CoNLL-X shared task on Multilingual Dependency Parsing

Sabine Buchholz
Speech Technology Group
Cambridge Research Lab
Toshiba Research Europe
Cambridge CB2 3NH, UK
sabine.buchholz@crl.toshiba.co.uk

Erwin Marsi
Communication & Cognition
Tilburg University
5000 LE Tilburg, The Netherlands
e.c.marsi@uvt.nl

Abstract

Each year the Conference on Computational Natural Language Learning (CoNLL)[1] features a shared task, in which participants train and test their systems on exactly the same data sets, in order to better compare systems. The tenth CoNLL (CoNLL-X) saw a shared task on Multilingual Dependency Parsing. In this paper, we describe how treebanks for 13 languages were converted into the same dependency format and how parsing performance was measured. We also give an overview of the parsing approaches that participants took and the results that they achieved. Finally, we try to draw general conclusions about multi-lingual parsing: What makes a particular language, treebank or annotation scheme easier or harder to parse and which phenomena are challenging for any dependency parser?

Acknowledgement

Many thanks to Amit Dubey and Yuval Krymolowski, the other two organizers of the shared task, for discussions, converting treebanks, writing software and helping with the papers.[2]

[1] see http://ilps.science.uva.nl/~erikt/signll/conll/

[2] Thanks also to Alexander Yeh for additional help with the paper reviews. His work was made possible by the MITRE Corporation's Sponsored Research Program.

1 Introduction

Previous CoNLL shared tasks focused on NP chunking (1999), general chunking (2000), clause identification (2001), named entity recognition (2002, 2003), and semantic role labeling (2004, 2005). This shared task on full (dependency) parsing is the logical next step. Parsing is an important preprocessing step for many NLP applications and therefore of considerable practical interest. It is a complex task and as it is not straightforwardly mappable to a "classical" segmentation, classification or sequence prediction problem, it also poses theoretical challenges to machine learning researchers.

During the last decade, much research has been done on data-driven parsing and performance has increased steadily. For training these parsers, syntactically annotated corpora (treebanks) of thousands to tens of thousands of sentences are necessary; so initially, research has focused on English. During the last few years, however, treebanks for other languages have become available and some parsers have been applied to several different languages. See Section 2 for a more detailed overview of related previous research.

So far, there has not been much comparison between different dependency parsers on exactly the same data sets (other than for English). One of the reasons is the lack of a de-facto standard for an evaluation metric (labeled or unlabeled, separate root accuracy?), for splitting the data into training and testing portions and, in the case of constituency treebanks converted to dependency format, for this conversion. Another reason are the various annotation

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 149–164, New York City, June 2006.

schemes and logical data formats used by different treebanks, which make it tedious to apply a parser to many treebanks. We hope that this shared task will improve the situation by introducing a uniform approach to dependency parsing. See Section 3 for the detailed task definition and Section 4 for information about the conversion of all 13 treebanks.

In this shared task, participants had two to three months[3] to implement a parsing system that could be trained for all these languages and four days to parse unseen test data for each. 19 participant groups submitted parsed test data. Of these, all but one parsed all 12 required languages and 13 also parsed the optional Bulgarian data. A wide variety of parsing approaches were used: some are extensions of previously published approaches, others are new. See Section 5 for an overview.

Systems were scored by computing the **labeled attachment score** (LAS), i.e. the percentage of "scoring" tokens for which the system had predicted the correct head and dependency label. Punctuation tokens were excluded from scoring. Results across languages and systems varied widely from 37.8% (worst score on Turkish) to 91.7% (best score on Japanese). See Section 6 for detailed results.

However, variations are consistent enough to allow us to draw some general conclusions. Section 7 discusses the implications of the results and analyzes the remaining problems. Finally, Section 8 describes possible directions for future research.

2 Previous research

Tesnière (1959) introduced the idea of a dependency tree (a "stemma" in his terminology), in which words stand in direct head-dependent relations, for representing the syntactic structure of a sentence. Hays (1964) and Gaifman (1965) studied the formal properties of **projective** dependency grammars, i.e. those where dependency links are not allowed to cross. Mel'čuk (1988) describes a multistratal dependency grammar, i.e. one that distinguishes between several types of dependency relations (morphological, syntactic and semantic). Other theories related to dependency grammar are word grammar (Hudson, 1984) and link grammar (Sleator and Temperley, 1993).

Some relatively recent rule-based full dependency parsers are Kurohashi and Nagao (1994) for Japanese, Oflazer (1999) for Turkish, Tapanainen and Järvinen (1997) for English and Elworthy (2000) for English and Japanese.

While phrase structure parsers are usually evaluated with the GEIG/PARSEVAL measures of precision and recall over constituents (Black et al., 1991), Lin (1995) and others have argued for an alternative, dependency-based evaluation. That approach is based on a conversion from constituent structure to dependency structure by recursively defining a head for each constituent.

The same idea was used by Magerman (1995), who developed the first "head table" for the Penn Treebank (Marcus et al., 1994), and Collins (1996), whose constituent parser is internally based on probabilities of bilexical dependencies, i.e. dependencies between two words. Collins (1997)'s parser and its reimplementation and extension by Bikel (2002) have by now been applied to a variety of languages: English (Collins, 1999), Czech (Collins et al., 1999), German (Dubey and Keller, 2003), Spanish (Cowan and Collins, 2005), French (Arun and Keller, 2005), Chinese (Bikel, 2002) and, according to Dan Bikel's web page, Arabic.

Eisner (1996) introduced a data-driven dependency parser and compared several probability models on (English) Penn Treebank data. Kudo and Matsumoto (2000) describe a dependency parser for Japanese and Yamada and Matsumoto (2003) an extension for English. Nivre's parser has been tested for Swedish (Nivre et al., 2004), English (Nivre and Scholz, 2004), Czech (Nivre and Nilsson, 2005), Bulgarian (Marinov and Nivre, 2005) and Chinese Cheng et al. (2005), while McDonald's parser has been applied to English (McDonald et al., 2005a), Czech (McDonald et al., 2005b) and, very recently, Danish (McDonald and Pereira, 2006).

3 Data format, task definition

The training data derived from the original treebanks (see Section 4) and given to the shared task participants was in a simple column-based format that is

[3] Some though had significantly less time: One participant registered as late as six days before the test data release (registration was a prerequisite to obtain most of the data sets) and still went on to submit parsed test data in time.

an extension of Joakim Nivre's Malt-TAB format[4] for the shared task and was chosen for its processing simplicity. All the sentences are in one text file and they are separated by a blank line after each sentence. A sentence consists of one or more tokens. Each token is represented on one line, consisting of 10 fields. Fields are separated from each other by a TAB.[5] The 10 fields are:

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

2) FORM: Word form or punctuation symbol. For the Arabic data only, FORM is a concatenation of the word in Arabic script and its transliteration in Latin script, separated by an underscore. This representation is meant to suit both those that do and those that do not read Arabic.

3) LEMMA: Lemma or stem (depending on the particular treebank) of word form, or an underscore if not available. Like for the FORM, the values for Arabic are concatenations of two scripts.

4) CPOSTAG: Coarse-grained part-of-speech tag, where the tagset depends on the treebank.

5) POSTAG: Fine-grained part-of-speech tag, where the tagset depends on the treebank. It is identical to the CPOSTAG value if no POSTAG is available from the original treebank.

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

7) HEAD: Head of the current token, which is either a value of ID, or zero ('0') if the token links to the virtual root node of the sentence. Note that depending on the original treebank annotation, there may be multiple tokens with a HEAD value of zero.

8) DEPREL: Dependency relation to the HEAD. The set of dependency relations depends on the particular treebank. The dependency relation of a token with HEAD=0 may be meaningful or simply 'ROOT' (also depending on the treebank).

9) PHEAD: Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all data sets), whereas the structure resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).

10) PDEPREL: Dependency relation to the PHEAD, or an underscore if not available.

As should be obvious from the description above, our format assumes that each token has exactly one head. Some dependency grammars, and also some treebanks, allow tokens to have more than one head, although often there is a distinction between primary and optional secondary relations, e.g. in the Danish Dependency Treebank (Kromann, 2003), the Dutch Alpino Treebank (van der Beek et al., 2002b) and the German TIGER treebank (Brants et al., 2002). For this shared task we decided to ignore any additional relations. However the data format could easily be extended with additional optional columns in the future. Cycles do not occur in the shared task data but are scored as normal if predicted by parsers. The character encoding of all data files is Unicode (specifically UTF-8), which is the only encoding to cover all languages and therefore ideally suited for multilingual parsing.

While the training data contained all 10 columns (although sometimes only with dummy values, i.e. underscores), the test data given to participants contained only the first 6. Participants' parsers then predicted the HEAD and DEPREL columns (any predicted PHEAD and PDEPREL columns were ignored). The predicted values were compared to the gold standard HEAD and DEPREL.[6] The official evaluation metric is the **labeled attachment score** (LAS), i.e. the percentage of "scoring" tokens for which the system has predicted the correct HEAD and DEPREL. The evaluation script defines a non-scoring token as a token where all characters of the FORM value have the Unicode category property "Punctuation".[7]

[4] http://w3.msi.vxu.se/ nivre/research/MaltXML.html

[5] Consequently, field values cannot contain TABs. In the shared task data, field values are also not supposed to contain any other whitespace (although unfortunately some spaces slipped through in the Spanish data).

[6] The official scoring script `eval.pl`, data sets for some languages and instructions on how to get the rest, the software used for the treebank conversions, much documentation, full results and other related information will be available from the permanent URL `http://depparse.uvt.nl` (also linked from the CoNLL web page).

[7] See `man perlunicode` for the technical details and the shared task website for our reasons for this decision. Note that an underscore and a percentage sign also have the Unicode "Punctuation" property.

We tried to take a test set that was representative of the genres in a treebank and did not cut through text samples. We also tried to document how we selected this set.[8] We aimed at having roughly the same size for the test sets of all languages: 5,000 scoring tokens. This is not an exact requirement as we do not want to cut sentences in half. The relatively small size of the test set means that even for the smallest treebanks the majority of tokens is available for training, and the equal size means that for the overall ranking of participants, we can simply compute the score on the concatenation of all test sets.

4 Treebanks and their conversion

In selecting the treebanks, practical considerations were the major factor. Treebanks had to be actually available, large enough, have a license that allowed free use for research or kind treebank providers who temporarily waived the fee for the shared task, and be suitable for conversion into the common format within the limited time. In addition, we aimed at a broad coverage of different language families.[9] As a general rule, we did not manually correct errors in treebanks if we discovered some during the conversion, see also Buchholz and Green (2006), although we did report them to the treebank providers and several got corrected by them.

4.1 Dependency treebanks

We used the following six dependency treebanks: **Czech**: Prague Dependency Treebank[10] (PDT) (Böhmová et al., 2003); **Arabic**: Prague Arabic Dependency Treebank[11] (PADT) (Hajič et al., 2004; Smrž et al., 2002); **Slovene**: Slovene Dependency Treebank[12] (SDT) (Džeroski et al., 2006); **Danish**: Danish Dependency Treebank[13] (Kromann, 2003); **Swedish**: Talbanken05[14] (Teleman, 1974; Einarsson, 1976; Nilsson et al., 2005); **Turkish**: Metu-Sabancı treebank[15] (Oflazer et al., 2003; Atalay et al., 2003).

The conversion of these treebanks was the easiest task as the linguistic representation was already what we needed, so the information only had to be converted from SGML or XML to the shared task format. Also, the relevant information had to be distributed appropriately over the CPOSTAG, POSTAG and FEATS columns.

For the Swedish data, no predefined distinction into coarse and fine-grained PoS was available, so the two columns contain identical values in our format. For the Czech data, we sampled both our training and test data from the official "training" partition because only that one contains gold standard PoS tags, which is also what is used in most other data sets. The Czech DEPREL values include the suffixes to mark coordination, apposition and parenthesis, while these have been ignored during the conversion of the much smaller Slovene data. For the Arabic data, sentences with missing annotation were filtered out during the conversion.

The Turkish treebank posed a special problem because it analyzes each word as a sequence of one or more inflectional groups (IGs). Each IG consists of either a stem or a derivational suffix plus all the inflectional suffixes belonging to that stem/derivational suffix. The head of a whole word is not just another word but a specific IG of another word.[16] One can easily map this representation to one in which the head of a word is a word but that

[8] See the shared task website for a more detailed discussion.

[9] That was also the reason why we decided not to include a fifth Germanic language (English) although the freely available SUSANNE treebank (Sampson, 1995) or possibly the Penn Treebank would have qualified otherwise.

[10] Many thanks to Jan Hajič for granting the temporary license for CoNLL-X and talking to LDC about it, to Christopher Cieri for arranging distribution through LDC and to Tony Castelletto for handling the distribution.

[11] Many thanks to Yuval Krymolowski for converting the treebank, Otakar Smrž for valuable help during the conversion and thanks again to Jan Hajič, Christopher Cieri and Tony Castelletto.

[12] Many thanks to the SDT people for granting the special license for CoNLL-X and to Tomaž Erjavec for converting the treebank for us.

[13] Many thanks to Matthias Trautner Kromann and assistants for creating the DDT and releasing it under the GNU General Public License and to Joakim Nivre, Johan Hall and Jens Nilsson for the conversion of DDT to Malt-XML.

[14] Many thanks to Jens Nilsson, Johan Hall and Joakim Nivre for the conversion of the original Talbanken to Talbanken05 and for making it freely available for research purposes and to Joakim Nivre again for prompt and proper respons to all our questions.

[15] Many thanks to Bilge Say and Kemal Oflazer for granting the license for CoNLL-X and answering questions and to Gülşen Eryiğit for making many corrections to the treebank and discussing some aspects of the conversion.

[16] This is a bit like saying that in "the usefulness of X for Y", "for Y" links to "use-" and not to "usefulness". Only that in Turkish, "use", "full" and "ness" each could have their own inflectional suffixes attached to them.

mapping would lose information and it is not clear whether the result is linguistically meaningful, practically useful, or even easier to parse because in the original representation, each IG has its own PoS and morphological features, so it is not clear how that information should be represented if all IGs of a word are conflated. We therefore chose to represent each IG as a separate token in our format. To make the result a connected dependency structure, we defined the HEAD of each non-word-final IG to be the following IG and the DEPREL to be "DERIV". We assigned the stem of the word to the first IG's LEMMA column, with all non-first IGs having LEMMA '_', and the actual word form to the last IG, with all non-last IGs having FORM '_'. As already mentioned in Section 3, the underscore has the punctuation character property, therefore non-last IGs (whose HEAD and DEPREL were introduced by us) are not scoring tokens. We also attached or reattached punctuation (see the README available at the shared task website for details.)

4.2 Phrase structure with functions for all constituents

We used the following five treebanks of this type: **German**: TIGER treebank[17] (Brants et al., 2002); **Japanese**: Japanese Verbmobil treebank[18] (Kawata and Bartels, 2000); **Portuguese**: The Bosque part of the Floresta sintá(c)tica[19] (Afonso et al., 2002); **Dutch**: Alpino treebank[20] (van der Beek et al., 2002b; van der Beek et al., 2002a); **Chinese**: Sinica treebank[21] (Chen et al., 2003).

Their conversion to dependency format required the definition of a head table. Fortunately, in contrast to the Penn Treebank for which the head table is based on POS[22] we could use the grammatical functions annotated in these treebanks. Therefore, head rules are often of the form: the head child of a VP/clause is the child with the HD/predicator/hd/Head function. The DEPREL value for a token is the function of the biggest constituent of which this token is the lexical head. If the constituent comprising the complete sentence did not have a function, we gave its lexical head token the DEPREL "ROOT".

For the Chinese treebank, most functions are not grammatical functions (such as "subject", "object") but semantic roles (such as "agent", "theme"). For the Portuguese treebank, the conversion was complicated by the fact that a detailed specification existed which tokens should be the head of which other tokens, e.g. the finite verb must be the head of the subject and the complementzier but the main verb must be the head of the complements and adjuncts.[23] Given that the Floresta sintá(c)tica does not use traditional VP constituents but rather verbal chunks (consisting mainly of verbs), a simple Magerman-Collins-style head table was not sufficient to derive the required dependency structure. Instead we used a head table that defined several types of heads (syntactic, semantic) and a link table that specified what linked to which type of head.[24]

Another problem existed with the Dutch treebank. Its original PoS tag set is very coarse and the PoS and the word stem information is not very reliable.[25] We therefore decided to retag the treebank automatically using the Memory-Based Tagger (MBT) (Daelemans et al., 1996) which uses a very fine-grained tag set. However, this created a problem with multiwords. MBT does not have the concept of multiwords and therefore tags all of their

[17] Many thanks to the TIGER team for allowing us to use the treebank for the shared task and to Amit Dubey for converting the treebank.

[18] Many thanks to Yasuhiro Kawata, Julia Bartels and colleagues from Tübingen University for the construction of the original Verbmobil treebank for Japanese and to Sandra Kübler for providing the data and granting the special license for CoNLL-X.

[19] Many thanks to Diana Santos, Eckhard Bick and other Floresta sint(c)tica project members for creating the treebank and making it publicly available, for answering many questions about the treebank (Diana and Eckhard), for correcting problems and making new releases (Diana), and for sharing scripts and explaining the head rules implemented in them (Eckhard). Thanks also to Jason Baldridge for useful discussions and to Ben Wing for independently reporting problems which Diana then fixed.

[20] Many thanks to Gertjan van Noord and the other people at the University of Groningen for creating the Alpino Treebank and releasing it for free, to Gertjan van Noord for answering all our questions and for providing extra test material and to Antal van den Bosch for help with the memory-based tagger.

[21] Many thanks to Academia Sinica for granting the temporary license for CoNLL-X, to Keh-Jiann Chen for answering our questions and to Amit Dubey for converting the treebank.

[22] containing rules such as: the head child of a VP is the leftmost "to", or else the leftmost past tense verb, or else etc.

[23] Eckhard Bick, p.c.

[24] See the conversion script `bosque2MALT.py` and the README file at the shared task website for details.

[25] http://www.let.rug.nl/vannoord/trees/Papers/diffs.pdf

components individually. As Alpino does not provide an internal structure for multiwords, we had to treat multiwords as one token. However, we then lack a proper PoS for the multiword. After much discussion, we decided to assign each multiword the CPOSTAG "MWU" (multiword unit) and a POSTAG which is the concatenation of the PoS of all the components as predicted by MBT (separated by an underscore). Likewise, the FEATS are a concatenation of the morphological features of all components. This approach resulted in many different POSTAG values for the training set and even in unseen values in the test set. It remains to be tested whether our approach resulted in data sets better suited for parsing than the original.

4.3 Phrase structure with some functions

We used two treebanks of this type: **Spanish**: Cast3LB[26] (Civit Torruella and Martí Antonín, 2002; Navarro et al., 2003; Civit et al., 2003); **Bulgarian**: BulTreeBank[27] (Simov et al., 2002; Simov and Osenova, 2003; Simov et al., 2004; Osenova and Simov, 2004; Simov et al., 2005).

Converting a phrase structure treebank with only a few functions to a dependency format usually requires linguistic competence in the treebank's language in order to create the head table and missing function labels. We are grateful to Chanev et al. (2006) for converting the BulTreeBank to the shared task format and to Montserrat Civit for providing us with a head table and a function mapping for Cast3LB.[28]

4.4 Data set characteristics

Table 1 shows details of all data sets. Following Nivre and Nilsson (2005) we use the following definition: "an arc (i, j) is projective iff all nodes occurring between i and j are dominated by i (where dominates is the transitive closure of the arc relation)".[29]

5 Approaches

Table 2 tries to give an overview of the wide variety of parsing approaches used by participants. We refer to the individual papers for details. There are several dimensions along which to classify approaches.

5.1 Top-down, bottom-up

Phrase structure parsers are often classified in terms of the parsing order: top-down, bottom-up or various combinations. For dependency parsing, there seem to be two different interpretations of the term "bottom-up". Nivre and Scholz (2004) uses this term with reference to Yamada and Matsumoto (2003), whose parser has to find all children of a token before it can attach that token to its head. We will refer to this as "bottom-up-trees". Another use of "bottom-up" is due to Eisner (1996), who introduced the notion of a "span". A span consists of a potential dependency arc r between two tokens i and j and all those dependency arcs that would be spanned by r, i.e. all arcs between tokens k and l with $i \leq k, l \leq j$. Parsing in this order means that the parser has to find all children and siblings on one side of a token before it can attach that token to a head on the same side. This approach assumes projective dependency structures. Eisner called this approach simply "bottom-up", while Nivre, whose parser implicitly also follows this order, called it "top-down/bottom-up" to distinguish it from the pure "bottom-up(-trees)" order of Yamada and Matsumoto (2003). To avoid confusion, we will refer to this order as "bottom-up-spans".

5.2 Unlabeled parsing versus labeling

Given that the parser needs to predict the HEAD as well as the DEPREL value, different approaches are possible: predict the (probabilities of the) HEADs of all tokens first, or predict the (probabilities of the) DEPRELs of all tokens first, or predict the HEAD and DEPREL of one token before predicting these values for the next token. Within the first approach, each dependency can be labeled independently (Corston-Oliver and Aue, 2006) or a

[26] Many thanks to Montserrat Civit and Toni Martí for allowing us to use Cast3LB for CoNLL-X and to Amit Dubey for converting the treebank.

[27] Many thanks to Kiril Simov and Petya Osenova for allowing us to use the BulTreeBank for CoNLL-X.

[28] Although unfortunately, due to a bug, the function list was not used and the Spanish data in the shared task ended up with many DEPREL values being simply '_'. By the time we discovered this, the test data release date was very close and we decided not to release new bug-fixed training material that late.

[29] Thanks to Joakim Nivre for explaining this.

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Bu
lang. fam.	Sem.	Sin.	Sla.	Ger.	Ger.	Ger.	Jap.	Rom.	Sla.	Rom.	Ger.	Ura.	Sla.
genres	1: ne	6	3	8+	5+	1: ne	1: di	1: ne	1: no	9	4+	8	12
annotation	d	c+f	d	d	dc+f	dc+f	c+f	dc+f	d	c(+f)	dc+f/d	d	c+t
training data													
tokens (k)	54	337	1249	94	195	700	151	207	29	89	191	58	190
%non-scor.	8.8	[a]0.8	14.9	13.9	11.3	11.5	11.6	14.2	17.3	12.6	11.0	[b]33.1	14.4
units (k)	1.5	57.0	72.7	5.2	13.3	39.2	17.0	9.1	1.5	3.3	11.0	5.0	12.8
tokens/unit	[c]37.2	[d]5.9	17.2	18.2	14.6	17.8	[e]8.9	22.8	18.7	27.0	17.3	11.5	14.8
LEMMA	[f](+)	–	+	–	+	–	–	+	+	+	–	+	–
CPOSTAGs	14	13+9	12	10	13	[g]52	20	15	11	15	37	14	11
POSTAGs	19	[h]294+9	63	24	[i]302	52	77	21	28	38	37	30	53
FEATS	19	–	61	47	81	–	4	146	51	33	–	82	50
DEPRELs	27	82	78	52	26	46	7	55	25	21	56	25	18
D.s H.=0	15	1	14	1	1	1	1	6	6	1	1	1	1
%HEAD=0	5.5	16.9	6.7	6.4	8.9	6.3	18.6	5.1	5.9	4.2	6.5	13.4	7.9
%H. preced.	82.9	24.8	50.9	75.0	46.5	50.9	8.9	60.3	47.2	60.8	52.8	6.2	62.9
%H. follow.	11.6	58.2	42.4	18.6	44.6	42.7	72.5	34.6	46.9	35.1	40.7	80.4	29.2
H.=0/unit	1.9	1.0	1.0	1.0	1.2	1.0	1.5	1.0	[j]0.9	1.0	1.0	1.0	1.0
%n.p. arcs	0.4	0.0	1.9	1.0	5.4	2.3	[k]1.1	1.3	1.9	[l]0.1	1.0	1.5	0.4
%n.p. units	11.2	0.0	23.2	15.6	36.4	27.8	5.3	18.9	22.2	1.7	9.8	11.6	5.4
test data													
scor. tokens	4990	4970	5000	5010	4998	5008	5003	5009	5004	4991	5021	5021	5013
%new form	17.3	9.3	5.2	18.1	20.7	6.5	0.96	11.6	22.0	14.7	18.0	41.4	14.5
%new lem.	4.3	n/a	1.8	n/a	15.9	n/a	n/a	7.8	9.9	9.7	n/a	13.2	n/a

Table 1: Characteristics of the data sets for the 13 languages (abbreviated by their first two letters): language family (Semitic, Sino-Tibetan, Slavic, Germanic, Japonic (or language isolate), Romance, Ural-Altaic); number of genres, and genre if only one (news, dialogue, novel); type of annotation (d=dependency, c=constituents, dc=discontinuous constituents, +f=with functions, +t=with types). For the training data: number of tokens (times 1000); percentage of non-scoring tokens; number of parse tree units (usually sentences, times 1000); average number of (scoring and non-scoring) tokens per parse tree unit; whether a lemma or stem is available; how many different CPOSTAG values, POSTAG values, FEATS components and DEPREL values occur for scoring tokens; how many different values for DEPREL scoring tokens with HEAD=0 can have (if that number is 1, there is one designated label (e.g. "ROOT") for tokens with HEAD=0); percentage of scoring tokens with HEAD=0, a head that precedes or a head that follows the token (this nicely shows which languages are predominantly head-initial or head-final); the average number of scoring tokens with HEAD=0 per parse tree unit; the percentage of (scoring and non-scoring) non-projective relations and of parse tree units with at least one non-projective relation. For the test data: number of scoring tokens; percentage of scoring tokens with a FORM or a LEMMA that does not occur in the training data.

[a] final punctuation was deliberately left out during the conversion (as it is explicitly excluded from the tree structure)

[b] the non-last IGs of a word are non-scoring, see Section 4.1

[c] in many cases the parse tree unit in PADT is not a sentence but a paragraph

[d] in many cases the unit in Sinica is not a sentence but a comma-separated clause or phrase

[e] the treebank consists of transcribed dialogues, in which some sentences are very short, e.g. just "Hai." ("Yes.")

[f] only part of the Arabic data has non-underscore values for the LEMMA column

[g] no mapping from fine-grained to coarse-grained tags was available; same for Swedish

[h] 9 values are typos; POSTAGs also encode subcategorization information for verbs and some semantic information for conjunctions and nouns; some values also include parts in square brackets which in hindsight should maybe have gone to FEATS

[i] due to treatment of multiwords

[j] probably due to some sentences consisting only of non-scoring tokens, i.e. punctuation

[k] these are all disfluencies, which are attached to the virtual root node

[l] from co-indexed items in the original treebank; same for Bulgarian

	algorithm	ver.	hor.	search	lab.	non-proj	learner	pre	post	opt
all pairs										
McD	MST/Eisner	b-s	irr.	opt/approx.	2nd	+ [a]	MIRA	–	–	–
Cor	MST/Eisner	b-s	irr.	optimal	2nd	–	BPM[b]+ME [SVM]	+ [c]	–	–
Shi	MST/CLE	irr.	irr.	optimal	1st	+, CLE	MIRA	–	–	–
Can	own algorithm	irr.	irr.	approx.(?)	int.	+ [d]	TiMBL	–	–	+
Rie	ILP	irr.	irr.	increment.	int.	+ [e]	MIRA	–	–	+
Bic	CG-inspired	mpf	mpf	backtrack(?)	int.	+ [f]	MLE(?)	+ [g]	+ [h]	–
stepwise										
Dre	hag[i]/Eisner/rerank	b-s	irr.	best 1st exh	2nd	–	MLE	–	–	+ [j]
Liu	own algorithm	b-t	mpf	det./local	int.	–	MLE	–	–	–
Car	Eisner	b-s	irr.	approx.	int.	–	perceptron	–	–	–
stepwise: classifier-based										
Att	Y&M	b-t	for.	determin.	int.	+ [k]	ME [MBL,SVM,...]	stem	–	–
Cha	Y&M	b-t	for.	local	2nd	– [l]	perceptron (SNoW)	proj	–	–
Yur	own algorithm	b-s	irr.	determin.	int.	–	decision list (GPA)[m]	–	–	–
Che	chunker+Nivre	b-s	for.	determin.	int.[n]	–	SVM + ME [CRF]	–	–	–
Niv	Nivre	b-s	for.	determin.	int.	+, ps-pr	SVM	proj	deproj	+
Joh	Nivre+MST/CLE	b-s	f+b[o]	N-best	int.[p]	+, CLE	SVM (LIBSVM)	–	–	
Wu	Nivre+root parser	b-s	f/b[q]	det.[+exh.]	int.	– [+]	SVM (SVMLight)	–	[+] [r]	–
other										
Sch	PCFG/CKY	b-t	irr.	opt.	int.	+, traces	MLE [ME]	d2c	c2d	–

Table 2: Overview of parsing approaches taken by participating groups (identified by the first three letters of the first author): algorithm (Y&M: Yamada and Matsumoto (2003), ILP: Integer Linear Programming), vertical direction (irrelevant, mpf: most probable first, bottom-up-spans, bottom-up-trees), horizontal direction (irrelevant, mpf: most probable first, forward, backward), search (optimal, approximate, incremental, best-first exhaustive, deterministic), labeling (interleaved, separate and 1st step, separate and 2nd step), non-projective (ps-pr: through pseudo-projective approach), learner (ME: Maximum Entropy; learners in brackets were explored but not used in the official submission), preprocessing (projectivize, d2c: dependencies to constituents), postprocessing (deprojectivize, c2d: constituents to dependencies), learner parameter optimization per language

[a] non-projectivity through approximate search, used for some languages
[b] 20 averaged perceptrons combined into a Bayes Point Machine
[c] introduced a single POS tag "aux" for all Swedish auxiliary and model verbs
[d] by having no projectivity constraint
[e] selective projectivity constraint for Japanese
[f] several approaches to non-projectivity
[g] using some FEATS components to create some finer-grained POSTAG values
[h] reattachment rules for some types of non-projectivity
[i] head automaton grammar
[j] determined the maximally allowed distance for relations
[k] through special parser actions
[l] pseudo-projectivizing training data only
[m] Greedy Prepend Algorithm
[n] but two separate learners used for unlabeled parsing versus labeling
[o] both foward and backward, then combined into a single tree with CLE
[p] but two separate SVMs used for unlabeled parsing versus labeling
[q] forward parsing for Japanese and Turkish, backward for the rest
[r] attaching remaining unattached tokens through exhaustive search (not for submitted runs)

sequence classifier can label all children of a token together (McDonald et al., 2006). Within the third approach, HEAD and DEPREL can be predicted simultaneously, or in two separate steps (potentially using two different learners).

5.3 All pairs

At the highest level of abstraction, there are two fundamental approaches, which we will call "all pairs" and "stepwise". In an "all pairs" approach, every possible pair of two tokens in a sentence is considered and some score is assigned to the possibility of this pair having a (directed) dependency relation. Using that information as building blocks, the parser then searches for the best parse for the sentence. This approach is one of those described in Eisner (1996). The definition of "best" parse depends on the precise model used. That model can be one that defines the score of a complete dependency tree as the sum of the scores of all dependency arcs in it. The search for the best parse can then be formalized as the search for the maximum spanning tree (MST) (McDonald et al., 2005b). If the parse has to be projective, Eisner's bottom-up-span algorithm (Eisner, 1996) can be used for the search. For non-projective parses, McDonald et al. (2005b) propose using the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967) and McDonald and Pereira (2006) describe an approximate extension of Eisner's algorithm. There are also alternatives to MST which allow imposing additional constraints on the dependency structure, e.g. that at most one dependent of a token can have a certain label, such as "subject", see Riedel et al. (2006) and Bick (2006). By contrast, Canisius et al. (2006) do not even enforce the tree constraint, i.e. they allow cycles. In a variant of the "all pairs" approach, only those pairs of tokens are considered that are not too distant (Canisius et al., 2006).

5.4 Stepwise

In a stepwise approach, not all pairs are considered. Instead, the dependency tree is built stepwise and the decision about what step to take next (e.g. which dependency to insert) can be based on information about, in theory all, previous steps and their results (in the context of generative probabilistic parsing, Black et al. (1993) call this the history). Stepwise approaches can use an explicit probability model over next steps, e.g. a generative one (Eisner, 1996; Dreyer et al., 2006), or train a machine learner to predict those. The approach can be deterministic (at each point, one step is chosen) or employ various types of search. In addition, parsing can be done in a bottom-up-constituent or a bottom-up-spans fashion (or in another way, although this was not done in this shared task). Finally, parsing can start at the first or the last token of a sentence. When talking about languages that are written from left to right, this distinction is normally referred to as left-to-right versus right-to-left. However, for multilingual parsing which includes languages that are written from right to left (Arabic) or sometimes top to bottom (Chinese, Japanese) this terminology is confusing because it is not always clear whether a left-to-right parser for Arabic would really start with the leftmost (i.e. last) token of a sentence or, like for other languages, with the first (i.e. rightmost). In general, starting with the first token ("forward") makes more sense from a psycholinguistic point of view but starting with the last ("backward") might be beneficial for some languages (possibly related to them being head-initial versus head-final languages). The parsing order directly determines what information will be available from the history when the next decision needs to be made. Stepwise parsers tend to interleave the prediction of HEAD and DEPREL.

5.5 Non-projectivity

All data sets except the Chinese one contain some non-projective dependency arcs, although their proportion varies from 0.1% to 5.4%. Participants took the following approaches to non-projectivity:

- Ignore, i.e. predict only projective parses. Depending on the way the parser is trained, it might be necessary to at least projectivize the training data (Chang et al., 2006).
- Always allow non-projective arcs, by not imposing any projectivity constraint (Shimizu, 2006; Canisius et al., 2006).
- Allow during parsing under certain conditions, e.g. for tokens with certain properties (Riedel et al., 2006; Bick, 2006) or if no alternative projective arc has a score above the threshold

(Bick, 2006) or if the classifier chooses a special action (Attardi, 2006) or the parser predicts a trace (Schiehlen and Spranger, 2006).

- Introduce through post-processing, e.g. through reattachment rules (Bick, 2006) or if the change increases overall parse tree probability (McDonald et al., 2006).

- The pseudo-projective approach (Nivre and Nilsson, 2005): Transform non-projective training trees to projective ones but encode the information necessary to make the inverse transformation in the DEPREL, so that this inverse transformation can also be carried out on the test trees (Nivre et al., 2006).

5.6 Data columns used

Table 3 shows which column values have been used by participants. Nobody used the PHEAD/PDEPREL column in any way. It is likely that those who did not use any of the other columns did so mainly for practical reasons, such as the limited time and/or the difficulty to integrate it into an existing parser.

5.6.1 FORM versus LEMMA

Lemma or stem information has often been ignored in previous dependency parsers. In the shared task data, it was available in just over half the data sets. Both LEMMA and FORM encode lexical information. There is therefore a certain redundancy. Participants have used these two columns in different ways:

- Use only one (see Table 3).

- Use both, in different features. Typically, a feature selection routine and/or the learner itself (through weights) will decide about the importance of the resulting features.

- Use a variant of the FORM as a substitute for a missing LEMMA. Bick (2006) used the lowercased FORM if the LEMMA is not available, Corston-Oliver and Aue (2006) a prefix and Attardi (2006) a stem derived by a rule-based system for Danish, German and Swedish.

	form	lem.	cpos	pos	feats
McD	++ [a]	+ [b]	–?	+	+, co+cr.pr.
Cor	+	+	+ [c]	++	+, co+cr.pr.[d]
Shi	+	–	+	–	–
Can	+	–	–	+	–
Rie	+ [e]	+	+	+ [f]	+ cr.pr.
Bic	(+)	+	+ [g]	+	(+)
Dre	++ [h]	+	rer.	rer.	–
Liu	(+)	+	++	+	–
Car	++	+	++	+	+ comp.
Att	(+)	+	+	–	(+)
Cha	–	+	–	+	+ atomic
Yur	+	+	+	+	+ comp.
Che	+	+	+	+	+ atomic?
Niv	+	+	+	+	+ comp.
Joh	+	–	+	+	+ comp.
Wu	+	–	+	+	–
Sch	?	(+)[i]	?	(+)	(+)

Table 3: Overview of data columns used by participating groups. '–': a column value was not used at all. '+': used in at least some features. '(+)': Variant of FORM used only if LEMMA is missing, or only parts of FEATS used. '++': used more extensively than another column containing related information (where FORM and LEMMA are related, as are CPOSTAG and POSTAG), e.g. also in combination features or features for context tokens in addition to features for the focus token(s). "rer.": used in the reranker only. For the last column: atomic, comp. = components, cr.pr. = cross-product.

[a] also prefix and suffix for labeler
[b] instead of form for Arabic and Spanish
[c] instead of POSTAG for Dutch and Turkish
[d] for labeler; unlab. parsing: only some for global features
[e] also prefix
[f] also 1st character of POSTAG
[g] only as backoff
[h] reranker: also suffix; if no lemma, use prefix of FORM
[i] LEMMA, POSTAG, FEATS only for back-off smoothing

5.6.2 CPOSTAG versus POSTAG

All data sets except German and Swedish had different values for CPOSTAG and POSTAG, although the granularity varied widely. Again, there are different approaches to dealing with the redundancy:

- Use only one for all languages.

- Use both, in different features. Typically, a feature selection routine and/or the learner itself (through weights) will decide about the importance of the resulting features.
- Use one or the other for each language.

5.6.3 Using FEATS

By design, a FEATS column value has internal structure. Splitting it at the '|'[30] results in a set of components. The following approaches have been used:

- Ignore the FEATS.
- Treat the complete FEATS value as atomic, i.e. do not split it into components.
- Use only some components, e.g. Bick (2006) uses only case, mood and pronoun subclass and Attardi (2006) uses only gender, number, person and case.
- Use one binary feature for each component. This is likely to be useful if grammatical function is indicated by case.
- Use one binary feature for each cross-product of the FEATS components of i and the FEATS components of j. This is likely to be useful for agreement phenomena.
- Use one binary feature for each FEATS component of i that also exists for j. This is a more explicit way to model agreement.

5.7 Types of features

When deciding whether there should be a dependency relation between tokens i and j, all parsers use at least information about these two tokens. In addition, the following sources of information can be used (see Table 4): token context (**tc**): a limited number (determined by the window size) of tokens directly preceding or following i or j; **ch**ildren: information about the already found children of i and j; **si**blings: in a set-up where the decision is not "is there a relation between i and j" but "is i the head of j" or in a separate labeling step, the siblings of i are the already found children of j; structural context (**sc**) other than children/siblings: neighboring subtrees/spans, or ancestors of i and j; **di**stance from i to j; information derived from all the tokens **in** between i and j (e.g. whether there is an intervening verb or how many intervening commas there are); **gl**obal features (e.g. does the sentence contain a finite verb); explicit feature **co**mbinations (depending on the learner, these might not be necessary, e.g. a polynomial kernel routinely combines features); for classifier-based parsers: the previous **ac**tions, i.e. classifications; whether information about **la**bels is used as input for other decisions. Finally, the precise set of features can be **op**timized per language.

[30] or for Dutch, also at the '_'

	tc	ch	si	sc	di	in	gl	co	ac	la	op
McD	+	l	+	l	?	l	l	+	–	l	(+)[a]
Cor	+	l[b]	l	+	p	–	+	+	–	–	(+)[c]
Shi	+	–	–	–	+	–	–	+	–	+	–
Can	+	–	–	–	+	–	–	–	–	–	–
Rie	+	–	+[d]	–	?	?	–	+	–	+[e]	+
Bic	+	+[f]	+[g]	–	+	+[h]	–	+	–	++	(+)[i]
Dre	r	r	+	r	+	r	–	+	–	r	r
Liu	–	+	–	+	+	–	–	+	–	–	–
Car	+	–	+	–	+	+	–	+	–	+	–
Att	–	+	+	+	–	–	–	–	+	+	(+)[j]
Cha	+	+	–	l	–	–	–	+	+	–	–
Yur	+	+	–	?	–	–	–	–	–	–	+
Che	–	+	+	+	+	–	–	–	–	–	–
Niv	+	+	–	+	–	–	–	–	–	+	+
Joh	+	+	–	+	–	–	–	–	–	+	–
Wu	+	+	–	+	–	–	–	+	–	+	–
Sch	–	+	–	–	–	–	–	–	–	+	–

Table 4: Overview of features used by participating groups. See the text for the meaning of the column abbreviations. For separate HEAD and DEPREL assignment: p: only for unlabeled parsing, l: only for labeling, r: only for reranking.

[a] FORM versus LEMMA
[b] number of tokens governed by child
[c] POSTAG versus CPOSTAG
[d] for arity constraint
[e] for arity constraint
[f] for "full" head constraint
[g] for uniqueness constraint
[h] for barrier constraint
[i] of constraints
[j] POS window size

6 Results

Table 5 shows the official results for submitted parser outputs.[31] The two participant groups with the highest total score are McDonald et al. (2006) and Nivre et al. (2006). As both groups had much prior experience in multilingual dependency parsing (see Section 2), it is not too surprising that they both achieved good results. It is surprising, however, how similar their total scores are, given that their approaches are quite different (see Table 2). The results show that experiments on just one or two languages certainly give an indication of the usefulness of a parsing approach but should not be taken as proof that one algorithm is better for "parsing" (in general) than another that performs slightly worse. The Bulgarian scores suggest that rankings would not have been very different had it been the 13th obligatory languages.

Table 6 shows that the same holds had we used another evaluation metric. Note that a negative number in both the third and fifth column indicates that errors on HEAD and DEPREL occur together on the same token more often than for other parsers. Finally, we checked that, had we also scored on punctuation tokens, total scores as well as rankings would only have shown very minor differences.

7 Result analysis

7.1 Across data sets

The average LAS over all data sets varies between 56.0 for Turkish and 85.9 for Japanese. Top scores vary between 65.7 for Turkish and 91.7 for Japanese. In general, there is a high correlation between the best scores and the average scores. This means that data sets are inherently easy or difficult, no matter what the parsing approach. The "easiest" one is clearly the Japanese data set. However, it would be wrong to conclude from this that Japanese in general is easy to parse. It is more likely that the effect stems from the characteristics of the data. The Japanese Verbmobil treebank contains dialogue within a restricted domain (making business appointments). As can be seen in Table 1, there are very few new FORM values in the test data, which is an indication of many dialogues in the treebank being similar. In addition, parsing units are short on average. Finally, the set of DEPREL values is very small and consequently the ratio between (C)POSTAG and DEPREL values is extremely favorable. It would be interesting to apply the shared task parsers to the Kyoto University Corpus (Kurohashi and Nagao, 1997), which is the standard treebank for Japanese and has also been used by Kudo and Matsumoto

	LAS	unlabeled		label acc.	
McD	80.3	=	86.6	−1	86.7
Niv	80.2	=	85.5	+1	86.8
O'N	78.4	=	85.3	−1	85.0
Rie	77.9	=	85.0	−1	84.9
Sag	77.8	−2	83.7	+2	85.6
Che	77.7	+1	84.6	=	84.2
Cor	76.9	+1	84.4	−1	84.0
Cha	76.8	=	83.5	+1	84.1
Joh	74.9	−1	80.4	=	83.7
Car	74.7	+1	81.2	=	83.5
Wu	71.7	−1	78.4	−1	79.1
Can	70.8	+1	78.4	−1	78.6
Bic	70.0	=	77.5	[a]+2	80.3
Dre	65.2	−1	74.5	−1	75.2
Yur	65.0	−1	73.5	−2	70.9
Liu	63.3	−2	70.7	=	73.6
Sch	62.8	=	72.1	[b]+3	75.7
Att	61.2	[c]+4	76.2	=	70.7
Shi	34.2	=	38.7	=	39.7

Table 6: Differences in ranking depending on the evaluation metric. The second column repeats the official metric (LAS). The third column shows how the ranking for each participant changes (or not: '=') if the unlabeled attachment scores, as shown in the fourth column, are used. The fifth column shows how the ranking changes (in comparison to LAS) if the label accuracies, as shown in the sixth column, are used.

[a] In Bick's method, preference is given to the assignment of dependency labels.

[b] Schiehlen derived the constituent labels for his PCFG approach from the DEPREL values.

[c] Due to the bug (see footnote with Table 5).

[31] Unfortunately, urgent other obligations prevented two participants (John O'Neil and Kenji Sagae) from submitting a paper about their shared task work. Their results are indicated by a smaller font. Sagae used a best-first probabilistic version of Y&M (p.c.).

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Tot	SD	Bu
McD	**66.9**	85.9	**80.2**	**84.8**	**79.2**	**87.3**	90.7	**86.8**	**73.4**	**82.3**	82.6	63.2	**80.3**	8.4	**87.6**
Niv	**66.7**	86.9	78.4	**84.8**	**78.6**	85.8	**91.7**	**87.6**	70.3	**81.3**	**84.6**	**65.7**	**80.2**	8.5	**87.4**
O'N	**66.7**	86.7	76.6	82.8	77.5	85.4	90.6	84.7	71.1	79.8	81.8	57.5	78.4	9.4	85.2
Rie	**66.7**	**90.0**	67.4	83.6	**78.6**	86.2	90.5	84.4	71.2	77.4	80.7	58.6	77.9	10.1	0.0
Sag	62.7	84.7	75.2	81.6	76.6	84.9	90.4	**86.0**	69.1	77.7	82.0	63.2	77.8	9.0	0.0
Che	65.2	84.3	76.2	81.7	71.8	84.1	89.9	85.1	71.4	80.5	81.1	61.2	77.7	8.7	86.3
Cor	63.5	79.9	74.5	81.7	71.4	83.5	90.0	84.6	**72.4**	80.4	79.7	61.7	76.9	8.5	83.4
Cha	60.9	85.1	72.9	80.6	72.9	84.2	89.1	84.0	69.5	79.7	82.3	60.5	76.8	9.4	0.0
Joh	64.3	72.5	71.5	81.5	72.7	80.4	85.6	84.6	66.4	78.2	78.1	63.4	74.9	7.7	0.0
Car	60.9	83.7	68.8	79.7	67.3	82.4	88.1	83.4	68.4	77.2	78.7	58.1	74.7	9.7	83.3
Wu	63.8	74.8	59.4	78.4	68.5	76.5	90.1	81.5	67.8	73.0	71.7	55.1	71.7	9.7	79.7
Can	57.6	78.4	60.9	77.9	74.6	77.6	87.4	77.4	59.2	68.3	79.2	51.1	70.8	11.1	78.7
Bic	55.4	76.2	63.0	74.6	69.5	74.7	84.8	78.2	64.3	71.4	74.1	53.9	70.0	9.3	79.2
Dre	53.4	71.6	60.5	66.6	61.6	71.0	82.9	75.3	58.7	67.6	67.6	46.1	65.2	9.9	74.8
Yur	52.4	72.7	51.9	71.6	62.8	63.8	84.4	70.4	55.1	69.6	65.2	60.3	65.0	9.5	73.5
Liu	50.7	75.3	58.5	77.7	59.4	68.1	70.8	71.1	57.2	65.1	63.8	41.7	63.3	10.4	67.6
Sch	44.4	66.2	53.3	76.1	72.1	68.7	83.4	71.0	50.7	47.0	71.1	49.8	62.8	13.0	0.0
Att	53.8	54.9	59.8	66.4	58.2	69.8	65.4	75.4	57.2	67.4	68.8	37.8	[a]61.2	9.9	72.9
Shi	62.8	0.0	0.0	75.8	0.0	0.0	0.0	0.0	64.6	73.2	79.5	54.2	34.2	36.3	0.0
Av	59.9	78.3	67.2	78.3	70.7	78.6	85.9	80.6	65.2	73.5	76.4	56.0			80.0
SD	6.5	8.8	8.9	5.5	6.7	7.5	7.1	5.8	6.8	8.4	6.5	7.7			6.3

Table 5: Labeled attachment scores of parsers on the 13 test sets. The total score (Tot) and standard deviations (SD) from the average per participant are calculated over the 12 obligatory languages (i.e. excluding Bulgarian). Note that due to the equal sizes of the test sets for all languages, the total scores, i.e. the LAS over the concatenation of the 12 obligatory test sets, are identical (up to the first decimal digit) to the average LAS over the 12 test sets. Averages and standard deviations per data set are calculated ignoring zero scores (i.e. results not submitted). The highest score for each column and those not significantly worse ($p < 0.05$) are shown in bold face. Significance was computed using the official scoring script `eval.pl` and Dan Bikel's Randomized Parsing Evaluation Comparator, which implements stratified shuffling.

[a] Attardi's submitted results contained an unfortunate bug which caused the DEPREL values of all tokens with HEAD=0 to be an underscore (which is scored as incorrect). Using the simple heuristic of assigning the DEPREL value that most frequently occured with HEAD=0 in training would have resulted in a total LAS of 67.5.

(2000), or to the domain-restricted Japanese dialogues of the ATR corpus (Lepage et al., 1998).[32]

Other relatively "easy" data sets are Portuguese (2nd highest average score but, interestingly, the third-longest parsing units), Bulgarian (3rd), German (4th) and Chinese (5th). Chinese also has the second highest top score[33] and Chinese parsing units are the shortest. and Chinese parsing units are the shortest. We note that all "easier" data sets offer large to middle-sized training sets.

The most difficult data set is clearly the Turkish one. It is rather small, and in contrast to Arabic and Slovene, which are equally small or smaller, it covers 8 genres, which results in a high percentage of new FORM and LEMMA values in the test set. It is also possible that parsers get confused by the high proportion (one third!) of non-scoring tokens

[32] Unfortunately, both these treebanks need to be bought, so they could not be used for the shared task. Note also that Japanese dependency parsers often operate on "bunsetsus" instead of words. Bunsetsus are related to chunks and consist of a content word and following particles (if any).

[33] Although this seems to be somewhat of a mystery compared to the ranking according to the average scores. Riedel et al. (2006)'s top score is more than 3% absolute above the second highest score and they offer no clear explanation for their success.

and the many tokens with '_' as either the FORM or LEMMA. There is a clear need for further research to check whether other representations result in better performance.

The second-most difficult data set is Arabic. It is quite small and has by far the longest parsing units. The third-most difficult data set is Slovene. It has the smallest training set. However, its average as well as top score far exceed those for Arabic and Turkish, which are larger. Interestingly, although the treebank text comes from a single source (a translation of Orwell's novel "1984"), there is quite a high proportion of new FORM and LEMMA values in the test set. The fourth-most difficult data set is Czech in terms of the average score and Dutch in terms of the top score. The diffence in ranking for Czech is probably due to the fact that it has by far the largest training set and ironically, several participants could not train on all data within the limited time, or else had to partition the data and train one model for each partition. Likely problems with the Dutch data set are: noisy (C)POSTAG and LEMMA, (C)POSTAG for multiwords, and the highest proportion of non-projectivity.

Factors that have been discussed so far are: the size of the training data, the proportion of new FORM and LEMMA values in the test set, the ratio of (C)POSTAG to DEPREL values, the average length of the parsing unit the proportion of non-projective arcs/parsing units. It would be interesting to derive a formula based on those factors that fits the shared task data and see how well it predicts results on new data sets. One factor that seems to be irrelevant is the head-final versus head-initial distinction, as both the "easiest" and the most difficult data sets are for head-final languages. There is also no clear proof that some language families are easier (with current parsing methods) than others. It would be interesting to test parsers on the Hebrew treebank (Sima'an et al., 2001), to compare performance to Arabic, the other Semitic language in the shared task, or on the Hungarian Szeged Corpus (Csendes et al., 2004), for another agglutinative language.

7.2 Across participants

For most parsers, their ranking for a specific language differs at most a few places from their overall ranking. There are some outliers though. For example, Johansson and Nugues (2006) and Yuret (2006) are seven ranks higher for Turkish than overall, while Riedel et al. (2006) are five ranks lower. Canisius et al. (2006) are six and Schiehlen and Spranger (2006) even eight ranks higher for Dutch than overall, while Riedel et al. (2006) are six ranks lower for Czech and Johansson and Nugues (2006) also six for Chinese. Some of the higher rankings could be related to native speaker competence and resulting better parameter tuning but other outliers remain a mystery. Even though McDonald et al. (2006) and Nivre et al. (2006) obtained very similar overall scores, a more detailed look at their performance shows clear differences. Taken over all 12 obligatory languages, both obtain a recall of more than 89% on root tokens (i.e. those with HEAD=0) but Nivre's precision on them is much lower than McDonald's (80.91 versus 91.07). This is likely to be an effect of the different parsing approaches.

7.3 Across part-of-speech tags

When breaking down by part-of-speech the results of all participants on all data sets, one can observe some patterns of "easy" and "difficult" parts-of-speech, at least in so far as tag sets are comparable across treebanks. The one PoS that everybody got 100% correct are the German infinitival markers (tag PTKZU; like "to" in English). Accuracy on the Swedish equivalent (IM) is not far off at 98%. Other easy PoS are articles, with accuracies in the nineties for German, Dutch, Swedish, Portuguese and Spanish. As several participants have remarked in their papers, prepositions are much more difficult, with typical accuracies in the fifties or sixties. Similarly, conjunctions typically score low, with accuracies even in the forties for Arabic and Dutch.

8 Future research

There are many directions for interesting research building on the work done in this shared task. One is the question which factors make data sets "easy" or difficult. Another is finding out how much of parsing performance depends on annotations such as the lemma and morphological features, which are not yet routinely part of treebanking efforts. In this respect, it would be interesting to repeat ex-

periments with the recently released new version of the TIGER treebank which now contains this information. One line of research that does not require additional annotation effort is defining or improving the mapping from coarse-grained to fine-grained PoS tags.[34] Another is harvesting and using large-scale distributional data from the internet. We also hope that by combining parsers we can achieve even better performance, which in turn would facilitate the semi-automatic enlargement of existing treebanks and possibly the detection of remaining errors. This would create a positive feedback loop. Finally one must not forget that almost all of the LEMMA, (C)POSTAG and FEATS values and even part of the FORM column (the multiword tokens used in many data sets and basically all tokenization for Chinese and Japanese, where words are normally not delimited by spaces) have been manually created or corrected and that the general parsing task has to integrate automatic tokenization, morphological analysis and tagging. We hope that the resources created and lessons learned during this shared task will be valuable for many years to come but also that they will be extended and improved by others in the future, and that the shared task website will grow into an informational hub on multilingual dependency parsing.

References

A. Arun and F. Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of French. In *Proc. of the 43rd Annual Meeting of the ACL*.

D. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proc. of the Human Language Technology Conf. (HLT)*.

E. Black, S. Abney, D. Flickenger, et al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California*.

E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of the 31rd Annual Meeting of the ACL*.

S. Buchholz and D. Green. 2006. Quality control of treebanks: documenting, converting, patching. In *LREC 2006 workshop on Quality assurance and quality measurement for language and speech resources.*

A. Chanev, K. Simov, P. Osenova, and S. Marinov. 2006. Dependency conversion and parsing of the BulTreeBank. In *Proc. of the LREC-Workshop Merging and Layering Linguistic Information*.

Y. Cheng, M. Asahara, and Y. Matsumoto. 2005. Chinese deterministic dependency analyzer: Examining effects of global features and root node finder. In *Proc. of SIGHAN-2005*.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Collins, J. Hajic, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proc. of the 37th Annual Meeting of the ACL*.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proc. of the 34th Annual Meeting of the ACL*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th Annual Meeting of the ACL*.

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

B. Cowan and M. Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

D. Csendes, J. Csirik, and T. Gyimóthy. 2004. The Szeged corpus: a POS tagged and syntactically annotated Hungarian natural language corpus. In *Proc. of the 5th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger-generator. In *Proc. of the 4th Workshop on Very Large Corpora (VLC)*.

A. Dubey and F. Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proc. of the 41st Annual Meeting of the ACL*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Einarsson. 1976. Talbankens skriftspråkskonkordans.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*, pages 340–345.

D. Elworthy. 2000. A finite-state parser with dependency structure output. In *Proc. of the 6th Intern. Workshop on Parsing Technologies (IWPT)*.

H. Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

D. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.

R. Hudson. 1984. *Word Grammar*. Blackwell.

[34] For the Swedish Talbanken05 corpus, that work has been done after the shared task (see the treebank's web site).

T. Kudo and Y. Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*.

S. Kurohashi and M. Nagao. 1994. KN parser: Japanese dependency/case structure analyzer. In *Proceedings of the Workshop on Sharable Natural Language*, pages 48–55.

S. Kurohashi and M. Nagao. 1997. Kyoto University text corpus project. In *Proc. of the 5th Conf. on Applied Natural Language Processing (ANLP)*, pages 115–118.

Y. Lepage, S. Ando, S. Akamine, and H. Iida. 1998. An annotated corpus in Japanese using Tesnière's structural syntax. In *ACL-COLING Workshop on Processing of Dependency-Based Grammars*, pages 109–115.

D. Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

D. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. of the 33rd Annual Meeting of the ACL*, pages 276–283.

M. Marcus, G. Kim, M. Marcinkiewicz, R. Mac-Intyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *Proc. of the Workshop on Human Language Technology (HLT)*.

S. Marinov and J. Nivre. 2005. A data-driven dependency parser for Bulgarian. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 89–100.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of the 11th Conf. of the European Chapter of the ACL (EACL)*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

I. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. The SUNY Press, Albany, N.Y.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of the 43rd Annual Meeting of the ACL*, pages 99–106.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING)*, pages 64–70.

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

K. Oflazer. 1999. Dependency parsing with an extended finite state approach. In *Proc. of the 37th Annual Meeting of the ACL*, pages 254–260.

G. Sampson. 1995. *English for the Computer: The SUSANNE Corpus and analytic scheme*. Clarendon Press.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a tree-bank of modern Hebrew text. In *Journal Traitement Automatique des Langues (t.a.l.) — Special Issue on Natural Language Processing and Corpus Linguistics*.

D. Sleator and D. Temperley. 1993. Parsing English with a link grammar. In *Proc. of the 3rd Intern. Workshop on Parsing Technologies (IWPT)*.

P. Tapanainen and T. Järvinen. 1997. A non-projective dependency parser. In *Proc. of the 5th Conf. on Applied Natural Language Processing (ANLP)*.

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

L. Tesnière. 1959. *Elément de syntaxe structurale*. Klincksieck, Paris.

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

Papers by participants of CoNLL-X (this volume)

G. Attardi. 2006. Experiments with a multilanguage non-projective dependency parser.

E. Bick. 2006. LingPars, a linguistically inspired, language-independent machine learner for dependency treebanks.

S. Canisius, T. Bogers, A. van den Bosch, J. Geertzen, and E. Tjong Kim Sang. 2006. Dependency parsing by inference over high-recall dependency predictions.

M. Chang, Q. Do, and D. Roth. 2006. A pipeline model for bottom-up dependency parsing.

S. Corston-Oliver and A. Aue. 2006. Dependency parsing with reference to Slovene, Spanish and Swedish.

M. Dreyer, D. Smith, and N. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision.

R. Johansson and P. Nugues. 2006. Investigating multilingual dependency parsing.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines.

S. Riedel, R. Çakıcı, and I. Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming.

M. Schiehlen and K. Spranger. 2006. Language independent probabilistic context-free parsing bolstered by machine learning.

N. Shimizu. 2006. Maximum spanning tree algorithm for non-projective labeled dependency parsing.

D. Yuret. 2006. Dependency parsing as a classication problem.