A Statistical Parser for Czech*

Michael Collins
AT&T Labs–Research,
Shannon Laboratory,
180 Park Avenue,
Florham Park, NJ 07932
mcollins@research.att.com

Jan Hajič
Institute of Formal and Applied Linguistics
Charles University,
Prague, Czech Republic
hajic@ufal.mff.cuni.cz

Lance Ramshaw
BBN Technologies,
70 Fawcett St.,
Cambridge, MA 02138
lramshaw@bbn.com

Christoph Tillmann
Lehrstuhl für Informatik VI,
RWTH Aachen
D-52056 Aachen, Germany
tillmann@informatik.rwth-aachen.de

Abstract

This paper considers statistical parsing of Czech, which differs radically from English in at least two respects: (1) it is a *highly inflected* language, and (2) it has relatively *free word order*. These differences are likely to pose new problems for techniques that have been developed on English. We describe our experience in building on the parsing model of (Collins 97). Our final results – 80% dependency accuracy – represent good progress towards the 91% accuracy of the parser on English (Wall Street Journal) text.

1 Introduction

Much of the recent research on statistical parsing has focused on English; languages other than English are likely to pose new problems for statistical methods. This paper considers statistical parsing of Czech, using the Prague Dependency Treebank (PDT) (Hajič, 1998) as a source of training and test data (the PDT contains around 480,000 words of general news, business news, and science articles annotated for dependency structure). Czech differs radically from English in at least two respects:

- It is a *highly inflected* (HI) language. Words in Czech can inflect for a number of syntactic features: case, number, gender, negation and so on. This leads to a very large number of possible word forms, and consequent sparse data problems when parameters are associated with lexical items. On the positive side, inflectional information should provide strong cues to parse structure; an important question is how to parameterize a statistical parsing model in a way that makes good use of inflectional information.
- It has relatively *free word order* (FWO). For example, a subject-verb-object triple in Czech can generally appear in all 6 possible surface orders (SVO, SOV, VSO etc.).

Other Slavic languages (such as Polish, Russian, Slovak, Slovene, Serbo-croatian, Ukrainian) also show these characteristics. Many European languages exhibit FWO and HI phenomena to a lesser extent. Thus the techniques and results found for Czech should be relevant to parsing several other languages.

This paper first describes a baseline approach, based on the parsing model of (Collins 97), which recovers dependencies with 72% accuracy. We then describe a series of refinements to the model, giving an improvement to 80% accuracy, with around 82% accuracy on newspaper/business articles. (As a point of comparison, the parser achieves 91% dependency accuracy on English (Wall Street Journal) text.)

* This material is based upon work supported by the National Science Foundation under Grant No. (#IIS-9732388), and was carried out at the 1998 Workshop on Language Engineering, Center for Language and Speech Processing, Johns Hopkins University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or The Johns Hopkins University. The project has also had support at various levels from the following grants and programs: Grant Agency of the Czech Republic grants No. 405/96/0198 and 405/96/K214 and Ministry of Education of the Czech Republic Project No. VS96151. We would also like to thank Eric Brill, Barbora Hladká, Frederick Jelinek, Doug Jones, Cynthia Kuo, Oren Schwartz, and Daniel Zeman for many useful discussions during and after the workshop.

2 Data and Evaluation

The Prague Dependency Treebank PDT (Hajič, 1998) has been modeled after the Penn Treebank (Marcus et al. 93), with one important exception: following the Praguian linguistic tradition, the syntactic annotation is based on *dependencies* rather than phrase structures. Thus instead of "nonterminal" symbols used at the non-leaves of the tree, the PDT uses so-called *analytical functions* capturing the type of relation between a dependent and its governing node. Thus the number of nodes is equal to the number of tokens (words + punctuation) plus one (an artificial root node with rather technical function is added to each sentence). The PDT contains also a traditional morpho-syntactic annotation (tags) at each word position (together with a lemma, uniquely representing the underlying lexical unit). As Czech is a HI language, the size of the set of possible tags is unusually high: more than 3,000 tags may be assigned by the Czech morphological analyzer. The PDT also contains machine-assigned tags and lemmas for each word (using a tagger described in (Hajič and Hladka, 1998)).

For evaluation purposes, the PDT has been divided into a training set (19k sentences) and a development/evaluation test set pair (about 3,500 sentences each). Parsing accuracy is defined as the ratio of correct dependency links vs. the total number of dependency links in a sentence (which equals, with the one artificial root node added, to the number of tokens in a sentence). As usual, with the development test set being available during the development phase, all final results has been obtained on the evaluation test set, which nobody could see beforehand.

3 A Sketch of the Parsing Model

The parsing model builds on Model 1 of (Collins 97); this section briefly describes the model. The parser uses a lexicalized grammar — each nonterminal has an associated head-word and part-of-speech (POS). We write non-terminals as $X(x)$: X is the non-terminal label, and x is a $\langle w, t\rangle$ pair where w is the associated head-word, and t as the POS tag. See figure 1 for an example lexicalized tree, and a list of the lexicalized rules that it contains.

Each rule has the form[1]:

$$P(h) \rightarrow L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m) \quad (1)$$

[1] With the exception of the top rule in the tree, which has the form `TOP` $\rightarrow$ `H(h)`.

H is the head-child of the phrase, which inherits the head-word h from its parent P. $L_1...L_n$ and $R_1...R_m$ are left and right modifiers of H. Either n or m may be zero, and $n = m = 0$ for unary rules. For example, in `S(bought,VBD)` $\rightarrow$ `NP(yesterday,NN) NP(IBM,NNP) VP(bought,VBD)`:

$n = 2$	$m = 0$
P = `S`	H = `VP`
L_1 = `NP`	L_2 = `NP`
l_1 = ⟨`IBM, NNP`⟩	l_2 = ⟨`yesterday, NN`⟩
h = ⟨`bought, VBD`⟩	

The model can be considered to be a variant of Probabilistic Context-Free Grammar (PCFG). In PCFGs each rule $\alpha \rightarrow \beta$ in the CFG underlying the PCFG has an associated probability $P(\beta|\alpha)$. In (Collins 97), $P(\beta|\alpha)$ is defined as a product of terms, by assuming that the right-hand-side of the rule is generated in three steps:

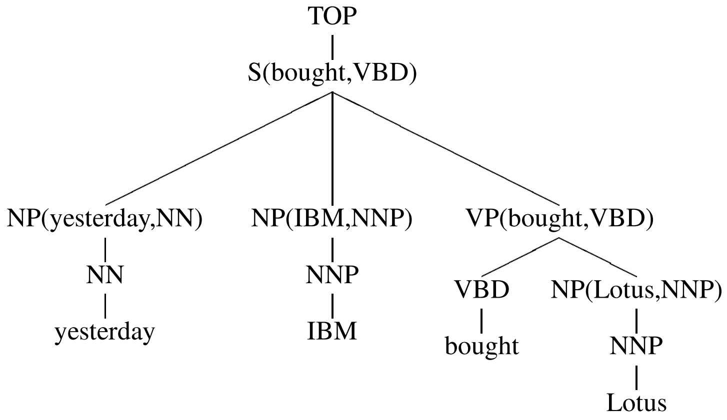
1. Generate the head constituent label of the phrase, with probability $\mathcal{P}_H(H \mid P, h)$.
2. Generate modifiers to the left of the head with probability $\prod_{i=1..n+1} \mathcal{P}_L(L_i(l_i) \mid P, h, H)$, where $L_{n+1}(l_{n+1}) =$ `STOP`. The `STOP` symbol is added to the vocabulary of nonterminals, and the model stops generating left modifiers when it is generated.
3. Generate modifiers to the right of the head with probability $\prod_{i=1..m+1} \mathcal{P}_R(R_i(r_i) \mid P, h, H)$. $R_{m+1}(r_{m+1})$ is defined as `STOP`.

For example, the probability of `S(bought,VBD) -> NP(yesterday,NN) NP(IBM,NNP) VP(bought,VBD)` is defined as

P_h(`VP` | `S,bought,VBD`) ×
P_l(`NP(IBM,NNP)` | `S,VP,bought,VBD`) ×
P_l(`NP(yesterday,NN)` | `S,VP,bought,VBD`) ×
P_l(`STOP` | `S,VP,bought,VBD`) ×
P_r(`STOP` | `S,VP,bought,VBD`)

Other rules in the tree contribute similar sets of probabilities. The probability for the entire tree is calculated as the product of all these terms.

(Collins 97) describes a series of refinements to this basic model: the addition of "distance" (a conditioning feature indicating whether or not a modifier is adjacent to the head); the addition of subcategorization parameters (Model 2), and parameters that model wh-movement (Model 3); estimation




```
TOP                ->  S(bought,VBD)
S(bought,VBD)      ->  NP(yesterday,NN)  NP(IBM,NNP)     VP(bought,VBD)
NP(yesterday,NN)   ->  NN(yesterday)
NP(IBM,NNP)        ->  NNP(IBM)
VP(bought,VBD)     ->  VBD(bought)       NP(Lotus,NNP)
NP(Lotus,NNP)      ->  NNP(Lotus)
```

Figure 1: A lexicalized parse tree, and a list of the rules it contains.

techniques that smooth various levels of back-off (in particular using POS tags as word-classes, allowing the model to learn generalizations about POS classes of words). Search for the highest probability tree for a sentence is achieved using a CKY-style parsing algorithm.

4 Parsing the Czech PDT

Many statistical parsing methods developed for English use lexicalized trees as a representation (e.g., (Jelinek et al. 94; Magerman 95; Ratnaparkhi 97; Charniak 97; Collins 96; Collins 97)); several (e.g., (Eisner 96; Collins 96; Collins 97; Charniak 97)) emphasize the use of parameters associated with dependencies between pairs of words. The Czech PDT contains dependency annotations, but no tree structures. For parsing Czech we considered a strategy of converting dependency structures in training data to lexicalized trees, then running the parsing algorithms originally developed for English. A key point is that the mapping from lexicalized trees to dependency structures is many-to-one. As an example, figure 2 shows an input dependency structure, and three different lexicalized trees with this dependency structure.

The choice of tree structure is crucial in determining the independence assumptions that the parsing model makes. There are at least 3 degrees of freedom when deciding on the tree structures:

1. How "flat" should the trees be? The trees could be as flat as possible (as in figure 2(a)), or binary branching (as in trees (b) or (c)), or somewhere between these two extremes.

2. What non-terminal labels should the internal nodes have?

3. What set of POS tags should be used?

4.1 A Baseline Approach

To provide a baseline result we implemented what is probably the simplest possible conversion scheme:

1. The trees were as flat as possible, as in figure 2(a).

2. The non-terminal labels were "XP", where X is the first letter of the POS tag of the headword for the constituent. See figure 3 for an example.

3. The part of speech tags were the major category for each word (the first letter of the Czech POS set, which corresponds to broad category distinctions such as verb, noun etc.).

The baseline approach gave a result of 71.9% accuracy on the development test set.

Input:
sentence with part of speech tags: I/N saw/V the/D man/N (N=noun, V=verb, D=determiner)

dependencies (*word* ⇒ *Parent*): ⟨I ⇒ saw⟩, ⟨saw ⇒ START⟩, ⟨the ⇒ man⟩, ⟨man ⇒ saw⟩

Output: a lexicalized tree

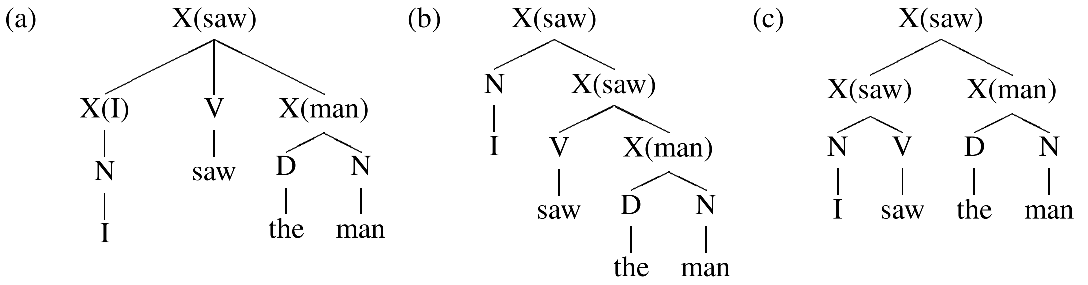


Figure 2: Converting dependency structures to lexicalized trees with equivalent dependencies. The trees (a), (b) and (c) all have the input dependency structure: (a) is the "flattest" possible tree; (b) and (c) are binary branching structures. Any labels for the non-terminals (marked X) would preserve the dependency structure.

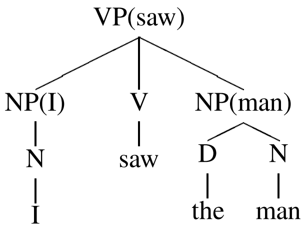


Figure 3: The baseline approach for non-terminal labels. Each label is XP, where X is the POS tag for the head-word of the constituent.

4.2 Modifications to the Baseline Trees

While the baseline approach is reasonably successful, there are some linguistic phenomena that lead to clear problems. This section describes some tree transformations that are linguistically motivated, and lead to improvements in parsing accuracy.

4.2.1 Relative Clauses

In the PDT the verb is taken to be the head of both sentences and relative clauses. Figure 4 illustrates how the baseline transformation method can lead to parsing errors in relative clause cases. Figure 4(c) shows the solution to the problem: the label of the relative clause is changed to `SBAR`, and an additional `VP` level is added to the right of the relative pronoun. Similar transformations were applied for relative clauses involving Wh-PPs (e.g., "the man *to whom* I gave a book"), Wh-NPs (e.g., "the man *whose book* I read") and Wh-Adverbials (e.g., "the place *where* I live").

4.2.2 Coordination

The PDT takes the conjunct to be the head of coordination structures (for example, *and* would be the head of the NP *dogs and cats*). In these cases the baseline approach gives tree structures such as that in figure 5(a). The non-terminal label for the phrase is `JP` (because the head of the phrase, the conjunct *and*, is tagged as J).

This choice of non-terminal is problematic for two reasons: (1) the `JP` label is assigned to *all* coordinated phrases, for example hiding the fact that the constituent in figure 5(a) is an `NP`; (2) the model assumes that left and right modifiers are generated independently of each other, and as it stands will give unreasonably high probability to two *unlike* phrases being coordinated. To fix these problems, the non-terminal label in coordination cases was altered to be the same as that of the second conjunct (the phrase directly to the right of the head of the phrase). See figure 5. A similar transformation was made for cases where a comma was the head of a phrase.

4.2.3 Punctuation

Figure 6 shows an additional change concerning commas. This change increases the sensitivity of the model to punctuation.

4.3 Model Alterations

This section describes some modifications to the parameterization of the model.

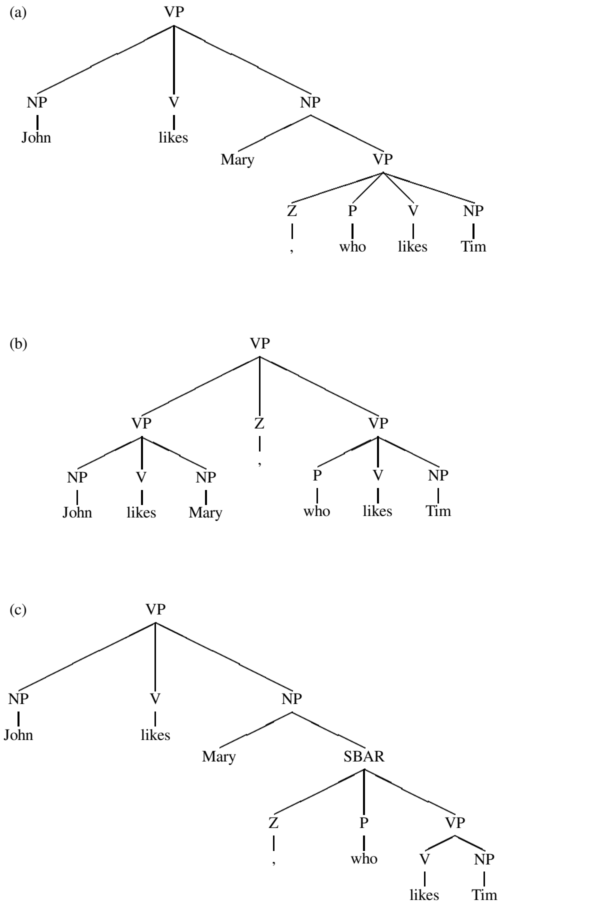


Figure 4: (a) The baseline approach does not distinguish main clauses from relative clauses: both have a verb as the head, so both are labeled VP. (b) A typical parsing error due to relative and main clauses not being distinguished. (note that two *main* clauses can be coordinated by a comma, as in *John likes Mary, Mary likes Tim*). (c) The solution to the problem: a modification to relative clause structures in training data.

4.3.1 Preferences for dependencies that do not cross verbs

The model of (Collins 97) had conditioning variables that allowed the model to learn a preference for dependencies which do not cross verbs. From the results in table 3, adding this condition improved accuracy by about 0.9% on the development set.

4.3.2 Punctuation for phrasal boundaries

The parser of (Collins 96) used punctuation as an indication of phrasal boundaries. It was found that if a constituent $Z \rightarrow \langle ...XY... \rangle$ has two children X and Y separated by a punctuation mark, then Y is generally followed by a punctuation mark or the end of

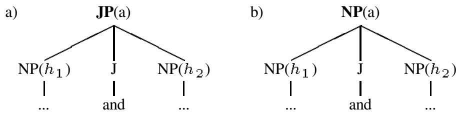


Figure 5: An example of coordination. The baseline approach (a) labels the phrase as a `JP`; the refinement (b) takes the second conjunct's label as the non-terminal for the whole phrase.

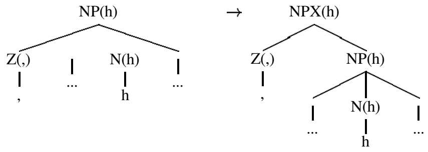


Figure 6: An additional change, triggered by a comma that is the left-most child of a phrase: a new non-terminal `NPX` is introduced.

sentence marker. The parsers of (Collins 96,97) encoded this as a hard constraint. In the Czech parser we added a cost of -2.5 (log probability)[2] to structures that violated this constraint.

4.3.3 First-Order (Bigram) Dependencies

The model of section 3 made the assumption that modifiers are generated independently of each other. This section describes a *bigram* model, where the context is increased to consider the previously generated modifier ((Eisner 96) also describes use of bigram statistics). The right-hand-side of a rule is now assumed to be generated in the following three step process:

1. Generate the head label, with probability
$$\mathcal{P}_{\mathcal{H}}(H \mid P, h)$$
2. Generate left modifiers with probability
$$\prod_{i=1..n+1} \mathcal{P}_{\mathcal{L}}(L_i(l_i) \mid L_{i-1}, \text{P,h,H})$$
where L_0 is defined as a special $NULL$ symbol. Thus the previous modifier, L_{i-1}, is added to the conditioning context (in the previous model the left modifiers had probability $\prod_{i=1..n+1} \mathcal{P}_{\mathcal{L}}(L_i(l_i) \mid \text{P,h,H})$.)
3. Generate right modifiers using a similar bigram process.

Introducing bigram-dependencies into the parsing model improved parsing accuracy by about 0.9 % (as shown in Table 3).

[2]This value was optimized on the development set

1.	main part of speech	8.	person
2.	detailed part of speech	9.	tense
3.	gender	10.	degree of comparison
4.	number	11.	negativeness
5.	case	12.	voice
6.	possessor's gender	13.	variant/register
7.	possessor's number		

Table 1: The 13-character encoding of the Czech POS tags.

4.4 Alternative Part-of-Speech Tagsets

Part of speech (POS) tags serve an important role in statistical parsing by providing the model with a level of generalization as to how classes of words tend to behave, what roles they play in sentences, and what other classes they tend to combine with. Statistical parsers of English typically make use of the roughly 50 POS tags used in the Penn Treebank corpus, but the Czech PDT corpus provides a much richer set of POS tags, with over 3000 possible tags defined by the tagging system and over 1000 tags actually found in the corpus. Using that large a tagset with a training corpus of only 19,000 sentences would lead to serious sparse data problems. It is also clear that some of the distinctions being made by the tags are more important than others for parsing. We therefore explored different ways of extracting smaller but still maximally informative POS tagsets.

4.4.1 Description of the Czech Tagset

The POS tags in the Czech PDT corpus (Hajič and Hladká, 1997) are encoded in 13-character strings. Table 1 shows the role of each character. For example, the tag `NNMP1-----A--` would be used for a word that had "noun" as both its main and detailed part of speech, that was masculine, plural, nominative (case 1), and whose negativeness value was "affirmative".

Within the corpus, each word was annotated with all of the POS tags that would be possible given its spelling, using the output of a morphological analysis program, and also with the single one of those tags that a statistical POS tagging program had predicted to be the correct tag (Hajič and Hladka, 1998). Table 2 shows a phrase from the corpus, with

Form	Dictionary Tags	Machine Tag

poslanci	`NNMP1-----A--`
	`NNMP5-----A--`
	`NNMP7-----A--`
	`NNMS3-----A--`
	`NNMS6-----A--`
		`NNMP1-----A--`

Parlamentu	`NNIS2-----A--`
	`NNIS3-----A--`
	`NNIS6-----A-1`
		`NNIS2-----A--`

schválili	`VpMP---XR-AA-`	`VpMP---XR-AA-`

Table 2: Corpus POS tags for "the representatives of the Parliament approved".

the alternative possible tags and machine-selected tag for each word. In the training portion of the corpus, the correct tag as judged by human annotators was also provided.

4.4.2 Selection of a More Informative Tagset

In the baseline approach, the first letter, or "main part of speech", of the full POS strings was used as the tag. This resulted in a tagset with 13 possible values.

A number of alternative, richer tagsets were explored, using various combinations of character positions from the tag string. The most successful alternative was a two-letter tag whose first letter was always the main POS, and whose second letter was the case field if the main POS was one that displays case, while otherwise the second letter was the detailed POS. (The detailed POS was used for the main POS values D, J, V, and X; the case field was used for the other possible main POS values.) This two-letter scheme resulted in 58 tags, and provided about a 1.1% parsing improvement over the baseline on the development set.

Even richer tagsets that also included the person, gender, and number values were tested without yielding any further improvement, presumably because the damage from sparse data outweighed the value of the additional information present.

4.4.3 Explorations toward Clustered Tagsets

An entirely different approach, rather than searching by hand for effective tagsets, would be to use clustering to derive them automatically. We explored two different methods, bottom-up and top-down, for automatically deriving POS tag sets based on counts of governing and dependent tags extracted from the parse trees that the parser constructs from the training data. Neither tested approach resulted in any improvement in parsing performance compared to the hand-designed "two letter" tagset, but the implementations of each were still only preliminary, and a clustered tagset more adroitly derived might do better.

4.4.4 Dealing with Tag Ambiguity

One final issue regarding POS tags was how to deal with the ambiguity between possible tags, both in training and test. In the training data, there was a choice between using the output of the POS tagger or the human annotator's judgment as to the correct tag. In test data, the correct answer was not available, but the POS tagger output could be used if desired. This turns out to matter only for unknown words, as the parser is designed to do its own tagging, for words that it has seen in training at least 5 times, ignoring any tag supplied with the input. For "unknown" words (seen less than 5 times), the parser can be set either to believe the tag supplied by the POS tagger or to allow equally any of the dictionary-derived possible tags for the word, effectively allowing the parse context to make the choice. (Note that the rich inflectional morphology of Czech leads to a higher rate of "unknown" word forms than would be true in English; in one test, 29.5% of the words in test data were "unknown".)

Our tests indicated that if unknown words are treated by believing the POS tagger's suggestion, then scores are better if the parser is also trained on the POS tagger's suggestions, rather than on the human annotator's correct tags. Training on the correct tags results in 1% worse performance. Even though the POS tagger's tags are less accurate, they are more like what the parser will be using in the test data, and that turns out to be the key point. On the other hand, if the parser allows all possible dictionary tags for unknown words in test material, then it pays to train on the actual correct tags.

In initial tests, this combination of training on the correct tags and allowing all dictionary tags for unknown test words somewhat outperformed the alternative of using the POS tagger's predictions both for training and for unknown test words. When tested with the final version of the parser on the full development set, those two strategies performed at the same level.

5 Results

We ran three versions of the parser over the final test set: the baseline version, the full model with all additions, and the full model with everything but the bigram model. The baseline system on the final test set achieved 72.3% accuracy. The final system achieved 80.0% accuracy[3]: a 7.7% absolute improvement and a 27.8% relative improvement.

The development set showed very similar results: a baseline accuracy of 71.9% and a final accuracy of 79.3%. Table 3 shows the relative improvement of each component of the model[4]. Table 4 shows the results on the development set by genre. It is interesting to see that the performance on newswire text is over 2% better than the averaged performance. The Science section of the development set is considerably harder to parse (presumably because of longer sentences and more open vocabulary).

Modification	Improvement
Coordination	+2.6%
Relative clauses	+1.5%
Punctuation	-0.1% ??
Enriched POS tags	+1.1%
Punctuation	+0.4%
Verb crossing	+0.9%
Bigram	+0.9%
Total change	+7.4%
Total Relative Error reduction	26%

Table 3: A breakdown of the results on the development set.

Genre	Proportion (Sentences/ Dependencies)	Accuracy
Newspaper	50%/44%	81.4%
Business	25%/19%	81.4%
Science	25%/38%	76.0%

Table 4: Breakdown of the results by genre. Note that although the Science section only contributes 25% of the *sentences* in test data, it contains much longer sentences than the other sections and therefore accounts for 38% of the *dependencies* in test data.

[3] The parser fails to give an analysis on some sentences, because the search space becomes too large. The baseline system missed 5 sentences; the full system missed 21 sentences; the full system minus bigrams missed 2 sentences. To score the full system we took the output from the full system minus bigrams when the full system produced no output (to prevent a heavy penalty due to the 21 missed sentences). The remaining 2 unparsed sentences (5 in the baseline case) had all dependencies attached to the root.

[4] We were surprised to see this slight drop in accuracy for the punctuation tree modification. Earlier tests on a different development set, with less training data and fewer other model alterations had shown a good improvement for this feature.

5.1 Comparison to Previous Results

The main piece of previous work on parsing Czech that we are aware of is described in (Kuboň 99). This is a rule-based system which is based on a manually designed set of rules. The system's accuracy is not evaluated on a test corpus, so it is difficult to compare our results to theirs. We can, however, make some comparison of the results in this paper to those on parsing English. (Collins 99) describes results of 91% accuracy in recovering dependencies on section 0 of the Penn Wall Street Journal Treebank, using Model 2 of (Collins 97). This task is almost certainly easier for a number of reasons: there was more training data (40,000 sentences as opposed to 19,000); Wall Street Journal may be an easier domain than the PDT, as a reasonable proportion of sentences come from a sub-domain, financial news, which is relatively restricted. Unlike model 1, model 2 of the parser takes subcategorization information into account, which gives some improvement on English and might well also improve results on Czech. Given these differences, it is difficult to make a direct comparison, but the overall conclusion seems to be that the Czech accuracy is approaching results on English, although it is still somewhat behind.

6 Conclusions

The 80% dependency accuracy of the parser represents good progress towards English parsing performance. A major area for future work is likely to be an improved treatment of morphology; a natural approach to this problem is to consider more carefully how POS tags are used as word classes by the model. We have begun to investigate this issue, through the automatic derivation of POS tags through clustering or "splitting" approaches. It might also be possible to exploit the internal structure of the POS tags, for example through incremental prediction of the POS tag being generated; or to exploit the use of word lemmas, effectively splitting word–word relations into syntactic dependencies (POS tag–POS tag relations) and more semantic (lemma–lemma) dependencies.

References

E. Charniak. 1997. Statistical Parsing with a Context-free Grammar and Word Statistics. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park (1997).

M. Collins. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184-191.

M. Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16-23.

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

J. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. *Proceedings of COLING-96*, pages 340-345.

Jan Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. Issues of Valency and Meaning (Festschrift for Jarmila Panevová). Carolina, Charles University, Prague. pp. 106-132.

Jan Hajič and Barbora Hladká. 1997. Tagging of Inflective Languages: a Comparison. In *Proceedings of the ANLP'97*, pages 136–143, Washington, DC.

Jan Hajič and Barbora Hladka. 1998. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of ACL/Coling'98*, Montreal, Canada, Aug. 5-9, pp. 483-490.

F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, S. Roukos. 1994. Decision Tree Parsing using a Hidden Derivation Model. *Proceedings of the 1994 Human Language Technology Workshop*, pages 272-277.

V. Kuboň. 1999. A Robust Parser for Czech. Technical Report 6/1999, ÚFAL, Matematicko-fyzikální fakulta Karlovy univerzity, Prague.

D. Magerman. 1995. Statistical Decision-Tree Models for Parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276-283.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.

A. Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, Rhode Island.