Semi-supervised Training for the Averaged Perceptron POS Tagger

Drahomíra "johanka" Spoustová Jan Hajič Jan Raab Miroslav Spousta
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University Prague, Czech Republic
{johanka,hajic,raab,spousta}@
ufal.mff.cuni.cz

Abstract

This paper describes POS tagging experiments with semi-supervised training as an extension to the (supervised) averaged perceptron algorithm, first introduced for this task by (Collins, 2002). Experiments with an iterative training on standard-sized supervised (manually annotated) dataset (10^6 tokens) combined with a relatively modest (in the order of 10^8 tokens) unsupervised (plain) data in a bagging-like fashion showed significant improvement of the POS classification task on typologically different languages, yielding better than state-of-the-art results for English and Czech (4.12 % and 4.86 % relative error reduction, respectively; absolute accuracies being 97.44 % and 95.89 %).

1 Introduction

Since 2002, we have seen a renewed interest in improving POS tagging results for English, and an inflow of results (initial or improved) for many other languages. For English, after a relatively big jump achieved by (Collins, 2002), we have seen two significant improvements: (Toutanova et al., 2003) and (Shen et al., 2007) pushed the results by a significant amount each time.[1]

[1]In our final comparison, we have also included the results of (Giménez and Màrquez, 2004), because it has surpassed (Collins, 2002) as well and we have used this tagger in the data preparation phase. See more details below. Most recently, (Suzuki and Isozaki, 2008) published their Semi-supervised sequential labelling method, whose results on POS tagging seem to be optically better than (Shen et al., 2007), but no significance tests were given and the tool is not available for download, i.e. for repeating the results and significance testing. Thus, we compare our results only to the tools listed above.

Even though an improvement in POS tagging might be a questionable enterprise (given that its effects on other tasks, such as parsing or other NLP problems are less than clear—at least for English), it is still an interesting problem. Moreover, the "ideal"[2] situation of having a single algorithm (and its implementation) for many (if not all) languages has not been reached yet. We have chosen Collins' perceptron algorithm because of its simplicity, short training times, and an apparent room for improvement with (substantially) growing data sizes (see Figure 1). However, it is clear that there is usually little chance to get (substantially) more manually annotated data. Thus, we have been examining the effect of adding a large monolingual corpus to Collins' perceptron, appropriately extended, for two typologically different languages: English and Czech. It is clear however that the features (feature templates) that the taggers use are still language-dependent.

One of the goals is also to have a fast implementation for tagging large amounts of data quickly. We have experimented with various classifier combination methods, such as those described in (Brill and Wu, 1998) or (van Halteren et al., 2001), and got improved results, as expected. However, we view this only as a side effect (yet, a positive one)—our goal was to stay on the turf of single taggers, which are both the common ground for competing on tagger accuracy today and also significantly faster at runtime.[3] Nevertheless, we have found that it is advantageous to use them to (pre-)tag the large amounts of plain text data dur-

[2]We mean easy to use for further research on problems requiring POS tagging, especially multilingual ones.

[3]And much easier to (re)implement as libraries in prototype systems, which is often difficult if not impossible with other people's code.

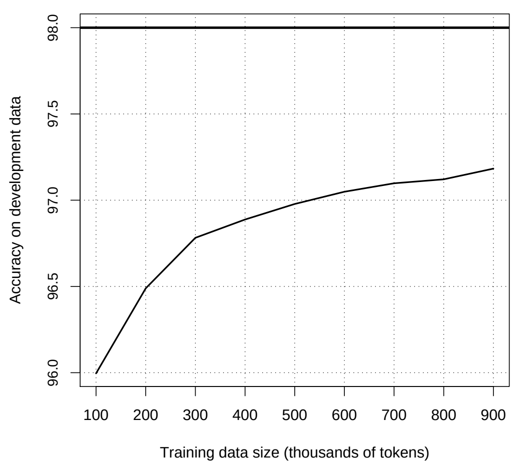

Figure 1: Accuracy of the original averaged perceptron, supervised training on PTB/WSJ (English)

ing the training phase.

Apart from feeding the perceptron by various mixtures of manually tagged ("supervised") and auto-tagged ("unsupervised")[4] data, we have also used various feature templates extensively; for example, we use lexicalization (with the added twist of lemmatization, useful especially for Czech, an inflectionally rich language), "manual" tag classification into large classes (again, useful especially for Czech to avoid the huge, still-to-be-overcome data sparseness for such a language[5]), and sub-lexical features mainly targeted at OOV words. Inspired i.a. by (Toutanova et al., 2003) and (Hajič and Vidová-Hladká, 1998), we also use "lookahead" features (however, we still remain in the left-to-right HMM world – in this respect our solution is closer to the older work of (Hajič and Vidová-Hladká, 1998) than to (Toutanova et al., 2003), who uses bidirectional dependencies to include the right-hand side *disambiguated* tags, which we cannot.)

To summarize, we can describe our system as follows: it is based on (Votrubec, 2006)'s implementation of (Collins, 2002), which has been fed at each iteration by a different dataset consisting of the supervised and unsupervised part: precisely, by a concatenation of the manually tagged training data (WSJ portion of the PTB 3 for English, morphologically disambiguated data from PDT 2.0 for Czech) and a chunk of automatically tagged unsupervised data. The "parameters" of the training process (feature templates, the size of the unsupervised chunks added to the trainer at each iteration, number of iterations, the combination of taggers that should be used in the auto-tagging of the unsupervised chunk, etc.) have been determined empirically in a number of experiments on a development data set. We should also note that as a result of these development-data-based optimizations, no feature pruning has been employed (see Section 4 for details); adding (even lexical) features from the auto-tagged data did not give significant accuracy improvements (and only made the training very slow).

The final taggers have surpassed the current state-of-the-art taggers by significant margins (we have achieved 4.12 % relative error reduction for English and 4.86 % for Czech over the best previously published results, single or combined), using a single tagger. However, the best English tagger combining some of the previous state-of-the-art ones is still "optically" better (yet not significantly—see Section 6).

2 The perceptron algorithm

We have used the Morče[6] tagger (Votrubec, 2006) as a main component in our experiments. It is a reimplementation of the averaged perceptron described in (Collins, 2002), which uses such features that it behaves like an HMM tagger and thus the standard Viterbi decoding is possible. Collins' `GEN(x)` set (a set of possible tags at any given position) is generated, in our case, using a morphological analyzer for the given language (essen-

[4]For brevity, we will use the terms "supervised" and "unsupervised" data for "manually annotated" and "(automatically annotated) plain (raw) text" data, respectively, even though these adjectives are meant to describe the *process* of learning, not the data themselves.

[5]As (Hajič, 2004) writes, Czech has 4400 plausible tags, of which we have observed almost 2000 in the 100M corpus we have used in our experiments. However, only 1100 of them have been found in the manually annotated PDT 2.0 corpus (the corpus on which we have based the supervised experiments). The situation with word forms (tokens) is even worse: Czech has about 20M different word forms, and the OOV rate based on the 1.5M PDT 2.0 data and measured against the 100M raw corpus is almost 10 %.

[6]The name "Morče" stands for "MORfologie ČEštiny" ("Czech morphology", see (Votrubec, 2006)), since it has been originally developed for Czech. We keep this name in this paper as the generic name of the averaged perceptron tagger for the English-language experiments as well. We have used the version available at `http://ufal.mff.cuni.cz/morce/`.

tially, a dictionary that returns all possible tags[7] for an input word form). The transition and output scores for the candidate tags are based on a large number of binary-valued features and their weights, which are determined during iterative training by the averaged perceptron algorithm.

The binary features describe the tag being predicted and its context. They can be derived from any information we already have about the text at the point of decision (respecting the HMM-based overall setting). Every feature can be true or false in a given context, so we can consider the true features at the current position to be the description of a tag and its context.

For every feature, the perceptron keeps its weight coefficient, which is (in its basic version) an integer number, (possibly) changed at every training sentence. After its final update, this integer value is stored with the feature to be later retrieved and used at runtime. Then, the task of the perceptron algorithm is to sum up all the coefficients of true features in a given context. The result is passed to the Viterbi algorithm as a transition and output weight for the current state.[8] We can express it as

$$w(C,T) = \sum_{i=1}^{n} \alpha_i.\phi_i(C,T) \qquad (1)$$

where $w(C,T)$ is the transition weight for tag T in context C, n is the number of features, α_i is the weight coefficient of the i^{th} feature and $\phi_i(C,T)$ is the evaluation of the i^{th} feature for context C and tag T. In the averaged perceptron, the values of every coefficient are added up at each update, which happens (possibly) at each training sentence, and their arithmetic average is used instead.[9] This trick makes the algorithm more resistant to weight oscillations during training (or, more precisely, at the end of it) and as a result, it substantially improves its performance.[10]

[7]And lemmas, which are then used in some of the features. A (high recall, low precision) "guesser" is used for OOV words.

[8]Which identifies unambiguously the corresponding tag.

[9]Implementation note: care must be taken to avoid integer overflows, which (at 100 iterations through millions of sentences) can happen for 32bit integers easily.

[10]Our experiments have shown that using averaging helps tremendously, confirming both the theoretical and practical results of (Collins, 2002). On Czech, using the best feature set, the difference on the development data set is 95.96 % vs. 95.02 %. Therefore, all the results presented in the following text use averaging.

The supervised training described in (Collins, 2002) uses manually annotated data for the estimation of the weight coefficients α. The training algorithm is very simple—only integer numbers (counts and their sums for the averaging) are updated for each feature at each sentence with imperfect match(es) found against the gold standard. Therefore, it can be relatively quickly retrained and thus many different feature sets and other training parameters, such as the number of iterations, feature thresholds etc. can be considered and tested. As a result of this tuning, our (fully supervised) version of the Morče tagger gives the best accuracy among all single taggers for Czech and also very good results for English, being beaten only by the tagger (Shen et al., 2007) (by 0.10 % absolute) and (not significantly) by (Toutanova et al., 2003).

3 The data

3.1 The "supervised" data

For English, we use the same data division of Penn Treebank (PTB) `parsed` section (Marcus et al., 1994) as all of (Collins, 2002), (Toutanova et al., 2003), (Giménez and Màrquez, 2004) and (Shen et al., 2007) do; for details, see Table 1.

data set	*tokens*	*sentences*
train (0-18)	912,344	38,220
dev-test (19-21)	131,768	5,528
eval-test (22-24)	129,654	5,463

Table 1: English supervised data set — WSJ part of Penn Treebank 3

For Czech, we use the current standard Prague Dependency Treebank (PDT 2.0) data sets (Hajič et al., 2006); for details, see Table 2.

data set	*tokens*	*sentences*
train	1,539,241	91,049
dev-test	201,651	11,880
eval-test	219,765	13,136

Table 2: Czech supervised data set — Prague Dependency Treebank 2.0

3.2 The "unsupervised" data

For English, we have processed the North American News Text corpus (Graff, 1995) (without the

WSJ section) with the Stanford segmenter and tokenizer (Toutanova et al., 2003). For Czech, we have used the SYN2005 part of Czech National Corpus (CNC, 2005) (with the original segmentation and tokenization).

3.3 GEN(x): The morphological analyzers

For English, we perform a very simple morphological analysis, which reduces the full PTB tagset to a small list of tags for each token on input. The resulting list is larger than such a list derived solely from the PTB/WSJ, but much smaller than a full list of tags found in the PTB/WSJ.[11] The English morphological analyzer is thus (empirically) optimized for precision while keeping as high recall as possible (it still overgenerates). It consists of a small dictionary of exceptions and a small set of general rules, thus covering also a lot of OOV tokens.[12]

For Czech, the separate morphological analyzer (Hajič, 2004) usually precedes the tagger. We use the version from April 2006 (the same as (Spoustová et al., 2007), who reported the best previous result on Czech tagging).

4 The perceptron feature sets

The averaged perceptron's accuracy is determined (to a large extent) by the set of features used. A feature set is based on feature templates, i.e. general patterns, which are filled in with concrete values from the training data. Czech and English are morphosyntactically very different languages, therefore each of them needs a different set of feature templates. We have empirically tested hundreds of feature templates on both languages, taken over from previous works for direct comparison, inspired by them, or based on a combination of previous experience, error analysis and linguistic intuition.

In the following sections, we present the best performing set of feature templates as determined on the development data set using only the supervised training setting; our feature templates have thus not been influenced nor extended by the unsupervised data.[13]

[11]The full list of tags, as used by (Shen et al., 2007), also makes the underlying Viterbi algorithm unbearably slow.

[12]The English morphology tool is also downloadable as a separate module on the paper's accompanying website.

[13]Another set of experiments has shown that there is not, perhaps surprisingly, a significant gain in doing so.

4.1 English feature templates

The best feature set for English consists of 30 feature templates. All templates predict the current tag as a whole. A detailed description of the English feature templates can be found in Table 3.

Context predicting whole tag	
Tags	Previous tag Previous two tags First letter of previous tag
Word forms	Current word form Previous word form Previous two word forms Following word form Following two word forms Last but one word form
Current word affixes	Prefixes of length 1-9 Suffixes of length 1-9
Current word features	Contains number Contains dash Contains upper case letter

Table 3: Feature templates for English

A total of 1,953,463 features has been extracted from the supervised training data using the templates from Table 3.

4.2 Czech feature templates

The best feature set for Czech consists of 63 feature templates. 26 of them predict current tag as a whole, whereas the rest predicts only some parts of the current tag separately (e.g., detailed POS, gender, case) to avoid data sparseness. Such a feature is true, in an identical context, for several different tags belonging to the same class (e.g., sharing a locative case). The individual grammatical categories used for such classing have been chosen on both linguistic grounds (POS, detailed fine-grained POS) and also such categories have been used which contribute most to the elimination of the tagger errors (based on an extensive error analysis of previous results, the detailed description of which can be found in (Votrubec, 2006)).

Several features can look ahead (to the right of the current position) - apart from the obvious word form, which is unambiguous, we have used (in case of ambiguity) a random tag and lemma of the first position to the right from the current position which might be occupied with a verb (based on dictionary and the associated morphological guesser restrictions).

A total of 8,440,467 features has been extracted from the supervised training data set. A detailed description is included in the distribution downloadable from the Morče website.

5 The (un)supervised training setup

We have extended the averaged perceptron setup in the following way: the training algorithm is fed, in each iteration, by a concatenation of the supervised data (the manually tagged corpus) and the automatically pre-tagged unsupervised data, different for each iteration (in this order). In other words, the training algorithm proper does not change at all: it is the data and their selection (including the selection of the way they are automatically tagged) that makes all the difference.

The following "parameters" of the (unsupervised part of the) data selection had to be determined experimentally:

- the tagging process for tagging the selected data
- the selection mechanism (sequential or random with/without replacement)
- the size to use for each iteration
- and the use and order of concatenation with the manually tagged data.

We have experimented with various settings to arrive at the best performing configuration, described below. In each subsection, we compare the result of our „winning“ configuration with results of the experiments which have the selected attributes omitted or changed; everything is measured on the development data set.

5.1 Tagging the plain data

In order to simulate the labeled training events, we have tagged the unsupervised data simply by a combination of the best available taggers. For practical reasons (to avoid prohibitive training times), we have tagged all the data in advance, i.e. no re-tagging is performed between iterations.

The setup for the combination is as follows (the idea is simplified from (Spoustová et al., 2007) where it has been used in a more complex setting):

1. run N different taggers independently;
2. join the results on each position in the data from the previous step — each token thus ends up with between 1 and N tags, a union of the tags output by the taggers at that position;
3. do final disambiguation (by a single tagger[14]).

Tagger	*Accuracy*
Morče	97.21
Shen	97.33
Combination	97.44

Table 4: Dependence on the tagger(s) used to tag the additional plain text data (English)[16]

Table 4 illustrates why it is advantageous to go through this (still)[16] complicated setup against a single-tagger bootstrapping mechanism, which always uses the same tagger for tagging the unsupervised data.

For both English and Czech, the selection of taggers, the best combination and the best overall setup has been optimized on the development data set. A bit surprisingly, the final setup is very similar for both languages (two taggers to tag the data in Step 1, and a third one to finish it up).

For English, we use three state-of-the-art taggers: the taggers of (Toutanova et al., 2003) and (Shen et al., 2007) in Step 1, and the SVM tagger (Giménez and Màrquez, 2004) in Step 3. We run the taggers with the parameters which were shown to be the best in the corresponding papers. The SVM tagger needed to be adapted to accept the (reduced) list of possible tags.[17]

For Czech, we use the Feature-based tagger (Hajič, 2004) and the Morče tagger (with the new feature set as described in section 4) in Step 1, and an HMM tagger (Krbec, 2005) in Step 3. This combination outperforms the results in (Spoustová et al., 2007) by a small margin.

5.2 Selection mechanism for the plain data

We have found that it is better to feed the training with different chunks of the unsupervised data at each iteration. We have then experimented with

[14]This tagger (possibly different from any of the N taggers from Step 1) runs as usual, but it is given a minimal list of (at most N) tags that come from Step 2 only.

[15]"Accuracy" means accuracy of the semi-supervised method using this tagger for pre-tagging the unsupervised data, not the accuracy of the tagger itself.

[16]In fact, we have experimented with other tagger combinations and configurations as well—with the TnT (Brants, 2000), MaxEnt (Ratnaparkhi, 1996) and TreeTagger (Schmid, 1994), with or without the Morče tagger in the pack; see below for the winning combination.

[17]This patch is available on the paper's website (see Section 7).

three methods of unsupervised data selection, i.e. generating the unsupervised data chunks for each training iteration from the „pool“ of sentences. These methods are: simple sequential chopping, randomized data selection with replacement and randomized selection without replacement. Table 5 demonstrates that there is practically no difference in the results. Thus, we use the sequential chopping mechanism, mainly for its simplicity.

Method of data selection	*English*	*Czech*
Sequential chopping	97.44	96.21
Random without replacement	97.44	96.20
Random with replacement	97.44	96.21

Table 5: Unsupervised data selection

5.3 Joining the data

We have experimented with various sizes of the unsupervised parts (from 500k tokens to 5M) and also with various numbers of iterations. The best results (on the development data set) have been achieved with the unsupervised chunks containing approx. 4 million tokens for English and 1 million tokens for Czech. Each training process consists of (at most) 100 iterations (Czech) or 50 iterations (English); therefore, for the 50 (100) iterations we needed only about 200,000,000 (100,000,000) tokens of raw texts. The best development data set results have been (with the current setup) achieved on the 44th (English) and 33th (Czech) iteration.

The development data set has been also used to determine the best way to “merge” the manually labeled data (the PTB/WSJ and the PDT 2.0 training data) and the unsupervised parts of the data. Given the properties of the perceptron algorithm, it is not too surprising that the best solution is to put (the full size of) the manually labeled data first, followed by the (four) million-token chunk of the automatically tagged data (different data in each chunk but of the same size for each iteration). It corresponds to the situation when the trainer is periodically “returned to the right track” by giving it the gold standard data time to time.

Figure 2 (English) and especially Figure 3 (Czech) demonstrate the perceptron behavior in cases where the supervised data precede the unsupervised data only in selected iterations. A subset of these development results is also present in Table 6.

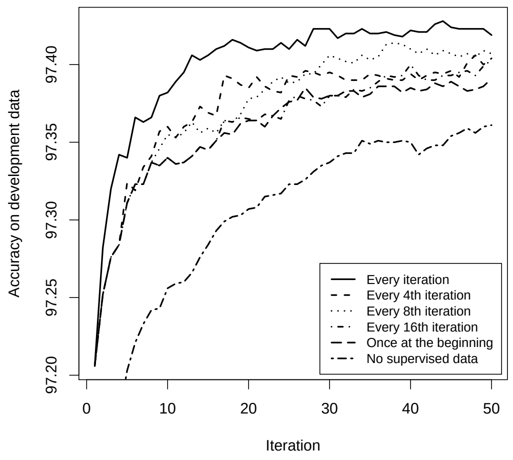

Figure 2: Dependence on the inclusion of the supervised training data (English)

	English	*Czech*
No supervised data	97.37	95.88
Once at the beginning	97.40	96.00
Every training iteration	97.44	96.21

Table 6: Dependence on the inclusion of the supervised training data

5.4 The morphological analyzers and the perceptron feature templates

The whole experiment can be performed with the original perceptron feature set described in (Collins, 2002) instead of the feature set described in this article. The results are compared in Table 7 (for English only).

Also, for English it is not necessary to use our morphological analyzer described in section 3.3 (other variants are to use the list of tags derived solely from the WSJ training data or to give each token the full list of tags found in WSJ). It is practically impossible to perform the unsupervised training with the full list of tags (it would take several years instead of several days with the default setup), thus we compare only the results with morphological analyzer to the results with the list of tags derived from the training data, see Table 8.

It can be expected (some approximated experiments were performed) that the results with the full list of tags would be very similar to the results with the morphological analyzer, i.e. the morphological analyzer is used mainly for technical reasons. Our expectations are based mainly (but not

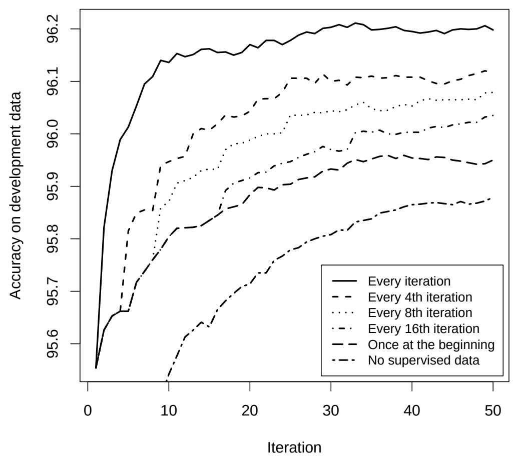

Figure 3: Dependence on the inclusion of the supervised training data (Czech)

only) on the supervised training results, where the performance of the taggers using the morphological analyzer output and using the full list of tags are nearly the same, see Table 9.

Feature set	*Accuracy*
Collins'	97.38
Our's	97.44

Table 7: Dependence on the feature set used by the perceptron algorithm (English)

GEN(x)	*Accuracy*
List of tags derived from train	97.13
Our morphological analyzer	97.44

Table 8: Dependence on the GEN(x)

6 Results

In Tables 10 and 11, the main results (on the eval-test data sets) are summarized. The state-of-the art taggers are using feature sets discribed in the corresponding articles ((Collins, 2002), (Giménez and Màrquez, 2004), (Toutanova et al., 2003) and (Shen et al., 2007)), Morče supervised and Morče semi-supervised are using feature set desribed in section 4.

For significance tests, we have used the paired Wilcoxon signed rank test as implemented in the R package (R Development Core Team, 2008) in `wilcox.test()`, dividing the data into 100 chunks (data pairs).

GEN(x)	*Accuracy*
List of tags derived from train	95.89
Our morphological analyzer	97.17
Full tagset	97.15

Table 9: Supervised training results: dependence on the GEN(x)

Tagger	accuracy
Collins	97.07 %
SVM	97.16 %
Stanford	97.24 %
Shen	97.33 %
Morče supervised	97.23 %
combination	97.48 %
Morče semi-supervised	**97.44 %**

Table 10: Evaluation of the English taggers

Tagger	accuracy
Feature-based	94.04 %
HMM	94.82 %
Morče supervised	95.67 %
combination	95.70 %
Morče semi-supervised	**95.89 %**

Table 11: Evaluation of the Czech taggers

6.1 English

The combination of the three existing English taggers seems to be best, but it is not significantly better than our semi-supervised approach.

The combination is significantly better than (Shen et al., 2007) at a very high level, but more importantly, Shen's results (currently representing the replicable state-of-the-art in POS tagging) have been significantly surpassed also by the semi-supervised Morče (at the 99 % confidence level).

In addition, the semi-supervised Morče performs (on single CPU and development data set) 77 times faster than the combination and 23 times faster than (Shen et al., 2007).

6.2 Czech

The best results (Table 11) are statistically significantly better than the previous results: the semi-supervised Morče is significantly better than both the combination and the supervised (original) variant at a very high level.

7 Download

We decided to publish our system for wide use under the name COMPOST (Common POS Tagger). All the programs, patches and data files are available at the website http://ufal.mff.cuni.cz/compost under either the original data provider license, or under the usual GNU General Public License, unless they are available from the widely-known and easily obtainable sources (such as the LDC, in which case pointers are provided on the download website).

The Compost website also contains easy-to-run Linux binaries of the best English and Czech single taggers (based on the Morče technology) as described in Section 6.

8 Conclusion and Future Work

We have shown that the "right"[18] mixture of supervised and unsupervised (auto-tagged) data can significantly improve tagging accuracy of the averaged perceptron on two typologically different languages (English and Czech), achieving the best known accuracy to date.

To determine what is the contribution of the individual ”dimensions” of the system setting, as described in Sect. 5, we have performed experiments fixing all but one of the dimensions, and compared their contribution (or rather, their loss when compared to the best ”mix” overall). For English, we found that excluding the state-of-the-art-tagger (in fact, a carefully selected combination of taggers yielding significantly higher quality than any of them has) drops the resulting accuracy the most (0.2 absolute). Significant yet smaller drop (less than 0.1 percent) appears when the manually tagged portion of the data is not used or used only once (or infrequently) in the input to the perceptron's learner. The difference in using various feature templates (yet all largely similar to what state-of-the-art taggers currently use) is not significant. Similarly, the way the unsupervised data is selected plays no role, either; this differs from the bagging technique (Breiman, 1996) where it *is* significant. For Czech, the drop in accuracy appears in all dimensions, except the unsupervised data selection one. We have used novel features inspired by previous work but not used in the standard perceptron setting yet (linguistically motivated tag classes in features, lookahead features). Interestingly, the resulting tagger is better than even a combination of the previous state-of-the-art taggers (for English, this comparison is inconclusive).

We are working now on parallelization of the perceptron training, which seems to be possible (based i.a. on small-scale preliminary experiments with only a handful of parallel processes and specific data sharing arrangements among them). This would further speed up the training phase, not just as a nice bonus per se, but it would also allow for a semi-automated feature template selection, avoiding the (still manual) feature template preparation for individual languages. This would in turn facilitate one of our goals to (publicly) provide single-implementation, easy-to-maintain state-of-the-art tagging tools for as many languages as possible (we are currently preparing Dutch, Slovak and several other languages).[19]

Another area of possible future work is more principled tag classing for languages with large tagsets (in the order of 10^3), and/or adding syntactically-motivated features; it has helped Czech tagging accuracy even when only the "introspectively" defined classes have been added. It is an open question if a similar approach helps English as well (certain grammatical categories can be generalized from the current WSJ tagset as well, such as number, degree of comparison, 3rd person present tense).

Finally, it would be nice to merge some of the approaches by (Toutanova et al., 2003) and (Shen et al., 2007) with the ideas of semi-supervised learning introduced here, since they seem orthogonal in at least some aspects (e.g., to replace the rudimentary lookahead features with full bidirectionality).

Acknowledgments

The research described here was supported by the projects *MSM0021620838* and *LC536* of *Ministry of Education, Youth and Sports* of the Czech Republic, *GA405/09/0278* of the Grant Agency of the Czech Republic and *1ET101120503* of Academy of Sciences of the Czech Republic.

[18]As empirically determined on the development data set.

[19]Available soon also on the website.

References

Thorsten Brants. 2000. TnT - a Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 224–231, Seattle, WA. ACL.

Leo Breiman. 1996. Bagging predictors. *Mach. Learn.*, 24(2):123–140.

Eric Brill and Jun Wu. 1998. Classifier Combination for Improved Lexical Disambiguation. In *Proceedings of the 17th international conference on Computational linguistics*, pages 191–195, Montreal, Quebec, Canada. Association for Computational Linguistics.

CNC, 2005. *Czech National Corpus – SYN2005*. Institute of Czech National Corpus, Faculty of Arts, Charles University, Prague, Czech Republic.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, pages 1–8, Philadelphia, PA.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A General POS Tagger Generator Based on Support Vector Machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46, Lisbon, Portugal.

David Graff, 1995. *North American News Text Corpus*. Linguistic Data Consortium, Cat. LDC95T21, Philadelphia, PA.

Jan Hajič and Barbora Vidová-Hladká. 1998. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of the 17th international conference on Computational linguistics*, pages 483–490. Montreal, Quebec, Canada.

Jan Hajič. 2004. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, Prague.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. 2006. Prague Dependency Treebank v2.0, CDROM, LDC Cat. No. LDC2006T01. Linguistic Data Consortium, Philadelphia, PA.

Pavel Krbec. 2005. *Language Modelling for Speech Recognition of Czech*. Ph.D. thesis, UK MFF, Prague, Malostranské náměstí 25, 118 00 Praha 1.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

R Development Core Team, 2008. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the 1st EMNLP*, pages 133–142, New Brunswick, NJ. ACL.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, page 9pp., Manchester, GB.

Libin Shen, Giorgio Satta, and Aravind K. Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic, June. Association for Computational Linguistics.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. 2007. The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing 2007*, pages 67–74, Prague, Czech Republic, June. Association for Computational Linguistics.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL-08: HLT*, pages 665–673, Columbus, Ohio, June. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Edmonton, Canada. Association for Computational Linguistics.

Hans van Halteren, Walter Daelemans, and Jakub Zavrel. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2):199–229.

Jan Votrubec. 2006. Morphological Tagging Based on Averaged Perceptron. In *WDS'06 Proceedings of Contributed Papers*, pages 191–195, Prague, Czech Republic. Matfyzpress, Charles University.