This paper is going to be part of a report whose full citation may once be like this:

[1] [HAJIČ ET AL. 1998]
Jan Hajič, Eric Brill, Michael Collins, Barbora Hladká, Doug Jones, Cynthia Kuo, Lance Ramshaw, Oren Schwartz, Christoph Tillmann, Daniel Zeman: *Core Natural Language Processing Technology Applicable to Multiple Languages. The Workshop 98 Final Report*. At: http://www.clsp.jhu.edu/ws98/projects/nlp/report/. Center for Language and Speech Processing, Johns Hopkins University, Baltimore 1998.

Workshop '98 Final Report: Zeman's Parser

Daniel Zeman
ÚFAL, MFF, Univerzita Karlova, Praha
zeman@ufal.mff.cuni.cz

The subpart of this project which is internally called 9802, involved a parser for Czech based on direct statistical modeling of tag / word dependencies in Czech. In comparison to the 9801 parser, this does not use any grammar, and works directly with dependency trees instead of parse trees.

Several techniques were developed in preparation for this workshop, that had not been published before and that help the tag-based part of the parser. Although they were prepared before the workshop, the workshop brought a great deal to their implementing for the Prague Dependency Treebank, and to testing them thoroughly.

Then the second part of the workshop was devoted to lexicalizing the parser, i.e. to developing a new statistical model that deals with dependencies between words rather than between the morphological tags. It is true that this part did not help the parser as much as expected (and as reported for other parsers for English) but the outcomes are still challenging and the model developed here enables to continue with the research in future.

Let us very briefly look at the structure of the parser. Its main task can be characterized by the following expression:

$$\arg\max_T p(T|S) = \arg\max_T \left(p(S|T) \cdot p(T)\right)$$

It means that the parser wants to find the dependency tree T that maximizes p(T|S) where S is the sentence being parsed. In other words, we want to construct the tree that most likely is a dependency structure of the sentence S. Because in no way we are able to decide among all possible trees in the universe, we have to decompose the tree probability into edge probabilities. These can be estimated from the relative frequencies of dependencies in training data. Then, using the Viterbi search algorithm, we try to find the tree with the highest possible product of probabilities of its edges. Here we take a significant simplification that the dependency probabilities are statistically independent, i.e.

$$p(T) = \prod_{i=1}^{n} p(d_i)$$

This obviously is not true and weakens the parser so that we had to introduce various constraints, additional models and techniques that help us a little to work around this weakness. A list of them follows; a more detailed description will be given later in this report.

- *Crossing dependencies (so-called non-projective constructions) are not allowed.*
- *A supervised reduction of morphological tag set was done. The number of different tags occurring in corpus decreased from about 1000 to about 400.*
- *A new model for valency was added. It says the parser how likely a node with a given tag has a particular number of child nodes.*
- *We take into account whether the words forming a dependency are adjacent in the sentence or not.*
- *We take into account whether the dependency goes to the right (the governing node precedes the dependent one in the sentence) or to the left.*

The following table gives a brief summary of the results in terms of parsing accuracy. That is, each number is the percentage of dependencies generated by the parser that were correct. Unless stated otherwise, all the numbers characterize parsers trained on over 19000 sentences (approx. 300000 words) and tested on one of the two test data sets, the development test data, and the evaluation test data. The e-test data was used only at the very end of the workshop to cross-validate the results, so most stages have been tested with the d-test only. The training set and both the test sets contained texts from three different sources, from a daily newspaper, from a

business weekly, and from a scientific magazine. It turned out to be much more difficult to parse the last one (mainly because of the sentence length) so it seems reasonable to give separate results for the scientific magazine (labeled "sci") and for the rest (labeled "norm").

The baseline parser includes all the techniques whose development started before the workshop so "baseline" may be read as "before lexicalization". A deeper description of the techniques and a more diversified summary of their contribution to the parsing accuracy will be given later in this report. The final results include the lexical part of the parser as well as some minor improvements that will be described later, too.

	D-test			E-test		
	norm	sci	all	norm	sci	all
Baseline	54	48	51	57	51	54
Final	57	52	55	58	53	56

1. The basic model

The backbone of the present parser is a statistical model that was first studied in [ZEMAN 1998 A]. (It is referred to as "the unigram model" there.) Let us now briefly summarize the fundaments of this model.

The main information the parser relies on (and thus the main thing to gather statistics about) is the dependencies between the words in a sentence. A **word** can be represented either by its **morphological tag** (containing the part-of-speech information as well as the inflection) or by some kind of **lexical information**. Or both.

During the training phase, the parser reads hand-annotated trees and saves the number of times a particular word was hanged on the other one. This way it estimates a probability for each dependency (tree edge) it sees in the data.

In the parsing phase, the parser tries to find the dependencies between tags so that the final tree has the highest possible probability. More formally, it searches for

$$\arg\max_{T} p(T|S) \tag{1}$$

where S is a sentence and T is its dependency tree. In the first approach we assumed that the only condition that bound $p(T)$ with S was that the nodes of the tree had to be exactly the words of the sentence (except of the root — see the format of the Prague Dependency Treebank (PDT)). Otherwise, we dealt with an unconditional probability $p(T)$. Now there is a few new conditions using the word order information; they will be described later.

For simplicity the parser also assumes that the probability of the whole tree is a product of the probabilities of all dependencies in it. In other words, the probability of each particular dependency shall be statistically independent on any other edge in the tree:

$$p(T) = \prod_{i=1}^{n} p(d_i) \tag{2}$$

Obviously this is not true, so there is a significant loss of accuracy which we try to compensate with additional models and features. These will be described later.

2. Ambiguous morphological tags

In our approach the parser needs the input sentences to have morphological tags already assigned to each of the words. Unfortunately, word forms in Czech are highly ambiguous, so for many words there may be a lot of tags, each of them potentially describing the word in some context. Ideally we would need to apply a dictionary (or a morphological analyzer) which would assign a set of all possible tags to each of the words, and then a tagger that would say which tag from the set is true in this particular sentence. Nevertheless, the parser actually does not need a tagger. Instead of a unique tag, automatically assigned by a tagger, it *can* use the whole set from the dictionary.

The solution is as follows: When training, in each dependency we count all possible tag combinations. But they represent only one occurrence of an edge, so we add their numbers in register by $1/n$ rather than by 1, where n is the number of tag combinations for the edge. When parsing, our estimation of the edge probability is the sum of the relative frequencies of all possible tag combinations for the given edge.

The tag ambiguity however increases the computational complexity. Most Czech words have ambiguous tags. Some words can have a relatively big set of them — for instance, there is a class of adjectives which all have an ambiguous form with 27 tags. Combined with lexical ambiguity (word form appurtenance to dictionary headwords), the number can yet increase, and a dependency between two such word forms is not very rare. So, to estimate one edge probability, the parser might need to access the table more than 700 times. Moreover, the tag ambiguity has influence on the accuracy of the results, as will be shown later.

3. Searching edges and building the tree

Obviously it is not trivial to select the optimal set of dependencies that maximizes the probability of the whole tree. Initially we used a greedy algorithm: in each step we added the edge that had the highest probability. Of course the set of allowed dependencies was constrained, the governing node had to be already added to the tree while the dependent node was not allowed to be there.

The greedy algorithm gives maximum probability only for non-oriented trees where $p(A,B) = p(B,A)$. For our case it is only an approximation, maybe too strong. It is impossible to find the real maximum (it needs exponential time) but we tried a Viterbi search. The set of all partially completed trees was truncated to N best of them. In each step we added M most promising (read: most probable) edges to each tree which gave $N \times M$ new trees; then the trees were sorted by their probabilities and the stack was truncated at the level N again. In most experiments (unless stated otherwise) we used $N=M=5$. Hopefully, in some cases this caused the most probable edge to be preserved for use in next rounds with even better gain.

The experiments showed that the initial model was so bad (31%[1], i.e. only some five points over a naive parser which creates chain-trees) that the Viterbi search did not improve anything. In fact almost all the generated trees had higher probability than the "truth" from test data because the probability distribution modeled the reality badly. But as we added new features and additional models, the influence of the Viterbi search increased.

4. Projectivity and crossing dependencies

Let us now discuss the first constraint we imposed on the model. As mentioned before, the version 1 parser did not use the word order of the sentence. But the actual word order information is important for parsing even in such a "free" word order language as Czech. For instance, in the generated parses it was quite often that a noun was connected to a preposition at the other end of the sentence, possibly with commas and verbs in-between, only because this particular dependency was relatively frequent in the training data. We cannot easily employ the absolute **distance** of the words to solve this problem because anytime there may be a phrase of an arbitrary length between the governing and the dependent node; however, there is another phenomenon that seems to be present with most of the ill-formed dependencies. It is called crossing dependencies or **non-projective constructions**.

Here is an example of a correct and a generated parse. The original sentence was *"Bohužel ale jednorázové, takže velkou část spolknou daně.*" which can be translated as "*Unfortunately, [the donation was] given whole at a time so that a big part will be swallowed by taxes.*". Note the crossing edges in the second tree. The floating point numbers in the second tree indicate the probabilities of the dependencies of every node on its parent.

[1] For details, see [ZEMAN 1998 A].

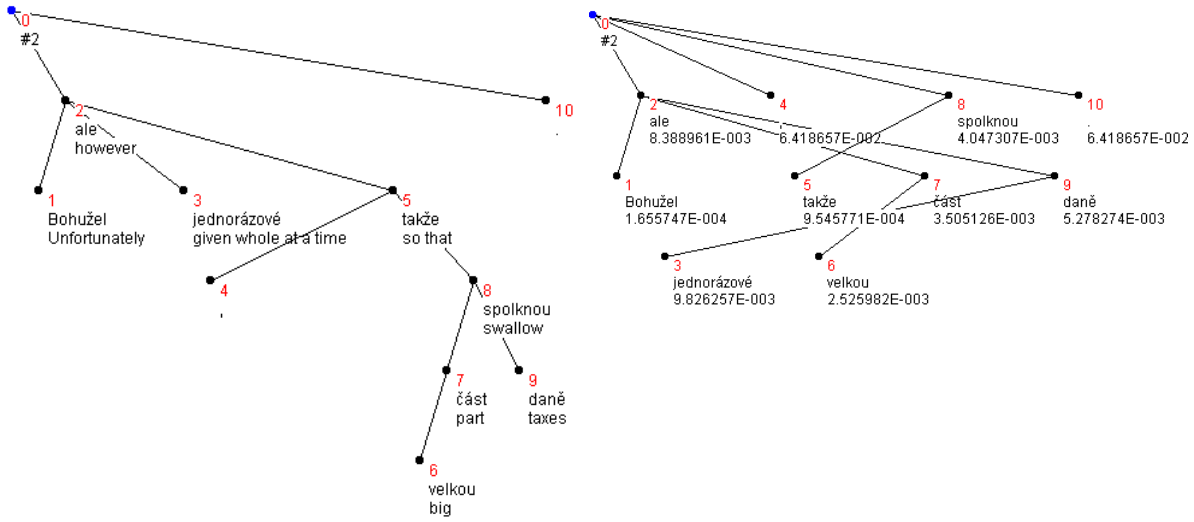


Figure 1: *A correct parse and a generated parse for the same sentence.*

Projectivity is a property that combines tree structure and the word order in sentence. A dependency A-B (where A is the governing node) is a **projective** construction if and only if all the words that are placed between A and B are included in the subtree of A. If you display the tree so that the x-coordinates of nodes correspond to the word order, each non-projective dependency will **cross** at least one perpendicular from another node. (It will not necessarily cross another dependency: in our example the nodes 3, 5, 7, and 9 are connected non-projectively.)

Let us mention that sometimes the non-projective constructions are *not* errors. The tree at the Figure 2 is an example of a *correct* parse with a crossing dependency. Such constructions are generally allowed in Czech but they are rare. Roughly 1.8% of all dependencies in the PDT are non-projective; the average length of a tree in the corpus is 16.3 words (dependencies). About 79.4% of all trees contain no crossing edge. The number of trees that contain one or no crossing dependency is 93.8%, and the number of trees with at most two such dependencies is 98.3%.

This investigation shows that it is a good idea to disable non-projectivities at all until the parser achieves an accuracy level beyond 90%. The experiments showed that using this constraint the parser improved its performance from 31 to 41%, resp. 42% (the first one using the greedy algorithm, the second one using the Viterbi search).

5. Proportional training

This chapter discusses an issue that we hoped might help the parser in case there is no tagger available and the morphological tags are not disambiguated.

As we mentioned before, the ambiguous tag combinations over an edge get a uniform distribution of probabilities. However, some of them are more probable than the other. The model may reflect this because particular tag combinations may have been present in various edges. But this is only an indirect binding and we wanted to emphasize the differences between particular tags, some of them being very frequent while the others were hardly to see at all.

So we built a small unigram probability distribution for the tags and then, training the tree structures, instead of increasing the frequency of each tag combination by *1/n* we weighted the frequency by the probability of the tags in the given combination. Of course it was normalized so that the values for all combinations in one edge summed to 1. However, this

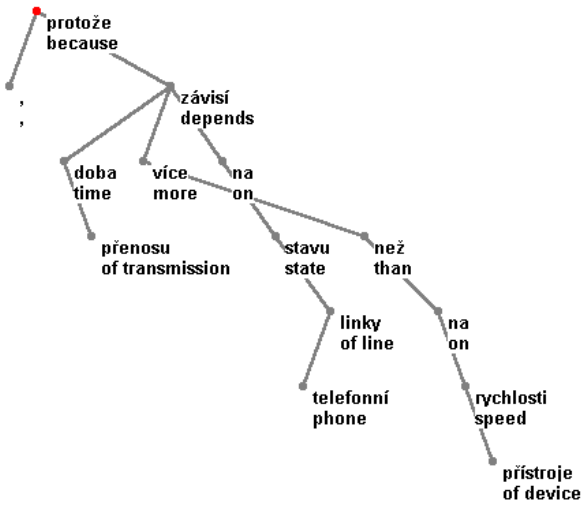
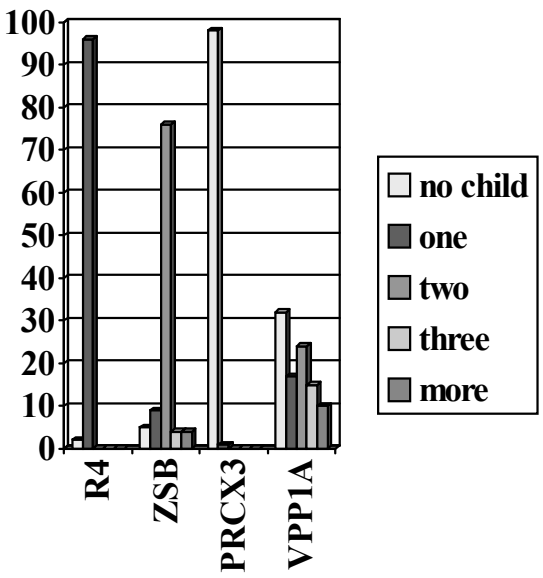


Figure 2: *An example of a correct parse with a crossing dependency.*



feature did not improve the results at all, they were even worse than before. So we omitted the proportional training in ongoing research.

6. Reduced set of morphological tags

Some of the morphological information encoded in the tags has no influence on syntax. For instance, all adjectives, verbs and adverbs can have positive and negative forms, whereas the negative form is constructed completely regularly by simply inserting the prefix "ne-" to the word beginning. This has no influence on the syntax because the negative forms behave syntactically exactly the same way that the positive ones do. So using such information in syntax training may damage the results. For instance, suppose we have seen a positive form of a word depending on a given other word n-times but we have not ever seen the negative form in this context. Then the edge with the negative form will get a probability of zero (or close to zero when smoothed) but in fact it should have the same probability as the one with positive form.

There are many other tags that we have merged together. Besides the negation, we also dropped degree of comparison of adjectives and adverbs, merged some classes of pronouns and numerals with nouns or adjectives, respectively, some classes of numerals match even adverbs. We also discarded information of some morphological and semantic subclasses (SUBPOS), and about the vocalization of prepositions. The VAR category (distinguishing some archaic, rarely used, and otherwise specific forms) had already been ignored. On the other hand, we split the tag for punctuation into several new tags in order to distinguish between commas, full-stops etc. Before the reduction there was 1279 tags present in the training part of PDT, out of about 3000 possible. After the reduction this number decreased to about 452 so the reduction rate is approximately 65%.

Note that this is a supervised reduction where a human has to say what shall be preserved and what shall not. There have been attempts to reduce the tag set using mutual information and a clustering algorithm[2]. These two approaches have not yet been studied together and compared.

7. Valency

Some of the words tend to have many dependents while some others are almost all the times leaves, and the prepositions take mostly exactly one dependent — a head of a noun phrase. The parser described so far does not reflect this observations and, e.g., if there is one preposition in the sentence but several nouns, it often wants to hang all the nouns to the single preposition. To avoid this, we introduced an additional model which knows how often a given tag is a leave, how often it has one child node, how often two, and how often three or more. This is what we call "valency". The parser multiplies the edge probability estimations by the probability of the governing node having $n+1$ children where n is the current number of children. With the ambiguous tags we use an average valency probability over all ambiguous tags for the governing node.

8. Direction and distance

Finally, we introduced two important features that reflect the mutual positions of the two dependency members in the sentence.

The first one is **direction**. The model assigns separate probabilities to a dependency where the depending node is (in the sentence) to the left of the governing node, and to the same dependency where it is to the right of the governing node. For instance, a preposition stands always to the left of its dependent. This feature (together with reduced tags and valency) pushes the accuracy to 49.6%; a more detailed summary of the results will be given later.

[2] See [RAMSHAW] in this report.

The second thing is distance, or better, **adjacency**. Now the parser only asks whether the two nodes are adjacent in the sentence or not. Separate probability estimates are kept for both choices. This feature (together with reduced tags and valency) pushes the accuracy to 48.5%.

Both direction and distance together bring the parser to 53% on the same small test data.

9. Summary of Version 2 results

Some of the features presented in the previous paragraphs immediately and significantly improve the results while some others work only when combined together. For this reason it was not always possible to specify the effect of a feature in terms of one number. Rather we present here a table of a number of feature combinations and the achieved accuracy levels. All experiments were run with the same small set of 199 sentences (trees) and 3036 words (dependencies).

Legend:

The column labeled N shows the number of correctly assigned dependencies, out of the 3036 total. The column numbered A shows the accuracy rate in %.

plain This model does not employ any of the following features. However, it disables the crossing dependencies, which is its only difference from the version 1 parser.

bt The Viterbi 5-best search was used instead of a greedy algorithm. (Searching for a tree which has the highest possible product of probabilities of its edges.)

red The reduced set of morphological tags was used both for training and parsing.

val Valency. A separate distribution of probabilities of number of child nodes for a given tag in the parent node.

prop Proportional training. In an edge that contained *n* possible tag combinations, each combination was counted with a weight corresponding to unconditional relative frequency of the tags participating on the combination. The former method counted each combination with a weight of *1/n* so there was a uniform distribution.

sou Adjacency (distance). For each particular tag dependency, both training and parsing phases distinguished the case, when both words were adjacent, from the other cases.

shr Dependency direction. For each particular tag dependency, both training and parsing phases distinguished the case, when the governing word lay left to the dependent word, from the case, when the governing word lay right to the dependent one.

Model	N	A
prop	1226	40.3821
val+prop	1235	40.6785
plain (only without non-proj)	1237	40.7444
bt+prop	1241	40.8762
red+val	1241	40.8762
red+prop	1242	40.9091
bt+red+prop	1253	41.2714
red	1256	41.3702
bt+red	1256	41.3702
red+val+prop	1258	41.4361
bt	1259	41.4690
bt+val	1271	41.8643
bt+val+prop	1313	43.2477
bt+red+val	1360	44.7958
bt+red+val+prop	1361	44.8287
bt+red+prop+val+sou	1458	48.0237
bt+red+val+sou	1472	48.4848
bt+red+prop+val+shr	1492	49.1436
bt+red+val+shr	1505	49.5718
bt+red+prop+val+sou+shr	1599	52.6680
bt+red+val+sou+shr	1620	53.3597

Figure 3: *Valency distributions of four different tags. R4 is a preposition with accusative (e.g. "na"). ZSB is the tag for the sentence root. PRCX3 is the reflexive pronoun "se". VPP1A is a verb in the present tense, plural, and the 1st person (e.g. "jedeme"). The cases like the first three motivated the valency effort.*

Note that not all possible feature combinations are included in this table.

The last table row shows the maximum performance of the "version 2" parser that was mostly implemented during the preparation phase before the Workshop 98. During the first half of the workshop, it was adjusted to be compatible with the workshop data, and it was tested on larger sets of data. With the workshop d-test data set, it gave **51% accuracy**, which was the baseline for the workshop's second part.

10. Different sources of morphological information

The version 1 parser worked with whole sets of ambiguous tags for each particular word, as assigned by the dictionary. The Workshop 98 version of the PDT allowed to use the disambiguated tags as well. Now there are two other sources of tags: a human annotation (available for training data only), and tags automatically assigned by a tagger (here a probabilistic one). We have mentioned that the ambiguous dictionary tags add computational complexity both in time and space. But we also confirmed the expectation that parsing with the dictionary tags is less accurate than with the disambiguated tags.

In these experiments we investigated different combinations of morphology sources for training and testing. Our training data set contained 13481 sentences. The tests were done on the workshop d-test data, i.e. 3697 sentences. The tagger was trained on approx. 200 000 words outside the treebank.

Training	Testing	Accuracy
13481 sentences	3697 sentences	
dictionary	dictionary	51.42
human	dictionary	52.59
human	tagger	53.44
dictionary	tagger	53.72
tagger	tagger	54.08

We were unable to test also the human-human combination because the test data are not annotated manually. However, we tried this with a small 124-sentence-set cut off the training data (of course, this portion had not been used to train the parsers involved in the experiment). Surprisingly, the human-human combination was by more than 3 percent points worse than the tagger-tagger combination. Eventually this might have been an effect of overtraining: the parser training dealt with some rare constructions that caused some errors in the average test data. Such strange constructions were however hidden by the tagger since it was not able to recognize them.

The better performance of the tagger-tagger combination over the human-tagger one can be explained more easily. The parser probably learned how to deal with the errors the tagger does. The similarity of the training and the test data turned out to be much more important than the accuracy of its morphological annotation (when the tagger accuracy is at least 92%, which is here the case).

The probability distributions for 13481 sentences

	Number of different dependencies	Maximum possible (uniform) entropy	Entropy	Perplexity
human	16163	13.98	11.30	2523
tagger	17391	14.09	11.42	2734
dictionary	82525	16.33	12.65	6442

11. Lexicalization

Finally, the second half of the workshop has been devoted to lexicalizing the parser. So far the parser made no use of lexical information at all: it relied completely on the morphological tags. But it is obvious that there are dependencies in which the lexical side could help a lot. Some notorious examples would be bindings of verbs, some preposition-noun pairs or idiomatic phrases at all. This expectation is supported by the results that are reported for other parsers with English: once they got lexicalized, their performance significantly improved (by about 10 percent points — see also [CHARNIAK 1997]).

We wanted to build a new statistical model that would be similar to the present one in terms of collecting frequencies of particular word pairs but it would identify a word by its lexical contents rather than by the morphological tag. Then we would combine this new model with the old (morphological), and we would hope we'd get better results.

As we mentioned in the beginning, there are two different lexical attributes for each word. One of them is the **word form** as it appeared in the sentence. The other is the **lemma** or the dictionary headword that the form belongs to. Similarly to the morphological tags, the lemmas can be ambiguous, i.e. there can be two different lemmas that, together with some particular tags, can create the same word form. For instance, the word form “je” can be either the singular 3rd person present tense of the verb “být” (“to be” — so the form means “he is”) or the plural accusative of the pronoun “on” (“he” — so the form means “them”). However, the lemma ambiguity is not as high as the tag ambiguity. A probabilistic lemmatizer can do at 98% accuracy.

Now the question was whether to use the forms or the lemmas. The lemmas are a little ambiguous while the forms are not at all. On the other hand, the forms are much sparser than the lemmas. According to an electronic dictionary of Czech, the potential of the language would be about 700K of lemmas but about 20M of forms! However, in our training data only about 20K lemmas and 50K forms were seen[3]. The ratio between possible and present will still increase to the power of two since we are working with word pairs rather than words. Another issue is that lemmas bear a different piece of information than tags while forms are in some sense subsets of tags. If we build our model with forms, we will be able to back off to tags whenever we fall against an unknown word pair. If we use lemmas, we shall treat the lexical and the morphological models as independent and combine them on a same-level basis.

We decided to use the lemmas. And similarly to the tags, for both training and parsing we used the lemmas automatically disambiguated by a lemmatizer.

12. Number of different lemmas and forms in the data

In the following table there are some figures supporting the arguments in the previous section. The columns represent different sorts of information: the lemmas assigned by a dictionary, by a lemmatizer or by a tagger, and the word forms. The rows give the numbers how many different entities of the respective category have been seen in the data *N* times or more; *N* is given in the first column. We can see that there are more word forms than the lemmas but they are sparser.

	Lemmas Dictionary	Lemmas Lemmatizer	Lemmas Human	Forms
≥ 1000	24	25	24	19
≥ 100	217	221	225	145
≥ 50	528	530	528	364
≥ 25	1207	1208	1207	879
≥ 5	5165	5051	5071	5964
> 0	23474	21711	21647	58347

Once we decided to use the lemmas, we trained a model of automatically-assigned-lemma dependencies which had the following parameters (cf. parameters of the morphological models presented above).

The probability distribution for 13481 sentences

	Number of different dependencies	Maximum possible (uniform) entropy	Entropy	Perplexity
automatic lemmas	118827	16.86	15.22	38083

13. How to combine the lexical model with the morphological one

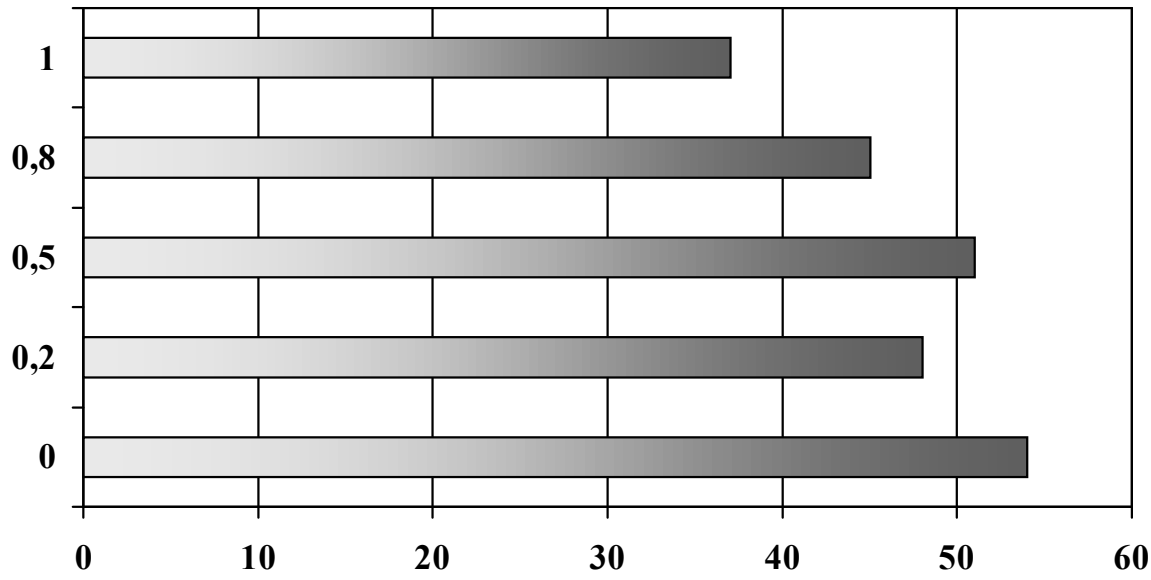
Keeping with the fact that lemmas and tags bear independent pieces of information, we did not want to use the old morphological model just as a back-off. Rather we tried to get a new probability distribution as a weighted interpolation of the two:

$$p(w_1, w_2) = \lambda \cdot p(l_1, l_2) + (1 - \lambda) \cdot p(t_1, t_2) \quad (3)$$

During the workshop we did not attempt to implement an automatic method of estimation of the λ weight from held-out data. Instead, we experimented with several weights estimated by hand.

[3] We have run several experiments to count the lemmas/forms and to extract their histograms. We do not want to diffuse this for-and-against-summary but we think the results could be of interest so we are presenting them in a separate section.

Unfortunately this use of the lexical information does not seem to help as expected. In the following table we can see the influence of the weight λ. The number 1 means that the lexical model was used alone. It is not surprising that id did badly because the lemma dependency distribution is sparse. We had hoped that the lexical model



would made different errors than the morphological one and that the combination of them would do much better. As we see from the table, the combination really did better but it was not able to exceed the accuracy of the morphological model alone. The result for the weight 0.5 forms a local maximum: this confirms that the both models really are good at different areas and that they are able to support each other. But the potential is eather not as great as expected or, more likely, it is not exploited entirely using this method. So the main task for the future work is to explore the methods of combination of two statistical model and to find a more suitable one.

In attempt to improve the accuracy at the workshop we introduced some minor changes. Firstly we stopped to trust lower counts. We considered unknown all dependencies that had been seen only five times or less. (The number five had been found experimentally: it gave the best results on the d-test data, together with four, while three and six worked worse.) We did not want to treat the unknown lexical dependencies the same way as impossible pairs. So in that cases the combined probability was taken as the morphological one with weight 1 (i.e. lexical weight zero) while for the "impossible" (not seen at all) dependencies we still used the old scheme, i.e. half-half weights, thus the result was half the morphological probability. This pushed the accuracy to 54% but it still was not *better* than the version 2 results.

Then, finally, we changed the beam width (stack size) for the Viterbi search from 5 to 50. This of course lengthened the run time by ten but it improved the accuracy by one point. The 55% accuracy is the best we are able to achieve at the moment.

14. Results

The following table gives a summary of the results. Unlike the other experiments, the final runs were done with a parser trained on all the 19 126 sentences (327 597 words) and tested on both the development test data, and the evaluation test data. The e-test data was used first here at the end to cross-validate the results. The training set and both the test sets contained texts from three different sources, from a daily newspaper (*Mladá fronta Dnes* and *Lidové noviny*), from a business weekly (*Československý Profit*), and from a scientific magazine (*Vesmír*). It turned out to be much more difficult to parse the last one (mainly because of the sentence length) so it seems reasonable to give separate results for the scientific magazine (labeled "sci") and for the rest (labeled "norm").

The baseline parser is actually the version 2 parser with all the features discussed in sections 1 – 9, except of the proportional training, which is now obsolete. Thus "baseline" may be read as "before lexicalization". The only middle step is the shift from the dictionary-generated ambiguous data to the automatically disambiguated data. The final results include the lexical part of the parser as well as the minor improvements discussed after the lexicalization.

	D-test	E-test

	norm	sci	all	norm	sci	all
Baseline	54	48	51	57	51	54
Disambiguated	56	51	54			
Final	57	52	55	58	53	56

The processing of the 3697 sentences of the d-test data by the version 2 parser takes on a 686 Intel equipped with Linux about $3\ ^{1}/_{2}$ hours. This number increases with the introduction of the lexical model and with making the Viterbi stack larger. Training the version 2 parser on 19000 sentences takes about 12 minutes. The time dramatically increases with the lexical model because the parser needs to build its lexicon during the training.

15. Acknowledgments

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 author and do not necessarily reflect the views of the National Science Foundation or The Johns Hopkins University.

The author's participation in the project, and the preparation of the annotated data, were also supported by the Grant Agency of the Czech Republic (Grant No. 405/96/K214), and by the Ministry of Education of the Czech Republic (project No. VS96151).

16. References

[1] [CHARNIAK 1997]
Eugene Charniak: *Statistical Techniques for Natural Language Parsing*. In: AI Magazine, Volume 18, No. 4. American Association for Artificial Intelligence, 1997.

[2] [COLLINS 1996]
Michael Collins: *A New Statistical Parser Based on Bigram Lexical Dependencies*. In: Proceedings of the 34th Annual Meeting of the ACL, Santa Cruz 1996.

[3] [COLLINS 1997]
Michael Collins: *Three Generative, Lexicalised Models for Statistical Parsing*. In: Proceedings of the 35th Annual Meeting of the ACL, Madrid 1997.

[4] [ZEMAN 1998 A]
Daniel Zeman: *A Statistical Approach to Parsing of Czech*. In: Prague Bulletin of Mathematical Linguistics 69. Univerzita Karlova, Praha 1998.