Part-of-Speech Tagging Using Parallel Weighted Finite-State Transducers

Department of Modern Languages
University of Helsinki
Helsinki, Finland
{miikka.silfverberg,krister.linden}@helsinki.fi

Abstract. We use parallel weighted finite-state transducers to implement a part-of-speech tagger, which obtains state-of-the-art accuracy when used to tag the Europarl corpora for Finnish, Swedish and English. Our system consists of a weighted lexicon and a guesser combined with a bigram model factored into two weighted transducers. We use both lemmas and tag sequences in the bigram model, which guarantees reliable bigram estimates.

Keywords: Weighted Finite-State Transducer, Part-of-Speech Tagging, Markov Model, Europarl.

1 Introduction

Part-of-Speech (POS) taggers play a crucial role in many language applications such as parsers, speech synthesizers, information retrieval systems and translation systems. Systems, which need to process a lot of data, benefit from fast taggers. Generally it is easier to find faster implementations for simple models than for complex ones, so simple models should be preferred, when tagging speed is crucial.

We demonstrate that a straightforward first order Markov model, is sufficient to obtain state-of-the-art accuracy when tagging English, Finnish and Swedish Europarl corpora [Koehn 2005]. The corpora were tagged using the Connexor fdg parsers [Järvinen et al. 2004] and we used the tagged corpora both for training and as a gold standard in testing. Our results indicate that bigram probabilities yield accurate tagging, if lemmas are included in POS analyzes.

Our model consists of a weighted lexicon, a guessing mechanism for unknown words, and two bigram models. We analyze each word in a sentence separately using the weighted lexicon and guesser. The analyzes are then combined into one acyclic minimal weighted finite-state transducer (WFST), whose paths correspond to possible POS analyzes of the sentence. The paths in the sentence WFST are re-scored using the bigram models.

The bigram models assign weights for pairs of successive word forms and corresponding POS analyzes including lemmas. One of the models assigns weight for POS analyzes of word form bigrams starting at even positions in the sentence

H. Loftsson, E. Rögnvaldsson, S. Helgadóttir (Eds.): IceTAL 2010, LNAI 6233, pp. 369–380, 2010.

and the other one assigns weights for bigrams starting at odd positions. Both bigram models are implemented as WFSTs.

The sentence WFST and bigram model WFSTs are combined using weighted intersecting composition [Silfverberg and Lindén 2009], which composes the sentence WFST with the simulated intersection of the bigram models. Finally the POS analysis of the sentence is obtained using a best paths algorithm [Mohri and Riley 2002]. The WFSTs and algorithms for parsing were implemented using an open source transducer library HFST [Linden et al. 2009].

The paper is structured as follows. We first review earlier relevant research in section 2. We then formalize the POS tagging task in section 3 and present our model for a POS tagger as an instance of the general formulation in section 4. In section 5 we demonstrate how to implement the model using WFSTs.

The remainder of the paper deals with training and testing the POS tagger. We present the corpora and parsers used in training and tests in section 6, describe training of the model in section 7, evaluate the implementation in section 8 and analyze the results of the evaluation and present future research directions in section 9. Lastly we conclude the paper in section 10.

2 Previous Research

Statistical POS tagging is a common task in natural language applications. POS taggers can be implemented using a variety of statistical models including Hidden Markov Models (HMM) [Church 1999] [Brants 2000] and Conditional Random Fields [Lafferty et al. 2001].

Markov models are probably the most widely used technique for POS tagging. Some older systems such as [Cutting 1992] used first order models, but the accuracies reported were not very good. E.g. [Cutting 1992] report an accuracy of 96 % for tagging English text. Newer systems like [Brants 2000] have used second order models, which generally lead to better tagging accuracy. [Brants 2000] reports accuracy of 96.46% for tagging the Penn Tree Bank. More recent second order models further improve on accuracy. [Collins 2002] reports 97.11% accuracy and [Shen et al. 2007] 97.33% accuracy on the Penn Tree Bank.

We use lemmas in our bigram model as did [Thede and Harper 1999], who used lexical probabilities in their second order HMM for tagging English and obtained improved accuracy (96% – 97%) w.r.t. a second order model using plain tag sequences. In contrast to this, our model uses only bigram probabilities and it is not an HMM, since we only use frequency counts of POS analyzes for word pairs. In addition we split our bigram model into two components, which reduces its size thus allowing us to use a larger training material.

The idea of syntactic parsing and POS tagging using parallel finite-state constraints was outlined by [Koskenniemi 1990]. The general idea in our system is the same, but instead of a rule-based morphological disambiguator, we implement a statistical tagger using WFSTs. Still, hand-crafted tagging constraints could be added to the system.

3 Formulation of the POS Tagging Task

In this section we formulate the task of Part-of-Speech (POS) tagging and describe probabilistic POS taggers formally.

By a sentence, we mean a sequence of syntactic tokens $s = (s_1 \ ... \ s_n)$ and by a POS analysis of the sentence s, we mean a sequence of POS analyzes $t = (t_1, \ ..., \ t_n)$. We include lemmas in POS analyzes. For each i, the analysis t_i corresponds to the token s_i in sentence s. We denote the set of all sentences by S and the set of all analyzes by T.

A *POS tagger* is a machine which associates each sentence s with its most likely *POS analysis* t_s. To find the most likely POS analyzes for the sentence s, the model estimates the probabilities for all possible analyzes of s using a distribution P. For the sentence s and every possible POS analysis t, the distribution P associates a probability $P(t, s)$. Keeping t fixed, the mapping $s \mapsto P(t, s)$ is a normalized probability distribution. The most likely analysis t_s of the sentence s is the analysis which maximizes the probability $P(t, s)$, i.e.

$$t_s = \arg\max_t P(t, s).$$

The distribution P can consist of a number of component distributions P_i, each giving probability $P_i(s, t)$ for sentence s and analysis t. The component probabilities are combined using some function $F : [0, 1]^n \to [0, 1]$ to obtain

$$P(s, t) = F\big(P_i(t, s), \ ..., \ P_n(t, s)\big).$$

The function F should be chosen in such a way that P is nonnegative and satisfies

$$\sum_{t \in T} P(t, s) = 1$$

for each sentence s.

Often a convex linear function F is used to combine estimates given by the component models. In such a case the model P is called a *linear interpolation* of the models P_i.

4 A Probabilistic First Order Model

In this section we describe the idea behind our POS tagger. We use a bigram model for POS tagging. Thus the probability of a given tagging of a sentence is estimated using analyzes of word pairs.

Since we make use of extensive training material, we may include lemmas in bigrams. Although the training material is extensive, the tagger will still encounter bigrams which did not occur in the training material or only occurred once or twice. In such cases we want to use unigram probabilities for estimating the best POS analysis. Hence we weight all analyzes using probabilities given by both the unigram and bigram models, but weight bigram probabilities heavily while only giving unigram probabilities a small weight. Hence unigram probabilities become significant only when bigram probabilities are very close to each other.

4.1 The Unigram Model

The unigram model emits plain unigram probabilities $p_u(t, s_x)$ for analyzes t given a word form s_x (we use the index x to signify that $p_u(t, s_x)$ is independent of the context of the word form s_x). Unigram probabilities are readily computed from training material. The probability of the analysis $t = (t_1 \ ... \ t_n)$ given the sentence $s = (s_1 \ ... \ s_n)$ assigned by the unigram model is

$$P_u(t, s) = \prod_{i=1}^{n} p_u(t_i, s_i).$$

In practice it is not possible to train the unigram model for all possible word forms in highly inflecting languages with productive compounding mechanism such as Finnish or Turkish. Instead the probabilities for analyzes given a word form need to be estimated using probabilities for words with similar suffixes. For instance, if the word form *foresaw* was not observed during training, we can give it a similar distribution of analyzes as the word *saw* receives, since *saw* shares a three-letter suffix with *foresaw*.

In practice such estimation relying on analogy is accomplished by a so called POS guesser, which seeks words with maximally long suffixes in common with an unknown word. It then assigns probabilities for POS analyzes of the unknown word on basis of the analyzes of the known words. [Linden 2009a] shows how a guesser can be integrated with a weighted lexicon in a consistent way.

4.2 The Bigram Models

We use two bigram models Q_o and Q_e giving probabilities for bigrams starting at even and odd positions in the sentence. The estimates are built using plain bigram probabilities for tagging a word-pair s_1 and s_2 with analyzes t_1 and t_2 respectively[1]. These probabilities $p_b(t_1, s_1, t_2, s_2)$ are easily computed from a training corpus.

For an analysis $t = t_1 \ ... \ t_{2k}$ and a sentence $s = s_1 \ ... \ s_{2k}$ of even length $2k$, the models Q_o and Q_e give bigram scores

$$Q_o(t, s) = \prod_{i=1}^{k} p_b(t_{2i-1}, s_{2i-1}, t_{2i}, s_{2i}),\ Q_e(t, s) = \prod_{i=1}^{k-1} p_b(t_{2i}, s_{2i}, t_{2i+1}, s_{2i+1})$$

For an analysis $t = t_1 \ ... \ t_{2k+1}$ and a sentence $s = s_1 \ ... \ s_{2k+1}$ of odd length $2k + 1$, the models Q_o and Q_e give bigram scores

$$Q_o(t, s) = \prod_{i=1}^{k} p_b(t_{2i-1}, s_{2i-1}, t_{2i}, s_{2i}),\ Q_e(t, s) = \prod_{i=1}^{k} p_b(t_{2i}, s_{2i}, t_{2i+1}, s_{2i+1})$$

[1] In literature, it is often suggested that one should instead compute probabilities of word form bigrams given POS analysis bigrams. We cannot do this, since we include lemmas in POS analyzes. This makes the probability of a word form given a POS analysis either 0 or 1 since most analyzes only have one realization as a word form.

4.3 Combining the Unigram and Bigram Models

The standard way of forming a model from P_u, Q_o and Q_e would be to use linear interpolation. We do not want to do this, since we aim to convert probabilities into penalty weights in the tropical semiring using the mapping $p \mapsto -\log p$, which is not compatible with sums. Instead we take a weighted product of powers of the component probabilities. Hence we get a model

$$P(t,s) = P_u(t,s)^{w_u} Q_o(t,s)^{w_o} Q_e(t,s)^{w_e}$$

where w_u, w_e and w_o are parameters, which need to be estimated.

If each of the models P_u, Q_e and Q_o agree on the probability p of an analysis t given a sentence s, we want P to give the same probability. This is accomplished exactly when $w_u + w_e + w_o = 1$. There does not seem to be any reason to prefer either of the models Q_e or Q_o, which makes it plausible to assume that $w_e = w_o$. Hence an implementation of the model only requires estimating two non-negative parameters: the unigram parameter w_u and the bigram parameter w_b. They should satisfy $w_u + 2w_b = 1$.

It is possible that $P(t,s)$ will not be a normalized distribution when s is kept fixed, but it can easily be normalized by scaling linearly with factor $\Sigma_t P(t,s)$. For the present implementation, it is not crucial that P is normalized.

5 Implementing the Statistical Model Using Weighted Finite-State Transducers

We describe the implementation of the POS tagger model using weighted finite-state transducers (WFSTs). We implement each of the components of the statistical model as a WFST, which are trained using corpus data.

In order to speed up computations and prevent roundoff errors, we convert probabilities p, given by the models, into penalty weights in the tropical semiring using the transformation $p \mapsto -\log p$. In the tropical semiring the product of probabilities pq translates to the sum of corresponding penalty weights $-\log p + -\log q$. The kth power of the probability p, namely p^k, translates to a scaling of its weight $-k \log p$. These observations follow from familiar algebraic rules for logarithms.

In our system, tagging of sentences is performed in three stages using four different WFSTs. The first two WFSTs, a weighted lexicon and a guesser for unknown words, implement a unigram model. They produce weighted suggestions for analyzes of individual word forms. The latter two WFSTs re-score the suggestions using bigram probabilities. The weights $-\log p$ given by the unigram model and the bigram model are scaled by multiplying with a constant in order to prefer analyzes which are strong bigrams. The scaled weights $-k \log p$ are then added to give the total scoring of the input sentence. This corresponds to multiplying the powers p^k of the corresponding probabilities.

In the first stage we use a weighted lexicon, which gives the five best analyzes for each known word form. In initial tests, the correct tagging for a known word

could be found among the five best analyzes in over 99% of tagged word forms, so we get sufficient coverage while reducing computational complexity.

For an unknown word x, we use a guesser which estimates the probability of analyzes using the probabilities for analyzes of known words. We find the set of known word forms W, whose words share the longest possible suffix with the word form x. We then determine the five best analyzes for the unknown word form x by finding the five best analyzes for words in the set W.

For each word s_i in a sentence $s = s_1 \ ... \ s_n$, we form a WFST W_i which is a disjunction of its five best analyzes $t_1 \ ... \ t_5$ according to the weights $w(s_i, t_i)$ given by the unigram model. In case there are less than five analyzes for a word, we take as many as there are. We then compute a weighted concatenation W_s of the individual WFSTs W_i. The transducer W_s is the disjunction of all POS analyzes of the sentence s, where each word receives one of its best five analyzes given by the unigram model.

To re-score the analysis suggestions given by the lexicon and the guesser, we use two WFSTs whose combined effect gives the bigram weighting for the sentence. One of the model scores bigrams starting at even positions in the sentence and the other one scores bigrams starting at odd positions. Thus we give a score for all bigrams in the sentence without having to compute a WFST equivalent to the intersection of the models which might be quite large.

Using weighted intersecting composition [Silfverberg and Lindén 2009] we simultaneously apply both bigram scoring WFSTs to the sentence WFST W_s. The POS analysis of the sentence s is the best path of the result of the composition.

The WFSTs and algorithms for parsing were implemented using the Helsinki Finite-State Technology (HFST) interface [Linden et al. 2009].

We now describe the lexicon, guesser and the bigram WFSTs in more detail.

5.1 The Weighted Lexicon

Using a tagged corpus, we form a weighted lexicon L which re-writes word forms to their lemmas and analyzes. POS analyzes for a word form s_i are weighted according to their frequencies, which are transformed into tropical weights.

In order to estimate the weights for words which were not seen in the training corpus, we construct a guesser. For an unknown word, the guesser will try to construct a series of analyzes relying on information about the analyzes of known similar words.

Figure 1 shows an example guesser, which can be constructed from a reversed weighted lexicon. Guessing begins at the end of the word. We allow guessing at a particular analysis for a word only if the word has a suffix agreeing with the analysis. See [Linden 2009a] for more information on guessers.

5.2 The Bigram Models

To re-score analyzes given by the unigram model, we use two WFSTs whose combination serves as a bigram model. The first one, B_e, scores each known word form/analysis bigram s_{2k}, s_{2k+1} and t_1, t_2 in the sentence starting at an

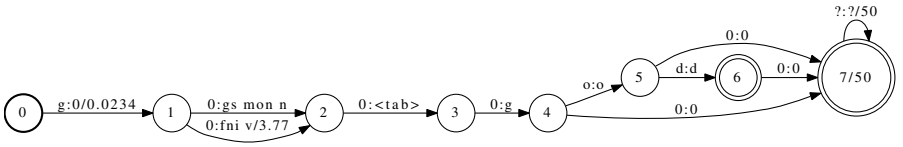

Fig. 1. Guesser constructed from a weighted lexicon. Guessing starts at the end of a word. Skipping letters gives a high penalty and analyzes, where equally many letters are skipped, are weighted according to the frequency of the analyzes.

even position $2k$ according to the maximum likelihood estimate of the tag bigram t_1t_2 w.r.t. the word form bigram $s_{2k}s_{2k+1}$. The WFST B_o is similar to B_e except it weights bigrams starting at odd positions $s_{2k-1}s_{2k}$.

Given a word form pair s_1, s_2, we compute the probability $P(t_1, s_1, t_2, s_2)$ for each POS analysis pair t_1, t_2. These sum to 1 when w_1 and w_2 remain fixed. Then we form a transducer B, whose paths transform word form pairs s_1s_2 into analysis pairs t_1t_2 with weight $-\log P(t_1, s_1, t_2, s_2)$. Lastly we disjunct B with a default bigram, which transforms arbitrary word form sequences to arbitrary analyzes with a penalty weight, which is greater than the penalty received by all other transformations.

In addition to the model B, we also compute a general word model W, which transforms an arbitrary sequence of symbols into an arbitrary lemma and an analysis. The word model W is used to skip words at the beginning and end of sentences.

From the transducers above, we form the models B_e and B_o using weighted finite state operations

$$B_e = WB^*W^{\{0,1\}} \text{ and } B_o = B^*W^{\{0,1\}}.$$

Here $W^{\{0,1\}}$ signifies an optional instance of W.

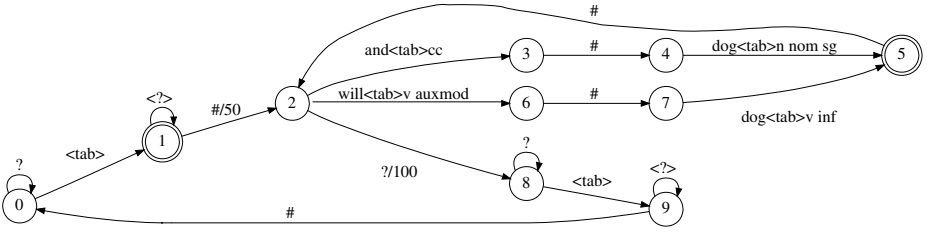

Fig. 2. A small example of an even bigram model B_e. ? signifies an arbitrary symbol and <?> signifies an arbitrary POS analysis symbol.

5.3 Parsing Using Weighted Intersecting Composition

In our system, parsing a sentence S is in principle equivalent to finding the best path of the transducer

$$(S \circ L) \circ (B_e \cap B_o).$$

Since the intersection of B_o and B_e could become prohibitively large, we instead use intersecting composition [Silfverberg and Lindén 2009] to simulate the intersection of B_e and B_o during composition with the unigram tagged sentence $S \circ L$.

Intersecting composition is an operation first used in compiling two-level grammars [Karttunen 1994]. We use a weighted version of the operation.

After the intersecting composition, we extract the best path from the resulting transducer. This is the tagged sentence.

6 Data

In this section we describe the data used for testing and training the POS tagger.

For testing and training, we used the Europarl parallel corpus [Koehn 2005].

The Europarl parallel corpus is a collection of proceedings of the European Parliament in eleven European languages. The corpus has markup to identify speaker and some html-markup, which we removed to produce a file in raw text format. We used the Finnish, English and Swedish corpora. Since the training and testing materials are the same for all three languages, the results we obtain for the different languages are comparable.

We parsed the Europarl corpora using Connexor functional dependency parsers fi-fdg for Finnish, sv-fdg for Swedish and en-fdg for English [Järvinen et al. 2004]. From the parses of the corpora we extracted word forms, lemmas and POS tags. For training and testing, we preserved the original tokenization of the fdg-parsers and removed *prop* tags marking proper nouns, *abbr* tags marking abbreviations and *heur* tags marking guesses made by the fdg-parser. The tag sequence counts in table 1 represent the number of tag sequences after *abbr*, *prop* and *heur* tags were removed.

Table 1. Some figures describing the test and training material for the POS tagger

Language	Syntactic tokens	Sentences	POS tag sequences
English	43 million	1 million	122
Finnish	25 million	1 million	2194
Swedish	38 million	1 million	243

Table 1 describes the data used in training and testing the POS tagger. We see that the fi-fdg parser for Finnish emitted more than ten times as many tag sequences as sv-fdg for Swedish or en-fdg for English. The en-fdg parse emitted clearly fewest tag sequences.

7 Training the Model

We now describe training the model, which consists of two phases. In the first phase we build the weighted lexicon and guesser and the bigram models. In the second phase we estimate experimentally coefficients w_u and w_b, which maximize the accuracy of the interpolated model

$$P(t,s) = P_u(t,s)^{w_u} Q_o(t,s)^{w_b} Q_e(t,s)^{w_b}$$

Using a small material covering 1000 syntactic tokens, we estimated $w_u = 0.1$ and $w_b = 0.45$. This shows that it is beneficial to weight the bigram model heavily, which seems natural, since bigrams provide more information than unigrams.

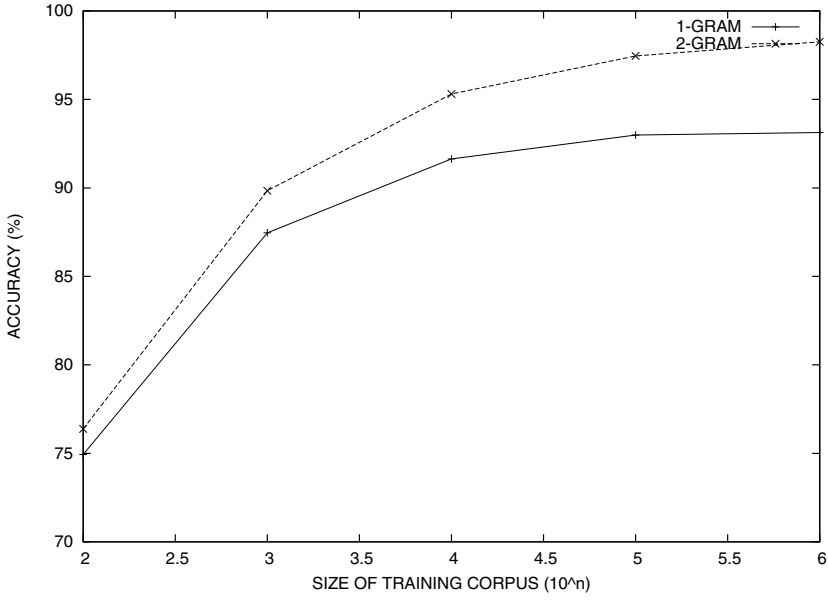

Fig. 3. The accuracy for the English POS tagger as a function of the size of training data. We used between 10^2 and 10^6 sentences for training. The lower curve displays the accuracy using only the unigram model, whilst the upper curve displays the accuracy of the combined unigram and bigram model.

Figure 3 shows learning curves for the English language POS tagger using 10^2 to 10^6 sentences for training. The lower curve displays accuracies for the unigram model and the upper curve shows the accuracy for the combined unigram and bigram model. For the unigram model, we can see that little improvement is obtained by increasing the training data from 10^4 sentences. In contrast, there is significant improvement ($\approx 0.82\%$) for the bigram model even when we move from 10^5 to 10^6 sentences.

8 Evaluation

We describe the methods we used to evaluate the POS tagger and the results we got.

We used ten-fold cross-validation to evaluate the POS tagger, that is we split the training material in ten equally sized parts and used nine parts for training the model and the remaining part for testing. Varying the tenth used for testing we trained ten POS taggers for each language.

For each of the languages we trained two sets of taggers. One set used only unigram probabilities for assigning POS tags. The other used both unigram and bigram probabilities. We may consider the unigram taggers as a baseline.

For each tree languages, we computed the average and standard deviation of the accuracy of the unigram and bigram taggers. In addition we computed the Wilcoxon matched-pairs signed-ranks test for the bigram and unigram accuracies in all three languages. The test does not assume that the data is normally distributed (unlike the paired t-test). The results of our tests can be seen in table 2.

Table 2. Average accuracies and standard deviations for POS taggers in Finnish, English and Swedish. The sixth column shows the improvement, which results for adding the bigram model. In the seventh column, we show the results of the Wilcoxon matched-pairs signed-ranks test between unigram and bigram accuracies.

Language	Unigram Acc.	σ	Bigram Acc.	σ	Diff.	Conf.
English	93.10%	0.09	98.29%	0.01	5.19%	$\geq 99.8\%$
Finnish	94.38%	0.07	96.63%	0.03	2.25%	$\geq 99.8\%$
Swedish	94.12%	0.20	97.31%	0.11	3.19%	$\geq 99.6\%$

9 Discussion and Future Work

It is interesting to see that a bigram tagger can perform equally well or better than trigram taggers at least on certain text genres. The mean accuracy 98.29%, we obtained for tagging the English Europarl corpus is exceptionally high (for example [Shen et al. 2007], report a 97.33% accuracy on tagging the Penn Tree Bank). The improvement of 5.19 percentage points from the unigram model to the combined unigram and bigram model is also impressive. There is also a clear improvement for Finnish and Swedish, when the bigram model is used in tagging and accuracy for these languages is also high. We had problems finding accuracies figures for statistical taggers of Finnish, but for Swedish [Megyesi 2001] reports accuracies between 94% and 96%, which means that we get state-of-the-art accuracy for Swedish.

Of course the Europarl corpus is probably more homogeneous than the Penn Tree Bank or the Brown Corpus, both of which include texts from a variety of genres. Furthermore tagging is easier because the en-fdg parser only emits 122 different POS analyzes. Still, Europarl texts represent an important genre,

because the EU is constantly producing written materials, which need to be translated into all official languages of the union.

The accuracy for Finnish shows less improvement than English and Swedish. We believe this is a result of the fact that Finnish words carry a lot of information but the bonds between words in sentences may be quite weak. This conclusion is supported by the fact that unigram accuracy for Finnish is best of all three languages.

We do not believe, that using trigram statistics would bring much improvement for Finnish. Instead we would like to write a set of linguistic rules which would cover most typically occurring tagging errors. Especially we would like to try out constraints, which would mark certain analyzes as illegal in some contexts. Such negative information is hard to learn using statistical methods. Still, it may be very useful, so it could be provided by hand-crafted rules.

Clearly our figures for accuracy need to be considered in relation to the tagging accuracy of the fdg parsers. We did not succeed in finding a study on the POS tagging accuracy of the fdg parsers. Instead we examined the POS tagging for one word per twenty thousand in the first tenth of the Europarl corpora for Finnish, English and Swedish. This amounted to 131 examined words for Finnish, 219 examined words for English and 191 examined words for Swedish. According to these tests, the POS tagging accuracy of the fdg parsers for Finnish is 95.4%, for English it is 97.3% and for Swedish it is 97.5%.

10 Conclusions

We introduced a model for a statistical POS tagger using bigram statistics with lemmas included. We showed how the tagger can be implemented using WFSTs. We also demonstrated a new way to factor a first order model into a model tagging bigrams at even positions in the sentence and another model tagging bigrams at odd positions.

In order to test our model, we implemented POS taggers for Finnish, English and Swedish, training them and evaluating them using Europarl corpora in the respective languages and Connexor fdg parsers.

We obtained a clear, statistically significant, improvement for all three languages when compared to the baseline unigram tagger. At least for English and Swedish, we obtain state-of-the-art accuracy.

Acknowledgements. We thank the anonymous referees. We also want thank our colleagues in the Hfst team. The first author is funded by Langnet Graduate School for Language Studies.

References

[Brants 2000] Brants, T.: TnT – A Statistical Part-of-Speech Tagger. In: ANLP - 2000 (2000)

[Church 1999] Church, K.: A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In: Proceedings of the Second Conference on Applied Natural Language Processing (1988)

[Collins 2002] Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In: EMNLP (2002)

[Cutting 1992] Cutting, D., Kupiec, J., Pedersen, J., Sibun, P.: A Practical Part-of-Speech Tagger. In: Proceedings of the Third Conference on Applied Natural Language Processing (1992)

[Järvinen et al. 2004] Järvinen, T., Laari, M., Lahtinen, T., Paajanen, S., Paljakka, P., Soininen, M., Tapanainen, P.: Robust Language Analysis Components for Practical Applications. In: Gambäck, B., Jokinen, K. (eds.) Robust and Adaptive Information Processing for Mobile Speech Interfaces (2004)

[Karttunen 1994] Karttunen, L.: Constructing Lexical Transducers. In: COLING 1994, pp. 406–411 (1994)

[Koehn 2005] Koehn, P.: Europarl: A Parallel Corpus for Statistical Machine Translation. In: Machine Translation Summit X, Phuket, Thailand, pp. 79–86 (2005)

[Koskenniemi 1990] Koskenniemi, K.: Finite-state parsing and disambiguation. In: 13th COLING (1990)

[Lafferty et al. 2001] Lafferty, J., MacCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML 2001 (2001)

[Linden et al. 2009] Lindén, K., Silfverberg, M., Pirinen, T.: Hfst Tools for Morphology – an Efficient Open-Source Package for Construction of Morphological Analyzers. In: Mahlow, C., Piotrowski, M. (eds.) SFCM 2009. LNCS, vol. 41, pp. 28–47. Springer, Heidelberg (2009)

[Linden 2009a] Lindén, K.: Entry Generation by Analogy Encoding New Words for Morphological Lexicons. NEJLT, vol. 1 (2009)

[Linden 2009b] Lindén, K.: Guessers for Finite-State Transducer Lexicons. In: Gelbukh, A. (ed.) CICLing 2009. LNCS, vol. 5449, pp. 158–169. Springer, Heidelberg (2009)

[Megyesi 2001] Megyesi, B.: Comparing data-driven learning algorithms for POS tagging of Swedish. In: EMNLP 2001 (2001)

[Mohri and Riley 2002] Mohri, M., Riley, M.: An Efficient Algorithm for the n-Best-Strings Problem. In: ICSLP 2002 (2002)

[Mikheev 1997] Mikheev, A.: Automatic Rule Induction for Unknown-Word Guessing. In: CL, vol. 23 (1997)

[Shen et al. 2007] Shen, L., Satta, G., Joshi, A.: Guided Learning for Bidirectional Sequence Classification. In: ACL 2007 (2007)

[Silfverberg and Lindén 2009] Silfverberg, M., Lindén, K.: Conflict Resolution Using Weighted Rules in HFST-TwolC. In: NODALIDA 2009 (2009)

[Thede and Harper 1999] Thede, S., Harper, M.: A Second-Order Hidden Markov Model for Part-of-Speech Tagging. In: 37th ACL (1999)