-level structural analysis such as this would also be very handy for excluding candidates for the sentence head. The segmentation system could be punctuation-based[1], perhaps like the one discussed by Kubon et al. [2006].

FUTURE: Split out quotations (and other parenthethicals which are complete on their on right) and parse them separately.

[1] As augmentation takes place *before* the syntactic analysis, it cannot rely on a the syntactic analysis for identifying the scopes to which the augmentation should apply. Punctuation and other superficial properties could be a good basis for the segmentation. There will likely be cases of ambiguity which require a deeper analysis to be resolved, so a joint segmentation and syntactic analysis could also prove useful.

Acknowledgements

- Lilja Øvrelid (animacy tagger)
- Bart Jongejan (danish lemmatizer)
- Anders Søgaard (idea, guidance)
- Sebastian Riedel and Ivan Vladimir (What's Wrong with my NLP?)

Bibliography

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta sintá(c)tica: a treebank for portuguese. In *Proc. of the 3rd Intern. Conf. on Language Resources and Evaluation (LREC)*, page 1698–1703, 2002.

N. B. Atalay, K. Oflazer, and B. Say. The annotation process in the turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, 2003.

Giuseppe Attardi and Felice Dell'Orletta. Chunking and dependency parsing. In *Proceedings of LREC 2008 Workshop on Partial Parsing*, Marrakech, 2008.

Jongejan Bart and Dorte Haltrup. The cst lemmatiser. Technical report, Center for Sprogteknologi, University of Copenhagen, 2008. URL `http://cst.dk/download/cstlemma/current/doc/cstlemma.pdf`.

Akshar Bharati, Samar Husain, Bharat Ambati, Sambhav Jain, Dipti M. Sharma, and Rajeev Sangal. Two semantic features make all the difference in parsing accuracy. In *Proceedings of the 6th International Conference on Natural Language Processing (ICON-08)*, CDAC Pune, India, 2008.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. The pdt: a 3-level annotation scenario. In A. Abeille, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers, 2001.

Dan Bikel. Randomized parsing evaluation comparator. http://www.cis.upenn.edu/ dbikel/software.html#comparator, 2004. URL `http://www.cis.upenn.edu/~dbikel/software.html\#comparator`.

Cristina Bosco, Leonardo Lesmo, Lombardo Vincenzo, Alessandro Mazzei, and Livio Robaldo. Linguistic notes for the turing university treebank - a detailed description of the annotation of the complex linguistic expressions. Technical report, Turin University, Turin, 2008. URL `http://www.di.unito.it/~tutreeb/documents/noteling-engl-15-11-08.pdf`.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The tiger treebank. In *Proc. of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*, 2002.

Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Tenth Conference on Computational Natural Language Learning*, page 149, 2006.

Sabine Buchholz, Erwin Marci, Yuval Krymolowski, and Amit Dubey. Conll 2006 shared task website. http://nextens.uvt.nl/ conll/, 2006. URL `http://nextens.uvt.nl/~conll/`.

Noam Chomsky. U.s. foreign policy in central america, May 1984. URL `http://en.wikiquote.org/wiki/Chomsky`.

Massimiliano Ciaramita and Giuseppe Attardi. Dependency parsing with second-order feature maps and annotated semantic information. In *Proc. of the 12th International Workshop on Parsing Technologies (IWPT), 2007*, 2007.

M. Civit, A. Martí, B. Navarro, N. Bufi, B. Fernández, and R. Marcos. Issues in the syntactic annotation of cast3lb. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, 2003.

Montserrat Civit and Ma Antònia Martí. Building cast3lb: a spanish treebank. *Research on Language and Computation*, 2(4):549–574, December 2004. doi: 10.1007/s11168-004-7429-x. URL `http://dx.doi.org.ep.fjernadgang.kb.dk/10.1007/s11168-004-7429-x`.

Montserrat Civit, Ma. Martí, and Núria Bufí. Cat3lb and cast3lb: from constituents to dependencies. In *Advances in Natural Language Processing*, pages 141–152. 2006. URL `http://dx.doi.org.ep.fjernadgang.kb.dk/10.1007/11816508_16`.

Michael A. Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, page 95–102, 2001.

Peter de Swart, Monique Lamers, and Sander Lestrade. Animacy, argument structure, and argument encoding. *Lingua*, 118(2):131–140, 2008.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. Towards a slovene dependency treebank. In *Proc. of the 5th Intern. Conf. on Language Resources and Evaluation (LREC)*, 2006.

J. Einarsson. *Talbankens skriftspraakskonkordans*. 1976.

Jason Eisner and Noah A. Smith. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*, page 30–41, 2005.

Hartmut Haberland. Reported speech in danish. *Direct and indirect speech*, page 218–54, 1986.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf , and E. Beška. Prague arabic dependency treebank: development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, page 110–117, 2004.

Johan Hall and Jens Nilsson. Conll-x shared task: multi-lingual dependency parsing, 2006. URL `http://w3.msi.vxu.se/users/jni/courses/ml2/ml2_jni_jha.pdf`.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beata Megyesi, Mattias Nilsson, and Markus Saers. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, page 933–939, 2007.

Ryoji Hamabe, Kiyotaka Uchimoto, Tatsuya Kawahara, and Hitoshi Isahara. Detection of quotations and inserted clauses and its application to dependency structure analysis in spontaneous japanese. 2006.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 2008. ISBN 0131873210, 9780131873216.

Y. Kawata and J. Bartels. Stylebook for the japanese treebank in verbmobil. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen, 2000.

Matthias Trautner Kromann. The danish dependency treebank and the dtag treebank tool. In *Proceedings of TLT*, page 217–220, 2003.

Vladislav Kubon, Marketa Lopatkova, Martin Platek, and Patrice Pognan. A linguistically-based segmentation of complex sentences. 2006.

Sandra Kübler and Jelena Prokić. Why is german dependency parsing more reliable than constituent parsing. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT)*, page 7–18, 2006.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127, 2009. URL `http://www.morganclaypool.com/doi/abs/10.2200/S00169ED1V01Y200901HLT002`.

B. Magnini and A. Cappelli. Evalita 2007: evaluating natural language tools for italian. *Intelligenza Artificiale*, 2007.

Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. 2006.

Tetsuji Nakagawa. Multilingual dependency parsing using global features. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, page 952–956, 2007.

J. Nilsson, J. Hall, and J. Nivre. Mamba meets tiger: reconstructing a swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*, 2005.

J. Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, page 149–160, 2003.

Joakim Nivre. Dependency grammar and dependency parsing. *MSI report*, 5133, 2005.

Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, December 2008. doi: 10.1162/coli.07-056-R1-07-027. URL `http://dx.doi.org/10.1162/coli.07-056-R1-07-027`.

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 CoNLL Shared Task Session of EMNLP-CoNLL*, volume 7, page 915–932, 2007.

Václav Novák and Zdeněk Žabokrtský. Feature engineering in maximum spanning tree dependency parser. In *Text, Speech and Dialogue: 10th International Conference, Tsd 2007, Pilsen, Czech Republic, September 3-7, 2007, Proceedings*, page 92. Springer, 2007.

G. Nunberg. *The linguistics of punctuation*. Cambridge University Press, 1990.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. Building a turkish treebank. 2003.

Sebastian Riedel and Ivan Vladimir. What's wrong with my nlp?: a visualizer for natural language processing problems, version 0.2.2. http://code.google.com/p/whatswrong/, 2009. URL `http://code.google.com/p/whatswrong/`.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with lr models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, page 1044–1050, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1111`.

K. Simov and P. Osenova. Practical annotation scheme for an hpsg treebank of bulgarian. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, page 17–24, 2003.

K. Simov, G. Popova, and P. Osenova. Hpsg-based syntactic treebank of bulgarian (bultreebank). In A. Wilson, P. Rayson, and T. McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, page 135–142. Lincom-Europa, Munich, 2002.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. Design and implementation of the bulgarian hpsg-based treebank. In *Journal of Research on Language and Computation – Special Issue*, page 495–522. Kluwer Academic Publishers, 2005.

O. Smrž, J. Šnaidauf , and P. Zemánek. Prague dependency treebank for arabic: multi-level annotation of arabic corpus. In *Proc. of the Intern. Symposium on Processing of Arabic*, page 147–155, 2002.

Robert E. Tarjan. Finding optimum branchings. *Networks*, 7(1), 1977.

U. Teleman. *Manual för grammatisk beskrivning av talad och skriven svenska (MAMBA)*. 1974.

Shisanu Tongchim, Virach Sornlertlamvanich, and Hitoshi Isahara. Experiments in base-np chunking and its role in dependency parsing for thai. 2008.

L. van der Beek, G. Bouma, J. Daciuk, T. Gaustad, R. Malouf, G. van Noord, R. Prins, and B. Villada. The alpino dependency treebank. In *Algorithms for Linguistic Processing*. 2002a.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. The alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*, 2002b.

Rebecca Watson and Ted Briscoe. Adapting the rasp system for the conll07 domain-adaptation task. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, page 1170–1174, 2007.

Andrea Weber and Karin Müller. Word order variation in german main clauses: a corpus analysis. In *Proceedings of the 20th International conference on Computational Linguistics*, page 71–77, 2004.

Ian H. Witten and Eibe Frank. *Data mining*. 2005. ISBN 0120884070, 9780120884070.

Ming Zhou. A block-based robust dependency parser for unrestricted chinese text. 2000.

Lilja Øvrelid . Finite matters. In *Proceedings of the 6th international conference on Advances in Natural Language Processing*, pages 500–509. Springer, 2008a.

Lilja Øvrelid . Linguistic features in data-driven dependency parsing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL 2008)*, 2008b.

Lilja Øvrelid . Cross-lingual porting of distributional semantic classification. In Kristiina Jokinen and Eckhard Bick, editors, *Proceedings of the 17th Nordic Conference on Computational Linguistics (NODALIDA)*, volume Vol. 4, pages 246–249, 2009. URL `http://hdl.handle.net/10062/9795`.

Lilja Øvrelid and Joakim Nivre. When word order and part-of-speech tags are not enough – swedish dependency parsing with rich linguistic features. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, page 447–451, 2007.

Appendix A

Source code

Include?

Appendix B

Example runtime output

Include?