Feature Engineering for Data-Driven Dependency Parsing

Christian Rishøj Jensen

August 2009

IT og Kognition
Institut for Nordiske Studier og Sprogvidenskab
Københavns Universitet

Under vejledning af Anders Østerskov Søgaard, Center for Sprogteknologi.

141.983 tegn (inkl. mellemrum, eksl. appendiks).

Dansk resumé

Dependensgrammatik til syntaktisk analyse er ingen ny opfindelse, men har været genstand for en øget interesse de senere år, særligt på grund af fremkomsten af datadrevne systemer, der forholdsvist let kan optrænes til at analysere sætninger med en nøjagtighed, der kan måle sig med systemer af den traditionelle slags, som drives af omfattende grammatikker, udfærdiget af eksperter ved håndkraft gennem års arbejde.

I dette speciale undersøges mulighederne for at forbedre de datadrevne dependensparsere gennem berigelse af træbanker med simple, lingvistisk motiverede træk. Visse sproglige konstruktioner, herunder citationer og parentesiske bemærkninger, volder i mange tilfælde problemer for de datadrevne systemer, antageligvis fordi sådanne konstruktioner ofte er strukturelt afvigende fra den typiske sætning.

Hypotesen for undersøgelsen er den, at tilføjelse af træk strategiske steder i træbanker kan tillade det datadrevne system at tilse problematiske konstruktioner en særlig behandling i den indlærte model, og derved højne parserens analytiske nøjagtighed.

I en indledende undersøgelse af en referenceparsers behandling af træbanker på forskellige sprog identificeres nogle mulige indsatsområder. En række konkrete muligheder for berigelse af træbankerne foreslås og afprøves. I adskillige tilfælde opnås ved berigelserne signifikante forbedringer i forhold til referenceparseren, særligt i forbindelse med indskydninger som citationer og parentesiske bemærkninger, men også ved indførsel af træk rettet mod atypisk brug af tegnsætning, eksempelvis i listeopstillinger.

Tilgangen er lovende, men også forbundet med visse udfordringer, herunder specificitet for sprog, konventioner for tegnsætning og i nogle tilfælde den anvendte opmærkning i træbankerne. Ikke desto mindre rummer flere af de foreslåede træk umiddelbar mulighed for at forbedre de mest avancerede af dagens datadrevne dependensparsere.

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, structurally 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 note 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 – with a practical example – when my mother reports on a heated discussion with a salesperson, a clear change in the tone of her voice provides me with an indication of when she is *directly reporting* the other person's speech.

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 seen an increasing interest in the last decade. While neither dependency parsing nor the underlying formalism of *dependency grammar* 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 analyses in genereal – including human comprehension. 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 course, 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

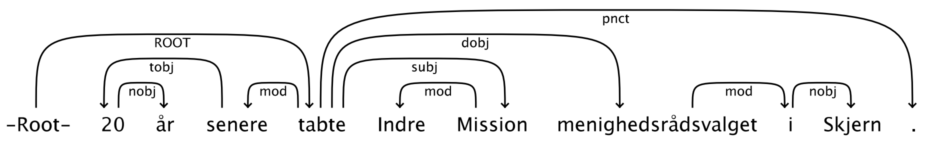
2.1 Dependency grammar

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 the syntactic structure of sentences 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 represents 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 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.



pnct
ROOT
dobj
tobj
subj
nobj
mod
mod
mod
nobj
-Root- 20 år senere tabte Indre Mission menighedsrådsvalget i Skjern .

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

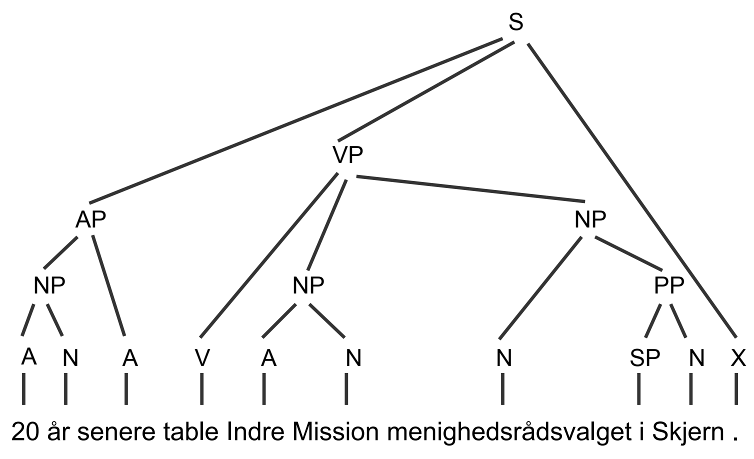

S
VP
AP
NP
NP
NP
NP
PP
A N A V A N N SP N X
20 år senere table Indre Mission menighedsrådsvalget i Skjern .

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

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 formalisms [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 maintain the single head constraint and 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 conjunction 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 is projective if and only if it satisfies the *planar property*, namely that it is possible to graphically configure all the dependency arcs in a plane 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 its own endpoints w_i, w_j – when regarding the linear order of tokens as they occur in 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 $j < i$.

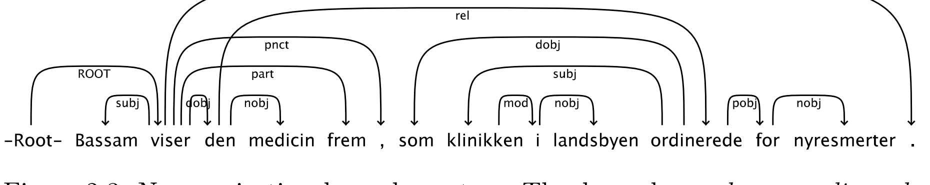

pnct
rel
pnct
dobj
ROOT
part
subj
subj
dobj
nobj
mod
nobj
pobj
nobj
-Root- Bassam viser den medicin frem , som klinikken i landsbyen ordinerede for nyresmerter .

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

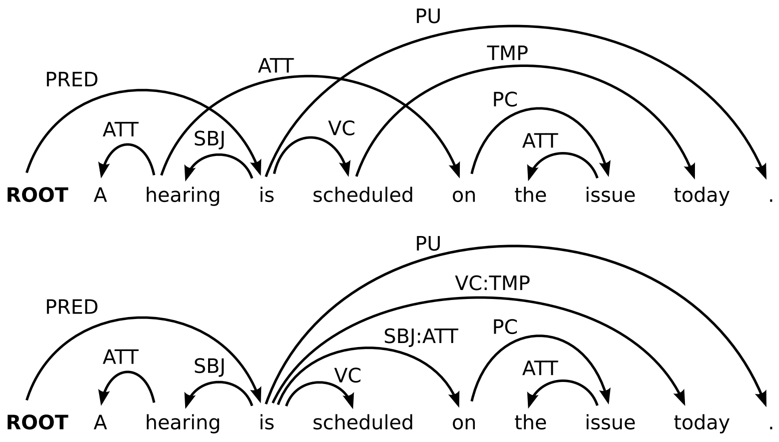

PU
TMP
ATT
PRED
PC
ATT
VC
SBJ
ATT
ROOT A hearing is scheduled on the issue today .
PU
VC:TMP
PRED
SBJ:ATT
PC
ATT
SBJ
VC
ATT
ROOT A hearing is scheduled on the issue today .

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

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 word 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 ones (*projectivization*) and back (*de-projectivization*) as steps of pre- and post-processing (see figure 2.4).

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 on the other hand, a method from *machine learning* is used to train a parsing model on a collection of example analyzed sentences, resulting in some *learned parameters*. The learned parameters are in turn 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 thus are indeed both data-driven and grammar-based.

In the terminology of Kübler et al. [2009], 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,

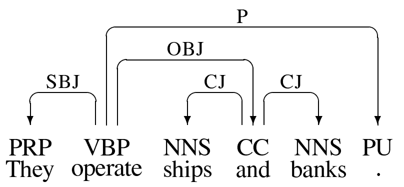
2. a fixed *algorithm* h, and

3. a – possibly empty – set of *parameters* λ that guide the parsing algorithm.

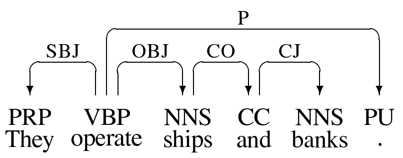
For a grammar-based system, the constraints would include the rules of grammar in use.

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 6, 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 locutions [Bosco et al., 2008], head-status of functional categories [Øvrelid , 2009] and coordinations [Nivre, 2005], which are particularly open to interpretation from a dependency viewpoint (figure 2.5 illustrates this for a coordinated phrase).

A data-driven parser must learn these annotational conventions when presented with the training material. In formal terms, the training material $\mathcal{D}$ is

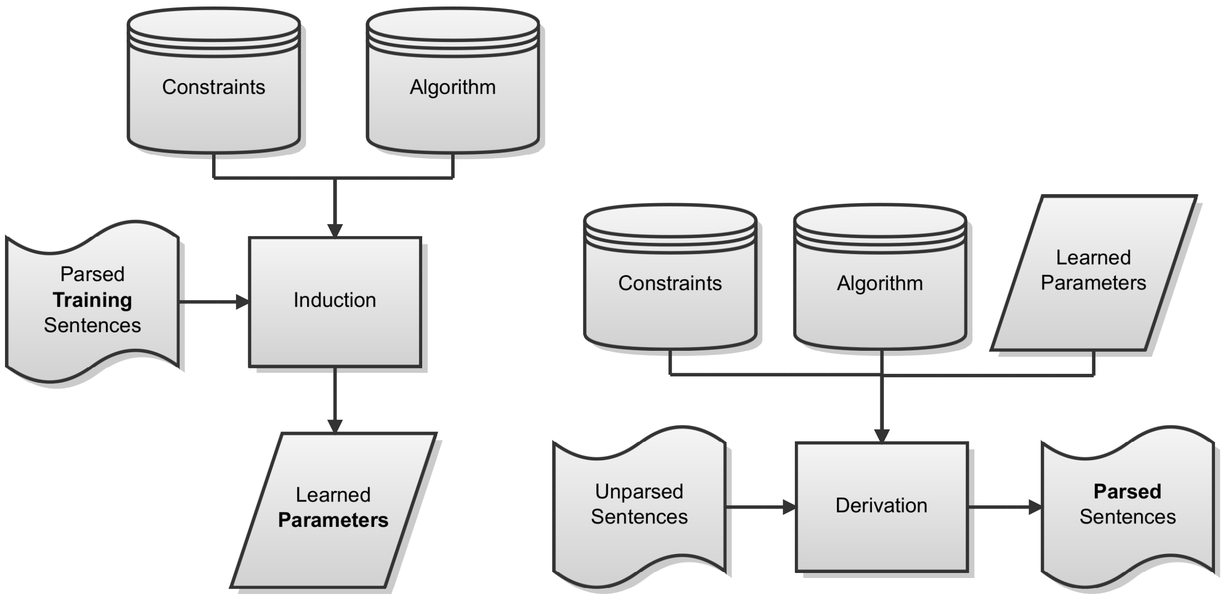


(a) Coordinating conjunction as head



(b) Coordinated conjunct as head

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



(a) Learning phase (b) Parsing phase

Figure 2.6: Phases of data-driven parsing.

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 specific dependency arc occurring in a sentence.

For reasons of notational and computational convenience, it is common to introduce an artificial token in the beginning of the sentence, denoted by ROOT, which has the special property of being headless.

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 one or more tokens 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 an 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 sentence and the *parsing history*, i.e., 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 shift-reduce models, 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 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*, *single-headed* and *spanning* (all tokens), but for some transition sequences, A may fail to satisfy the *connectedness* criteria of well-formed dependency graphs. In Nivre's [2003] parser, such disconnected trees in A are simply connected to the root node at the completion of the transition sequence.

With a sentence of n tokens, the number of transitions needed for this shift-reduce parser to complete is bounded by n. If determining the transition to take can be done in a constant time, the parser operates with linear time complexity $O(n)$.

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.

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 popular approach, namely 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 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, but a common property is that the scoring function is decomposed into smaller functions that score local properties of the graph being evaluated. For for *first-order* or *arc-factored* models, the scored properties are single attachments, whereas *second-order* models score pairs of attachments. 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 found by combining the local property scores[1]. The particular way in which subgraph scores are combined may differ between graph-based models, but a fundamental assumption is shared among all: The score of a graph G factors through the scores of the subgraphs of G [Kübler et al., 2009]. Let us assume local property scores are combined by summation. Then, for an arc-factored model, in which each arc is scored individually by the function $\lambda(w_i, r, w_j)$, we can write:

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

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:

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

2.2.3.2 Parsing as a graph search problem

Given the sentence S and dependency types R, let us construct a graph $G_S = (A_S, V_S)$ which contains all possible of all possible dependency types between all pairs of tokens. For this purpose, V_S is the original set of tokens $\{w_0, w_1, ..., w_n\}$, and we want a full set of arcs for all but 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.* Implicitly, G_S contains all possible parse trees for S. In G_S, each possible parse tree is a *spanning tree*, that is, a subgraph which is forms a directed tree and all the nodes of G_S.

Using the scoring function from before to assign scores to spanning trees, the *maximum spanning tree* (MST) for graph G_S is simply the highest scoring spanning tree for G_S. Or, in other words, the highest scoring parse tree. Parsing the sentence is then a matter of finding the maximum spanning tree of G_S.

As arc-factored models score each arc individually, we can simplify the search problem a bit by removing all but the highest scoring arc between any two nodes in G_S. This will effectively reduce the multi digraph G_S to a digraph $G'_S = (V'_S, A'_S)$:

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

This reduction is risk-free with arc-factored models, because the arcs kept in the reduced graph G'_S must include those of the maximum spanning tree. This

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

can be shown by contradiction: If a maximum spanning tree of G_S contains an arc not in A'_S, then it would be possible to get a higher scoring tree by substituting arcs in it from A'_S, as A'_S by definition contains the highest scoring arcs between all pairs of nodes.

When reducing G_S to the digraph G'_S, we stripped the arc labels. In order to recover these again, it is necassary to keep track of them, simply by noting the label r that yielded the highest arc score $\lambda(w_i, r, w_j)$ for each pair of nodes.

Now, with the set of possible dependency trees for the sentence S being identical to the set of spanning trees for G'_S, we can use the scoring function score(G) in a search for the maximum spanning tree of G_S (best parse). Graph theory provides several maximum spanning tree finding algorithms. One such algorithm is Chu-Liu-Edmons, which proceeds by iteratively

1. selecting the highest scoring arc for each node in greedy fashion, and
2. using 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 previous page, an arc score is 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 the whole dependency graph $G = (V, A)$ the score amounts to the sum of 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 $G \in \mathcal{G}_S$ which has the highest score:

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

So at training time, the graph-based model must learn the weight vector $\mathbf{w}$ from training data. A simple approach is to use a perceptron [Rosenblatt, 1958]. With the perceptron, learning proceeds by iteratively looking at 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.
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.

	# 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 used at the CoNLL shared task [Hall and Nilsson, 2006]

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. Several transtition-based and graph-based parsing systems were submitted, including the arc-factored MST-Parser [McDonald et al., 2006] and the shift-reduce based MaltParser [Nivre, 2003]. These two jointly ranked 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]. The treebanks used differ not only in language, but also in 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 (the *CoNLL format*). Having a

	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]

common file format was instrumental for comparing systems across the whole spectrum of corpora.

2.3.1 CoNLL format

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

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.

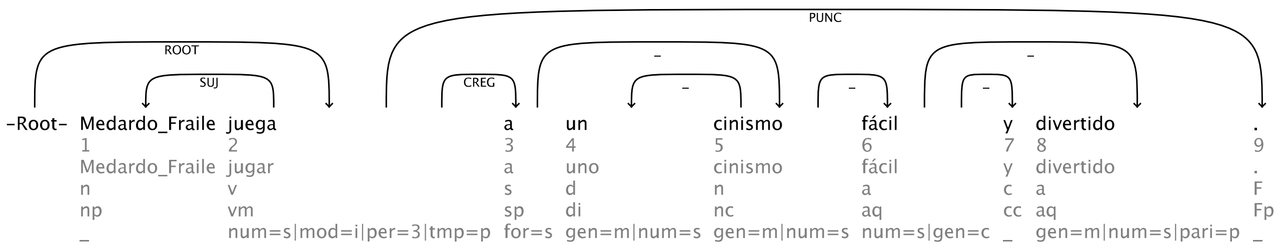
CPOSTAG Coarse-grained part-of-speech tag, where tagset depends on the language.

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.

```
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


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

(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]

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 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'.

For the shared task at the conference, each treebank was split into a training set and a test set. While both parts contain parse trees, the parse trees of the test sets were not revealed until after the teams had finished development of their systems. The term *gold standard* refers to the parse trees of a test set which the systems are meant to predict.

2.3.2 Evaluation metrics

Systems were evaluated by comparing predicted parse trees to the gold standard, and calculating the *labeled accuracy score* (LAS), i.e. the percentage of tokens in where both the predicted 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].

[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.

2.3.3 CoNLL results

Scores achieved by the participating systems are shown in table 2.4 on page 21. 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 2.3.

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 errors made:

> [...] 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 [...]

The authors also find differences in accuracy in relation to linguistic categories of tokens (part of speech) and dependency types, but explain most of these differences 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 the *coordinating conjunction as head* (CCH) convention, which MSTParser predicts nearly 20 percentage points more accurately (figure 2.8 on the following page). As briefly described in section 2.2,

	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*
McDonald	90.71	86.82	73.44	82.25	82.55	63.19	*80.83*
Nivre	91.65	87.60	70.30	81.29	84.58	65.68	*80.75*

Table 2.3: Scores for the top two systems: The graph-based MSTParser [McDonald et al., 2006] and the transition-based MaltParser [Nivre, 2003].

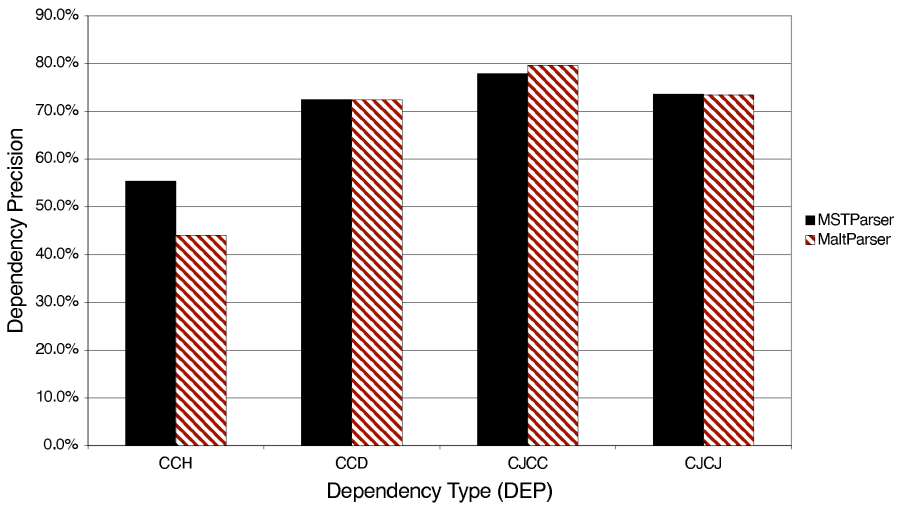


Figure 2.8: Precision across different annotational conventions for coordination [McDonald and Nivre, 2007].

this is the convention where the conjunction itself it head of the coordination, thus somewhat occluding the function of the conjuncts themselves 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.

	Arabic	**Bulgarian**	**Chinese**	**Czech**	**Danish**	**Dutch**	**German**	**Japanese**	**Portuguese**	**Slovene**	**Spanish**	**Swedish**	**Turkish**	*Average*
Canisius	57.64	78.74	78.37	60.92	77.90	74.59	77.56	87.41	77.42	59.19	68.32	79.15	51.07	*70.80*
Attardi	53.81	72.89	54.89	59.76	66.35	58.24	69.77	65.38	75.36	57.19	67.44	68.77	37.80	*61.23*
Wu	63.81	79.73	74.81	59.36	78.38	68.45	76.52	90.11	81.47	67.83	72.99	71.72	55.09	*71.71*
Carreras	60.94	83.30	83.68	68.82	79.74	67.25	82.41	88.13	83.37	68.43	77.16	78.65	58.06	*74.72*
Yuret	52.42	73.49	72.72	51.86	71.56	62.75	63.82	84.35	70.35	55.06	69.63	65.23	60.31	*65.01*
Bick	55.37	79.21	76.18	63.02	74.61	69.51	74.74	84.75	78.18	64.31	71.37	74.09	53.87	*70.00*
Nivre	66.71	87.41	86.92	78.42	84.77	78.59	85.82	91.65	87.60	70.30	81.29	84.58	65.68	*80.19*
Schiehlen	44.39	0.00	66.20	53.34	76.05	72.11	68.73	83.35	71.01	50.72	46.96	71.10	49.81	*62.81*
Ma	50.74	67.64	75.29	58.52	77.70	59.36	68.11	70.84	71.13	57.21	65.08	63.83	41.72	*63.29*
Dreyer	53.37	74.81	71.63	60.54	66.61	61.56	70.97	82.87	75.28	58.73	67.62	67.58	46.05	*65.23*
O'Neil	66.71	85.24	86.70	76.60	82.83	77.51	85.36	90.57	84.69	71.08	79.82	81.78	57.52	*78.43*
Do	60.92	0.00	85.05	72.88	80.60	72.91	84.17	89.07	83.99	69.52	79.72	82.31	60.51	*76.80*
Johansson	64.29	0.00	72.49	71.46	81.54	72.67	80.43	85.63	84.57	66.43	78.16	78.13	63.39	*74.93*
McDonald	66.91	87.57	85.90	80.18	84.79	79.19	87.34	90.71	86.82	73.44	82.25	82.55	63.19	*80.27*
Riedel	66.65	0.00	89.96	67.44	83.63	78.59	86.24	90.51	84.43	71.20	77.38	80.66	58.61	*77.94*
Sagae	62.71	0.00	84.73	75.24	81.56	76.61	84.92	90.37	86.01	69.06	77.68	82.00	63.21	*77.84*
Shimizu	62.83	0.00	0.00	0.00	75.81	0.00	0.00	0.00	0.00	64.57	73.17	79.49	54.23	*34.18*
Corston-Oliver	63.53	83.36	79.92	74.48	81.74	71.43	83.47	89.95	84.59	72.42	80.36	79.69	61.74	*76.94*
Cheng	65.19	86.34	84.27	76.24	81.72	71.77	84.11	89.91	85.07	71.42	80.46	81.08	61.22	*77.70*
Average	*59.94*	*79.98*	*78.32*	*67.17*	*78.31*	*70.73*	*78.58*	*85.86*	*80.63*	*65.16*	*73.52*	*76.44*	*55.95*	
Std. dev.	*6.53*	*6.30*	*8.82*	*8.93*	*5.45*	*6.66*	*7.51*	*7.09*	*5.83*	*6.78*	*8.41*	*6.46*	*7.71*	

Table 2.4: Scores (LAS) achieved by each system at CoNLL-X [Buchholz et al., 2006].

Chapter 3

Feature Engineering in Dependency Parsing

In this chapter I will start out by elaborating on the intentions of this project, and then proceed to take a look at similar endeavors of others. Finally, I will present the main work of this project: A number of schemes for augmenting treebanks.

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 exclusively on the strengths of each model? To this end, McDonald and Nivre [2007] simulated an *oracle parser*, which for each sentence chose the most accurate parse of all the CoNLL systems. With the best parses achieved for each sentence, the labeled attachment score rose to 86.9%. This can be taken as a rough[1] upper bound on how much improvement can be achieved by the strengths all models in unison.

In this project, instead of trying to improve parsing models themselves, I will turn attention to the input data for the data-driven systems, namely the 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 quotations and 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 what features of which tokens – and in

[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.

which combinations – to let their parser consider, and partly from researchers wanting to improve performance of existing systems. I will not attempt a systematic and exhaustive overview – instead, I will 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 improving 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 Maximum Spanning Tree parser [...].

However, the chunking is done on the basis of gold standard (manual) annotation. When using a statistical chunker instead, the effect is reduced, likely due to propagation of errors from 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 processing steps and hence using a chunker leads to a more complex layered

> parser architecture. But since the parser itself may have access to the same information that the chunker 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, in order to identify QUOTATIONs and INSERTED CLAUSEs, and find that augmenting the treebank with 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 in parsing of Swedish, identification of subjects and objects is problematic, due to limited case marking and ambiguous word order patterns in Swedish. Referring to a statistical corpus study by 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 ANIMACY 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.

Along similar lines,Bharati et al. [2008] augment an Indian treebank with features to distinguish HUMAN from NONHUMAN and ANIMATE from INANIMATE nominals [2]. They are motivated by the fact that certain Indian verbs only take human subjects, and find that introducing these features in the treebank 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 pragmatic compromise with a fair improvement in accuracy and a much smaller feature space.

3.2.3 Structural features

Another class of features relate to the structure of dependency graphs, most commonly dependency length. Noting that a word's modifiers tend to fall near it in the sentence, Eisner and Smith [2005] experiment with introducing soft and hard constraints on dependency lengths. The distance measure used is simply the number of words between two tokens, but they also note:

> Other distance measures could be substituted or added (following the literature on heavy-shift and sentence comprehension), including the phonological, morphological, syntactic, or referential (given/new) complexity of the intervening material.

The distance feature applies to dependencies rather than tokens. As such, it must be introduced in the parsing process by the parser itself, and cannot be regarded as a treebank augmentation. I am mentioning it here just to point out that an interesting class feature engineering lies beyond pure treebank augmentation.

[2]Yielding a total of three classes, as the combination of INANIMATE and HUMAN is impossible

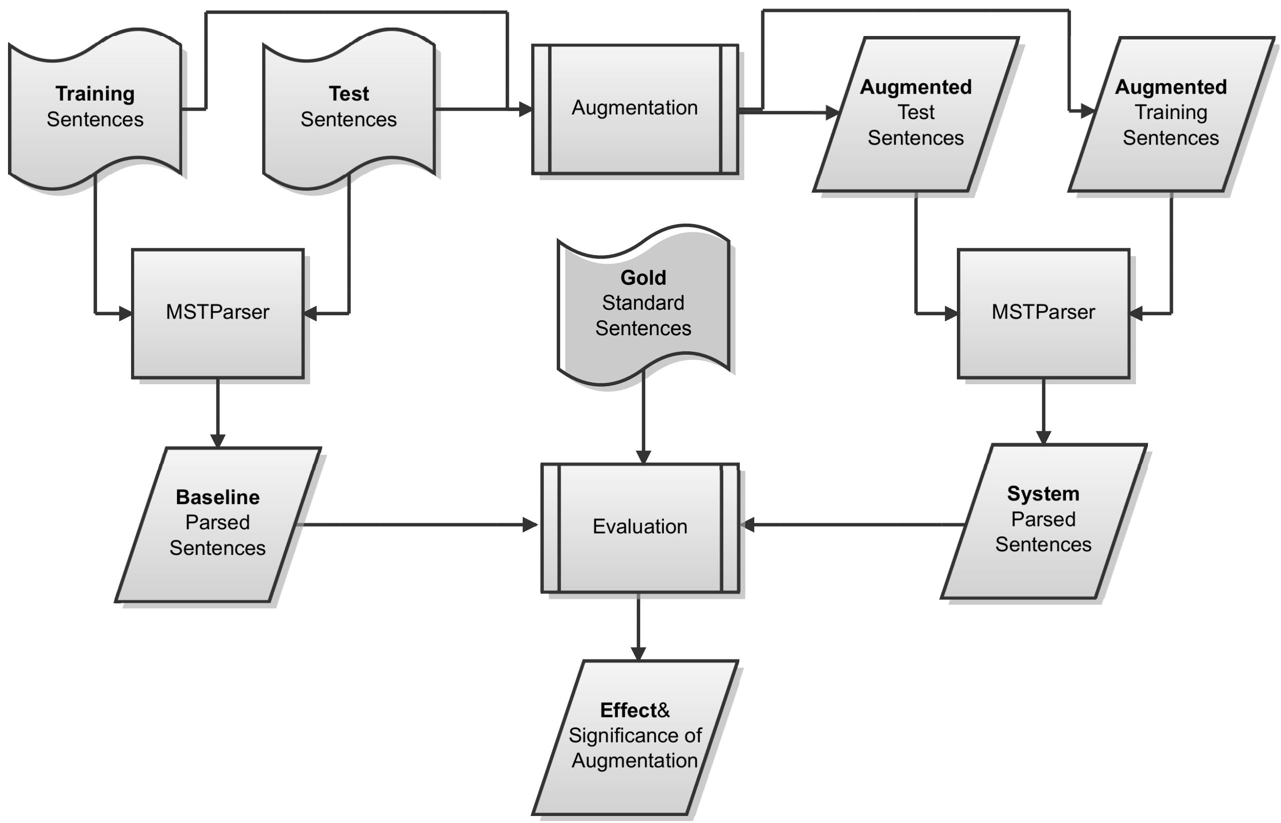

Training Sentences
Test Sentences
Augmentation
Augmented Test Sentences
Augmented Training Sentences
MSTParser
Gold Standard Sentences
MSTParser
Baseline Parsed Sentences
Evaluation
System Parsed Sentences
Effect& Significance of Augmentation

Figure 3.1: Overview of experiment setup.

3.3 Experiment design

In the following I will describe how I intend to augment treebanks with new features, and how I will 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.

3.3.1 Choice of parser

Two systems gave the best results in the CoNLL shared task: The graph-based MSTParser and the transition-based MaltParser. Of all the systems, it would be most satisfactory to improve the performance of one of these two. Most of the proposed feature designs involve longer dependencies, which the graph-based of the two systems appears to handle better. Therefore, I am focusing on this system. However, the case for the MSTParser in relation to this work is not empirically founded, and it would be natural to conduct the same experiments with the MaltParser in future work.

3.3.2 Experiment setup

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

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[3].

3.3.3 Treebank augmentation

In augmenting a treebank, new features may be added to sentence tokens, or existing properties (such as POS tags) may be 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 matching pairs of symmetrical punctuation (e.g. quotation marks). 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.

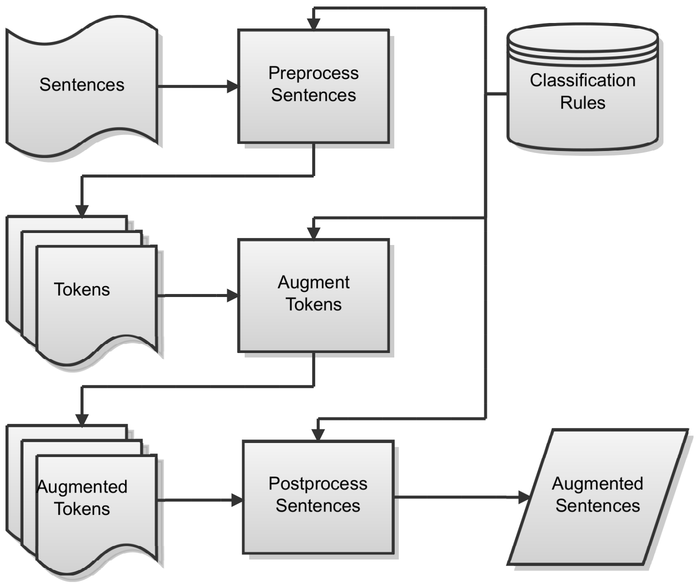


Figure 3.2: Augmentation process for a treebank.

3.3.4 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[4] `eval.pl` is used. For each parser in turn, its predicted parses of the test set are compared to the gold standard (correct) parses given in the treebank. The obtained labeled accuracy score (LAS) for the system is compared to the score of the baseline parser, resulting in a measure of the overall effect on parsing accuracy of adding the feature in question.

3.3.4.1 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 tokens *affected* by the augmentation? And also interesting: What effect does it have on the remaining, *unaffected* tokens in the treebank?

With only a part of the tokens affected by the augmention, one option is to count dependency errors for these 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

[3]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 being a common choice). 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.

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

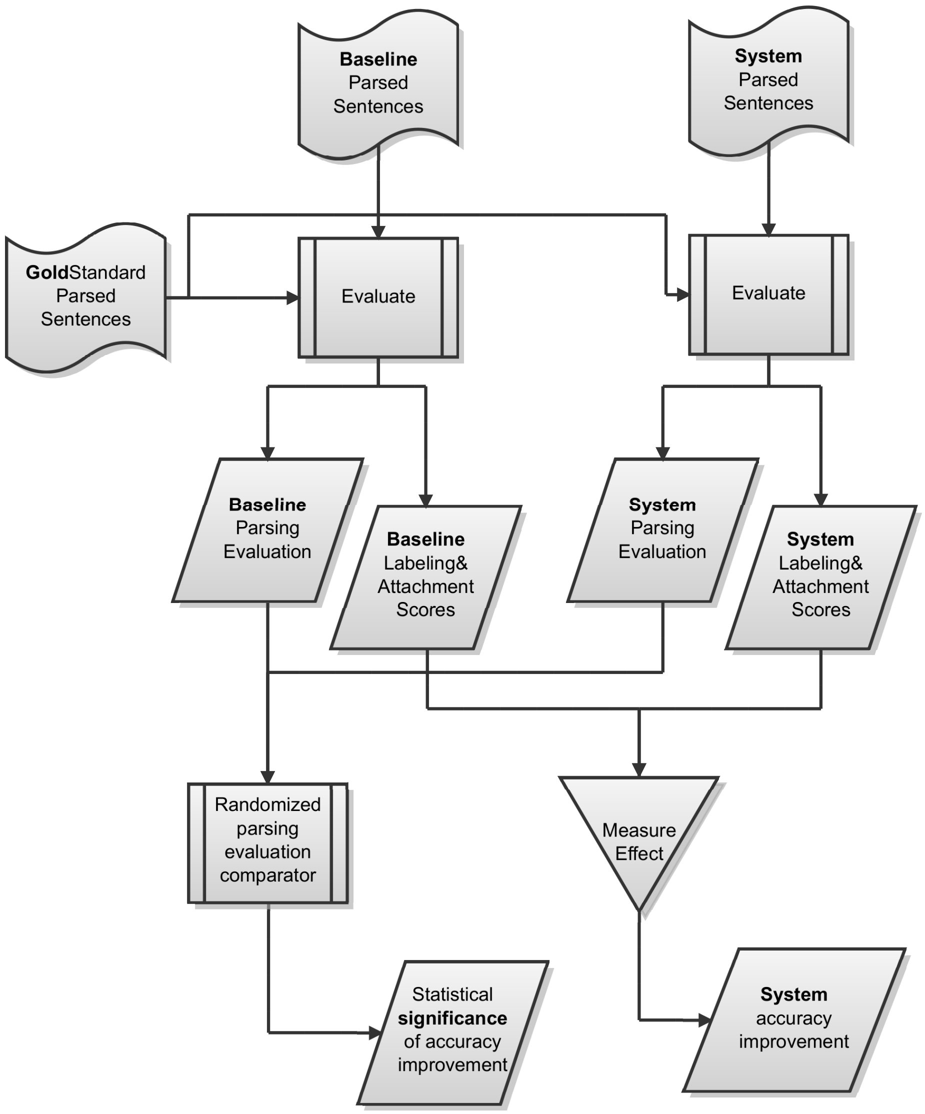

Baseline Parsed Sentences
System Parsed Sentences
GoldStandard Parsed Sentences
Evaluate
Evaluate
Baseline Parsing Evaluation
Baseline Labeling& Attachment Scores
System Parsing Evaluation
System Labeling& Attachment Scores
Randomized parsing evaluation comparator
Measure Effect
Statistical **significance** of accuracy improvement
System accuracy improvement

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.

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, because 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 evaluation with token- or dependency-level categorization, I have chosen to categorize on the *sentence-level* instead when estimating effects of a feature on targeted versus untargeted parts of the text. I thus categorize a given sentence by whether the added feature applied to *any* token of the sentence. I then evaluate each category separately, yielding two ratios, namely one for *affected sentences* (in which augmented tokens occur), and one for *unaffected 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 linguistic 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?

3.3.4.2 Punctuation

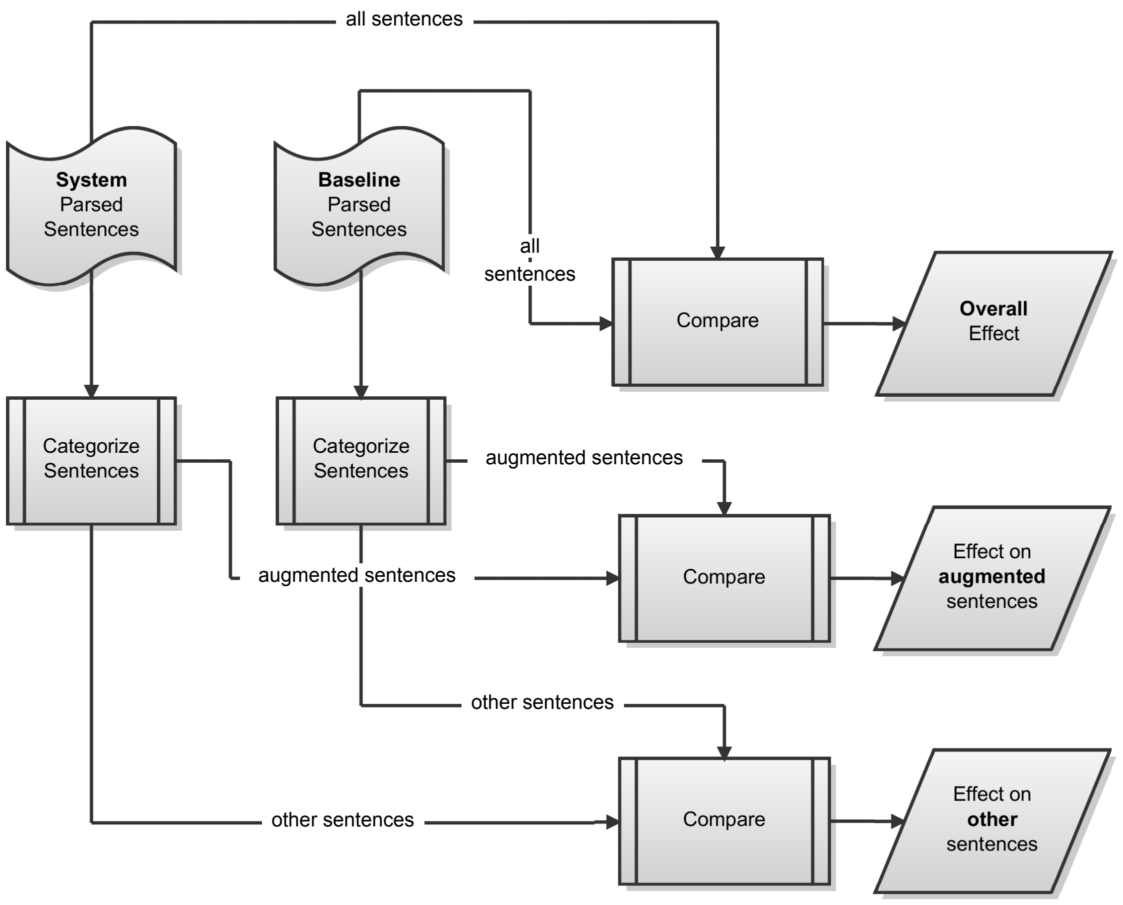
Contrary to the convention of the CoNLL shared task [Buchholz and Marsi, 2006], I am *including* punctuation tokens when evaluating parser accuracy. As several of the phenomena in my experiments involve constructions marked by punctuation (e.g. quotation marks and parentheses), I expect the inclusion of the punctuation tokens themselves to contribute a valuable indication of a parser's 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 for punctuation – as was the case at the second of the CoNLL shared tasks in dependency parsing – it would perhaps make sense to exclude punctuation when scoring.

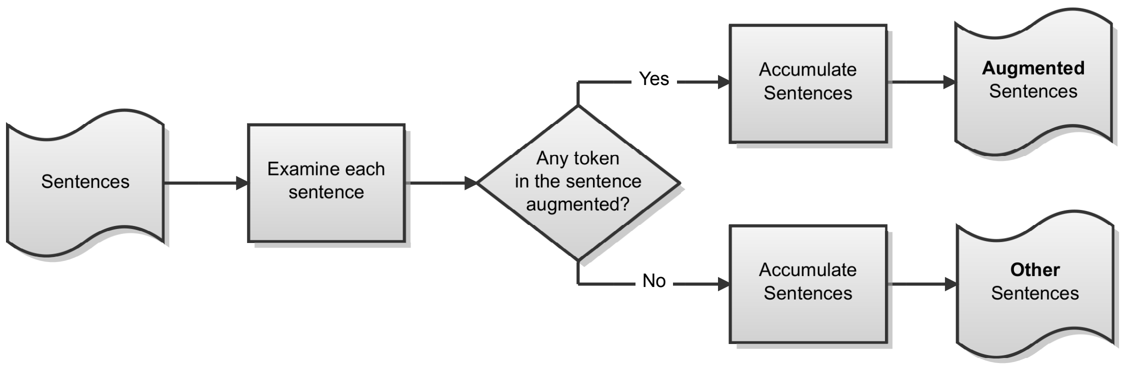
3.3.5 Statistical significance

To quantify the likelihod of observed effects occurring as a result of the augmentation, and not merely by chance, Dan Bikel's "Randomized Parsing Evaluation Comparator" [Bikel, 2004] is used to compute p-values if the differences in parsing accurary. 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



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

> 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.

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.

3.3.6 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. Except for the Italian treebank, they are all identical to those that were used at CoNLL. Some of the treebanks were originally phrase structure treebanks converted that were converted to dependency treebanks, either specifically for CoNLL or for other purporses.

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 Treebank for the BioNLP'09 shared task [Tsujii et al., 2009]

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

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

Japanese Verbmobil [Kawata and Bartels, 2000]

		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.

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]

Several of the treebanks have a size which made the training task 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 in a managable size by truncating the training part to 6000 sentences. When referring to these truncated treebanks, I will use a "(6K)" suffix.

[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 $\times$ 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.

3.4 Feature designs

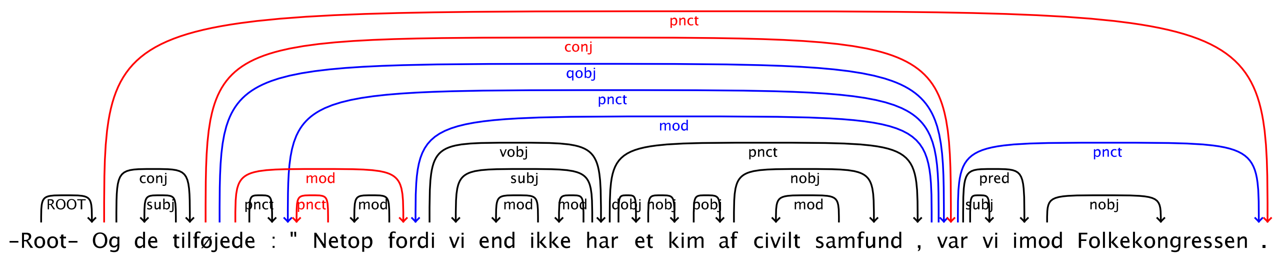
Acting on the hunch that problematic constructions will involve punctuation, a coarse grained survey of the treebanks with the focus on various punctuation was made (table 3.2) in order to quantify the suspicion. Taking for example quotation marks, the survey gives an indication of how widespread quotations are in each treebank, and how well sentences with quotations were parsed in comparison to the treebank average.

3.4.1 Quotation

The prelimenary survey (table 3.2) 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 baseline parses indicate that errors often involve dependencies erroneously crossing the boundary of a quoted phrase, as well as incorrect attachment of quotation marks themselves (an example baseline parse is shown on figure 3.5 on the next page).

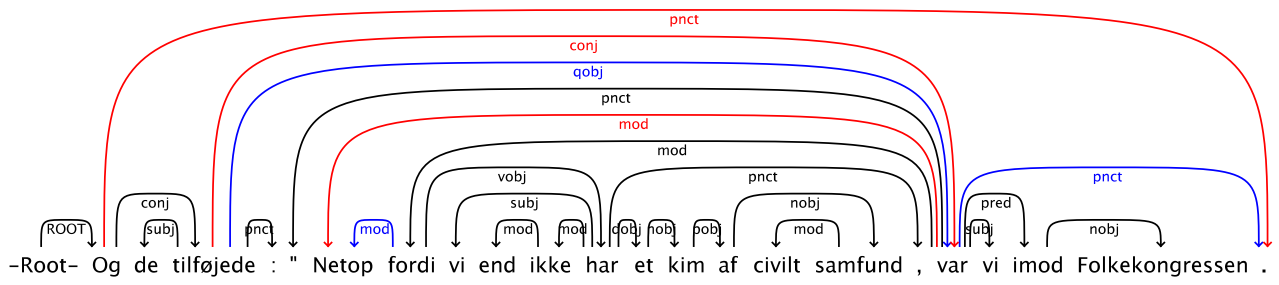
Intuitively, it seems reasonable that quotations require special treatment. Quoted phrases are often complete in themselves: Frequently containing a subject and predicate, conveying a statement, question or exclamation, and sometimes comprising a main clause and one or more subordinate clauses.

From a linguistic viewpoint, quoted phrases exhibit special behavior in some languages. Apart from having an internal structure on their own right,Haberland



-Root- Og de tilføjede : " Netop fordi vi end ikke har et kim af civilt samfund , var vi imod Folkekongressen .
pnct
conj
qobj
pnct
mod
vobj
pnct
subj
nobj
pred
pnct
ROOT
conj
subj
pnct
mod
mod
mod
mod
dobj
nobj
pobj
mod
subj
nobj

(a) Baseline parse



-Root- Og de tilføjede : " Netop fordi vi end ikke har et kim af civilt samfund , var vi imod Folkekongressen .
pnct
conj
qobj
pnct
mod
mod
vobj
pnct
pnct
subj
nobj
pred
ROOT
conj
subj
pnct
mod
mod
mod
dobj
nobj
pobj
mod
subj
nobj

(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].

[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 singled out at the word form level. The 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

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 ▶).

by introducing a separate part of speech tag for these, as well as for tokens in their vicinity. This gave rise to an improvement of 0.50 in LAS ($p < 0.1$) for a truncated part of the Danish treebank.

A 0-1-2 window was used when modifying the POS tags, meaning that the POS tag was changed for the quotation mark itself as well the two tokens following it. Such a window will inevitably mark some tokens immediately following a quoted phrase, as well as tokes occuring in the beginning of one, so this augmentation does not clearly indicate which tokens are actually quoted. Intuitively, a clear indication would be valuable information to the parser.

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. Determining which tokens to augment 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 figure 3.6 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.

3. *Incontinuity.* In at least one case, the a sentence in the Danish treebank with an open-ended quotation was followed by an unrelated sentence[7], which did not continue the quotation.

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:

[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.

[7]This incontinuity is entirely expectable: Treebanks sample sentences from a variety of sources, and treebank authors inevitably have to "cut" somewhere. Additional cuts are made when introducing train/test splits. The bottom line is that intra-sentential continuity is not a valid assumption.

- First parts of multi-sentence quotations, when introduced with a colon preceeding a quotation mark. This 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."[8]

- 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 scheme yields a smaller but still significant ($p < 0.007$) overall improvement of 0.67 on the Danish corpus. The QuotationMark scheme is similar to that of Søgaard, although not identical[9].

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 perhaps 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 for the remaining treebanks were not obtained. Conducting experiments is a computationally intensive task[10], and it was deemed futile to conduct experiments on treebanks containing very few quotations. This principle also applies to the following experiments.

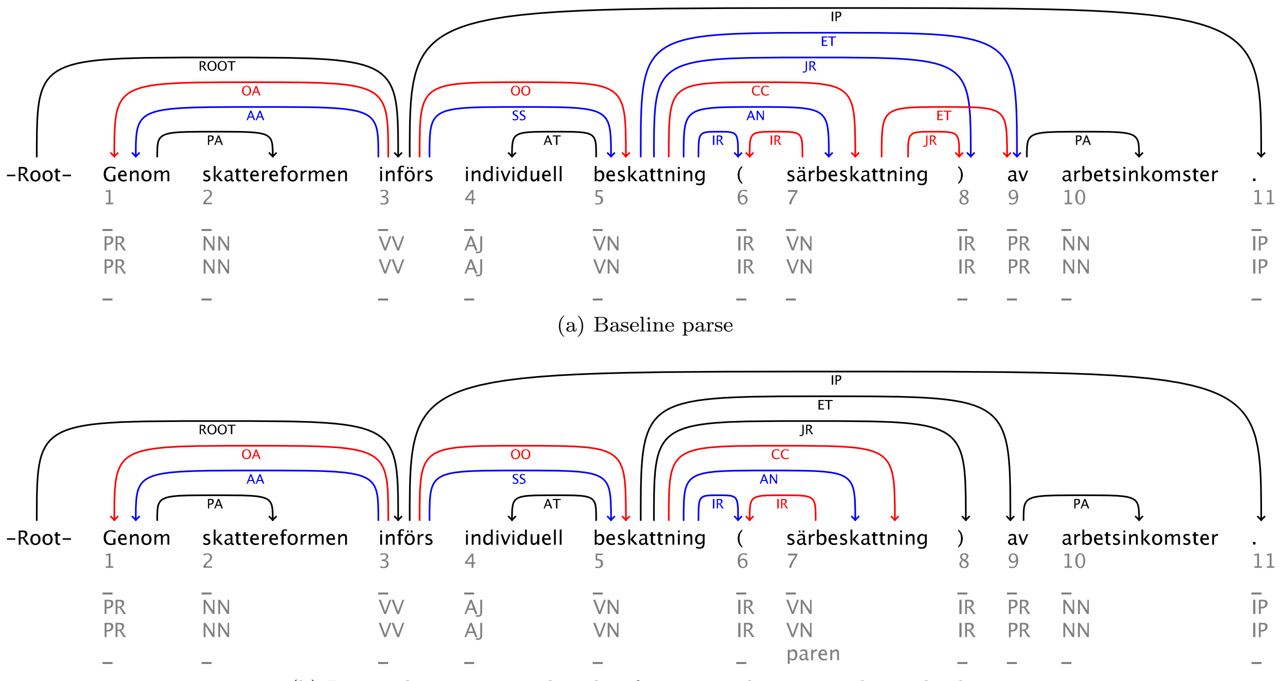
[8]With the proposed augmentation scheme, a quote feature is added to the tokens marked by *italics*, i.e. only the first of the quoted sentences, which in the treebank would occur as part of the outer (quoting) sentence.

[9]In addition to this, Søgaard used a truncated treebank for training, so the results are not comparable.

[10]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.

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.



(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.

3.4.2 Parenthesis

The prelimenary survey of precision for sentences with various punctuation (table 3.2 on page 34) 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 seems to suffer (5.36 worse), and the same 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[11] Italian treebank in a trial

[11] As noted on page 28, it is not really methodically sound to base experiments on a small treebank with a fixed train/test split. The smaller the treebank, the greater the variance of results obtainable with the 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.

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*

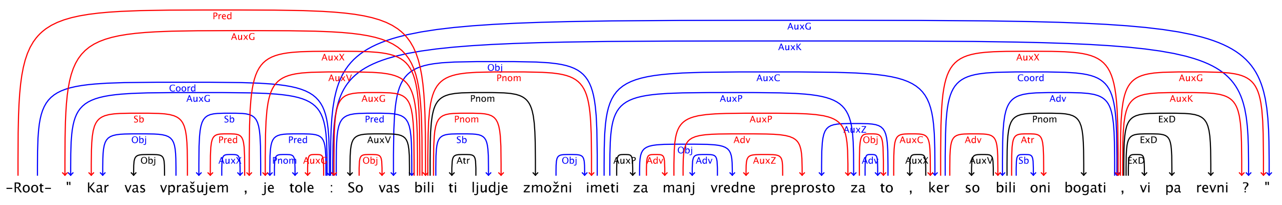
Table 3.4: Results for the Parenthesis augmentation scheme, which marks parenthesized tokens with the feature paren. *Newspaper* and *Law* are the Italian treebanks, and *Both* are the two concatenated.

where *Newspaper* and *Law* are concatenated to form a single treebank (*Both*).

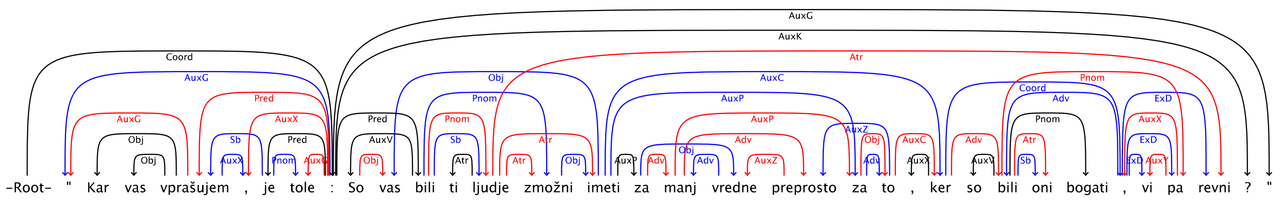
Augmenting parenthesized tokens with a paren feature yields a modest, yet statistically insignificant[12], improvement in overall labeled attachment score for both Italian treebanks, and for the combined treebank. For the *Newspaper* treebank, the augmentation has a positive effect on the sentences with 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 sentences *without* parentheses, but the effect on the augmented sentences is next to none. This absence of effect 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 is practically no overall effect of the augmentation, 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.

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


-Root- " Kar vas vprašujem , je tole : So vas bili ti ljudje zmožni imeti za manj vredne preprosto za to , ker so bili oni bogati , vi pa revni ? "

(a) Baseline parse


-Root- " Kar vas vprašujem , je tole : So vas bili ti ljudje zmožni imeti za manj vredne preprosto za to , ker so bili oni bogati , vi pa revni ? "

(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 right, is introduced with colon. Again, black arcs were correctly predicted by the system, red arcs erroneously predicted, and blue arcs should have been predicted, but were not. Comparing the system parse to the baseline, the separation 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.

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. These parentheticals may be complete sentences on their own right, and as such, should most 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.

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

> [...] lön, pension, livränta, undantagsförmåner och övrig tjänsteinkomst (undantag: *periodiskt understöd eller därmed jämförlig periodisk inkomst*), inkomst av jordbruksfastighet - om den skattskyldige arbetat i jordbruket i ej blott ringa omfattning [...]

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*.

In this particular example, only tokens within the parenthesis should be considered. Tokens thus marked by the scheme are shown in *italics*.

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 test part of the Italian treebank, they occur exclusively in a sentence final position. 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 cases in training corpus where the augmentation applied[13].

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

[13]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.

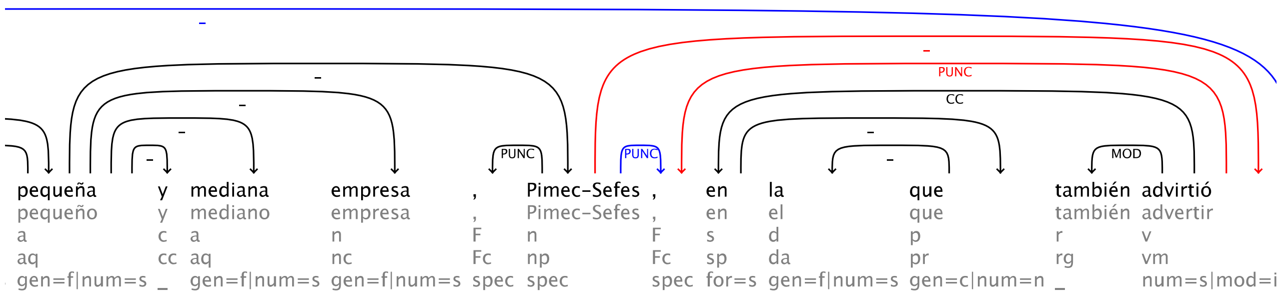

-
-
PUNC
CC
-
-
-
-
PUNC
PUNC
-
MOD
pequeña y mediana empresa , Pimec-Sefes , en la que también advirtió
pequeño y mediano empresa , Pimec-Sefes , en el que también advertir
a c a n , n , s d p r v
aq cc aq nc F np F sp da pr rg vm
gen=f|num=s _ gen=f|num=s gen=f|num=s Fc spec Fc for=s gen=f|num=s gen=c|num=n _ num=s|mod=i
spec spec

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

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 - .
```

Being syntactically parallel constructions, appositions involve some of the same difficulties as other syntactically parallel constructions (e.g. coordinations), 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[14] tokens, that 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 on the next page also shows the

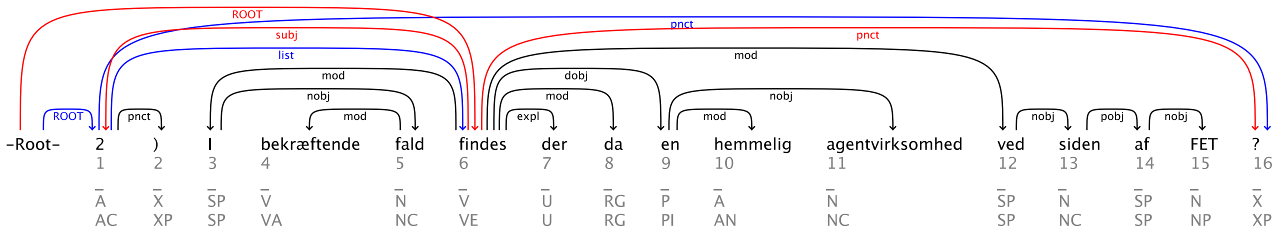
[14]In the case of the Spanish corpus, tokens with a prefix of "N" in the part of speech tag are taken to be nominal.

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

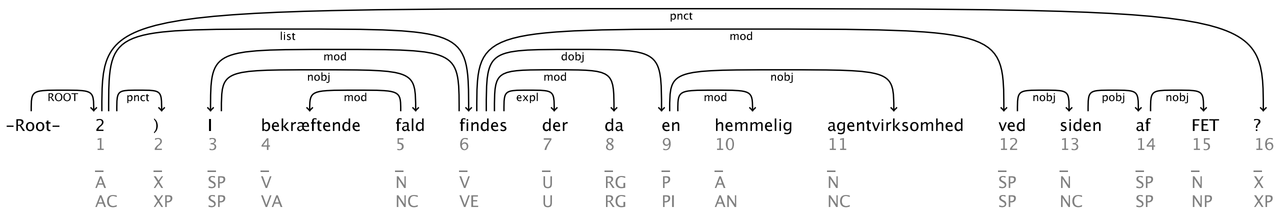
Table 3.6: Results for the augmentation schemes for appositions.

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 certainly too simplistic, 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. For these treebanks, deriving training data from the dependency structure and training an HMM- or CRF-based tagger [Jurafsky and Martin, 2008] to more reliably identify appositions in unparsed text seems feasible.



(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.

3.4.5 List items

Another challenge to the baseline parser is posed by items of numbered or bulleted lists. In particular, 3% of the test part of the Danish Dependency Treebank are sentences of the following kind:

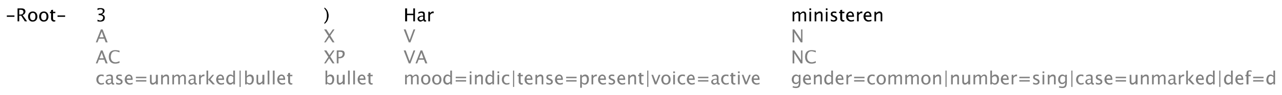
> 3) Har ministeren løjet om sit og regeringens kendskab til virksomheden?

Two-thirds of these 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 verb) being a dependant of the list counter. The baseline parser rarely gets this right (an example of baseline error is shown in figure 3.10).

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). A significant ($p <$ 0.03) improvement in LAS is seen for sentences with numerical list counters,



-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.

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[15], and this result could be construed as helping the parser discriminate these use cases.

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).

3.4.6 Nominal enumerations

When mentioning a number of things one by one, a coordinating conjunction (e.g. *and*) would normally be used before the last thing mentioned, but left out before this (and replaced by a comma instead in writing). Depending on the annotational conventions of a treebank, enumerations can have different dependency structure. In the case of the Bulgarian treebank[16], the initial conjunct acts as head for every other tokens in the coordination. The baseline parser rarely gets this right. Instead, it seems to build locally rooted subtrees and mix dependency labels up (figure 3.13a). Intuitively, this is because the parser "looses sight" of the extent of the coordination.

The commas involved are simply annotated as PUNCT, although they are functionally quite different from clause-separating commas.

Two augmentation schemes are proposed: One (ENUMERATION) to indicate the extent of coordinations by introducing an ENUM feature on all tokens participating in an enumeration, and one (POSENUMERATION) to clarify the function of the commas involved, by simply changing their part of speech tags to that of the final coordinating conjunction (e.g. *and*), as illustrated in figure 3.12.

As with previous schemes, it can be non-trivial to identify enumerations for augmentation in a reliable way. A sequence of commas followed by a coordinating conjunction is to be expected, but the conjuncts themselves can comprise multiple tokens and belong to numerous parts of speech, let alone be small nonfinite verbal phrases.

A cautious heuristic is employed, which matches nominal conjuncts consisting of a single token only. In the following example, this heuristic will match only the first of the two emphasized phrases:

> Syncopation is used in many musical styles, if not all, and is fundamental in such styles as *funk, ska, reggae, ragtime, rap, blues, jazz, breakbeat* and often in *dubstep, heavy metal, and classical music.*

The second emphasized phrase is only covered if *heavy metal* is tokenized as a single, composite token. Obviously, as was the case with appositions in section3.4.4, such a heuristic is really too simplistic. However, for enumerations, results with this simple heuristic are quite encouraging:

[16]Note that the Bulgarian treebank is one of the large ones which brought my experimental system to its knees. The treebank used in experiments is a truncated version with only 6000 sentences in the training set (down from 12823 in the original CoNLL setup).

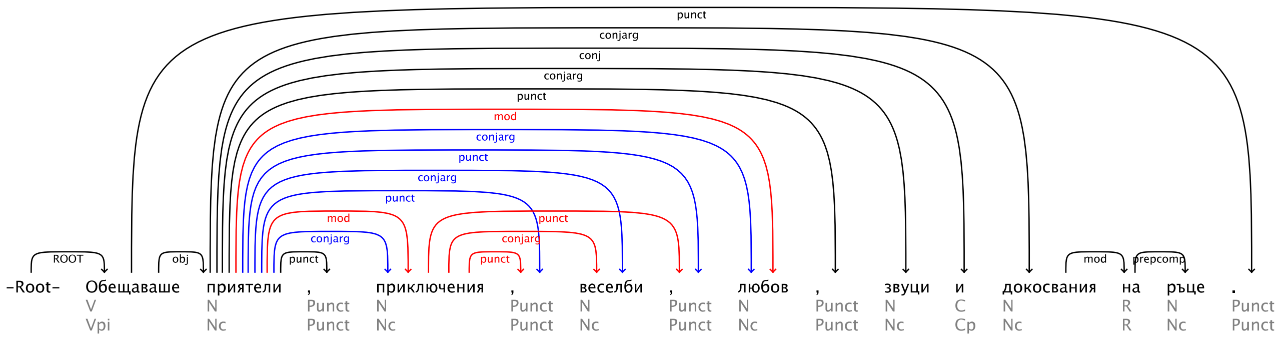
веселби	,	любов	,	звуци	и	докосвания
N	Punct	N	Punct	N	C	N
Nc	Punct	Nc	Punct	Nc	Cp	Nc
gen=f|num=p|def=i|enum	enum	enum	enum	gen=m|num=p|def=i|enum	enum	gen=n|num=p|def=i|enum

(a) The ENUMERATION scheme adds an ENUM feature to each token in the chained coordination.

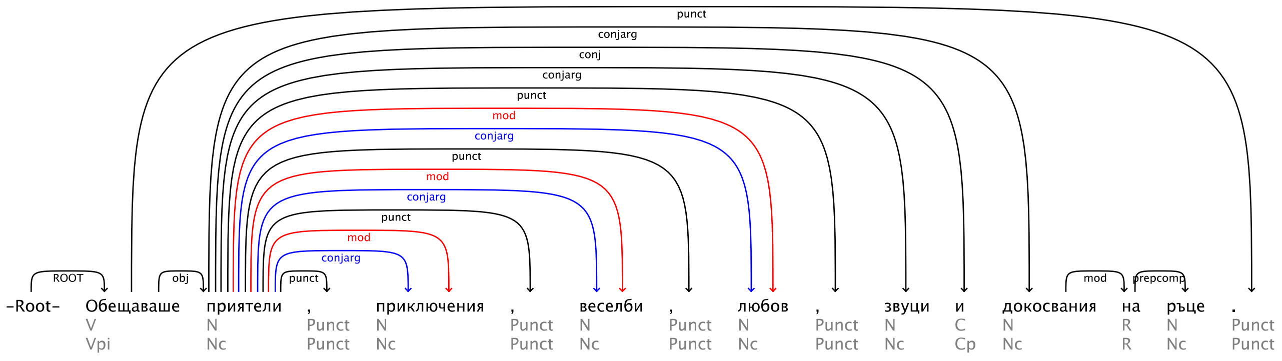
веселби	,	любов	,	звуци	и	докосвания
N	C	N	C	N	C	N
Nc	Cp	Nc	Cp	Nc	Cp	Nc
gen=f|num=p|def=i|enum	enum	enum	enum	gen=m|num=p|def=i|enum	enum	gen=n|num=p|def=i|enum

(b) The POSENUMERATION scheme also changes the POS of commas involved to that of the final coordinating conjunction.

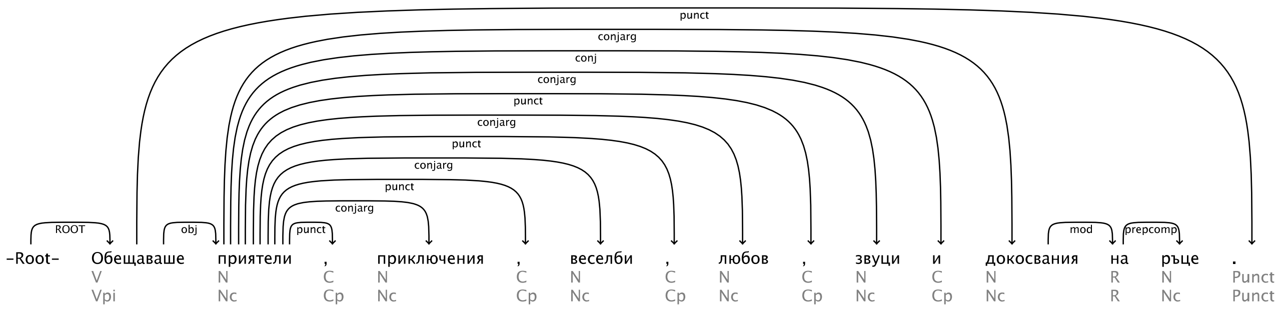
Figure 3.12: Augmentation schemes for chains of enumerated nominals.



(a) Baseline parse. Several dependencies erroneously rooted at the second conjunct.



(b) Parse with the ENUMERATION sheme (tokens in the coordination are augmented with an ENUM feature). Dependency arcs are in order now, but the parser still gets some labels wrong.



(c) POSENUMERATION sheme parse (POS of commas changed to that of the conjunction). Labels are correctly predicted now.

Figure 3.13: Baseline and system parses of an enumeration of nominals. For this particular sentence, the augmentation scheme in subfigure 3.13c (POSENUMERATION) achives a LAS of 100 (perfect parse).

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
Bul (6K)	Enumeration	overall			89.91%	85.24%	90.09%	85.20%	0.18	*-0.04*	0.00%	0.4508
		affected	9	2%	85.16%	80.00%	93.55%	86.45%	8.39	6.45	*32.25%*	*0.0622*
		unaffected	389	98%	90.03%	85.38%	90.00%	85.17%	*-0.03*	*-0.21*	0.00%	0.2590
Bul (6K)	PosEnumeration	overall			89.91%	85.24%	90.12%	85.52%	0.22	0.28	*1.92%*	0.1715
		affected	9	2%	85.16%	80.00%	92.90%	90.97%	7.74	10.97	*54.85%*	*0.0319*
		unaffected	389	98%	90.03%	85.38%	90.05%	85.38%	0.02	0.00	0.00%	0.4909

Table 3.8: Results for the two simple schemes for augmenting enumerations (in the Bulgarian treebank).

Augmenting the (truncated) Bulgarian treebank with the Enumeration scheme results in a LAS gain of 6.45 for sentences with matches enumerations ($p \approx 0.06$), equivalent to a 32.2% reduction in error rate. The PosEnumeration scheme yields an even larger gain (10.97) for the affected sentences ($p < 0.04$), and apparently has no effect on other sentences.

Experiments with a better heuristic (or a probabilistic model) for identifying enumerations is left for future research.

Corpus	Augmentation applied	Sentence focus	Baseline scores		System scores		Improvement		Error reduction	
			UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
DDT	Lemma	overall	88.12%	82.89%	87.37%	82.23%	*-0.75*	*-0.67*	0.00%	*0.0235*
DDT	LemmaOnly	overall	88.12%	82.89%	88.72%	83.68%	0.60	0.79	*4.60%*	*0.0117*

Table 3.9: Results from the LEMMA and LEMMAONLY augmentation schemes. Note that this augmentation scheme applies uniformly to every sentence, so no focusing in on specific parts of the test set is done for this evaluation.

3.4.7 Lemmatization

The CoNLL treebank format provides a column for specifying a token's *lemma*, i.e. the canonical form of a set of wordforms being realizations of the same abstract conceptual form. As such, it can be said that the lemma uniquely specifcies the conceptual identity of a token, regardless of inflection and other contextual influence.

Presumably, language users identify token lemmas when interpreting an utterance. *Laughing* refers to the same sort of act as *laugh* or *laughed*, regardless of inflection, and the very identity of this act (the laugh *lemma*) imposes restrictions on plausible interpretations in the context of a sentence. It rarely makes sense for *cheese* to laugh, and this insight discredits an analysis attaching *cheese* to *laughs* as subject.

Lemma restrictions apply to realizations of the lemma (word forms) as well, but focusing on the lemmas allow for further generalization – resulting in a *simpler yet sufficient* model. If the principle of parsimony[17] applies to parser modelling, this would be also a *better* model, if one were to apply Occam's Razor.

At least in the edition that was used at CoNLL, the Danish Dependency Treebank does not contain lemmas. With the hypothesis that lemmas are valuable for data-driven parsing, I augmented the treebank with lemmas obtained with the CST Lemmatizer [Jongejan and Haltrup, 2008]. To my great surprise, the LEMMA augmentation had a rather strong *adverse* effect on parsing accuracy (-0.67), and significantly so ($p < 0.03$).

However, a second augmentation scheme (LEMMAONLY), which simultaneously *strips* the word forms away, thus leaving only the lemmas in the treebank, yields a substantial (0.79) and significant ($p < 0.02$) *improvement* in parsing accuracy (see table 3.9).

The adverse effect of introducing lemmas in the first place was rather surprising, but the significant gain in LAS achieved by simultanously removing word forms is consistent with the hypothesis the lemmas provide a valuable opportunity for further generalization in the parsing model. Note that virtually no information is lost by stripping word forms, as all morphological information manifest in these surface forms is thoroughly specified in the *features* column of DDT.

[17]The principle that things are usually connected or behave in the simplest or most economical way.

Corpus	Augmentation	Sentence focus	Sentences		Baseline		System		Difference		Effect	
					UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
DDT	Animacy	overall			88.12%	82.89%	88.35%	83.15%	0.22	0.26	*1.50%*	0.2098
		affected	83	26%	87.27%	81.60%	86.92%	81.50%	*-0.35*	*-0.10*	0.00%	0.4329
		unaffected	239	74%	88.57%	83.56%	89.08%	84.00%	0.51	0.44	*2.68%*	*0.1331*

Table 3.10: Results for ANIMACY augmentation.

3.4.8 Animacy

In parsing the Swedish treebank, Øvrelid and Nivre [2007] report of positive effects of including a number of linguistically motivated features hypothesized to target certain consistent errors in dependency assignments (stemming from word order ambiguity and lack of case marking in Swedish). These features include one denoting the *animacy* class of nouns, as well as *case* and *definiteness*. It is demonstrated that the animacy feature can be obtained for common nouns by means of a classifier trained on distributional features of a parsed corpus. The feature space, which includes a number of syntactic and morphological features, enables animacy classification of common nouns in general with a pretty good accuracy [Øvrelid , 2008b].

Given the similarities of Swedish and Danish (lack of case marking, ambiguity in word order), it is natural to assume that parsing of the Danish treebank may benefit from a similar augmentation. To our luck, Øvrelid [2009] reports of successfully applying the Swedish animacy tagger to Danish nouns, by mapping the Danish nouns into the same space of distributional features used for Swedish.

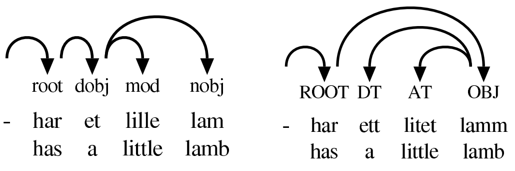
In her experiment of porting the classifier, 18,240 Danish noun lemmas was extracted from the Danish corpus Korpus2000[18] (using a frequency threshold of 10 to ensure sufficient distributional data). Distributional features for the lemmas were obtained by parsing Korpus2000 with a dependency parser trained on the Danish Dependency Treebank. Subsequently, the lemmas were classified using the Swedish animacy classifier.

The list of Danish noun lemmas and corresponding animacy classes (as kindly provided by Øvrelid) enables augmentation schemes for marking animacy of nouns and pronouns in the Danish Dependency Treebank. The list provides animacy classes on the lemma-level, so the augmentation is preceded by an invokation of the lemmatizer from section 3.4.7. For each noun and pronoun in DDT, the animacy class is looked up and inserted as a feature. Finally, lemmas are removed again (to prevent lemmatization of influencing the result).

Thus augmented with ANIMACY, a moderate overall improvement in LAS of 0.26 is achieved for DDT (table 3.10), but with a p-value of about 0.21, the observed effect does not qualify as statistically significant.

The low significance (and relatively small effect) may in part be a result of the different head status that functional categories have in DDT. As observed by Øvrelid [2009], the DDT convention is to let the determiner of a noun phrase act as head, while having the noun itself as a dependent of the determiner. In the Swedish treebank, it goes the other way around, so nouns act as heads (see figure 3.14 on the following page). Intuitively, with the animacy feature

[18]http://korpus.dsl.dk/korpus2000/



(a) DDT (b) Talbanken05

Figure 3.14: 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].

residing on the noun token, it occurs to me that this information is somewhat "stashed away" in DDT, at least for purposes of verb attachment. Information about animacy may be significant for determining the syntactic function of a noun phrase, but in DDT the information is not available on the phrase head, whereas it would be in Talbanken05, because here the noun acts as head.

A factor that which is certainly reducing the observed effect is the relatively low coverage of animacy tagging obtained for Danish. As I am using a fixed word list (rather than an actual tagger), many lemmas (about 45%) are simply not found in the lookup list.

Additionally, there is probably some error propagation from the Danish animacy classification, which is not nearly as accurate as the Swedish. In particular, Øvrelid reports a precision of 74.5 and recall of 45.5 (F-score 56.5).

Finally, in the animacy list I was able to obtain, many special and accented letters (including the Danish vowels *æ*, *ø* and *å*) had been stripped[19]. Accordingly, these were ignored in lookup, thus further lowering the coverage.

In future research, it would be interesting to replace the list-based animacy classification with a proper classifier, as in the Swedish case. One approach is to map the danish lemmas into the feature space used by Øvrelid and Nivre [2007] and then use the Swedish classifier. Another approach is to traini an animacy tagger specifically for Danish. For the latter approach, animacy categories for a training corpus could possibly be obtained by looking for hypernyms with a known animacy class in the evolving Danish wordnet project [Pedersen et al., 2006].

[19]To be specific, all non-ASCII characters had been replaced by the byte sequence "EF BF BD", corresponding to the UTF-8 Replacement Character, which according to http://www.fileformat.info/info/unicode/char/fffd/ is "used to replace an incoming character whose value is unknown or unrepresentable in Unicode".

Corpus	Augmentation	Baseline		System		Difference		Effect	
		UAS	LAS	UAS	LAS	Δ UAS	Δ LAS	LA	p-value
DDT	Bullet	88.12%	82.89%	88.55%	83.58%	0.43	0.69	*4.01%*	*0.0094*
DDT	QuotedPhrase	88.12%	82.89%	88.88%	83.65%	0.75	0.75	*4.40%*	*0.0014*
DDT	QuotationMark	88.12%	82.89%	88.55%	83.56%	0.43	0.67	*3.90%*	*0.0066*
DDT	QuotedPhrase + QuotationMark	88.12%	82.89%	88.74%	83.56%	0.62	0.67	*3.90%*	*0.0125*
DDT	Bullet + QuotedPhrase + LemmaOnly	88.12%	82.89%	88.72%	83.77%	0.60	0.87	*5.09%*	*0.0068*

Table 3.11: Accumulative effects of selected augmentation scheme combinations.

3.5 Feature combinations

When several different augmentation schemes give promising results for a given language, it begs the question: What will the combined effect of applying multiple schemes simultaneously be? Or, in other words, will we see a still greater improvement for each added scheme? And what can be said of the way effects combine – do they simply add up?

Due to the time-consuming nature of the experiments (all involve training a parser from scratch), the effects of feature combinations were not extensively examined in this study. However, I did carry out a few prelimenary experiments of combining some of the more promising augmentation schemes for the Danish treebank. The results (table 3.11) provide some indications which might be telling of what can be expected of further combinations.

In particular, in the combinations tested, gains of the different schemes did accumulate, but the combined effects were lower than simply a sum of the gains of individual schemes.

When combining schemes targeted at the *same* phenomena, the accumulative effects are more dubious. In particular, the combined effect of both QuotationMark and QuotedPhrase for Danish is *lower* than of each schema by itself.

A concise account of the effects of combining augmentation schemes can probably not be given. It depends entirely on the nature of the augmentations in question. Furthermore, the possibility of a synergistic case where the combined effect is *greater* than the sum of the particular effects alone should not be ruled out. Imagine for example that adding a gender feature on nouns and verbs yield moderate effects individually. The two in combination, however, would allow the parser to model agreement in gender and likely yield a much greater effect.

3.6 Unreported features

A number of other augmentation schemes were explored, including features indicating capitalization, verb-counting features and features with n-gram suffixes of word forms. These schemes did not yield convincing results, and are not reported.

Chapter 4

Conclusions

I set out to examine the possibility of enhancing state of the art data-driven dependency parsing systems by augmenting treebanks. After brief overview of dependency grammar and dependency parsing, a closer look was taken on two data-driven approaches to the parsing problem. MSTParser, a model of the graph-based approach, was chosen for augmentation experiments.

A litterature review revealed that the idea of augmenting treebanks is not entirely new. Nonetheless, in this project, several novel augmentation schemes were proposed and tested, several of which were found to significantly improveme parsing accuracy of relevant treebanks. Additionally, some previously proposed augmentations (or variations hereof) were tested on new languages. To quantify effects of treebank augmentation, an evaluation method involving sentence-level categorization of treebank test sets was proposed, and employed in evaluating feature designs.

In the following, I will briefly summarize the findings of the work with applying proposed augmentation schemes to the various treebanks.

4.1 Results

Two augmentation schemes for quotations were evaluated: QuotedPhrase, which focuses on the tokens that are quoted, and QuotationMark, which instead modifies the punctuation tokens indicating quotations. Both were found to significantly improve parsing accuracy of the Danish Dependency Treebank, with the former showing the strongest effect. QuotedPhrase had a significant effect for Slovene as well, but the improvements for Bulgarian and Italian were insignificant.

Enhancements in parsing accuracy from the Parenthesis scheme were observed for Swedish and Italian, but were not statistically significant. The same goes for the Colon scheme, aimed at secondary phrases occurring after a colon.

A scheme targeting nominal appositions (CommaApposition) was found to improve parsing of Spanish sentences with appositions, but the observed effect was bordering on the 0.05-level of statistical significance. The approach seems promising, but could benefit from a better method of identifying appositions for augmentation.

Parsing of the Danish treebank was significantly improved by an augmentation scheme targeting list items arranged as sentences with prefixed list bullets (BULLET).

The parser coped with intra-sentencial enumerations of nominals significantly better when these were marked as such, and the commas separating list items were given a conjunctional part of speech (POSENUMERATION).

Introducing lemmas in the Danish treebank had an *adverse effect* on accuracy (LEMMA), but if word forms were simultaneously stripped off (LEMMAONLY), parsing accuracy increased significantly over the baseline.

An attempt of augmenting nominals in the Danish treebank with ported animacy catetegories resulted in a small but insignificant improvement, most likely due to a number of weaknesses in the chain of procedures for obtained the animacy class.

4.2 Considerations of applicability

While the treebank augmentations explored clearly have potential, they also come with apparent problems. It might be objected that the rules driving the augmentations are often language specific, or more precisely, highly specific to the punctuational and annotational conventions in use in the treebank, as well as being specific to the kind of text in the treebank. An augmentation for quotations for example must be able to handle the quotation style of the text, and will not be of much use when an unexpected quotation style is encountered – or when there are very few or no quotations at all, for that matter.

That being said, it seems only fair to regard this as a task to be handled by augmentation systems, if they are to be of any value in production grade data-driven systems. In other words, augmentation systems for use in data-driven parsing should be sensitive to and *detect* punctuational and annotational conventions, and be able to *accommodate*. As a trivial example, an augmentation scheme for quotations should detect whether guillemets occur as *« ... »* or as *» ... «* or even as *» ... »* [1].

4.3 Future research

Working on this project gave rise to a number of ideas worth pursuing in possible future research:

Other parsing systems Working exclusively with the graph-based MSTParser really begs the question: How would the proposed augmentation schemes work with a transition-based parser, such as the MaltParser? It would be straight forward to plug the MaltParser into to the experimental pipeline set up for this project, and see how the obtained effects compare.

Optimize parser configurations While the MSTParser was the highest scoring system at CoNLL for several of the languages, the results at the conference were obtained with a version of the MSTParser which predicts unlabeled dependencies only. Dependencies were then labeled using a separate

[1]The second being predominantly a Danish and Slavic convention, and the last being used in Finnish, according to a Wikipedia entry on the matter: http://en.wikipedia.org/wiki/Quotation_mark,_non-English_usage

component. This configuration lead to slightly better results than those obtainable with the publicly available MSTParser. As such, the *baseline* results obtained for this project (using the publicly available MSTParser in standard configuration) are – regrettably – not comparable to those at CoNLL. It could be interesting to use the actual CoNLL setup, so that the results reported at CoNLL could be reproduced, and the baselines of experiments brought on par with the state of the art[2].

Transform syntactic annotation As discussed in section 3.4.8, the Danish Dependency Treebank differs from the Swedish with respect to the head status of so-called functional categories [Øvrelid , 2009]. It could be interesting to transform the said annotation of either treebank into using the convention of the other, and determine the impact, if any, that these annotational conventions have on parsing accuracy (using at least one parser from each of the data-driven paradigms). This would also allow for an attempt to substantiate the intuition that these annotational conventions affect the parser's ability to exploit the animacy feature.

Transform coordination structure Along similar lines, it could prove efficacious to transform coordinations using the *coordinated conjunction as head* (CCH) convention into using one of the other conventions. As discussed in relation to the CoNLL results in section 2.3.3, the CCH convention for coordinations seems to cause difficulties in parsing. An augmentation scheme for transforming this annotational convention would allow an attempt to substantiate this suspicion without having to compare across treebanks[3].

Extrasentential features Weber and Müller [2004] report that *givenness* affects word order. The notion of givenness is highly related to definiteness, and Øvrelid and Nivre [2007] exploit the readily available definiteness to help disambiguate argument structure in Swedish. Although highly related to definiteness, the givenness essentially reflects the information status of an entity: Has it been introduced previously, or is this the first mention in the discourse? It could be interesting to obtain givenness automatically by examining a window of preceding sentences[4]. One would probably have to employ a WordNet or other lexical database and traverse hyponymy relations in identifying previous mentions of an entity.

Typeface feature*s* *Italics* are often used for emphasis, but also to *mention* a word or a phrase, as opposed to *using* it [Saka, 1998]. Mentioned words do not participate in sentence structure in quite the same way as words that are actually being used. As an example of this, the mentioned verb in "to the editor *explode* sounds exaggerated" should not subcategorize any subject or object. Retaining typeface features such as italization in treebanks could prove useful for data-driven parsing.

[2]For direct comparison to CoNLL results, one would also have to disregard punctuation in the evalutation

[3]Both of the treebank transforming augmentations suggested alter the *dependency structure* of treebanks, and thus presuppose that the same transformation is made on the test part containing the gold standard (correct) parses.

[4]For this to be feasible, one would have to use a treebank where continuity of sentences is a reasonable assumption.

High-level segmentation Several of the proposed augmentation schemes depend on the high-level segmentation implied by the punctuation of a sentence. Symmetrical markers for 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 indicating 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 e.g. parenthetical tokens from being candidates for the sentence head. The segmentation system could be punctuation-based[5], perhaps like the one discussed by Kubon et al. [2006].

Discriminate appositions and parentheticals Pairs of hyphens and commas can introduce appositions as well as parentheticals. While appositions function in parallel to another sentence element, parenthetical remarks exhibit a more varied attachment. The proposed augmentations do not clearly distinguish the two ([6]. It might be feasible to discriminate the different usages of hyphen or comma pairs, thus allowing differentiated treatment by the parser.

Separate parsing Quotations and other parenthethicals which are complete on their own right, might benefit from being parsed *out of context*, i.e. as separate sentences, without interference from (and interfering with) the outer sentence.

Unification hybrid This final idea is rather speculative and not necassarily theoretically sound. In cases where potentially useful features are "out of reach" for the parser (e.g. deep within a subtree), it might be beneficial to expose such features to the parser, possibly through a unification-like process similar to that of LFG [Kaplan and Bresnan, 1982] or HPSG [Sag et al., 1994], in which certain features progagate upward. Take for example a nominal subtree that is governed by a determiner. Depending on the dependency relations of the subtree, it could make sense for the whole subtree to assume e.g. a semantic feature such a animacy or givenness from the noun. Of course, such unification-like processes are not achievable by treebank augmentation. It would require modification of the parsing algorithm, and a small grammar-like set set of rules stating in which cases feature should propagate upward. As such, it borders on a hybrid between a data-driven and a grammar-based system.

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

[6]Although the simple heuristic used for identifying appositions matches single tokens only, thus effectively ruling out most parentheticals.

4.4 Final thoughts

Overall, several of the proposed augmentation schemes were shown to be effective. Occurring as a pre-processing step to training and parsing, the augmentations are highly modular and practically independent of the parsing system used. As such, the proposed augmentations provide an immediate opportunity to increase parsing accuracy of existing parsing systems.

Acknowledgements

I would like to thank Bart Jongejan for kindly granting access to the Danish lemmatizer for duration of this project. This not only led to interesting observations in regard to lemmatization, but also made experiments with Lilja Øvrelids ported animacy classification possible. I am grateful to Lilja for letting me use her ported animacy classification and for providing guidelines for usage. Furthermore, I would like to extend my gratitude to all treebank authors, especially those who made the results their efforts publicly available, and to Sebastian Riedel and Ivan Vladimir for the utility *What's Wrong with my NLP?*, which has been very handy for inspecting parse trees. Finally, a big trank you to Anders Søgaard, who not only gave invaluable guidance, but also provided the research idea and lots of encouraging feedback.

References

Susana Afonso, Eckhard Bick, Renato Haber, and Diana 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.

Nart B. Atalay, Kemal Oflazer, and Bilge 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.

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.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. The pdt: a 3-level annotation scenario. In Anne 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`.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George 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.

Montserrat Civit and M$^{\underline{a}}$ 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, M$^{\underline{a}}$ Antonia Martí, Borja Navarro, Núria Bufi, Belén Fernández, and Raquel Marcos. Issues in the syntactic annotation of cast3lb. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, 2003.

Montserrat Civit, M$^{\underline{a}}$ Antònia 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.

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, and Andreja Žele. Towards a slovene dependency treebank. In *Proc. of the 5th Intern. Conf. on Language Resources and Evaluation (LREC)*, 2006.

Jan 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.

Jan Hajič, Otakar Smrž, Patr Zemánek, Jan Šnaidauf , and Emanuel 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.

Bart Jongejan 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`.

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

R. M Kaplan and J. Bresnan. Lexical functional grammar. *Bresnan, J., editor*, page 173–281, 1982.

Yasuhiro Kawata and Julia 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`.

Bernardo Magnini and Amedeo 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.

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

Joakim 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.

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

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gokhan Tür. Building a turkish treebank. 2003.

Bolette Sandford Pedersen, Sanni Nimb, Jörg Asmussen, and Nicolai Hartvig Sørensen. Dannet - a wordnet project for danish. 2006.

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. ISSN 0033-295X.

I. A Sag, J. Nerbonne, K. Netter, and C. Pollard. Head-driven phrase structure grammar. *Computational Linguistics*, 22(1), 1994.

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`.

Paul Saka. Quotation and the use-mention distinction. *Mind*, 107:425, 1998.

Kiril Simov and Petya 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.

Kiril Simov, Gergana Popova, and Petya Osenova. Hpsg-based syntactic treebank of bulgarian (bultreebank). In Andrew Wilson, Paul Rayson, and Tony McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, page 135–142. Lincom-Europa, Munich, 2002.

Kiril Simov, Petya Osenova, Alexander Simov, and Milen 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.

Otakar Smrž, Jan Šnaidauf , and Patr 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.

Ulf Teleman. *Manual för grammatisk beskrivning av talad och skriven svenska (MAMBA).* Lund, Sweden, 1974.

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

Jun'ichi Tsujii, Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, and Yoshinobu Kano. Bionlp'09 shared task. http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/, 2009. URL `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/`.

Leonoor van der Beek, Gosse Bouma, Jan Daciuk, Tanja Gaustad, Robert Malouf, Gertjan van Noord, Robbert Prins, and Begoña Villada. The alpino dependency treebank. In *Algorithms for Linguistic Processing.* 2002a.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan 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

Example runtime session

What follows here is a few examples to illustrate the usage of the augmentation tools.

A.1 Augmentation

With a treebank laid out in two or three subdirectories containing a training part, a test part and possibly system output, we can apply e.g. the Bullet augmentation to all parts as follows:

```
Christians-MacBook% bin/corpus.rb annotate --kind bullet danish/ddt
     [mkdir] danish/ddt-bullet
  [baseline] danish/ddt/system/out.conll
     [mkdir] danish/ddt-bullet/baseline
   [augment] danish/ddt/system/out.conll (bullet)
      [save] danish/ddt/system/out.conll
              => danish/ddt-bullet/baseline/out.conll
     [mkdir] danish/ddt-bullet/test
   [augment] danish/ddt/test/danish_ddt_test.conll (bullet)
      [save] danish/ddt/test/danish_ddt_test.conll
              => danish/ddt-bullet/test/danish_ddt_test.conll
     [mkdir] danish/ddt-bullet/train
   [augment] danish/ddt/train/danish_ddt_train.conll (bullet)
      [save] danish/ddt/train/danish_ddt_train.conll
              => danish/ddt-bullet/train/danish_ddt_train.conll
```

This results in a new directory (`danish-bullet`) being created, containing augmented copies of the treebank files. Several augmentations can also be applied using the option `--kinds scheme1,scheme2[,...]`.

A.2 Training and testing

We can now train and test a parser on the augmented treebank:

```
Christians-MacBook% bin/experiment/run danish/ddt-bullet
Creating Alphabet ...
[...lots of output...]
```

Depending on the size of the treebank and the feature space, training can take several hours. Parsed sentences are placed in the file `danish-bullet/system/out.conll`.

A.3 Comparison to baseline

Let us compare the performance of the system to the baseline:

```
Christians-MacBook% bin/corpus.rb compare danish/ddt-bullet
[gold] danish/ddt-bullet/test/danish_ddt_test.conll
[eval] danish/ddt-bullet/baseline/out.conll
  Labeled   attachment score: 4851 / 5852 * 100 = 82.89 %
  Unlabeled attachment score: 5157 / 5852 * 100 = 88.12 %
  Label accuracy score:       5269 / 5852 * 100 = 90.04 %
[eval] danish/ddt-bullet/system/out.conll
  Labeled   attachment score: 4891 / 5852 * 100 = 83.58 %
  Unlabeled attachment score: 5182 / 5852 * 100 = 88.55 %
  Label accuracy score:       5284 / 5852 * 100 = 90.29 %
[evalb] danish/ddt-bullet/baseline/out.conll
        => danish/ddt-bullet/baseline/evalb.out
[evalb] danish/ddt-bullet/system/out.conll
        => danish/ddt-bullet/system/evalb.out
[compare] danish/ddt-bullet/baseline/evalb.out danish/ddt-bullet/system/evalb
    .out
model1: recall=89.3285013399299, precision=82.8947812713602
model2: recall=89.7542404416275, precision=83.5782501708817
model2 recall - model1 recall = 0.425739101697587
model2 precision - model1 precision = 0.683468899521529
number of random recall diferences equal to or greater than
        original observed difference: 581
number of random precision diferences equal to or greater than
        original observed difference: 107
p-value for recall diff: 0.0581941805819418
p-value for precision diff: 0.0107989201079892
```

This applies the CoNLL eval script to both the system and the baseline, and uses the produced evalb to estimate statistical significance using Dan Bikel's Randomized Parsing Evaluation Comparator.

A.4 Focused evaluation

If we wish to evaluate performance on only the sentences that were affected by the augmentation, we first need to categorize the baseline, system and gold standard sentences based on the augmentation:

```
Christians-MacBook% bin/corpus.rb categorize --kind bullet danish/ddt-bullet
      [gold] danish/ddt-bullet/test/danish_ddt_test.conll
[categorize] danish/ddt-bullet/baseline/out.conll
     [mkdir] danish/ddt-bullet/baseline/no-bullet
      [part] no-bullet
             => danish/ddt-bullet/baseline/no-bullet/out.conll
     [mkdir] danish/ddt-bullet/baseline/has-bullet
      [part] has-bullet
             => danish/ddt-bullet/baseline/has-bullet/out.conll
[categorize] danish/ddt-bullet/test/danish_ddt_test.conll
     [mkdir] danish/ddt-bullet/test/no-bullet
      [part] no-bullet
             => danish/ddt-bullet/test/no-bullet/danish_ddt_test.conll
     [mkdir] danish/ddt-bullet/test/has-bullet
      [part] has-bullet
             => danish/ddt-bullet/test/has-bullet/danish_ddt_test.conll
[categorize] danish/ddt-bullet/system/out.conll
     [mkdir] danish/ddt-bullet/system/no-bullet
      [part] no-bullet
             => danish/ddt-bullet/system/no-bullet/out.conll
     [mkdir] danish/ddt-bullet/system/has-bullet
      [part] has-bullet
             => danish/ddt-bullet/system/has-bullet/out.conll
```

We can now repeat the comparison procedure for each of the categories *has-bullet* and *no-bullet*.