Projective and Non-Projective Turkish Parsing

Ruket Çakıcı
University of Edinburgh
ICCS, School of Informatics
2 Buccleuch Place
Edinburgh EH8 9LW, UK
R.Cakici@sms.ed.ac.uk

Jason Baldridge
University of Texas at Austin
Department of Linguistics
1 University Station B5100
Austin, TX 78712-0198, USA
jbaldrid@mail.utexas.edu

Abstract

We evaluate several dependency parsing models on the METU-Sabancı Turkish Treebank through two main approaches: generative probabilistic phrase-structure parsers and discriminative dependency parsers. We find that the non-projective Maximum Spanning Tree parser of McDonald et al. (2005) is the best parsing model, in part because it recovers crossed dependencies more effectively than projective algorithms. We also show that the choice of tag set is very important and that using morphological information boosts performance, especially when the parser is not given gold standard POS tags. We also discuss some improvements to the treebank itself.

1 Introduction

In the rapidly growing body of work regarding building treebanks and parsers, there is an increasing emphasis on dependency structures rather than phrase structures. There has been a consequent surge in interest in dependency parsing, e.g. for Swedish (Nivre, Hall, and Nilsson, 2004), Czech (McDonald et al., 2005; Nivre and Nilsson, 2005), and Turkish (Eryiğit and Oflazer, 2006). Early work on wide-coverage dependency parsing made use of both dependency-based models (Eisner, 1996) and phrase structure based models (Collins et al., 1999). These models could naturally handle analyses that involved only projective dependencies. Subsequent work has focused on algorithms that capture the crossed dependencies that occur in languages like Czech and Turkish; e.g. pseudo-projective methods (Nivre and Nilsson, 2005) and fully non-projective methods (McDonald et al., 2005).

In this paper, we develop parsing models for Turkish using the METU-Sabancı dependency treebank (Atalay, Oflazer, and Say, 2003; Oflazer et al., 2003). We

investigate different representations of analyses and we use both a phrase structure parser and a non-projective dependency parser. We show that representing distinctions about derivational morphology and nominal case provides large improvements in the accuracy of recovering word-word dependencies. The non-projective maximum spanning tree parser of McDonald et al. (2005) is superior under all conditions, and is especially well suited for crossed dependencies.

2 The METU-Sabancı Treebank

Like Czech, Turkish is highly-inflected and has more word order flexibility than languages like English. It is an agglutinating language, so a single word can be a sentence with tense, modality, polarity, and voice. It allows both local and long-distance scrambling. The former means that arguments of verbs may swap order within a clause, and the latter means that an argument may appear in a higher clause than that of the verb which subcategorises for it.

The METU-Sabancı Treebank is a subcorpus of the METU Turkish Corpus (Atalay, Oflazer, and Say, 2003; Oflazer et al., 2003) that includes material taken from three daily newspapers, 87 journal issues and 201 books. The treebank has 5620 sentences and 53,798 tokens. The average sentence length is about eight words.[1] We exclude nine sentences that are annotated incorrectly, since there was no way to correct them within the current design principles of the treebank. Figure 1 provides an example of how sentences are represented in the treebank. The dependencies are surface ones, so phenomena such as traces and pro-drop are not modelled. Each word can can have only one parent, but words can have more than one dependent. Links are represented from dependent to head in this paper.

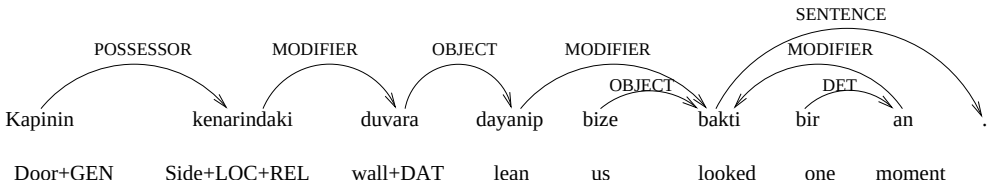


Figure 1: The graphical representation of word-word dependencies for the sentence *(He) looked at us leaning on the wall next to the door, for a moment.*

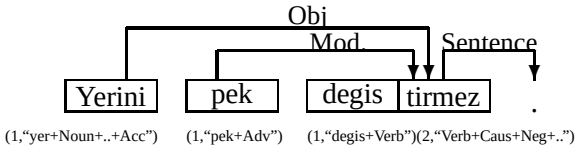
The syntactic relations used to model the dependency relations include labels

[1] One Turkish word typically corresponds to several English words, since the morphological information which exists in the treebank represents additional information including part-of-speech, modality, tense, person, case, causativity, passive voice etc.

such as SUBJECT, OBJECT, POSSESSOR, MODIFIER, SENTENCE, and COORDINATION. Punctuation marks are excluded from dependency structures unless they participate in a relation, such as the use of comma in coordination. The label SENTENCE links the head of the sentence to the punctuation mark or a conjunct in case of coordination.[2]

Morphology is represented in *Inflectional Groups* (IGs). Words with more than one IG either have derivational morphology or valency altering suffixes (e.g. causative and passive forming morphemes for verbs). Figure 1 shows the OBJECT linked to the second IG of *değiştirmez* because *değiş* is intransitive; it only has an object because of the causative morpheme.

(1)



IGs thus play a role in dependency structure. Different IGs can be heads of different dependents. Dependencies always emanate from the final IG of a word. We believe a parser that is capable of recovering the correct IG-IG dependencies for *raw* Turkish text is desirable, but difficult to accomplish. It is not trivial to obtain accurate IG information from raw text since both morphological analysis and disambiguation are necessary. We thus focus on word-word dependencies, like nearly all work in dependency parsing.

We have made many changes to the treebank in order to have improve consistency and correctness, including: (a) fixing incorrect morphological analyses of common words (e.g. *evet* “yes”, *bile* “even”); (b) connecting tokens that previously were not part of the overall dependency structure; (c) changing dependency links or labels of some relatively non-frequent types (e.g. intensifiers, appositions) so they are consistently annotated; and (d) fixing some incorrect dependency links. There are some sentence boundary and tokenisation errors which we did not change in order to keep the number of sentences in the treebank unchanged.

3 Constituency Models

In order to use a phrase-structure parser with the treebank, it is necessary to create constituent trees out of the annotated dependency structures.

[2]This is essentially like identifying the final punctuation mark as the *root* symbol, which is how we treat it when evaluating parser output – see section 5.

3.1 Creating constituent structures

Collins et al. (1999) outline three choices when creating constituent structures from dependencies: (a) branching factor, (b) choice of non-terminal labels, and (c) the set of POS tags to be used. A fourth choice is how to handle non-projective dependencies. We create the flattest possible trees and use the POS tags to create non-terminal labels as explained in Collins et al. (1999). Tags are derived from the morphological analyses in the treebank; there are 15 in the most basic tag set.

In 344 sentences, there is at least one crossed dependency. This makes the mapping process non-trivial. These dependencies could be faithfully represented in constituent structures using mechanisms such as traces, but doing so would involve considerable effort and care. One of our goals is to compare the *straightforward* application of a phrase-structure approach to a natively non-projective dependency-based model. The way we map dependency structures to constituent trees, puts such dependents immediately adjacent to their heads. This gives the correct head-child dependencies but changes the word order from that in the original sentence.

Punctuation is ignored by the translation process unless it is sentence final punctuation that the sentence head is dependent on or a dependency link emanates from it (e.g commas in coordination). The treebank is modified for the sake of consistency such that if there are both a conjunctive word and a punctuation mark next to each other, the word is taken to be the head of the conjunction. In our experiments, we have excluded punctuation without dependency links from all scoring.

3.2 Modifications to the baseline constituent structures

The POS tags we use for the parser are derived from the IGs, which we use to create four distinct tag sets. Our basic tag set uses only the POS tags in a word's last IG. For example the POS tag is *Verb* for (2), and *Noun* for (3).

(2) *istemiyorum* "*I don't want...*": IG='[(1,"iste+Verb+Neg+Prog1+A1sg")]'

(3) *kurtulmak* "*escaping*":
IG='[(1,"kurtul+Verb+Pos") (2,"Noun+Inf+A3sg+Pnon+Nom")]'

However, this is inadequate for representing subordination and extraction because it is not possible to discriminate between a subordinated clause and a *NounP* (*noun phrase*) in the constituent structures we derive from the dependencies.

We thus create enriched POS tags for our second configuration by concatenating the original tag of the morphological stem and the final tag. Words with only one inflectional group are unaffected by this change. This gives *Verb* for (2) and *Verb_Noun* for (3). This kind of information is expected to help subcategorisation choices for some words such as subordinated verbs and thereby help with

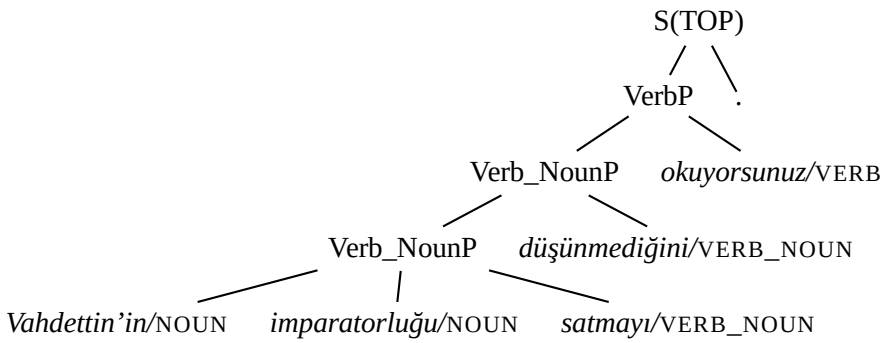


Figure 2: The tree produced using extended tags for the sentence *Vahdettin'in imparatorluğu satmayı düşünmediğini okuyorsunuz*, "You read that Vahdettin was not planning to sell the empire".

predicting the relation between such words and their dependents. This means that *Vahdettin'in* and *imparatorluğu* will be correctly identified as being dependent on the subordinated verb *satmayı* instead of being clustered as a noun group. The same holds for the subordinated verb *düşünmediğini* and the rest as shown in Figure 2.

The third configuration has only the case information for nouns. In this configuration all nouns (derived and root) gets this information via tags such as *Noun_Acc, Noun_Dat, Noun_Loc, Noun_Abl, Noun_Gen, etc.*

Finally, the fourth configuration includes both case information for nouns and the extended tags in the first configuration. We only alter the root nouns (ie. we do not include the case information for the derived nouns.)

4 Parsing Models

4.1 Head-driven generative parsing

Collins (1997) describes several lexicalised head-driven generative parsing models that are now widely known and used. They incorporate varying levels of structural information, such as distance features, the complement/adjunct distinction, subcategorisation and gaps. The core idea is to decompose the calculation of context-free rule probabilities by first generating a head and then generating its left and right modifiers independently.

We use Dan Bikel's multi-lingual parsing engine (Bikel, 2002) to train such models for parsing Turkish. We use Collins' model 1, so the features are standard ones: words, tags and distance over heads and modifiers. We also use the first-order bigram dependencies described in (Collins et al., 1999). With this extension,

the generation of a modifier is dependent on the previous modifier as well as the parent and the head. We use Bikel's default approximation of the previous modifier, where it is either (a) the START symbol (no previous modifiers), (b) a coordinating conjunction, (c) a punctuation mark, or (d) MISC for all other modifiers.

We train the parser on the trees mapped from the dependencies, as described in section 3, and then parse unseen sentences with and without their POS tags. Dependencies are then recovered from the trees derived by the parser by reversing the dependency structure to constituent tree mapping.

4.2 Discriminative dependency parsing

Eisner's cubic generative algorithm (Eisner, 1996) solves the dependency parsing task directly. McDonald et al. (2005) provide a discriminative version of Eisner's dependency parser that scores alternative analyses using large-margin constraints determined with the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003). For English, this parser performs on par with using a Collins model to recover dependencies.

McDonald et al. (2005) formalise dependency parsing as the problem of finding a maximum spanning tree in a directed graph. Again, MIRA is used to determine the weights of dependency links as part of this computation. This algorithm has two major advantages: it runs in $O(n^2)$ time and it handles non-projective dependencies directly. McDonald et al. show that this algorithm significantly improves performance on dependency parsing for Czech, especially on sentences which contain at least one crossed dependency.

We use McDonald's MSTParser with the same four tag sets as we do with the Bikel parser. It uses features that incorporate almost all the different ways in which the words and POS tags of a head and dependent (and words/tags in between them) can be related (McDonald, Crammer, and Pereira, 2005). The basic features are unigram features for the words and part-of-speech for both the parent and the child on their own, and bigram features mixing the words and parts-of-speech of both the parent and the child. Furthermore, there are extended features which encode trigrams of the parts-of-speech of the parent, child and words between them, and there are similar extended features for the words surrounding the parent and child.

Because MIRA is a discriminative approach, many more features can be included without running into problems due to independence assumptions. We thus extended the parser to optionally use a wider range of features; specifically, we use word stems and suffixes to create many new combined features. Examples include parent and child unigram features containing parts-of-speech with suffixes and words with suffixes, and likewise for bigram features. Extended features contain the various combinations of stems and affixes of words in the context along

with the parent and child. We obtain the stems from the treebank itself, and as suffixes we use the remainder of the word after removing the stem.[3] Performance with these features should indicate whether even such a basic morphological analysis is useful for parsing morphologically rich languages like Turkish.

5 Experiments

We report on experiments comparing various configurations which vary the parser, the tagger, and the tag-sets. We use two parser configurations: Collins' phrase-structural model (Bikel's implementation) and the non-projective MIRA model (McDonald's implementation). There are four different tag sets (see Section 3.2): (a) the basic ones [BAS], (b) extended tags (tag of the stem plus tag of final inflectional group) [EXT], (c) case for nouns [CAS] and (d) the combination of (b) and (c) [EC]. Furthermore, we consider an enriched feature set for MSTParser that incorporates stems and suffixes. For tagging, we use either tags produced by a tagger[4] [TT] or gold tags from the treebank [GT]. Note that the tagger is trained on the relevant tag set; for example, it produces full tags like *Noun_Acc* for the CAS set.

We perform 10-fold cross-validation over all 5611 sentences in the treebank. Model performance is given for both word-level and sentence-level dependency accuracy.[5] We provide unlabelled scores for the Collins parser, we give both labelled and unlabelled for MSTParser. Unlabelled word and sentence accuracy are abbreviated as UP and SUP, respectively. LP and SLP are likewise used for labelled accuracy. Scores are globally determined rather than averaged over all individual folds. We take word-final punctuation in the Turkish treebank to constitute the *root* symbol (familiar from other work on dependency parsing) in our evaluation. We do this because the word-final punctuation is given a dependency link to a dummy root symbol, but this happens unambiguously for all sentences. This link is thus trivial to identify, so we exclude it from consideration for scoring all our models.

Turkish is a predominantly head-final language, so the toughest baseline is one for which all dependencies point to the word immediately to the right.[6] This baseline correctly captures 63.2% of the unlabelled dependencies. The left branching baseline is 6.1%, highlighting the rarity of *adjacent* leftward links.

[3] More precisely, we remove a prefix with the same number of characters as the stem in order to handle sound changes. For example, the word *tutsağım* has the stem *tutsak*; from this we get the suffix *ım*.

[4] We use the OpenNLP tagger (`opennlp.sf.net`).

[5] We score our models with the evaluation script used for the CoNLL-X dependency parsing shared task, and evaluate significance with Dan Bikel's significance tester.

[6] Recall that in the Turkish treebank, dependencies point *toward* the head, not away from it as in some other dependency formats.

Table (a) in Figure 3 shows the performance of the Collins model under the various configurations. Even with the most basic tag set, the parser easily beats the right-linking baseline. Using the richer tag sets helps considerably, mirroring the results of Collins et al. (1999) for Czech based on similar strategies. Because many Turkish words convey what would require several words in English, it is too crude to just label them with simple tags like *Noun*. The extended tags (EXT), such as *Noun_Verb*, are crucial for getting the syntactic distribution of such words correct. Case information on tags (CAS) is also fundamental; for example, nominative and genitive nouns appear in very different contexts, so collapsing them as in the basic tag set keeps the parser from being able to handle them appropriately. From the basic tag set BAS to the most complete EC, performance is improved by 6%.

Unsurprisingly, performance suffers when using tags from the tagger rather than the gold standard tags. However, the drop is not great, and the 77.4% accuracy achieved by the model using the EC tags is well above the 63.2% baseline — and it is obtained with only access to the raw words. Note that the parser is capable of tagging for itself – for the same configuration using parser tags instead of the tagger's, the performance is 74.6%. This is actually *not* in line with many previous results, where it is often found to be better to let the parser tag for itself. This is probably due to the fact that both the corpus and the tag sets are small, so the maximum entropy tagger is able to model the tags themselves more effectively than the parser, which obtains its probabilities directly from frequencies and is thus more reliant on larger amounts of data.

Model	UP	SUP
BAS-TT	71.9	35.6
BAS-GT	73.6	37.5
EXT-TT	74.0	36.3
EXT-GT	76.2	39.0
CAS-TT	76.2	38.5
CAS-GT	77.8	40.7
EC-TT	77.4	38.9
EC-GT	79.3	41.5

(a)

Model	UP	SUP	LP	SLP
BAS-TT	79.0	39.1	61.5	19.6
BAS-GT	81.5	42.7	65.9	22.9
EXT-TT	80.0	40.5	62.6	20.1
EXT-GT	83.2	44.6	67.9	23.5
CAS-TT	79.8	40.7	64.7	21.7
CAS-GT	82.5	44.5	70.0	26.6
EC-TT	80.6	41.7	65.3	22.3
EC-GT	84.5	46.9	72.1	28.2

(b)

Figure 3: Performance of the Collins model (a) and the maximum spanning tree model (b) with different tag sets.

Table (b) in Figure 3 shows the results for the maximum spanning tree parser. Across the board, this parser clearly beats the Collins parser on recovering unlabelled dependencies. When given gold tags, the MST parser given access to the

Model	UP	SUP	LP	SLP
BAS-TT	80.6	41.2	62.3	19.4
BAS-GT	83.3	45.5	66.7	22.9
EC-TT	81.3	42.6	65.8	22.5
EC-GT	84.9	47.7	72.3	28.2

Figure 4: Performance of MST non-projective model with the BAS and EC tag sets using stem and suffix features.

same tag set beats the Collins parser by over 5%. It also shows less variance to the choice of tag set, with only a 3% difference between BAS and EC, compared to 6% for the Collins parser. However, its performance when using tags from the tagger rather than gold tags is relatively more affected than the Collins parser. Nonetheless, its absolute performance even with tagger tags is still well above that of the Collins parser.[7]

The labelled scores for the MST parser also show some interesting patterns. Most obvious is that labelled performance is more heavily affected than unlabelled when the parser is given tags from the tagger. This is not surprising since some tags correlate closely with some labels, such as the tag *Noun_Nom* (nominative-case noun) and the label SUBJECT. On a similar note, we see that the CAS tag set (where case is given) improves labelled accuracy from 65.9% for the basic set to 70.0%, a more significant jump than the 67.9% provided by the EXT tag set.

Table 4 provides the results for when the MST parser is given the stems and suffix features in addition to the word and tag features that come out of the box. The additional features provide a significant boost in performance ($p < 0.05$) for all configurations. Most interestingly, the performance when using the tagger tags is a more marked improvement over the model with stems and suffixes. The stems and suffixes essentially provide a means to lexicalise the model with less sensitivity to data sparsity than full words on their own. They thus help keep the model from choosing poorly when it is given an incorrect tag from the tagger. This indicates that lexical information is both useful and sufficient despite the small size of the treebank, in contrast with Eryiğit and Oflazer (2006), whose statistical dependency model pays attention only to tags and distance measures.

The projective dependency parser (Eisner's algorithm) actually performs very similarly to the non-projective one. For example, with gold tags, the EC tag set and the stems and suffixes features, it achieves 84.8%/48.1% UP/SUP and 72.2%/28.5%

[7]Performance would presumably not be as degraded if the parser was *trained* on tags from the tagger rather than gold tags. That way, the material that the parser trains on is deficient in similar ways to the material it is tested on.

LP/SLP, not significantly different from the performance attained by the non-projective parser (see EC-GT in 4). This is actually not very surprising, given that only 2.5% of the dependencies in the treebank are crossed. Nonetheless, we can see the importance of the non-projective algorithm more clearly by scoring both models on just the 344 sentences that had at least one crossed dependency. For these, the non-projective parser with the EC tag set and the stem and suffix features achieves 76.3%/64.0% unlabelled/labelled accuracy. The projective parser with the same tags and features obtains 75.1%/62.9%. This mirrors what McDonald et al. (2005) found for Czech, though the difference they found was greater: 81.5% for the non-projective versus 74.8% for the projective.

The first result on Turkish dependency parsing is Oflazer (2003) who tests a finite state parser on 200 sentences in the METU-Sabancı Treebank. Eryiğit and Oflazer (2006) present a statistical dependency parser for Turkish evaluated on a subset of the treebank. Our work is different from theirs in four ways. First, we focus on word-word dependencies whereas they are concerned with dependencies that include the sub-word level (we argue that morphological segmentation and analysis must be handled automatically to produce a useable parser for the IG level). Second, we parse *all* sentences rather than just projective ones with only rightward links. Third, we provide results for parsing with automatic POS tagging as well as for gold-standard POS tags. Finally, we provide results for labeled as well as unlabeled dependencies.

In the treebank, there are 3501 sentences which have only rightward links.[8] These are the sentences Eryiğit and Oflazer (2006) used in their evaluation. Their best model achieved 81.2% word-word UP on these sentences. Our best *projective* model (EC-GT) gets 85.6%/53.1% UP/SUP and 72.6%/31.6% LP/SLP on these sentences. This large improvement should be considered in the light that our model *can* posit leftward links, a degree of freedom that is not granted to Eryiğit and Oflazer's model. Not surprisingly, on these test sentences, the projective parser is slightly, but significantly, better than the non-projective one's performance of 85.4%/52.3% UP/SUP and 72.2%/30.9% LP/SLP. The Collins model achieves 81.6%/47.8% UP/SUP on the rightward-linking sentences.

It should be noted that some work on dependency parsing uses only gold standard POS tags as input to the parser. Our absolute best result of 84.9% by EC-GT with stems and suffixes can thus be compared to other work which assumes information beyond just the word string, including the results presented in Eryiğit and Oflazer (2006) for Turkish. Yet for a parser to be useful outside the context of the experimental sandbox, it needs to be able to deal with just the words. Our best con-

[8] This is a slightly different number than that given by Eryiğit and Oflazer (2006) (3398) This might be because we do not count crossed dependencies on the IG level.

figuration for this more stringent criterion is the MST parser with the EC tag set, tags from the tagger, and without features based on stems and suffixes (the former of which we obtain from the treebank, not automatically). This model, shown as EC-TT in Figure 3, obtains 80.6% UP.

6 Conclusion and Future Work

We have demonstrated a range of dependency parsing regimes for Turkish. All our models perform well above a right-linking baseline, even when using tags from a tagger rather than gold standard ones. Simple extensions to the tag set provided large improvements to parsing accuracy for all models. The discriminative dependency parsers of McDonald et al. (2005) easily outperform the Collins-style phrase-structure parser. This can in part be attributed to the small size of the corpus – generative models need lots of examples to count, whereas discriminative training iteratively targets a direct reduction in classification error. But we also think the difference can be attributed to the fact that the dependency parsers attack the problem directly and do not need the extra level of indirection of phrase structure trees, which can have complications such as training on phrase structures with uncrossed orders and then testing on sentences with crossed dependencies.

We also extended the MST parser with features based on word stems and suffixes in addition to full words and tags. These features infuse morphology into the parsing model, which would be expected to be important in a morphologically rich language like Turkish. Our results show significant improvements with these features, especially when the parser was supplied tags from a tagger.

Even though the non-projective MST algorithm and the projective Eisner algorithm (both using MIRA) achieve similar performance overall, we showed that the former is significantly better on the subset of sentences in the Turkish treebank which have at least one crossed dependency.

Our results are the first for parsers evaluated on word-word dependencies on *all* sentences in the Turkish treebank. Our focus was on providing a broad comparison of different approaches, so we did not attempt to optimise parameters such as as training iterations and beam widths. We expect improvements could be gained by paying attention to these factors, but leave that for future work.

We are planning to focus on how proper treatment of morphology would affect performance on dependency parsing. We will also train a supertagger with a CCG lexicon derived from the treebank so that we can use CCG categories for morphemes as further features in parsers.

The corrected and extended version of the treebank will be made available to the academic community in the near future.

References

Atalay, Nart B., Kemal Oflazer, and Bilge Say. 2003. The annotation process in the Turkish Treebank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

Bikel, Daniel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, San Francisco.

Collins, M., J. Hajic, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, Maryland, USA.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th Annual Meeting of the ACL*, pages 16–23, Madrid, Spain.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*.

Eisner, Jason. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.

Eryiğit, Gülşen and Kemal Oflazer. 2006. Statistical dependency parsing of Turkish. In *Proceedings of the* 11^{th} *Annual Meeting of the EACL*, pages 89–96, Trento, Italy.

McDonald, Ryan, Koby Crammer, and Fernanda Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, Ann Arbor, MI, USA.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP 2005*, Vancouver, B.C.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *CoNNL 2004*, pages 49–56, Boston, Massachusetts, USA.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *ACL 2005*, pages 99–106, Ann Arbor, MI, USA.

Oflazer, Kemal. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544.

Oflazer, Kemal, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gokhan Tür. 2003. Building a Turkish Treebank. In Abeille Anne, editor, *Treebanks: Building and Using Parsed Corpora*. Kluwer, Dordrecht, pages 261–277.