Natural Language Engineering **13** (2): 95–135. © 2007 Cambridge University Press
doi:10.1017/S1351324906004505 First published online 12 January 2007 Printed in the United Kingdom

MaltParser: A language-independent system for data-driven dependency parsing

JOAKIM NIVRE

Växjö University, School of Mathematics and Systems Engineering, 35195 Växjö, Sweden
Uppsala University, Department of Linguistics and Philology, Box 635, 75126 Uppsala, Sweden
e-mail: joakim.nivre@msi.vxu.se

JOHAN HALL, JENS NILSSON

Växjö University, School of Mathematics and Systems Engineering, 35195 Växjö, Sweden
e-mail: {johan.hall,jens.nilsson}@msi.vxu.se

ATANAS CHANEV

University of Trento, Dept. of Cognitive Sciences, 38068 Rovereto, Italy
ITC-irst, 38055 Povo-Trento, Italy
e-mail: chanev@form.unitn.it

GÜLŞEN ERYİĞİT

Istanbul Technical University, Dept. of Computer Engineering, 34469 Istanbul, Turkey
e-mail: gulsen.cebiroglu@itu.edu.tr

SANDRA KÜBLER

University of Tübingen, Seminar für Sprachwissenschaft, Wilhelmstr. 19, 72074 Tübingen, Germany
e-mail: kuebler@sfs.uni-tuebingen.de

SVETOSLAV MARINOV

University of Skövde, School of Humanities and Informatics, Box 408, 54128 Skövde, Sweden
Göteborg University & GSLT, Faculty of Arts, Box 200, 40530 Göteborg, Sweden
e-mail: svetoslav.marinov@his.se

ERWIN MARSI

Tilburg University, Communication and Cognition, Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: e.c.marsi@uvt.nl

(*Received 16 February 2006; revised 15 August 2006*)

Abstract

Parsing unrestricted text is useful for many language technology applications but requires parsing methods that are both robust and efficient. MaltParser is a language-independent system for data-driven dependency parsing that can be used to induce a parser for a new language from a treebank sample in a simple yet flexible manner. Experimental evaluation confirms that MaltParser can achieve robust, efficient and accurate parsing for a wide range of languages without language-specific enhancements and with rather limited amounts of training data.

1 Introduction

One of the potential advantages of data-driven approaches to natural language processing is that they can be ported to new languages, provided that the necessary

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

linguistic data resources are available. In practice, this advantage can be hard to realize if models are overfitted to a particular language or linguistic annotation scheme. Thus, several studies have reported a substantial increase in error rate when applying state-of-the-art statistical parsers developed for English to other languages, such as Czech (Collins *et al.* 1999), Chinese (Bikel and Chiang 2000; Levy and Manning 2003), German (Dubey and Keller 2003), and Italian (Corazza *et al.* 2004). Another potential obstacle to successful reuse is that data-driven models may require large amounts of annotated training data to give good performance, while for most languages the availability of such resources is relatively limited. This is also a problem when porting parsers to new domains, even for languages where large amounts of annotated data are available (Titov and Henderson 2006). Given that approaches based on completely unsupervised learning are still vastly inferior in terms of accuracy, there is consequently a need for supervised approaches that are resilient against data sparseness.

In this article, we present a data-driven approach to dependency parsing that has been applied to a range of different languages, consistently giving a dependency accuracy in the range 80–90%, usually with less than a 5% increase in error rate compared to state-of-the-art parsers for the language in question. All these results have been obtained without any language-specific enhancements and in most cases with fairly modest data resources.

The methodology is based on three essential techniques:

1. Deterministic parsing algorithms for building dependency graphs (Kudo and Matsumoto 2002; Yamada and Matsumoto 2003; Nivre 2003)
2. History-based feature models for predicting the next parser action (Black *et al.* 1992; Magerman 1995; Ratnaparkhi 1997; Collins 1999)
3. Discriminative machine learning to map histories to parser actions (Veenstra and Daelemans 2000; Kudo and Matsumoto 2002; Yamada and Matsumoto 2003; Nivre *et al.* 2004)

The system uses no grammar but relies completely on inductive learning from treebank data for the analysis of new sentences and on deterministic parsing for disambiguation. This combination of methods guarantees that the parser is both robust, producing a well-formed analysis for every input sentence, and efficient, deriving this analysis in time that is linear or quadratic in the length of the sentence (depending on the particular algorithm used).

This methodology has been implemented in the MaltParser system, which can be applied to a labeled dependency treebank in order to induce a labeled dependency parser for the language represented by the treebank. MaltParser is freely available for research and educational purposes[1] and has been designed primarily as a tool for research on data-driven dependency parsing, allowing users to flexibly combine different parsing algorithms, feature models, and learning algorithms. However, given that the necessary data resources are available, MaltParser can also be used

[1] URL: `http://www.msi.vxu.se/users/nivre/research/MaltParser.html`.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

for rapid development of robust and efficient dependency parsers, which can be used in language technology applications that require parsing of unrestricted text.

In this article, we begin by describing the general methodology of deterministic dependency parsing with history-based feature models and discriminative machine learning (section 2). We then describe the implemented MaltParser system, focusing on its functionality with respect to parsing algorithms, feature models, and learning algorithms (section 3). To support our claims about language-independence and resilience against data sparseness, we then present an experimental evaluation based on data from ten different languages, with treebanks of different sizes and with different annotation schemes (section 4). Finally, we draw some general conclusions and make some suggestions for future work (section 5).

2 Inductive dependency parsing

Mainstream approaches in statistical parsing are based on nondeterministic parsing techniques, usually employing some kind of dynamic programming, in combination with generative probabilistic models that provide an *n*-best ranking of the set of candidate analyses derived by the parser. This methodology is exemplified by the influential parsers of Collins (1997; 1999) and Charniak (2000), among others. The accuracy of these parsers can be further improved by reranking the analyses output by the parser, typically using a discriminative model with global features that are beyond the scope of the underlying generative model (Johnson *et al.* 1999; Collins 2000; Collins and Duffy 2002; Collins and Koo 2005; Charniak and Johnson 2005).

A radically different approach is to perform disambiguation deterministically, using a greedy parsing algorithm that approximates a globally optimal solution by making a series of locally optimal choices, guided by a classifier trained on gold standard derivation sequences derived from a treebank. Although this may seem like a futile strategy for a complex task like parsing, it has recently been used with some success especially in dependency-based parsing.[2] It was first applied to unlabeled dependency parsing by Kudo and Matsumoto (2002) (for Japanese) and by Yamada and Matsumoto (2003) (for English). It was later extended to labeled dependency parsing by Nivre *et al.* (2004) (for Swedish) and Nivre and Scholz (2004) (for English). More recently, it has also been applied with good results to lexicalized phrase structure parsing by Sagae and Lavie (2005).

One of the advantages of the deterministic, classifier-based approach is that it is straightforward to implement and has a very attractive time complexity, with parsing time being linear or at worst quadratic in the size of the input, although the constant associated with the classifier can sometimes become quite large. Moreover, while the accuracy of a deterministic parser is normally a bit lower than what can be attained with a more complex statistical model, trained and tuned on large amounts of data, the deterministic parser will often have a much steeper learning curve,

[2] In fact, essentially the same methodology has been proposed earlier for other frameworks by Berwick (1985), Simmons and Yu (1992), Zelle and Mooney (1993) and Veenstra and Daelemans (2000), among others, although these approaches have typically been evaluated only on artificially generated or very small data sets.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

which means that it may in fact give higher accuracy with small training data sets. This is a natural consequence of the fact that the deterministic model has a much smaller parameter space, where only the mode of the distribution for each distinct history needs to be estimated, whereas a traditional generative model requires a complete probability distribution. Finally, and for essentially the same reason, the deterministic model can be less sensitive to differences in linguistic structure and annotation style across languages and should therefore be more easily portable without substantial adaptation.

In this study, we investigate these issues by applying the deterministic, classifier-based approach, as implemented in the MaltParser system for inductive dependency parsing, to a wide range of languages with varying annotation schemes and with data sets of varying sizes. By way of background, this section presents the theoretical foundations of inductive dependency parsing, defining syntactic representations, parsing algorithms, feature models, and learning algorithms.[3] In section 3, we then describe the implemented MaltParser system that has been used for the experiments reported in section 4.

2.1 Dependency graphs

In dependency parsing, the syntactic analysis of a sentence is represented by a dependency graph, which we define as a labeled directed graph, the nodes of which are indices corresponding to the tokens of a sentence. Formally:

Definition 1
Given a set R of dependency types (arc labels), a *dependency graph* for a sentence $x = (w_1, \ldots, w_n)$ is a labeled directed graph $G = (V, E, L)$, where:

1. $V = \mathbf{Z}_{n+1}$
2. $E \subseteq V \times V$
3. $L : E \rightarrow R$

Definition 2
A dependency graph G is *well-formed* if and only if:

1. The node 0 is a root (Root).
2. G is connected (Connectedness).[4]

The set V of *nodes* (or *vertices*) is the set $\mathbf{Z}_{n+1} = \{0, 1, 2, \ldots, n\}$ ($n \in \mathbf{Z}^+$), i.e., the set of non-negative integers up to and including n. This means that every token index i of the sentence is a node ($1 \leq i \leq n$) and that there is a special node 0, which does not correspond to any token of the sentence and which will always be a root of the dependency graph (normally the only root).

[3] For an in-depth discussion of inductive dependency parsing and its relation to other parsing methods, see Nivre (2006).

[4] Strictly speaking, we require the graph to be *weakly connected*, which entails that the corresponding undirected graph is connected, whereas a *strongly connected* graph has a *directed* path between any pair of nodes.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

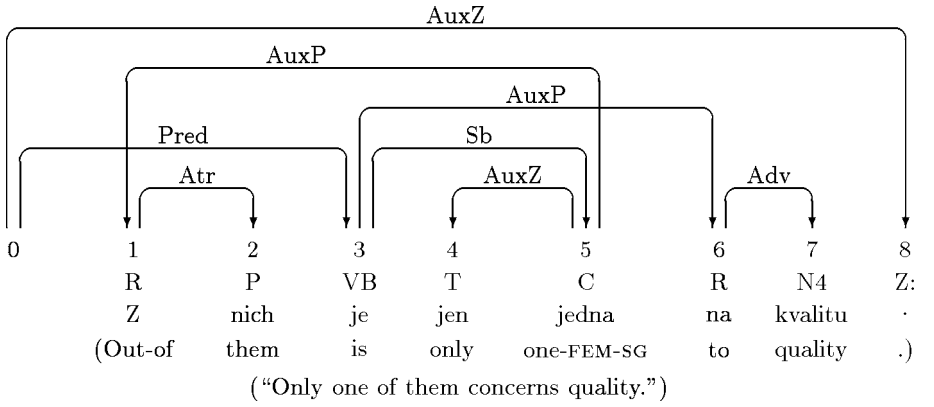


Fig. 1. Dependency graph for Czech sentence from the Prague Dependency Treebank.

In the following, we will reserve the term *token node* for a node that corresponds to a token of the sentence, and we will use the symbol V^+ to denote the set of token nodes of a sentence for which the set of nodes is V, i.e., $V^+ = V - \{0\}$. When necessary, we will write V_x and V_x^+ to indicate that V and V^+ are the nodes corresponding to a particular sentence $x = (w_1, \ldots, w_n)$. Note, however, that the only requirement imposed by x is that the number of nodes matches the length of x, i.e., $|V^+| = n$ and $|V| = n + 1$.

The set E of *arcs* (or *edges*) is a set of ordered pairs (i, j), where i and j are nodes. Since arcs are used to represent dependency relations, we will say that i is the *head* and j is the *dependent* of the arc (i, j). As usual, we will use the notation $i \rightarrow j$ to mean that there is an arc connecting i and j (i.e., $(i, j) \in E$) and we will use the notation $i \rightarrow^* j$ for the reflexive and transitive closure of the arc relation E (i.e., $i \rightarrow^* j$ if and only if $i = j$ or there is a path of arcs connecting i to j).

The function L assigns a dependency type (arc label) $r \in R$ to every arc $e \in E$. We will use the notation $i \xrightarrow{r} j$ to mean that there is an arc labeled r connecting i to j (i.e., $(i, j) \in E$ and $L((i, j)) = r$).

Figure 1 shows a Czech sentence from the Prague Dependency Treebank with a well-formed dependency graph according to Definition 1–2. Note that the use of a special root node (0) is crucial for the satisfaction of CONNECTEDNESS, since the graph would otherwise have consisted of two connected components rooted at nodes 3 and 8, respectively. The use of a special root node is thus a convenient way of ensuring CONNECTEDNESS, regardless of whether a particular annotation scheme requires that a single token node should dominate all the others. More importantly, it is a way of achieving robustness in parsing, since there will always be a single entry point into the graph even if the parser produces fragmented output.

The only conditions so far imposed on dependency graphs is that the special node 0 be a root and that the graph be connected. Here are three further constraints that are common in the literature:

3. Every node has at most one head, i.e., if $i \rightarrow j$ then there is no node k such that $k \neq i$ and $k \rightarrow j$ (SINGLE-HEAD).

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

4. The graph G is acyclic, i.e., if $i \rightarrow j$ then not $j \rightarrow^* i$ (ACYCLICITY).
5. The graph G is projective, i.e., if $i \rightarrow j$ then $i \rightarrow^* k$, for every node k such that $i < k < j$ or $j < k < i$ (PROJECTIVITY).

The SINGLE-HEAD constraint, together with the basic well-formedness conditions, entails that the graph is a tree rooted at the node 0, which means that any well-formed graph satisfying SINGLE-HEAD also satisfies ACYCLICITY. And whereas it is possible to require ACYCLICITY without SINGLE-HEAD, the two conditions are jointly assumed in almost all versions of dependency grammar, especially in computational systems.

By contrast, PROJECTIVITY is much more controversial. Broadly speaking, we can say that whereas most practical systems for dependency parsing do assume projectivity, most dependency-based linguistic theories do not. More precisely, most theoretical formulations of dependency grammar regard projectivity as the norm but also recognize the need for non-projective representations to capture non-local dependencies and discontinuities arising from free or flexible word order (Mel'čuk 1988; Hudson 1990). This theoretical preference for non-projective dependency graphs is usually carried over into treebank annotation schemes, so that virtually all treebanks annotated with dependency graphs contain non-projective structures. This is true, for example, of the Prague Dependency Treebank of Czech (Hajič *et al.* 2001), the Danish Dependency Treebank (Kromann 2003), and the Turkish Treebank (Oflazer *et al.* 2003), all of which are used in this study.

2.2 Deterministic parsing algorithms

The most commonly used deterministic algorithms for dependency parsing can be seen as variants of the basic shift-reduce algorithm, analyzing the input from left to right using two main data structures, a queue of remaining input tokens and a stack storing partially processed tokens. One example is the arc-eager algorithm introduced in Nivre (2003), which is used in all the experiments in this article and which we describe in detail in this section. Like most of the algorithms used for practical dependency parsing, this algorithm is restricted to projective dependency graphs. We begin by defining a parser *configuration* for a sentence $x = (w_1, \ldots, w_n)$, relative to a set R of dependency types (including a special symbol r_0 for dependents of the root):

Definition 3
Given a set $R = \{r_0, r_1, \ldots r_m\}$ of dependency types and a sentence $x = (w_1, \ldots, w_n)$, a *parser configuration* for x is a quadruple $c = (\sigma, \tau, h, d)$, where:

1. σ is a stack of token nodes i ($1 \leq i \leq j$ for some $j \leq n$).
2. τ is a sorted sequence of token nodes i ($j < i \leq n$).
3. $h : V_x^+ \rightarrow V_x$ is a function from token nodes to nodes.
4. $d : V_x^+ \rightarrow R$ is a function from token nodes to dependency types.
5. For every token node $i \in V_x^+$, $d(i) = r_0$ only if $h(i) = 0$.

The idea is that the sequence τ represents the remaining input tokens in a left-to-right pass over the input sentence x; the stack σ contains partially processed nodes

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

that are still candidates for dependency arcs, either as heads or dependents; and the functions h and d represent the (partially constructed) dependency graph for the input sentence x.

Representing the graph by means of two functions in this way is possible if we assume the SINGLE-HEAD constraint. Since, for every token node j, there is at most one arc (i, j), we can represent this arc by letting $h(j) = i$. Strictly speaking, h should be a partial function, to allow the possibility that there is no arc (i, j) for a given node j, but we will avoid this complication by assuming that every node j for which there is no token node i such that $i \rightarrow j$ is headed by the special root node 0, i.e., $h(j) = 0$. Formally, we establish the connection between configurations and dependency graphs as follows:

Definition 4
A configuration $c = (\sigma, \tau, h, d)$ for $x = (w_1, \ldots, w_n)$ defines the dependency graph $G_c = (V_x, E_c, L_c)$, where:

1. $E_c = \{(i, j) \mid h(j) = i\}$
2. $L_c = \{((i, j), r) \mid h(j) = i, d(j) = r\}$

We use the following notational conventions for the components of a configuration:

1. Both the stack σ and the sequence of input tokens τ will be represented as lists, although the stack σ will have its head (or top) to the right for reasons of perspicuity. Thus, $\sigma|i$ represents a stack with top i and tail σ, while $j|\tau$ represents a list of input tokens with head j and tail τ, and the operator $|$ is taken to be left-associative for the stack and right-associative for the input list. We use ϵ to represent the empty list/stack.
2. For the functions h and d, we will use the notation $f[x \mapsto y]$, given a specific function f, to denote the function g such that $g(x) = y$ and $g(z) = f(z)$ for all $z \neq x$. More formally, if $f(x) = y'$, then $f[x \mapsto y] = (f - \{(x, y')\}) \cup \{(x, y)\}$.

Initial and terminal parser configurations are defined in the following way:

Definition 5
A configuration c for $x = (w_1, \ldots, w_n)$ is *initial* if and only if it has the form $c = (\epsilon, (1, \ldots, n), h_0, d_0)$, where:

1. $h_0(i) = 0$ for every $i \in V_x^+$.
2. $d_0(i) = r_0$ for every $i \in V_x^+$.

A configuration c for $x = (w_1, \ldots, w_n)$ is *terminal* if and only if it has the form $c = (\sigma, \epsilon, h, d)$ (for arbitrary σ, h and d).

In other words, we initialize the parser with an empty stack, with all the token nodes of the sentence remaining to be processed, and with a dependency graph where all token nodes are dependents of the special root node 0 and all arcs are labeled with the special label r_0, and we terminate whenever the list of input tokens is empty, which happens when we have completed one left-to-right pass over the sentence. We use C for the set of all possible configurations (relative to some set

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

R of dependency types) and C^n for the set of non-terminal configurations, i.e., any configuration $c = (\sigma, \tau, h, d)$ where $\tau \neq \epsilon$.

A *transition* is a partial function $t : C^n \rightarrow C$. In other words, a transition maps non-terminal configurations to new configurations but may be undefined for some non-terminal configurations. The parsing algorithm uses four transitions, two of which are parameterized by a dependency type $r \in R$.

Definition 6
Given a set of dependency types R, the following transitions are possible for every $r \in R$:

1. LEFT-ARC(r):
   $(\sigma|i, j|\tau, h, d) \rightarrow (\sigma, j|\tau, h[i \mapsto j], d[i \mapsto r])$
   if $h(i) = 0$

2. RIGHT-ARC(r):
   $(\sigma|i, j|\tau, h, d) \rightarrow (\sigma|i|j, \tau, h[j \mapsto i], d[j \mapsto r])$
   if $h(j) = 0$

3. REDUCE:
   $(\sigma|i, \tau, h, d) \rightarrow (\sigma, \tau, h, d)$
   if $h(i) \neq 0$

4. SHIFT:
   $(\sigma, i|\tau, h, d) \rightarrow (\sigma|i, \tau, h, d)$

The transition LEFT-ARC(r) makes the top token i a (left) dependent of the next token j with dependency type r, i.e., $j \xrightarrow{r} i$, and immediately pops the stack. This transition can apply only if $h(i) = 0$, i.e., if the top token is previously attached to the root 0. The node i is popped from the stack because it must be complete with respect to left and right dependents at this point (given the assumption of projectivity).

The transition RIGHT-ARC(r) makes the next token j a (right) dependent of the top token i with dependency type r, i.e., $i \xrightarrow{r} j$, and immediately pushes j onto the stack. This transition can apply only if $h(j) = 0$, i.e., if the next token is previously attached to the root 0.[5] The node j is pushed onto the stack since it must be complete with respect to its left dependents at this point, but it cannot be popped because it may still need new dependents to the right.

The transition REDUCE pops the stack. This transition can apply only if $h(i) \neq 0$, i.e., if the top token i is already attached to a token node. This transition is needed for popping a node that was pushed in a RIGHT-ARC(r) transition and which has since found all its right dependents.

The transition SHIFT pushes the next token i onto the stack. This transition can apply unconditionally as long as there are input tokens remaining. It is needed for

[5] This condition is in fact superfluous, since it is impossible for the next input token to be attached to any other node, but it is included for symmetry.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

processing nodes that have their heads to the right, as well as nodes that are to remain attached to the special root node.

The transition system just defined is nondeterministic in itself, since there is normally more than one transition applicable to a given configuration. In order to perform deterministic parsing, the transition system needs to be supplemented with a mechanism for predicting the next transition at each nondeterministic choice point, as well as choosing a dependency type r for the transitions LEFT-ARC(r) and RIGHT-ARC(r). Such a mechanism can be called an *oracle* (Kay 2000). Assuming that we have an oracle $o : C^n \rightarrow (C^n \rightarrow C)$, the algorithm for deterministic dependency parsing is very simple and straightforward:

```
PARSE(x = (w_1, ..., w_n))
1  c ← (ϵ, (1, ..., n), h_0, d_0)
2  while c = (σ, τ, h, d) is not terminal
3      if σ = ϵ
4          c ← SHIFT(c)
5      else
6          c ← [o(c)](c)
7  G ← (V_x, E_c, L_c)
8  return G
```

As long as the parser remains in a non-terminal configuration, it applies the SHIFT transition if the stack is empty and otherwise the transition $o(c)$ predicted by the oracle. When a terminal configuration is reached, the dependency graph defined by this configuration is returned.

The notion of an oracle is useful for the theoretical analysis of parsing algorithms and allows us to show, for example, that the parsing algorithm just described derives a well-formed projective dependency graph for any input sentence in time that is linear in the length of the input, and that any projective dependency graph can be derived by the algorithm (Nivre 2006). In practice, the oracle can only be approximated, but the fundamental idea in inductive dependency parsing is that we can achieve a good approximation using history-based feature models and discriminative machine learning, as described in the following subsections.

An alternative to the algorithm described in this section is to use an arc-standard strategy, more directly corresponding to the strict bottom-up processing in traditional shift-reduce parsing. In this scheme, the RIGHT-ARC(r) and REDUCE transitions are merged into a single transition that immediately pops the dependent in the same way as LEFT-ARC(r), which means that right dependents can only be attached after they have found all their descendants. This is the strategy used by Kudo and Matsumoto (2002), Yamada and Matsumoto (2003) and Cheng *et al.* (2004), although they also modify the algorithm by allowing multiple passes over the input. There are few studies comparing the performance of different algorithms, but Cheng *et al.* (2005) found consistently better accuracy for the arc-eager, single-pass strategy (over the arc-standard, multi-pass algorithm) in parsing the CKIP Treebank of Chinese.

A somewhat different approach is to use the incremental algorithms described by Covington (2001), where the stack is replaced by an open list where any token can be linked to the next input token. This allows non-projective graphs to be

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

derived at the cost of making parsing time quadratic in the length of the input. This is a technique that has not yet been evaluated on a large scale, and attempts at recovering non-projective dependencies within this tradition have so far relied on post-processing of projective dependency graphs, e.g., using the pseudo-projective technique proposed by Nivre and Nilsson (2005).

2.3 History-based feature models

History-based models for natural language processing were first introduced by Black *et al.* (1992) and have been used extensively for part-of-speech tagging and syntactic parsing. The basic idea is to map each pair (x, y) of an input string x and an analysis y to a sequence of decisions $D = (d_1, \ldots, d_n)$. In a generative probabilistic model, the joint probability $P(x, y)$ can then be expressed using the chain rule of probabilities as follows:

$$P(x, y) = P(d_1, \ldots, d_n) = \prod_{i=1}^{n} P(d_i \mid d_1, \ldots, d_{i-1}) \tag{1}$$

The conditioning context for each d_i, $(d_1, \ldots, d_{i-1})$, is referred to as the *history* and usually corresponds to some partially built structure. In order to get a tractable learning problem, histories are grouped into equivalence classes by a function Φ:

$$P(x, y) = P(d_1, \ldots, d_n) = \prod_{i=1}^{n} P(d_i \mid \Phi(d_1, \ldots, d_{i-1})) \tag{2}$$

Early versions of this scheme were integrated into grammar-driven systems. For example, Black *et al.* (1993) used a standard PCFG but could improve parsing performance considerably by using a history-based model for bottom-up construction of leftmost derivations. In more recent developments, the history-based model has replaced the grammar completely, as in the parsers of Collins (1997; 1999) and Charniak (2000).

With a generative probabilistic model, the parameters that need to be estimated are the conditional probabilities $P(d_i \mid \Phi(d_1, \ldots, d_{i-1}))$, for every possible decision d_i and non-equivalent history $H_i = \Phi(d_1, \ldots, d_{i-1})$. With a deterministic parsing strategy, we only need to estimate the mode of each conditional distribution, i.e., $\arg\max_{d_i} P(d_i \mid \Phi(d_1, \ldots, d_{i-1}))$. This reduces the parameter estimation problem to that of learning a classifier, where the classes are the possible decisions of the parser, e.g., the possible transitions of the algorithm described in the previous section.

Distinct parser histories are normally represented as sequences of attributes, so-called *feature vectors*, and the function Φ, referred to as the *feature model*, can therefore be defined in terms of a sequence $\Phi_{1,p} = (\phi_1, \ldots, \phi_p)$ of *feature functions*, where each function ϕ_i identifies some relevant feature of the history. The most important features in dependency parsing are the attributes of input tokens, such as their word form, part-of-speech or dependency type, and we will in fact limit ourselves in this article to features that can be defined as simple attributes of tokens.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Token attributes can be divided into *static* and *dynamic* attributes, where static attributes are properties that remain constant during the parsing of a sentence. This primarily includes the actual word form of a token, but also any kind of annotation that is the result of preprocessing, such as part-of-speech tag, lemma, or word sense annotation. In this article, we restrict our attention to two kinds of static attributes, word form and part-of-speech. Given a sentence $x = (w_1, \ldots, w_n)$, with part-of-speech annotation, we use $w(i)$ and $p(i)$ to refer to the word form and part-of-speech, respectively, of the ith token. We will also make use of fixed-length suffixes of word forms and write $s_m(w(i))$ for the m-character suffix of $w(i)$ (where $s_m(w(i)) = w(i)$ if $w(i)$ has length $l \leq m$).

Dynamic attributes, by contrast, are attributes that are defined by the partially built dependency graph, which in this article will be limited to the dependency type by which a token is related to its head, given by the function d of the current parser configuration $c = (\sigma, \tau, h, d)$.

To define complex history-based feature models, we need to refer to attributes of arbitrary tokens in the parser history, represented by the current parser configuration. For this purpose, we introduce a set of address functions.

Definition 7
Given a sentence $x = (w_1, \ldots, w_n)$ and a parser configuration $c = (\sigma, \tau, h, d)$ for x:

1. σ_i is the ith token from the top of the stack (starting at index 0).
2. τ_i is the ith token in the remaining input (starting at index 0).
3. $h(i)$ is the head of token i in the graph defined by h.
4. $l(i)$ is the leftmost child of token i in the graph defined by h.
5. $r(i)$ is the rightmost child of token i in the graph defined by h.

By combining these functions, we can define arbitrarily complex functions that identify tokens relative to a given parser configuration c. For example, while σ_0 is the token on top of the stack, $h(\sigma_0)$ is the head of the token on top of the stack, and $l(h(\sigma_0))$ is the leftmost dependent of the head of the token on top of the stack. It should be noted that these functions are generally partial functions on token nodes, which means that if one of the inner functions in a chain of applications returns 0 (because $h(i) = 0$) or is undefined (because the stack is empty, or a token does not have a leftmost child, etc.), then the outermost function is always undefined.

Finally, we can now define feature functions by applying attribute functions to complex combinations of address functions. For example, $p(\tau_0)$ is the part-of-speech of the next input token, while $d(h(\sigma_0))$ is the dependency type of the head of the token on top of the stack, which may or may not be defined in a given configuration. Any feature function that is undefined for a given configuration, because the complex address function fails to identify a token, is assigned a special *nil* value. Feature models used for inductive dependency parsing typically combine static part-of-speech features and lexical features (or suffix features) with dynamic dependency type features. The kind of models used in the experiments later on are described in section 3.2 below.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

2.4 Discriminative machine learning

Given a function approximation problem with labeled training data from target function $f : X \rightarrow Y$, discriminative learning methods attempt to optimize the mapping from inputs $x \in X$ to outputs $y \in Y$ directly, without estimating a full generative model of the joint distribution of X and Y. Discriminatively trained models have in recent years been shown to outperform generative models for many problems in natural language processing, including syntactic parsing, by directly estimating a conditional probability distribution $P(Y|X)$ (Johnson *et al.* 1999; Collins 2000; Collins and Duffy 2002; Collins and Koo 2005; Charniak and Johnson 2005). With a deterministic parsing strategy, the learning problem can be further reduced to a pure classification problem, where the input instances are histories (represented by feature vectors) and the output classes are parsing decisions.

Thus, the training data for the learner consists of pairs $(\Phi(c), t)$, where $\Phi(c)$ is the representation of a parser configuration defined by the feature model $\Phi(c)$ and t is the correct transition out of c. Such data can be generated from a treebank of gold standard dependency graphs, by reconstructing the correct transition sequence for each dependency graph in the treebank and extracting the appropriate feature vectors for each configuration, as described in detail by Nivre (2006) for the parsing algorithm discussed in section 2.2.

Although in principle any learning algorithm capable of inducing a classifier from labeled training data can be used to solve the learning problem posed by inductive dependency parsing, most of the work done in this area has been based on support vector machines (SVM) and memory-based learning (MBL).[6]

SVM is a hyperplane classifier that relies on the maximum margin strategy introduced by Vapnik (1995). Furthermore, it allows the use of kernel functions to map the original feature space to a higher-dimensional space, where the classification problem may be (more) linearly separable. In dependency parsing, SVM has been used primarily by Matsumoto and colleagues (Kudo and Matsumoto 2002; Yamada and Matsumoto 2003; Cheng *et al.* 2004; Cheng *et al.* 2005).

MBL is a lazy learning method, based on the idea that learning is the simple storage of experiences in memory and that solving a new problem is achieved by reusing solutions from similar previously solved problems (Daelemans and Van den Bosch 2005). In essence, this is a k nearest neighbor approach to classification, although a variety of sophisticated techniques, including different distance metrics and feature weighting schemes can be used to improve classification accuracy. In dependency parsing, MBL has been used primarily by Nivre and colleagues (Nivre *et al.* 2004; Nivre and Scholz 2004; Nivre and Nilsson 2005), and it is also the learning method that is used for the experiments in this article.

3 MaltParser

MaltParser is an implementation of inductive dependency parsing, as described in the previous section, where the syntactic analysis of a sentence amounts to

[6] In addition, maximum entropy modeling was used in the comparative evaluation of Cheng *et al.* (2005).

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

the deterministic derivation of a dependency graph, and where discriminative machine learning is used to guide the parser at nondeterministic choice points, based on a history-based feature model. MaltParser can also be characterized as a data-driven parser-generator. While a traditional parser-generator constructs a parser given a grammar, a data-driven parser-generator constructs a parser given a treebank.

The system can be run in two basic modes. In learning mode, it takes as input a (training) set of sentences with dependency graph annotations, derives training data by reconstructing the correct transition sequences, and trains a classifier on this data set according to the specifications of the user. In parsing mode, it takes as input a (test) set of sentences and a previously trained classifier and parses the sentences using the classifier as a guide.

3.1 Parsing algorithms

MaltParser provides two main parsing algorithms, each with several options:

- The linear-time algorithm of Nivre (2003) can be run in arc-eager or arc-standard mode. The arc-standard version is similar to but not identical to the algorithm of Yamada and Matsumoto (2003), since the latter also uses multiple passes over the input (Nivre 2004). In both versions, this algorithm is limited to projective dependency graphs.
- The incremental algorithm of Covington (2001) can be run in projective or non-projective mode. In the latter case, graphs are still guaranteed to obey the constraints Root, Connectedness, Single-Head and Acyclicity.

The experiments reported in this article are all based on the arc-eager version of Nivre's algorithm.

3.2 Feature models

MaltParser allows the user to define arbitrarily complex feature models, using address functions and attribute functions as described in section 2.3.[7] The standard model used in most of the experiments reported below combines part-of-speech features, lexical features and dependency type features in the following way:

$p(\sigma_1)$	$w(h(\sigma_0))$	$d(l(\sigma_0))$
$p(\sigma_0)$	$w(\sigma_0)$	$d(\sigma_0)$
$p(\tau_0)$	$w(\tau_0))$	$d(r(\sigma_0))$
$p(\tau_1)$	$w(\tau_1)$	$d(l(\tau_0))$
$p(\tau_2)$		
$p(\tau_3)$		

[7] The feature models supported by MaltParser are in fact slightly more general in that they also allow address functions that refer to siblings. This option is not exploited in the experiments reported below and has therefore been excluded from the presentation in section 2.3.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

This model includes six part-of-speech features, defined by the part-of-speech of the two topmost stack tokens ($p(\sigma_0)$, $p(\sigma_1)$) and by the first four tokens of the remaining input ($p(\tau_0)$, $p(\tau_1)$, $p(\tau_2)$, $p(\tau_3)$). The dependency type features involve the top token on the stack ($d(\sigma_0)$), its leftmost and rightmost dependent ($d(l(\sigma_0))$, $d(r(\sigma_0))$), and the leftmost child of the next input token ($d(l(\tau_0))$).[8] Finally, the standard model includes four lexical features, defined by the word form of the top token on the stack ($w(\sigma_0)$), the head of the top token ($w(h(\sigma_0))$), and the next two input tokens ($w(\tau_0)$, $w(\tau_1)$).

The standard model can be seen as the prototypical feature model used in the experiments reported below, although the tuned models for some languages deviate from it by adding or omitting features, or by replacing lexical features by suffix features (the latter not being used at all in the standard model). Deviations from the standard model are specified in table 3 below.

3.3 Learning algorithms

MaltParser provides two main learning algorithms, each with a variety of options:

- Memory-based learning (MBL) using TiMBL, a software package for memory-based learning and classification developed by Daelemans, Van den Bosch and colleagues at the Universities of Tilburg and Antwerp (Daelemans and Van den Bosch 2005).
- Support vector machines (SVM) using LIBSVM, a library for SVM learning and classification developed by Chang and Lin at National Taiwan University (Chang and Lin 2001).

The experiments reported in this paper are all based on MBL and make crucial use of the following features of TiMBL:

- Varying the number k of nearest neighbors
- Using the Modified Value Difference Metric (MVDM) for distances between feature values (for values seen more than l times)
- Distance-weighted class voting for determining the majority class

The optimal values for these parameters vary for different feature models, languages and data sets, but typical values are $k = 5$, MVDM down to $l = 3$ (with the simple Overlap metric for lower frequencies), and class voting weighted by inverse distance (ID). For more information about these and other TiMBL features, we refer to Daelemans and Van den Bosch (2005).

3.4 Auxiliary tools

MaltParser is supported by a suite of freely available tools for, among other things, parser evaluation and treebank conversion. Of special interest in this context are

[8] It is worth pointing out that, given the nature of the arc-eager parsing algorithm, the dependency type of the next input token and its rightmost child will always be undefined at decision time (hence their omission in the standard model and all other models).

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

the tools for pseudo-projective dependency parsing (Nivre and Nilsson 2005). This is a method for recovering non-projective dependencies through a combination of data-driven projective dependency parsing and graph transformation techniques in the following way:

1. Dependency graphs in the training data sets are transformed (if necessary) to projective dependency graphs, by minimally moving non-projective arcs upwards towards the root and encoding information about these transformations in arc labels.
2. The projective parser is trained as usual, except that the dependency graphs in the training set are labeled with the enriched arc labels.
3. New sentences are parsed into projective dependency graphs with enriched arc labels.
4. Dependency graphs produced by the parser are transformed (if possible) to non-projective dependency graphs, using an inverse transformation guided by information in the arc labels.

This methodology has been used in a few of the experiments reported below, in particular for the parsing of Czech (section 4.2.5).

4 Experimental evaluation

In this section, we summarize experiments with the MaltParser system on data from ten different languages: Bulgarian, Chinese, Czech, Danish, Dutch, English, German, Italian, Swedish and Turkish.[9] Although the group is dominated by Indo-European languages, in particular Germanic languages, the languages nevertheless represent fairly different language types, ranging from Chinese and English, with very reduced morphology and relatively inflexible word order, to languages like Czech and Turkish, with rich morphology and flexible word order, and with Bulgarian, Danish, Dutch, German, Italian and Swedish somewhere in the middle. In addition, the treebank annotation schemes used to analyze these languages differ considerably. Whereas the treebanks for Czech, Danish, Italian and Turkish are proper dependency treebanks, albeit couched in different theoretical frameworks, the annotation schemes for the remaining treebanks are based on constituency in combination with grammatical functions, which necessitates a conversion from constituent structures to dependency structures.

Below we first describe the general methodology used to evaluate the system, in particular the evaluation metrics used to assess parsing accuracy, and give an overview of the different data sets and experiments performed for different languages (section 4.1). This is followed by a presentation of the results (section 4.2), with specific subsections for each language (section 4.2.1–4.2.10), where we also give a more detailed description of the respective treebanks and the specific settings

[9] Results have been published previously for Swedish (Nivre *et al.* 2004; Nivre 2006), English (Nivre and Scholz 2004; Nivre 2006), Czech (Nivre and Nilsson 2005), Bulgarian (Marinov and Nivre 2005), Danish (Nivre and Hall 2005) and Italian (Chanev 2005) but not for Chinese, German and Turkish.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 1. *Data sets. AS = Annotation scheme (C = Constituency, D = Dependency, G = Grammatical functions); Pro = Projective; #D = Number of dependency types; #P = Number of PoS tags; TA =Tagging accuracy; #W = Number of words; #S = Number of sentences; SL = Mean sentence length; EM = Evaluation method (T = Held-out test set, CV_k = k-fold cross-validation)*

Language	AS	Pro	#D	#P	TA	#W	#S	SL	EM
Bulgarian	C	no	14	51	93.5	72k	5.1k	14.1	CV_8
Chinese	CG	yes	12	35	100.0	509k	18.8k	27.1	T
Czech	D	no	26	28	94.1	1507k	87.9k	17.2	T
Danish	D	no	54	33	96.3	100k	5.5k	18.2	T
Dutch	CD	no	23	165	95.7	186k	13.7k	13.6	T
English	CG	yes	12	48	96.1	1174k	49.2k	23.8	T
German	CG	no	31	55	100.0	382k	22.1k	17.3	CV_{10}
Italian	D	no	17	89	93.1	42k	1.5k	27.7	CV_{10}
Swedish	CG	yes	17	46	95.6	98k	6.3k	15.5	T
Turkish	D	no	24	484	100.0	48k	5.6k	8.6	CV_{10}

used for individual experiments, followed by a general discussion, where we bring together the results from different languages and try to discern some general trends (section 4.3).

4.1 Method

Table 1 gives an overview of the data sets for the ten languages. The first column characterizes the annotation scheme and the second indicates whether the (possibly converted) annotation is restricted to projective dependency graphs. The next two columns contain the number of distinct dependency types and part-of-speech tags, respectively, where the latter refers to the tagset actually used in parsing, which may be a reduced version of the tagset used in the original treebank annotation. The fifth column gives the mean accuracy of the part-of-speech tagging given as input to the parser, where 100.0 indicates that experiments have been performed using gold standard tags (i.e., manually assigned or corrected tags) rather than the output of an automatic tagger. The next three columns give the number of tokens and sentences, and the mean number of words per sentence. These figures refer in each case to the complete treebank, of which at most 90% has been used for training and at least 10% for testing (possibly using *k*-fold cross-validation).

Table 2 gives a little more information about the syntactic analysis adopted in the different treebank annotation schemes. Whereas all the schemes agree on basic structures such as verbs taking their core arguments as dependents and adjuncts being dependents of the heads they modify, there are a number of constructions that have competing analyses with respect to their dependency structure. This holds in particular for constructions involving function words, such as auxiliary verbs, prepositions, determiners, and complementizers, but also for the ubiquitous phenomenon of coordination. Table 2 shows the choices made for each of these cases in the different treebanks, and we see that there is a fair amount of variation

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 2. *Annotation style (choice of head). VG = Verb group (Aux = Auxiliary verb, MV = Main verb); AP = Adpositional phrase (Ad = Adposition, N = Nominal head); NP = Noun phrase (Det = Determiner, N = Noun); SC = Subordinate clause (Comp = Complementizer, V = Verb); Coord = Coordination (CC = Coordinating conjunction, $Conj_1$ = First conjunct, $Conj_n$ = Last conjunct); NA = Not applicable*

Language	VG	AP	NP	SC	Coord
Bulgarian	MV	Ad	N	Comp	$Conj_1$
Chinese	Aux	Ad	N	Comp	$Conj_1/Conj_n$
Czech	MV	Ad	N	V	CC
Danish	Aux	Ad	Det	Comp	$Conj_1$
Dutch	Aux	Ad	N	Comp	CC
English	Aux	Ad	N	Comp	$Conj_1/Conj_n$
German	Aux	Ad	N	V	$Conj_1$
Italian	MV	Ad	Det	Comp	$Conj_1$
Swedish	Aux	Ad	N	Comp	$Conj_1$
Turkish	NA	(Ad)	N	NA	$Conj_n$

especially with respect to verb groups and coordination.[10] It is worth noting that for Turkish, which is a richly inflected, agglutinative language, some of the distinctions are not applicable, since the relevant construction is encoded morphologically rather than syntactically.[11] It is also important to remember that, for the treebanks that are not originally annotated with dependency structures, the analysis adopted here only represents one conversion out of several possible alternatives. More information about the conversions are given for each language below.

All the experiments reported in this article have been performed with the parsing algorithm described in Nivre (2003; 2004; 2006) and with memory-based learning and classification as implemented in the TiMBL software package by Daelemans and Van den Bosch (2005). A variety of feature models have been tested, but we only report results for the optimal model for each language, which is characterized in relation to the standard model defined in section 3.2. Standard settings for the TiMBL learner include $k = 5$ (number of nearest distances), MVDM metric down to a threshold of $l = 3$, and distance weighted class voting with Inverse Distance weights (ID).

Final evaluation has been performed using either k-fold cross-validation or a held-out test set, as shown in the last column in table 1. Evaluation on held-out data has in turn been preceded by a tuning phase using either k-fold cross-validation or a development test set, as described for each language below. The diversity in evaluation methods is partly a result of practical circumstances and partly motivated by the concern to make results comparable to previously published results for a

[10] The notation $Conj_1/Conj_n$ under Coord for Chinese and English signifies that coordination is analyzed as a head-initial or head-final construction depending on whether the underlying phrase type is head-initial (e.g., verb phrases) or head-final (e.g., noun phrases).

[11] Whereas postpositions generally appear as suffixes on nouns, there are marginal cases where they occur as separate words and are then treated as heads. Hence, the brackets around Ad in the AP column for Turkish.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

given language. Thus, while results on the Penn Treebank are customarily obtained by training on sections 2–21 and testing on section 23 (using any of the remaining sections as a development test set), results on the Turkish Treebank have so far been based on ten-fold cross-validation, which is well motivated by the limited amount of data available. It should also be noted that the amount of work devoted to model selection and parameter optimization varies considerably between the languages, with Swedish and English being most thoroughly investigated while the results for other languages, notably Dutch, German and Turkish, are still preliminary and can probably be improved substantially.

The evaluation metrics used throughout are the *unlabeled attachment score* AS_U, which is the proportion of tokens that are assigned the correct head (regardless of dependency type), and the *labeled attachment score* AS_L, which is the proportion of tokens that are assigned the correct head and the correct dependency type, following the proposal of Lin (1998). All results are presented as mean scores per token, with punctuation tokens excluded from all counts.[12] For each language, we also provide a more detailed breakdown with (unlabeled) attachment score, precision, recall and F measure for individual dependency types.

Before we turn to the experimental results, a caveat is in order concerning their interpretation and in particular about cross-linguistic comparability. The main point of the experimental evaluation is to corroborate the claim that MaltParser is language-independent enough to achieve reasonably accurate parsing for a wide variety of languages, where the level of accuracy is related, whenever possible, to previously obtained results for that language. In order to facilitate this kind of comparison, we have sometimes had to sacrifice comparability between languages, notably by using training sets of different size or different procedures for obtaining accuracy scores as explained earlier. This means that, even though we sometimes compare results across languages, such comparisons must be taken with a pinch of salt. Although a more controlled cross-linguistic comparison would be very interesting, it is also very difficult to achieve given that available resources are very diverse with respect to standards of annotation, the amount of annotated data available, the existence of accurate part-of-speech taggers, etc. Faced with this diversity, we have done our best to come up with a reasonable compromise between the conflicting requirements of ensuring cross-linguistic comparability and being faithful to existing theoretical and practical traditions for specific languages and treebanks. This means, for example, that we retain the original arc labels for all treebanks, so that users of these treebanks can easily relate our results to theirs, even though this has the consequence that, e.g., subjects will be denoted by a variety of labels such as SUB, SBJ, SUBJ and Sb, but all arc labels will be accompanied by descriptions that should make them understandable also for readers who are not familiar with a given treebank annotation scheme.

[12] Although punctuation tokens are excluded in the calculation of accuracy scores, they are *included* during parsing. No changes have been made to the tokenization or sentence segmentation found in the respective treebanks, except for Turkish (see section 4.2.10).

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 3. *Overview of results. Model = Best feature model (− = omitted, + = added, → = replaced by); Settings = TiMBL settings; AS_U = Unlabeled attachment score; AS_L = Labeled attachment score*

Language	Model	Settings	AS_U	AS_L
Bulgarian	$\forall a[w(a) \rightarrow s_6(w(a))]$	Standard	81.3	73.6
Chinese	Standard	$k = 6$, $l = 8$	81.1	79.2
Czech	Standard	Standard	80.1	72.8
Danish	$[w(h(\sigma_0)) \rightarrow s_6(w(h(\sigma_0)))); -w(\tau_1)]$	Standard	85.6	79.5
Dutch	Standard	$k = 10$	84.7	79.2
English	Standard	$k = 7$, $l = 5$	88.1	86.3
German	$[-w(h(\sigma_0)); -w(\tau_1); +p(\sigma_2)]$	$k = 13$, IL	88.1	83.4
Italian	Standard	Standard	82.9	75.7
Swedish	Standard	Standard	86.3	82.0
Turkish	$[-p(\sigma_1); -p(\tau_2); -p(\tau_3); -w(h(\sigma_0)); -w(\tau_1)]$	Standard	81.6	69.0

4.2 Results

Table 3 gives an overview of the results, summarizing for each language the optimal feature model and TiMBL parameter settings, as well as the best unlabeled and labeled attachment scores. In the following subsections, we analyze the results for each language in a little more detail, making state-of-the-art comparisons where this is possible. The earliest experiments were those performed on Swedish and English and the standard models and settings are mainly based on the results of these experiments. It is therefore natural to treat Swedish and English first, with the remaining languages following in alphabetical order.

4.2.1 Swedish

The Swedish data come from Talbanken (Einarsson 1976), a manually annotated corpus of both written and spoken Swedish, created at Lund University in the 1970s. We use the professional prose section, consisting of material taken from textbooks, newspapers and information brochures. Although the original annotation scheme is an eclectic combination of constituent structure, dependency structure, and topological fields (Teleman 1974), it has been possible to convert the annotated sentences to dependency graphs with very high accuracy. In the conversion process, we have reduced the original fine-grained classification of grammatical functions to a more restricted set of 17 dependency types, mainly corresponding to traditional grammatical functions such as *subject*, *object* and *adverbial*. We have used a pseudo-randomized data split, dividing the data into 10 sections by allocating sentence i to section $i \bmod 10$. We have used sections 1–9 for 9-fold cross-validation during development and section 0 for final evaluation.

The overall accuracy scores for Swedish, obtained with the standard model and standard settings, are $AS_U = 86.3\%$ and $AS_L = 82.0\%$. Table 4 gives unlabeled attachment score (AS_U), labeled precision (P), recall (R) and F measure (F) for individual dependency types in the Swedish data. These types can be divided into three groups according to accuracy. In the high-accuracy set, with a labeled F

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 4. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Swedish (held-out test set, section 0)*

Dependency Type	n	AS_U	P	R	F
Adverbial (ADV)	1607	79.8	75.8	76.8	76.3
Apposition (APP)	42	23.8	38.1	19.0	25.4
Attribute (ATT)	950	81.3	79.9	78.5	79.2
Coordination (CC)	963	82.5	78.1	79.8	78.9
Determiner (DET)	947	92.6	88.9	90.2	89.5
Idiom (ID)	254	72.0	72.5	58.3	64.6
Infinitive marker (IM)	133	98.5	98.5	98.5	98.5
Infinitive complement (INF)	10	100.0	100.0	30.0	46.2
Object (OBJ)	585	88.0	78.2	77.3	77.7
Preposition dependent (PR)	985	94.2	88.6	92.7	90.6
Predicative (PRD)	244	90.6	76.7	77.0	76.8
Root (ROOT)	607	91.3	84.6	91.3	87.8
Subject (SUB)	957	89.8	86.7	82.5	84.5
Complementizer dependent (UK)	213	85.0	89.4	83.6	86.4
Verb group (VC)	238	93.7	82.1	90.6	86.1
Other (XX)	29	82.8	85.7	20.7	33.3
Total	8764	86.3	82.0	82.0	82.0

measure from 84% to 98%, we find all dependency types where the head is a closed class word: IM (marker → infinitive), PR (preposition → noun), UK (complementizer → verb) and VC (auxiliary verb → main verb). We also find the type DET (noun → determiner), which has similar characteristics although the determiner is not treated as the head in the Swedish annotation. The high-accuracy set also includes the central dependency types ROOT and SUB, which normally identify the finite verb of the main clause and the grammatical subject, respectively.

In the medium-accuracy set, with a labeled F measure in the range of 75–80%, we find the remaining major dependency types, ADV (adverbial), ATT (nominal modifier), CC (coordination), OBJ (object) and PRD (predicative). However, this set can be divided into two subsets, the first consisting of ADV, ATT and CC, which have an unlabeled attachment score not much above the labeled F measure, indicating that parsing errors are mainly due to incorrect attachment. This is plausible since ADV and ATT are the dependency types typically involved in modifier attachment ambiguities and since coordination is also a source of attachment ambiguities. The second subset contains OBJ and PRD, which both have an unlabeled attachment score close to 90%, which means that they are often correctly attached but may be incorrectly labeled. This is again plausible, since these types identify nominal arguments of the verb (other than the subject), which can often occur in the same structural positions.

Finally, we have a low-accuracy set, with a labeled F measure below 70%, where the common denominator is mainly low frequency: INF (infinitive complements), APP (appositions), XX (unclassifiable). The only exception to this generalization is the type ID (idiom constituent), which is not that rare but which is rather special for other reasons. All types in this set except APP have a relatively high unlabeled attachment score, but their labels are seldom used correctly.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 5. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for English (held-out test set, section 23)*

Dependency Type	*n*	AS_U	P	R	F
Adjective/adverb modifier (AMOD)	2072	78.2	80.7	73.0	76.7
Other (DEP)	259	42.9	56.5	30.1	39.3
Noun modifier (NMOD)	21002	91.2	91.1	90.8	91.0
Object (OBJ)	1960	86.5	78.9	83.5	81.1
Preposition modifier (PMOD)	5593	90.2	87.7	89.5	88.6
Predicative (PRD)	832	90.0	75.9	71.8	73.8
Root (ROOT)	2401	86.4	78.8	86.4	82.4
Complementizer dependent (SBAR)	1195	86.0	87.1	85.1	86.1
Subject (SBJ)	4108	90.0	90.6	88.1	89.3
Verb group (VC)	1771	98.8	93.4	96.6	95.0
Adverbial (VMOD)	8175	80.3	76.5	77.1	76.8
Total	49368	88.1	86.3	86.3	86.3

Relating the Swedish results to the state of the art is rather difficult, since there is no comparable evaluation reported in the literature, let alone based on the same data. Voutilainen (2001) presents a partial and informal evaluation of a Swedish FDG parser, based on manually checked parses of about 400 sentences from newspaper text, and reports F measures of 95% for subjects and 92% for objects. These results clearly indicate a higher level of accuracy than that attained in the experiments reported here, but without knowing the details of the data selection and evaluation procedure it is very difficult to draw any precise conclusions.

4.2.2 English

The data set used for English is the standard data set from the Wall Street Journal section of the Penn Treebank, with sections 2–21 used for training and section 23 for testing (with section 00 as the development test set). The data has been converted to dependency trees using the head percolation table of Yamada and Matsumoto (2003), and dependency type labels have been inferred using a variation of the scheme employed by Collins (1999), which makes use of the nonterminal labels on the head daughter, non-head daughter and parent corresponding to a given dependency relation. However, instead of simply concatenating these labels, as in the Collins scheme, we use a set of rules to map these complex categories onto a set of 10 dependency types, including traditional grammatical functions such as *subject*, *object*, etc. More details about the conversion can be found in Nivre (2006).

The best performing model for English is the standard model and the TiMBL parameter settings deviate from the standard ones only by having a higher k value ($k = 7$) and a higher threshold for MVDM ($l = 5$). The overall accuracy scores for English are $AS_U = 88.1\%$ and $AS_L = 86.3\%$. The relatively narrow gap between unlabeled and labeled accuracy is probably due mainly to the coarse-grained nature of the dependency type set and perhaps also to the fact that these labels have been inferred automatically from phrase structure representations. Table 5 shows the accuracy for individual dependency types in the same way as for Swedish in

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

table 4, and again we can divide dependency types according to accuracy into three sets. In the high-accuracy set, with a labeled F measure from 86% to 95%, we find SBJ (subject) and three dependency types where the head is a closed class word: PMOD (preposition → complement/modifier), VC (auxiliary verb → main verb) and SBAR (complementizer → verb). In addition, this set includes the type NMOD, which includes the noun-determiner relation as an important subtype.

In the medium-accuracy set, with a labeled F measure from 74% to 82%, we find the types AMOD, VMOD, OBJ, PRD and ROOT. The former two dependency types mostly cover adverbial functions, and have a labeled accuracy not too far below their unlabeled attachment score, which is an indication that the main difficulty lies in finding the correct head. By contrast, the argument functions OBJ and PRD have a much better unlabeled attachment score, which shows that they are often attached to the correct head but misclassified. This tendency is especially pronounced for the PRD type, where the difference is more than 15 percentage points, which can probably be explained by the fact that this type is relatively infrequent in the annotated English data. The low-accuracy set for English only includes the default classification DEP. The very low accuracy for this dependency type can be explained by the fact that it is both a heterogeneous category and the least frequent dependency type in the data.

Compared to the state of the art, the unlabeled attachment score is about 4% lower than the best reported results, obtained with the parser of Charniak (2000) and reported in Yamada and Matsumoto (2003).[13] For the labeled attachment score, we are not aware of any strictly comparable results, but Blaheta and Charniak (2000) report an F measure of 98.9% for the assignment of grammatical role labels to phrases that were correctly parsed by the parser described in Charniak (2000), using the same data set. If null labels are excluded, the F score drops to 95.6%. The corresponding F measures for MaltParser are 98.0% and 97.8%, treating the default label DEP as the equivalent of a null label. The experiments are not strictly comparable, since they involve different sets of functional categories (where only the labels SBJ and PRD are equivalent) and one is based on phrase structure and the other on dependency structure, but it nevertheless seems fair to conclude that MaltParser's labeling accuracy is close to the state of the art, even if its capacity to derive correct structures is not.

4.2.3 Bulgarian

For the current experiments we used a subset of BulTreeBank (Simov *et al.* 2002), since the complete treebank is not officially released and still under development. The set contains 71703 words of Bulgarian text from different sources, annotated with constituent structure. Although the annotation scheme is meant to be compatible with the framework of HPSG, syntactic heads are not explicitly annotated, which

[13] The score for the Charniak parser has been obtained by converting the output of the parser to dependency structures using the same conversion as in our experiments, which means that the comparison is as exact as possible. For further comparisons, see Nivre (2006).

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 6. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Bulgarian (mean of 8-fold cross-validation, frequency counts rounded to whole integers)*

Dependency Type	*n*	AS_U	P	R	F
Adverbial (ADV)	914	67.2	59.4	51.2	55.0
Apposition (APP)	120	65.5	54.1	49.0	51.9
Attribute (ATT)	1297	79.6	74.0	75.4	74.7
Coordination (CC)	555	53.6	52.8	48.5	50.6
Determiner (DET)	259	82.9	80.2	76.5	78.3
Idiom (ID)	214	94.6	90.2	89.5	89.8
Object (OBJ)	949	85.9	66.9	70.4	68.6
Preposition dependent (PR)	1137	93.6	91.8	93.2	92.5
Predicative (PRD)	254	89.8	65.7	73.3	69.3
Root (ROOT)	635	88.7	76.8	88.7	82.3
Subject (SUBJ)	600	82.7	68.9	66.8	67.8
Complementizer dependent (UK)	418	88.1	87.5	88.7	88.1
Verb group (VC)	397	79.8	71.2	72.5	71.8
Total	7748	81.3	73.6	73.6	73.6

means that the treebank must be converted to dependency structures using the same kind of head percolation tables and inference rules that were used for the English data, except that for Bulgarian the converted treebank also contains non-projective dependencies. In most cases, these involve subordinate *da*-clauses, where we often find subject-to-object or object-to-object raising. In these cases, we have taken *da* to be the head of the subordinate clause with the main verb dependent on *da* and the raised subject or object dependent on the main verb. More details about the conversion can be found in Marinov and Nivre (2005).

Experiments were performed with several models but the highest accuracy was achieved with a variant of the standard model, where all lexical features are based on suffixes of length 6, rather than the full word forms. That is, every lexical feature $w(a)$ (with address function a) is replaced by $s_6(w(a))$ (cf. section 2.3). The overall accuracy scores for Bulgarian are 81.3% (AS_U) and 73.6% (AS_L). Using suffixes instead of full forms makes the data less sparse, which can be an advantage for languages with limited amounts of data, especially if the endings of content words can be expected to carry syntactically relevant information. The optimal suffix length can be determined using cross-validation, and a length of 6 seems to work well for several languages, presumably because it captures the informative endings of content words while leaving most function words intact.

Table 6 gives accuracy, precision, recall and balanced F measures for individual dependency types. The overall trend is the same as for Swedish and English in that dependency relations involving function words tend to have higher accuracy than relations holding primarily between content words. Thus, the highest ranking dependency types with respect to the F measure are PR (preposition → noun) and UK (complementizer → verb), together with ID (multi-word unit), which in the Bulgarian data includes verbs taking the reflexive/possessive pronouns *se* and *si*. Further down the list we find as expected the major verb complement types OBJ (object) and PRD (predicative complement) but also SUBJ (subject), which has

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

considerably lower accuracy than the corresponding type in Swedish and English. This is a reflection of the more flexible word order in Bulgarian.

Other dependency types that are ranked lower for Bulgarian than for the other languages considered so far are DET (noun → determiner) and VC (auxiliary verb ← main verb). In the former case, since Bulgarian lacks free-standing determiners like English *the*, this category was reserved for demonstratives (*this*, *that*, etc.), which occurred infrequently. In the latter case, this again seems to be related to word order properties, allowing the verbs to be separated by adverbials or even subordinate clauses (which will also lead the parser to erroneously connect verbs that belong to different clauses). Finally, we note that coordinate structures (CC) and adverbials (ADV) have very low accuracy (with an F measure below 60%). For adverbials, one possible error source is the fact that many adverbs coincide in form with the third person singular form of adjectives.

There are no other published results for parsing Bulgarian, except for a paper by Tanev and Mitkov (2002), who report precision and recall in the low 60s for a rule-based parser. However, this parser has only been tested on 600 syntactic phrases, as compared to the 5080 sentences used in the present study, so it is very difficult to draw any conclusions about the relative quality of the parsers.

4.2.4 Chinese

The Chinese data are taken from the Penn Chinese Treebank (CTB), version 5.1 (Xue *et al.* 2005), and the texts are mostly from Xinhua newswire, Sinorama news magazine and Hong Kong News. The annotation of CTB is based on a combination of constituent structure and grammatical functions and has been converted in the same way as the data for English and Bulgarian, with a head percolation table created by a native speaker for the purpose of machine translation. Dependency type labels have been inferred using an adapted version of the rules developed for English, which is possible given that the treebank annotation scheme for CTB is modeled after that for the English Penn Treebank. More details about the conversion can be found in Hall (2006).

One often underestimated parameter in parser evaluation is the division of data into training, development and evaluation sets. Levy and Manning (2003) report up to 10% difference in parsing accuracy for different splits of CTB 2.0. We have used the same pseudo-randomized split as for Swedish (cf. section 4.2.1), with sections 1–8 for training, section 9 for validation, and section 0 for final evaluation. The results presented in this article are based on gold-standard word segmentation and part-of-speech tagging.

The best performing model for Chinese is the standard one and the same goes for TiMBL settings except that $k = 6$ and $l = 8$. Table 7 presents the unlabeled attachment score (AS_U), labeled precision (P), recall (R) and F measure (F) for individual dependency types in the Chinese data. We see that the overall accuracy scores for Chinese are $AS_U = 81.1\%$ and $AS_L = 79.2\%$, and the difference between labeled and unlabeled accuracy is generally very small also on the level of individual dependency types, with a few notable exceptions. Both SBJ (subject) and VC (verb

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 7. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Chinese (held-out test set, section 0)*

Dependency Type	*n*	AS_U	P	R	F
Adjective/adverb modifier (AMOD)	1503	95.2	95.8	94.5	95.1
Other (DEP)	2999	90.5	92.4	89.5	90.9
Noun modifier (NMOD)	13046	85.4	86.3	85.2	85.7
Object (OBJ)	2802	86.0	82.8	85.3	84.0
Preposition modifier (PMOD)	1839	77.3	81.3	77.2	79.2
Predicative (PRD)	467	78.8	81.4	76.0	78.6
Root (ROOT)	1880	70.5	55.4	70.5	62.0
Complementizer dependent (SBAR)	1296	83.6	83.6	83.3	83.4
Subject (SBJ)	3242	83.2	73.3	78.5	75.8
Verb group (VC)	940	80.0	76.0	75.1	75.5
Adverbial (VMOD)	12043	72.6	71.3	68.8	70.0
Total	42057	81.1	79.2	79.2	79.2

chain) have considerably lower labeled F measure than unlabeled attachment score, which indicates that these relations are difficult to classify correctly even if the head-dependent relations are assigned correctly. For the special ROOT label, we find a very low precision, which reflects fragmentation in the output (since too many tokens remain attached to the special root node), but even the recall is substantially lower than for any other language considered so far. This may indicate that the feature model has not yet been properly optimized for Chinese, but it may also indicate a problem with the arc-eager parsing strategy adopted in all the experiments.

It is rather difficult to compare results on parsing accuracy for Chinese because of different data sets, word segmentation strategies, dependency conversion methods, and data splits. But the unlabeled attachment score obtained in our experiments is within 5% of the best reported results for CTB (Cheng *et al.* 2005).

4.2.5 Czech

The Prague Dependency Treebank (PDT) consists of 1.5M words of newspaper text, annotated on three levels, the morphological, analytical and tectogrammatical levels (Hajič *et al.* 2001). Our experiments all concern the analytical annotation, which uses a set of 28 surface-oriented grammatical functions (Böhmová *et al.* 2003). Unlike the treebanks discussed so far, PDT is a genuine dependency treebank also including non-projective dependencies.

The best results for Czech are again based on the standard model with standard settings, although it should be acknowledged that the sheer size of the Czech data sets makes it hard to perform extensive optimization of feature model and learning algorithm parameters. The experiments are based on the designated training and development sets in the treebank distribution, with final evaluation on the separate test set (Hajič *et al.* 2001).

Although less than 2% of all arcs in the training data are non-projective, they are distributed over as many as 23% of the sentences. It follows that the configuration of

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 8. *Attachment score (AS_U), precision (P), recall (R) and F measure for selected dependency types for Czech (held-out test set, etest section)*

Dependency Type	n	AS_U	P	R	F
Adverbial (Adv)	12948	88.0	75.3	74.2	74.7
Attribute (Atr)	36239	86.9	82.8	83.6	83.2
Subordinate conjunction (AuxC)	2055	75.9	80.5	75.8	78.1
Preposition (AuxP)	12658	72.0	73.7	71.7	72.4
Auxiliary Verb (AuxV)	1747	85.6	91.3	85.1	88.2
Rhematizer (AuxZ)	1962	76.9	70.0	73.9	71.9
Coordination node (Coord)	2716	31.4	39.0	31.0	34.5
Ellipsis handling (ExD)	2529	59.9	43.6	31.2	36.4
Object (Obj)	10480	81.6	66.5	62.6	64.5
Nominal predicate's nominal part (Pnom)	1668	80.2	63.8	70.3	66.9
Main predicate (Pred)	2892	58.2	45.7	53.1	49.1
Root node (ROOT)	7462	77.0	61.5	77.0	68.4
Subject (Sb)	9364	79.8	68.6	69.8	69.3
Total	108128	80.1	72.8	72.8	72.8

MaltParser used for all languages, constructing only projective graphs, cannot even in theory achieve an exact match for these sentences. To cope with non-projectivity, the concept of pseudo-projective parsing was introduced and evaluated in Nivre and Nilsson (2005). An overview of this approach is given in section 3.4.

Using non-projective training data, i.e., without applying any tree transformations and encodings, the overall accuracy scores are $AS_U = 78.5\%$ and $AS_L = 71.3\%$. By simply transforming all non-projective sentences to projective, without encoding the transformations in dependency type labels (baseline), an improvement is achieved for both $AS_U = 79.1\%$ and $AS_L = 72.0\%$. This indicates that it helps to make the input conform to the definition of projectivity, despite the fact that the trees are distorted and that it is not possible to recover non-projective arcs in the output of the parser.

In Nivre and Nilsson (2005), three types of encoding schemes were evaluated in order to recover the non-projective structure by an inverse transformation. The encodings increase the burden on the parser, since it now also has to distinguish between pseudo-projective arcs and the original projective arcs. The differences between different encodings are small and not statistically significant, but all three encodings increase both labeled and unlabeled attachment score in comparison both to the projectivized baseline and to the use of non-projective training data (all differences being significant beyond the 0.01 level according to McNemar's test). Compared to the projectivized baseline, the improvement is as high as 1 precentage point for $AS_U = 80.1\%$ and 0.8 percentage points for $AS_L = 72.8\%$.

A closer look at the 13 most frequent dependency types in table 8 reveals a larger drop from unlabeled to labeled accuracy compared to other languages such as English and Chinese. This is partly a result of the more fine-grained set of dependency types for Czech, but the more flexible word order for major clause constituents like Sb (subject) and Obj (object) is probably important as well. On the other hand, dependents of the types AuxC (subordinate conjunction), AuxP

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

(preposition), AuxV (auxiliary verb) or Coord (conjunction) actually have a higher F measure than AS_U, due to higher precision. In contrast to Sb and Obj, these dependents all come from closed word classes, which often uniquely identifies the dependency type. In addition, it is worth noting the surprisingly low accuracy for Coord, lower than for most other languages. This may indicate that the analysis of coordination in PDT, treating the coordinating conjunction as the head, does not interact well with the parsing strategy and/or feature models adopted in the experiments.[14]

We are not aware of any published results for labeled accuracy, but the unlabeled attachment score obtained is about 5% lower than the best results reported for a single parser, using the parser of Charniak (2000), adapted for Czech, with corrective post-processing to recover non-projective dependencies (Hall and Novák 2005).

4.2.6 *Danish*

The Danish experiments are based on the Danish Dependency Treebank (DDT), which is based on a subset of the Danish PAROLE corpus and annotated according to the theory of Discontinuous Grammar (Kromann 2003). This annotation involves *primary* dependencies, capturing grammatical functions, and *secondary* dependencies, capturing other relations such as co-reference. Our experiments only concern primary dependencies, since including secondary dependencies as well would have violated the SINGLE-HEAD constraint (cf. section 2.1), but the dependency type set is still the most fine-grained of all, with 54 distinct dependency types. The annotation is not restricted to projective dependency graphs, and while only about 1% of all dependencies are non-projective, the proportion of sentences that contain at least one non-projective dependency is as high as 15%.

The treebank has been divided into training, validation and test sets using the same pseudo-randomized splitting method described earlier for Swedish and Chinese. The training data for the experiments have been projectivized in the same way as the Czech data, with a similar improvement compared to the use of non-projective training data. However, none of the encoding schemes for recovering non-projective dependencies in the output of the parser led to any improvement in accuracy (nor to any degradation), which is probably due to the fact that the training data for non-projective dependencies are much more sparse than for Czech.

The best performing model for Danish is a modification of the standard model, where the feature $w(\tau_1)$ (the word form of the first lookahead token) is omitted, and the feature $w(h(\sigma_0))$ (the word form of the head of the top token) is replaced by the suffix feature $s_6(w(h(\sigma_0)))$. The TiMBL settings are standard. The overall accuracy scores for Danish are $AS_U = 85.6\%$ and $AS_L = 79.5\%$.[15] The relatively wide gap between unlabeled and labeled accuracy is probably due mainly to the fine-grained

[14] In more recent work, Nilsson *et al.* (2006) have shown how parsing accuracy for coordination in Czech can be improved by transforming the representations so that coordinating conjunctions are not treated as heads internally.

[15] The labeled attachment score is slightly lower than the one published in Nivre and Hall (2005), where results were based on the development test set.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 9. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Danish, $n \geq 10$ (held-out test set, section 0)*

Dependency Type	n	AS_U	P	R	F
Elliptic modifier (<MOD>)	11	45.5	0.0	0.0	–
Root (ROOT)	554	91.2	87.5	91.2	89.3
Adjectival object (AOBJ)	17	70.6	50.0	17.6	26.0
Parenthetical apposition (APPA)	20	50.0	53.8	35.0	42.4
Restrictive apposition (APPR)	23	43.5	69.2	39.1	50.0
Adverbial object (AVOBJ)	19	78.9	30.8	21.1	25.0
Conjunct (CONJ)	399	80.7	77.4	77.4	77.4
Coordinator (COORD)	299	75.6	75.4	74.9	75.1
Direct object (DOBJ)	504	90.1	77.5	77.8	77.6
Expletive subject (EXPL)	36	100.0	89.5	94.4	91.9
Indirect object (IOBJ)	13	100.0	66.7	15.4	25.0
List item (LIST)	17	29.4	57.1	23.5	33.3
Locative object (LOBJ)	117	88.0	53.0	45.3	48.8
Modifier (MOD)	1809	77.9	70.6	71.0	70.8
Parenthetical modifier (MODP)	15	26.7	0.0	0.0	–
Modifying proper name (NAME)	13	30.8	22.2	15.4	18.2
Modifying first name (NAMEF)	96	91.7	79.8	90.6	84.9
Nominal object (NOBJ)	1831	92.6	88.5	91.6	90.0
Verbal particle (PART)	21	85.7	62.5	23.8	34.5
Prepositional object (POBJ)	501	79.6	64.4	66.7	65.5
Possessed (POSSD)	171	90.1	91.3	85.4	87.1
Predicative (PRED)	251	86.5	62.0	65.7	63.8
Quotation object (QOBJ)	37	78.4	51.9	75.7	61.6
Relative clause modification (REL)	131	59.5	62.7	56.5	59.4
Subject (SUBJ)	892	93.6	90.7	90.7	90.7
Title of person (TITLE)	19	78.9	63.6	73.7	68.3
Temporal adjunct (TOBJ)	16	50.0	62.5	31.3	41.7
Verbal object (VOBJ)	635	95.1	92.7	93.4	93.0
Direct quotation (XPL)	12	0.25	0.0	0.0	–
Total	8530	85.6	79.5	79.5	79.5

nature of the dependency type set in combination with a relatively small training data set.

Table 9 shows the unlabeled attachment score (AS_U), precision (P), recall (R) and F measure (F) for dependency types occurring at least 10 times in the test set. It is clear that low-frequency types ($n < 100$) generally have very low labeled precision and recall, despite sometimes having a quite high unlabeled accuracy. A striking example is indirect object (IOBJ), which has perfect unlabeled accuracy but only 15% labeled recall. Concentrating on types that occur at least 100 times in the test set, we see a pattern that is very similar to the one observed for the closely related language Swedish, despite important differences in the style of annotation. Thus, we can observe a very high accuracy ($F \geq 90$) for dependencies involving function words, notably VOBJ, which includes dependencies linking verbs to function words (auxiliary verb → main verb, marker → infinitive, complementizer → verb), and NOBJ, which includes dependencies linking prepositions and determiners to nominals, but also for subjects, both normal subjects (SUBJ) and the much less frequent expletive subjects (EXPL), and roots (ROOT). Furthermore, we see that other arguments of the verb (DOBJ, IOBJ, LOBJ, PRED) have a high unlabeled accuracy but (sometimes substantially) lower labeled accuracy, while the generic

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

adjunct type MOD has lower accuracy, both labeled and unlabeled. Finally, both Danish and Swedish have comparatively high accuracy for coordination, which in Danish is split into CC (conjunct → coordinator) and COORD (conjunct$_i$ → conjunct$_{i+1}$). Compared to the results for Czech, this indicates that an analysis of coordination where a conjunct, rather than the coordinator, is treated as the head is easier to cope with for the parser.

McDonald and Pereira (2006) report an unlabeled attachment score for primary dependency types in DDT of 86.8%.[16] However, these results are based on gold standard part-of-speech tags, whereas our experiments use automatically assigned tags with an accuracy rate of 96.3%. Replicating the experiment with gold standard tags, using the same feature model and parameter settings, results in an unlabeled attachment score of 87.3%, which indicates that MaltParser gives state-of-the-art performance for Danish.

4.2.7 Dutch

The Dutch experiments are based on the Alpino Treebank (Beek *et al.* 2003). The text material (186k non-punctuation tokens) consists primarily of two sections of newspaper text (125k and 21k), plus two smaller segments containing questions (21k) and (in part) manually constructed sentences for parser development and annotation guide examples (19k). As the latter type of material is atypical, it is only used for training purposes, whereas the smaller newspaper text section is used as held out material for final testing.

The syntactic annotation of the Alpino Treebank is a mix of constituent structure and dependency relations, nearly identical to the syntactic annotation of the Spoken Dutch Corpus (Wouden *et al.* 2002). It was converted to a pure dependency structure employing a head percolation table, removing secondary relations as indicated by traces. Multi-word units, consisting of a sequence of words without any further syntactic analysis, were concatenated into a single word using underscores. Finally, non-projective structures were projectivized using the same baseline procedure as for Danish (i.e., without extending the dependency type labels or attempting to recover non-projective dependencies in the output of the parser). Since the original part-of-speech tags in the Alpino Treebank are coarse-grained and lack any additional feature information besides the word class, all tokens were retagged with the memory-based tagger for Dutch (Daelemans *et al.* 2003).

Ten-fold cross-validation was used to manually optimize the TiMBL settings. Experimentation confirmed that the standard settings with MVDM, no feature weighting, and distance weighted class voting generally performs best. However, choosing a higher value for k ($k = 10$) usually gives an improvement of one to two percentage points for Dutch. The results obtained on held out data using the standard model are $AS_U = 84.7\%$ and $AS_L = 79.2\%$. The relatively large

[16] It should be pointed out that McDonald and Pereira (2006) also consider secondary dependency arcs, which are beyond the reach of MaltParser in its current configuration, and that the result reported is actually the highest precision of their parser when restricted to primary dependencies.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 10. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Dutch (held-out test set)*

Dependency Type	*n*	AS_U	P	R	F
Apposition (APP)	299	73.6	78.8	71.9	75.2
Body of embedded clause (BODY)	88	85.8	83.9	84.3	84.1
Conjunct (CNJ)	997	70.0	72.8	68.6	70.6
Coordinator (CRD)	9	44.4	28.6	22.2	25.0
Determiner (DET)	3239	97.2	96.1	96.9	97.0
Closing element of circumposition (HDF)	13	53.8	70.0	53.8	60.8
Locative/directional complement (LD)	239	68.6	40.2	21.3	27.9
Measure complement (ME)	33	72.7	69.2	54.5	61.0
Modifier (MOD)	5069	78.3	71.1	73.9	72.5
Object of adjective or adverb (OBCOMP)	51	74.5	90.0	52.9	66.6
Direct object (OBJ1)	3392	90.3	86.0	86.4	86.2
Indirect object (OBJ2)	56	80.4	77.8	12.5	21.5
Prepositional complement (PC)	344	73.8	51.6	28.5	36.7
Suppletive object (POBJ1)	14	78.6	33.3	35.7	34.5
Predicative complement (PREDC)	428	79.4	65.6	56.1	60.5
Predicate modifier (PREDM)	61	65.6	54.5	9.8	16.6
Root (ROOT)	1874	82.7	70.8	82.7	76.3
Obligatory reflexive object (SE)	53	83.0	72.2	73.6	72.9
Subject (SU)	186	85.2	80.8	78.1	79.4
Suppletive subject (SUP)	19	89.5	45.0	47.4	46.2
Separable verbal particle (SVP)	259	85.3	69.6	61.8	65.5
Verbal complement (VC)	1074	89.0	80.4	85.6	82.9
Total	20263	84.7	79.2	79.2	79.2

gap between the labeled and unlabeled scores may be attributed to the relatively fine-grained set of dependency labels. Table 10 gives unlabeled attachment score (AS_U), labeled precision (P), recall (R) and F measure (F) for individual dependency types. We can observe a general trend towards better scores for the more frequent dependency labels, but there are notable exceptions such as the relatively high score for the infrequently occurring SE (reflexive object) and the low score on the more frequent PC (prepositional complement) and LD (locative/directional complement).

As for several other languages, we can distinguish three groups with high, medium and low F measures respectively. The high score set ($F > 80\%$) includes the dependency relations indicated by closed class words: DET (determiner → noun), VC (auxiliary verb → main verb), and BODY (complementizer → verb). Somewhat surprisingly, this group also includes OBJ1 (direct object), perhaps because this is the second most frequent dependency relation.

The low score group ($F < 60\%$) includes the rather infrequent suppletive subject (SUP) and object (POBJ1). Furthermore, it involves four classes which are canonically expressed in the form of a prepositional phrase – PC (prepositional complement), OBJ2 (indirect object), LD (locative/directional complement) and PREDM (predicate modifier) – and where the sometimes subtle distinction is often of a semantic rather than a syntactic nature. The fact that coordinator (CRD) is also in the low score group is somewhat counter-intuitive, because it is indicated by a closed word class, normally the word *en* 'and', but the result is consistent with

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

the low accuracy for coordination in Czech, given that both treebanks treat the coordinating conjunction as the head of a coordinate structure.

The remaining 11 types belong to the medium score group ($60\% < F < 80\%$), which includes the by far most frequent class MOD (modifier). It is interesting to note that the scores for a conceptually difficult class like APP (apposition) are still quite good. The same goes for the potentially highly ambiguous CONJ (conjunct), although there seems to be a trade-off here with the low scores noted for CRD earlier.

The Alpino parser is a rule-based, HPSG-style parser that is currently the state-of-art parser for Dutch (Bouma *et al.* 2001). It has an extensive and detailed lexicon (including, e.g., subcategorization information) and a MaxEnt-based disambiguation module. Its output is in the same format as the Alpino Treebank. We used it to parse the held out material and converted the parse trees to dependency structures, using exactly the same procedure as for converting the treebank, which includes transforming non-projective to projective structures. Evaluation resulted in the scores $AS_U = 93.2\%$ and $AS_L = 91.2\%$. Clearly, there is still a substantial gap between the two parsers. Also, the Alpino parser provides additional information, e.g., traces and non-projective analyses, which is ignored here. Yet, given all the effort invested in building the Alpino grammar, lexicon, and disambiguation strategy, it is interesting to see that its performance can be approximated by a purely inductive approach using fairly limited amounts of data.

4.2.8 German

The experiments for German are based on the Tübingen Treebank of Written German (TüBa-D/Z) (Telljohann *et al.* 2005). The treebank is based on issues of the German daily newspaper 'die tageszeitung' (taz) that appeared in April and May of 1999. The annotation of the treebank is constituency-based, but it is augmented by function-argument structure on all levels, which allows a straightforward conversion to dependencies for most phenomena. Heuristics are used only for apposition, embedded infinitive clauses, and nominal postmodifications. Long-distance relations, which are annotated in the constituency model via special labels, are translated into non-projective dependencies. The set of dependency types is modeled after the one used for the Constraint Dependency Grammar for German (Foth *et al.* 2004), a manually written dependency grammar for German.

The best performing model for German modifies the standard model by omitting the two lexical features $w(h(\sigma_0))$ and $w(\tau_1)$ and by adding the part-of-speech of an additional stack token $p(\sigma_2)$. The TiMBL settings for German deviate from the standard settings by using $k = 13$ and voting based on inverse linear weighting (IL).

The overall accuracy scores for German are $AS_U = 88.1\%$ and $AS_L = 83.4\%$. The (unlabeled) results are comparable to results by Foth *et al.* (2004), who reached 89.0% accuracy when parsing the NEGRA treebank (Skut *et al.* 1997), another treebank for German, which is also based on newspaper texts (but which uses a different constituency-based annotation scheme). The labeled results are considerably

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 11. *Attachment score (AS_U), precision (P), recall (R) and F measure for selected dependency types for German (mean of 10-fold cross-validation, frequency counts rounded to whole integers)*

Dependency Type	n	AS_U	P	R	F
Adverbial (ADV)	2762	80.5	78.7	79.3	79.0
Determiner (DET)	4485	99.1	99.1	99.0	99.0
Genitive modifier (GenMOD)	571	79.5	59.1	66.3	62.5
Accusative Object (AccOBJ)	1466	82.4	66.6	73.4	69.8
Dative Object (DatOBJ)	219	79.0	62.4	16.4	26.0
Genitive Object (GenOBJ)	4	78.0	16.7	6.8	9.7
Predicate (PRED)	549	84.8	69.6	64.3	66.8
PP complement (PPOBJ)	399	83.1	54.7	41.6	47.3
Relative clause (RelCL)	241	54.1	56.9	52.6	54.7
Subject (SUBJ)	2931	92.0	85.7	86.3	86.0
Total	32555	88.1	83.4	83.4	83.4

higher than constituency parsing results reported for German, which reach a labeled F measure of 75.3% when constituent nodes also include grammatical functions (Kübler *et al.* 2006).

Table 11 gives unlabeled attachment scores (AS_U), labeled precision (P), recall (R), and F measure (F) for selected dependency types. The overall trends are very similar to what we have observed for other languages, notably Germanic languages like Swedish and Danish. For example, both determiners (DET) and adverbials (ADV) have labeled and unlabeled accuracy at about the same level (although considerably higher for DET than for ADV), while arguments of the verb (AccOBJ, DatOBJ, GenOBJ, PRED and PPOBJ) have substantially better unlabeled than labeled accuracy. One difference, compared to Danish and Swedish, is that the lower labeled accuracy also affects subjects (SUBJ), which is probably a reflection of the fact that German exhibits freer word order thanks to case marking. The relatively low labeled accuracy for different case-marked arguments is also an indication that the parser would benefit from morphological information, which is currently not included in the German part-of-speech tags.

Contrary to expectations that, with growing data size, adding more lexical features would improve performance, experiments with all the lexical features of the standard model showed a decrease in performance by 1.5 percentage points. The hypothesis that this decrease is due to data sparseness is refuted by experiments with only 2000 sentences for training, where the decrease in performance is only 7.5%. These results are consistent with those of Dubey and Keller (2003), who found that lexicalizing a PCFG grammar for NEGRA results in a decrease in performance, although it should be remembered that the first two lexical features are beneficial in the case of MaltParser.

4.2.9 Italian

The Italian treebank used in the experiments is the Turin University Treebank (TUT) (Bosco 2004), consisting of 1500 sentences and 41771 tokens. It is balanced

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 12. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Italian (mean of 10-fold cross-validation, frequency counts rounded to whole integers)*

Dependency Type	*n*	AS_U	P	R	F
Apposition (APPOSITION)	69	44.4	54.2	47.8	50.8
Argument (ARG)	1351	95.0	92.7	94.5	93.6
Auxiliary verb (AUX)	96	92.1	90.5	94.2	92.3
Part of expression (CONTIN)	75	86.9	78.2	54.4	64.2
Coordination (COORDINATOR)	271	66.6	63.6	63.6	63.6
Other (DEPENDENT)	1	40.0	0.0	0.0	–
Reflexive complement (EMPTYCOMPL)	15	94.5	35.7	50.0	41.7
Indirect complement (INDCOMPL)	82	85.9	70.4	47.5	56.7
Indirect object (INDOBJ)	18	81.5	33.3	33.3	33.3
Interjection (INTERJECTION)	1	20.0	0.0	0.0	–
Object (OBJ)	222	84.9	33.3	33.3	33.3
Predicative complement (PREDCOMPL)	52	78.4	54.3	37.3	44.2
Restrictive modifier (RMOD)	1013	74.3	69.5	70.2	69.8
Subject (SUBJ)	256	75.5	64.8	58.6	61.5
Root (TOP)	150	75.5	63.5	77.2	69.7
Adverbial extraction (VISITOR)	13	74.6	0.0	0.0	–
Total	3683	82.9	75.7	75.7	75.7

over genres with 60% newspaper text, 30% legal text, and 10% from novels and academic literature. The dependency annotation involves traces in order to avoid non-projective structures, although there is in fact a certain number of non-projective trees in the treebank.

The treebank has been converted to the format required by MaltParser without significant loss of linguistic information, as described in Chanev (2005), replacing traces if necessary by (possibly non-projective) dependency arcs. The dependency tag set was reduced from 283 to 17 distinct tags, keeping only information about syntactic dependency relations. The training data were projectivized using the same procedure as for Danish and Dutch and tagged for part-of-speech using TnT (Brants 2000). All experiments were performed using 10-fold cross-validation with a randomized split.

The best performing feature model for Italian is the standard model, although several simpler models give nearly the same results. The accuracy scores for Italian are $AS_U = 82.9\%$ and $AS_L = 75.7\%$, and table 12 shows the accuracy obtained for different dependency types. It is striking that there are only two types that obtain a really high accuracy in the Italian data, the type ARG, which is usually used for relations between articles and nouns or prepositions and articles, and the type AUX, which is used for auxiliary verbs. While these two types have a labeled F measure well above 90%, no other type has a score higher than 70%. There is also a set of low-frequency types that all have zero recall and precision. The relatively low labeled accuracy for most dependency types in Italian is undoubtedly due partly to sparse data, but it is also relevant that the inventory of dependency types is more semantically oriented than for most other languages.

For Italian there are not any published results for statistical dependency parsing except the preliminary results for MaltParser reported in Chanev (2005). Compared

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

to Corazza *et al.* (2004), where state-of-the-art constituency parsers were tested on the Italian Syntactic-Semantic Treebank (Montemagni *et al.* 2003), an improvement seems to have been achieved, although it is not straightforward to compare evaluation metrics for constituency and dependency parsing. A more relevant comparison is the rule-based parser of Lesmo *et al.* (2002), which uses the TUT dependency type set and which has been reported to achieve a labeled attachment score of 76.65% when evaluated during the development of the treebank. Since this is within a percentage point of the results reported in this article and the evaluation is based on the same kind of data, it seems clear that MaltParser achieves highly competitive results for Italian.

4.2.10 Turkish

The Turkish Treebank (Oflazer *et al.* 2003), created by Metu and Sabancı Universities is used in the experiments for Turkish. This treebank is composed of 5635 sentences, annotated with dependency structures, of which 7.2% are non-projective (not counting punctuation that is not connected to a head). As can be seen from table 1, even though the number of sentences in the Turkish Treebank is in the same range as for Danish, Swedish and Bulgarian, the number of words is considerably smaller (54k as opposed to 70–100k for the other treebanks). This significant difference arises from the very rich morphological structure of the language due to which a word may sometimes correspond to a whole sentence in another language.

As a result of their agglutinative morphology, Turkish words can change their main part-of-speech after the concatenation of multiple suffixes. This structure is represented in the treebank by dividing words into inflectional groups (IG). The root and derived forms of a word are represented by different IGs separated from each other by derivational boundaries (DB). Each IG is annotated with its own part-of-speech and inflectional features, as illustrated in the following example:[17]

okulunuzdaydı
(he was at your school)
okulunuzda DB ydı
okul+Noun+A3sg+P2pl+Loc DB +Verb+Zero+Past+A3sg
IG_1 IG_2

The part-of-speech of the stem of the word *okulunuzdaydı* is a noun, from which a verb is derived in a separate IG. In the treebank, dependencies hold between specific IGs of the dependent and head word.

For the parsing experiments, we have concatenated IGs into word forms to get a word-based tokenization and used a reduced version of the part-of-speech tagset given by the treebank, very similar to the reduced tagset used in the parser of Eryiğit and Oflazer (2006). For each word, we use the part-of-speech of each IG and in addition include the case and possessive information if the stem is a noun or pronoun. Using this approach, the tag of the word *okulunuzdaydı* becomes

[17] A3sg = 3sg number agreement, P2pl = 2pl possessive agreement, Loc = locative case.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Table 13. *Attachment score (AS_U), precision (P), recall (R) and F measure per dependency type for Turkish (mean of 10-fold cross-validation, frequency counts rounded to whole integers)*

Dependency Type	*n*	AS_U	P	R	F
ABLATIVE.ADJUNCT	52	82.8	58.8	54.7	56.7
APPOSITION	190	40.6	8.5	5.9	7.0
CLASSIFIER	205	87.0	72.8	70.4	71.6
COLLOCATION	5	41.2	25.0	5.9	9.5
COORDINATION	81	53.6	56.0	48.6	52.0
DATIVE.ADJUNCT	136	86.8	55.0	54.9	54.9
DETERMINER	195	91.1	83.7	85.8	84.7
EQU.ADJUNCT	2	62.5	0.0	0.0	–
ETOL	1	70.0	0.0	0.0	–
FOCUS.PARTICLE	2	78.3	0.0	0.0	–
INSTRUMENTAL.ADJUNCT	27	71.6	34.7	18.8	24.4
INTENSIFIER	90	93.9	82.9	86.0	84.4
LOCATIVE.ADJUNCT	114	73.0	59.5	58.1	58.8
MODIFIER	1168	76.5	68.7	68.2	68.4
NEGATIVE.PARTICLE	16	90.0	89.6	80.6	84.9
OBJECT	796	88.3	68.6	69.4	69.0
POSSESSOR	152	80.0	81.7	69.9	75.3
QUESTION.PARTICLE	29	93.8	85.9	80.2	83.0
RELATIVIZER	8	91.8	54.5	49.4	51.8
ROOT	2	0.0	0.0	0.0	–
SENTENCE.MODIFIER	59	52.4	33.8	47.6	39.5
SENTENCE	725	91.2	84.4	89.2	86.7
SUBJECT	448	72.0	50.7	50.8	50.7
VOCATIVE	24	51.0	20.4	19.1	19.7
Total	4357	81.6	69.0	69.0	69.0

Noun+P2pl+Loc+Verb. Even after this reduction, the tagset contains 484 distinct tags, making it by far the biggest tagset used in the experiments.

The best performing model for Turkish omits five of the features of the standard model, three part-of-speech features ($p(\sigma_1)$, $p(\tau_2)$, $p(\tau_3)$) and two lexical features ($w(h(\sigma_0))$, $w(\tau_1)$). In addition, the stem of a word is used as its word form in lexical features. This leads to an accuracy of $AS_U = 81.6\%$ and $AS_L = 69.0\%$. These are the mean results obtained after 10-fold cross-validation.

Table 13 gives unlabeled attachment scores (AS_U), labeled precision (P), recall (R), and F measure (F) for individual dependency types. First of all, we see that types with a frequency below 5 in the test set have very low labeled accuracy, which is consistent with results reported for other languages earlier. Secondly, we may note that the frequency of tokens analyzed as roots (ROOT) is very low, which is a consequence of the fact that punctuation tokens are excluded in evaluation, since final punctuation is generally treated as the root node of a sentence in the Turkish Treebank.[18] Therefore, the closest correspondent to ROOT for other languages is SENTENCE, which is the type assigned to a token dependent on the final punctuation token (normally the final verb of the sentence) and which has a very

[18] The few roots that do occur are unconnected words that give rise to non-projective dependency structures.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

high accuracy, on a par with the ROOT type for most other languages. Finally, there is a clear tendency that dependency types with high accuracy (INTENSIFIER, QUESTION.PARTICLE, RELATIVIZER, SENTENCE, DETERMINER, NEGATIVE.PARTICLE) are types that are generally adjacent to their head, whereas types with lower accuracy (COORDINATION, SENTENCE.MODIFIER, APPOSITION, COLLOCATION, VOCATIVE) are types that are either more distant or hard to differentiate from other types.

The only comparable results for Turkish are for the unlexicalized dependency parser of Eryiğit and Oflazer (2006). These results are based on a selected subset of the treebank sentences containing only projective dependencies with the heads residing on the right side of the dependents and the main evaluation metrics are based on IGs rather than words, but word-based scores are presented for the purpose of comparison with a top score of $AS_U = 81.2\%$. Applying MaltParser with the best feature model to the same subset of the treebank resulted in an unlabeled attachment score of 84.0%, which is a substantial improvement.[19]

4.3 Discussion

Although MaltParser achieves an unlabeled dependency accuracy above 80% for all languages, there is also a considerable range of variation, which seems to correlate fairly well with the linguistic dimensions of morphological richness and word order flexibility, exemplified by high accuracy for English and lower accuracy for Czech, which represent extreme positions on these scales. Given that English and Czech are also the languages with the largest data sets, the linguistic properties seem to be more important than the amount of data available. Another influencing factor is the level of detail of the dependency annotation, as given by the number of dependency types used, where Czech has a more fine-grained classification than English. However, Danish has an even more fine-grained classification but still comes out with higher parsing accuracy than Czech, despite a much smaller training data set.

If morphological richness and word order flexibility are indeed the most important factors determining parsing accuracy, the results for German are surprisingly good, given that German has both richer morphology and freer word order than English. On the other hand, the results for Chinese are on the low side. This points to another important factor, namely the complexity of the sentences included in the treebank data, which can be roughly approximated by considering the mean sentence length in the sample. Here we see that Chinese has the second highest value of all languages, while the sentence length for German is at least considerably lower than for English. At the same time, we have to remember that the number of words per sentence is not strictly comparable between languages with different morphological properties, as illustrated especially by the data for Turkish (cf. section 4.2.10).

[19] Strictly speaking, the subset used by Eryiğit and Oflazer (2006) only contains non-crossing dependencies, although it does contain punctuation that is not connected to other tokens. In order to make these graphs projective, the punctuation tokens were attached to the immediately following word. However, since punctuation is excluded in all evaluation scores, this nevertheless seems like a fair comparison.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Comparing individual dependency types across languages is very difficult, given the diversity in annotation, but a few recurrent patterns are clearly discernible. The first is that dependencies involving function words generally have the highest accuracy. The second is that core arguments of the verb often have high unlabeled accuracy but lower labeled accuracy, with the possible exception of subjects, which have high labeled accuracy in languages where they are distinguished configurationally. The third is that the parsing accuracy for coordinate structures tends to be higher if the dependency analysis treats conjuncts, rather than coordinating conjunctions, as heads.

Needless to say, a more detailed error analysis will be needed before we can draw any reliable conclusions about the influence of different factors, so the tentative conclusions advanced here are best regarded as conjectures to be corroborated or refuted by future research. However, given the fact that unlabeled dependency accuracy is consistently above 80%, the parsing methodology has proven to be relatively insensitive to differences in language typology as well as in annotation schemes. Moreover, respectable results can be obtained also with fairly limited amounts of data, as illustrated in particular by the results for Italian and Turkish.

Finally, we note that MaltParser achieves state-of-the-art performance for most of the languages investigated in this article, although the possibility of comparison differs widely between languages. For English, Chinese, Czech and Dutch, parsing accuracy does not quite reach the highest level, but the difference is never more than about 5% (slightly more for Dutch).[20]

5 Conclusion

We have presented MaltParser, a data-driven system for dependency parsing that can be used to construct syntactic parsers for research purposes or for practical language technology applications. Experimental evaluation using data from ten different languages shows that MaltParser generally achieves good parsing accuracy without language-specific enhancements and with fairly limited amounts of training data. Unlabeled dependency accuracy is consistently above 80% and the best results are normally within a 5% margin from the best performing parsers, where such comparisons are possible. MaltParser is freely available for research and educational purposes.

Acknowledgments

We want to express our gratitude for assistance with data sets, conversions and many other things to Christina Bosco, Yuchang Cheng, Yuan Ding, Jan Hajič,

[20] More recent work using SVM, rather than MBL, for discriminative learning has shown that this gap can be closed, and in the recent shared task of multilingual dependency parsing at the Tenth Conference on Computational Natural Language Learning (CoNLL-X), MaltParser was one of the two top performing systems (Buchholz and Marsi 2006; Nivre *et al.* 2006; Hall 2006).

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Matthias Trautner Kromann, Alberto Lavelli, Haitao Liu, Yuji Matsumoto, Ryan McDonald, Kemal Oflazer, Petya Osenova, Kiril Simov, Yannick Versley, Hiroyasu Yamada, and Daniel Zeman. We are also grateful for the support of GSLT (Swedish National Graduate School of Language Technology), TÜBİTAK (The Scientific and Technical Research Council of Turkey), and The Swedish Research Council. Finally, we want to thank our three anonymous reviewers for insightful comments and suggestions that helped us improve the final version of the article.

References

Van der Beek, L., Bouma, G., Malouf, R. and Van Noord, G. 2003. The Alpino Dependency Treebank. In Gaustad, T. (ed.) *Computational Linguistics in the Netherlands 2002. Selected Papers from the Thirteenth CLIN Meeting*, pp. 8–22. Rodopi.

Berwick, R. C. 1985. *The Acquisition of Syntactic Knowledge*. MIT Press.

Bikel, D. and Chiang, D. 2000. Two statistical parsing models applied to the Chinese Treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, pp. 1–6.

Black, E. and Garside, R. and Leech, G. (eds.) 1993. *Statistically-Driven Computer Grammars of English: The IBM/Lancaster Approach*. Rodopi.

Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R. and Roukos, S. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, pp. 31–37.

Blaheta, D. and Charniak, E. 2000. Assigning function tags to parsed text. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 234–240.

Böhmová, A., Hajič, J., Hajičová, E. and Hladká, B. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Abeillé, A. (ed.), *Treebanks: Building and Using Parsed Corpora*, pp. 103–127. Dordrecht: Kluwer.

Bosco, C. 2004. *A Grammatical Relation System for Treebank Annotation*. PhD thesis, Turin University.

Bouma, G., Van Noord, G. and Malouf, R. 2001. Alpino: Wide-coverage computational analysis of Dutch. In Daelemans, W., Sima'an, K., Veenstra, J. and Zavrel, J. (eds.) *Computational Linguistics in the Netherlands 2000. Selected Papers from the Eleventh CLIN Meeting*, pp. 45-59. Rodopi.

Brants, T. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP'2000)*, pp. 224–231.

Buchholz, S. and Marsi, E. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pp. 149–164.

Chanev, A. 2005. Portability of dependency parsing algorithms – an application for Italian. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 29–40.

Chang, C.-C. and Lin, C.-J. 2001. LIBSVM: A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Charniak, E. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 132–139.

Charniak, E. and Johnson, M. 2005. Coarse-to-fine *n*-best parsing and discriminative MaxEnt reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 173–180.

Cheng, Y., Asahara, M. and Matsumoto, Y. 2004. Deterministic dependency structure analyzer for Chinese. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 500–508.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Cheng, Y., Asahara, M. and Matsumoto, Y. 2004. Machine learning-based dependency analyzer for Chinese. In *Proceedings of International Conference on Chinese Computing (ICCC)*, pp. 66–73.

Cheng, Y., Asahara, M. and Matsumoto, Y. 2005. Chinese deterministic dependency analyzer: Examining effects of global features and root node finder. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pp. 17–24.

Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 16–23.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Collins, M. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 175–182.

Collins, M. and Duffy, N. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 263–270.

Collins, M., Hajič, J., Ramshaw, L. and Tillmann, C. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 505–512.

Collins, M. and Duffy, N. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1), 25–70.

Corazza, A., Lavelli, A., Satta, G. and Zanoli, R. 2004. Analyzing an Italian treebank with state-of-the-art statistical parsers. In *Proceedings of the Third Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 39–50.

Covington, M. A. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pp. 95–102.

Daelemans, W. and Van den Bosch, A. 2005. *Memory-Based Language Processing*. Cambridge University Press.

Daelemans, W., Zavrel, J., Van den Bosch, A. and Van der Sloot, K. 2003. MBT: Memory Based Tagger, version 2.0, Reference Guide. ILK Technical Report 03-13, Tilburg University.

Dubey, A. and Keller, F. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 96–103.

Einarsson, J. 1976. Talbankens skriftspråkskonkordans. Lund University, Department of Scandinavian Languages.

Eryiğit, G. and Oflazer, K. 2006. Statistical dependency parsing of Turkish. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 89–96.

Foth, K., Daum, M. and Menzel, W. 2004. A broad-coverage parser for German based on defeasible constraints. In *KONVENS 2004, Beiträge zur 7. Konferenz zur Verarbeitung natürlicher Sprache*, pp. 45–52.

Hajič, J., Vidova Hladka, B., Panevová, J., Hajičová, E., Sgall, P. and Pajas, P. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Hall, J. 2006. *MaltParser – An Architecture for Labeled Inductive Dependency Parsing*. Licentitate thesis, Växjö University.

Hall, K. and Novák, V. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pp. 42–52.

Hudson, R. A. 1990. *English Word Grammar*. Blackwell.

Johnson, M., Geman, S., Canon, S., Chi, Z. and Riezler, S. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 535–541.

Kay, M. 2000. Guides and oracles for linear-time parsing. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT)*, pp. 6–9.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Kromann, M. T. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 217–220. Växjö University Press.

Kübler, S., Hinrichs, E. W. and Maier, W. 2006. Is it really that difficult to parse German? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 111–119.

Kudo, T. and Matsumoto, Y. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL)*, pp. 63–69.

Lesmo, L., Lombardo, V. and Bosco, C. 2002. Treebank development: The TUT approach. In Sangal, R. and Bendre, S. M. (eds.) *Recent Advances in Natural Language Processing*, pp. 61–70. New Delhi: Vikas Publishing House.

Levy, R. and Manning, C. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 439–446.

Lin, D. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering* 4, 97–114.

Magerman, D. M. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 276–283.

Marinov, S. and Nivre, J. 2005. A data-driven parser for Bulgarian. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 89–100.

Maruyama, H. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Meeting of the Association for Computational Linguistics (ACL)*, pp. 31–38.

McDonald, R. and Pereira, F. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 81–88.

Mel'čuk, I. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Montemagni, S., Barsotti, F., Battista, M., Calzolari, N., Corazzari, O., Lenci, A., Zampolli, A., Fanciulli, F., Massetani, M., Raffaelli, R., Basili, R., Pazienza, M. T., Saracino, D., Zanzotto, F., Pianesi, F., Mana, N. and Delmonte, R. 2003. Building the Italian syntactic-semantic treebank. In Anne Abeillé (ed.) *Building and Using Syntactically Annotated Corpora*, pp. 189–210. Dordrecht: Kluwer.

Nilsson, J., Nivre, J. and Hall, J. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pp. 257–264.

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

Nivre, J. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pp. 50–57.

Nivre, J. 2006. *Inductive Dependency Parsing*. Springer.

Nivre, J. and Hall, J. 2005. MaltParser: A language-independent system for data-driven dependency parsing. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 137–148.

Nivre, J., Hall, J. and Nilsson, J. 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*, pp. 49–56.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G. and Marinov, S. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pp. 221–225.

Nivre, J. and Nilsson, J. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 99–106.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press

Nivre, J. and Scholz, M. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 64–70.

Oflazer, K., Say, B., Hakkani-Tür, D. Z. and Tür, G. 2003. Building a Turkish treebank. In Abeillé, A. (ed.) *Treebanks: Building and Using Parsed Corpora*, pp. 261–277. Dordrecht: Kluwer.

Ratnaparkhi, A. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–10.

Sagae, K. and Lavie, A. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pp. 125–132.

Simov, K., Popova, G. and Osenova, P. 2002. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In Wilson, A., Rayson, P. and McEnery, T. (eds), *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, pp. 135–142. Lincon-Europa.

Simmons, R. F. and Yu, Y.-H. 1992. The acquisition and use of context-dependent grammars for English. *Computational Linguistics* 18, 391–418.

Skut, W., Krenn, B., Brants, T. and Uszkoreit, H. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, Washington, D.C.

Tanev, H. and Mitkov, R. 2002. Shallow language processing architecture for Bulgarian. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING)*, pp. 995–1001.

Teleman, U. 1974. *Manual för grammatisk beskrivning av talad och skriven svenska*. Lund: Studentlitteratur.

Telljohann, H., Hinrichs, E. W., Kübler, S. and Zinsmeister, H. 2005. Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany.

Titov, I. and Henderson, J. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pp. 6–13.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. Springer.

Veenstra, J. and Daelemans, W. 2000. A memory-based alternative for connectionist shift-reduce parsing. Technical Report ILK-0012, University of Tilburg.

Voutilainen, A. 2001. Parsing Swedish. Extended Abstract for the 13th Nordic Conference of Computational Linguistics, Uppsala University, May, 20-22, 2001.

Van der Wouden, T., Hoekstra, H., Moortgat, M., Renmans, B. and Schuurman, I. 2002. Syntactic analysis in the spoken Dutch corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pp. 768–773.

Xue, N., Fei Xia, F.-D. and Palmer, M. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11(2), 207–238.

Yamada, H. and Matsumoto, Y. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 195–206.

Zelle, J. M. and Mooney, R. J. 1993. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference of the American Association for Artificial Intelligence (AAAI)*, pp. 817–899.

https://doi.org/10.1017/S1351324906004505 Published online by Cambridge University Press