Different Approaches to Unknown Words in a Hidden Markov Model Part-of-Speech Tagger

Martin Haulrich

May 17, 2009

Abstract

We present an implementation of a first-order Hidden Markov Model part-of-speech tagger and suggest different approaches to dealing with unknown words. We show that estimation of unknown word probabilities using a log-linear model provides better accuracy than simpler models.

1 Introduction

Part-of-speech (POS) tagging is the task of assigning part of speech tags to words in a text. This is a well studied problem, but is still of interest because many applications performing other NLP-tasks, e.g. syntactic parsing, assumes input that are tagged with POS. For this reason it is crucial to have good POS-taggers, otherwise one risks that the errors in the tagging propagates to the following applications.

Here we will describe the implementation of a POS-tagger based on Hidden Markov Models. We will only consider supervised training of the models, and the focus will be on how to deal with words that are not encountered during the training of the parser. The tagger will be tested on a small corpus of Swedish test extracted from the Stockholm Umeå Corpus[1].

2 Hidden Markov Models

One way of solving the problem of assigning POS-tags to the words in a text is using a *sequence classifier*. **Hidden Markov Models** are sequence classifiers and in the following we will describe **HMMs** in more detail. The description and definition of **HMMs** follows Jurafsky and Martin (2008, chap. 6).

A **HMM** has the following components:

[1]`http://www.ling.su.se/staff/sofia/suc/suc.html`

$Q = q1q2 \dots qN$	a set of **states**
$A = a_{01}a_{02} \dots a_{n1} \dots a_{nm}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_N$	a set of **observations**, each drawn from a vocabulary $V = v_1, v_2, \dots, v_V$.
$B = b_i(o_t)$	A set of **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
$\pi = \pi_1, \pi_2, \dots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$
$QA = q_x, q_y, \dots$	a set $AQ \in Q$ of legal **accepting states**

Training a **HMM** consists of finding the transition probabilities A and the symbol emission probabilities B. We will return to how this is done for POS-tagging in section 3.

When A and B are given determining the best (hidden) sequence of states given a sequence of observations is called **decoding**. The decoding task can be solved using the **Viterbi** algorithm (Jurafsky and Martin, 2008, chap. 6).

3 Tagging with HMMs

In this section we will describe how to use HMMs for part-of-speech tagging.

In POS-tagging the known observations are the words in the text and the hidden states are the POS-tags corresponding to these words. So if we have:

P	set of allowed **part-of-speech** tags
V	possible **words-forms** in language

and we let $QA = P$, we can create HMMs that can be used for POS-tagging.

Of course we cannot let V be all the possible word forms in the language because this set cannot be determined. Instead the word forms from a dictionary or simply the word forms present the text used to train the HMM-tagger can be used. Here we will do the latter.

3.1 Transition probabilities

When the states are POS-tags as described above a transition probability a_{km} from a state q_k to a state q_m is the probability of the word with POS-tag q_m following a word with POS-tag q_k. Using a tagged training corpus we can find a maximum-likelihood estimate of this probability simply by counting occurrences.

The maximum-likelihood estimate is then given by:

$$a_{km} = \frac{|q_k q_m|}{|q_k|}$$

Using the maximum-likelihood estimate can cause problems because unseen transitions are assigned zero-probability. If we also allow zero-probabilities in the symbol emission probabilities we can encounter sentences that have no tag-sequences with non-zero probability. To avoid this we use Laplace (add-one) smoothing when estimating the transition probabilities. This implies that:

$$a_{km} = \frac{|q_k q_m| + 1}{|q_k| + |QA|}$$

The accepting states QA are the possible POS-tags and therefore $|QA|$ is the number of possible POS-tags.

3.2 Symbol emission probabilities

The symbols are taken from a vocabulary of words which means that a symbol emission probability is the probability of a given word being 'emitted' given a particular POS-tag. Therefore a maximum-likelihood estimate for $b_i(o_t)$ is given by:

$$b_i(o_t) = \frac{|o, q_i|}{|q_i|}$$

I.e. the probability for a word o (at time t) being emitted given a state q_i is calculated being dividing the number of times the word occurs with this POS-tag q_i in the training set with the number of times the POS-tag occurs with any word.

3.2.1 Unknown words

If o is not in the training text then $b_i(o_t) = 0$. Despite of course being a very bad estimate of the true probability, zero-probabilities like this will also result in that all possible state-sequences for an observation sequence containing an unknown word will have probability 0 and therefore we cannot choose between them.

One way of solving this problem is using Laplace smoothing. This gives us:

$$b_i(o_t) = \frac{|o_t, q_i| + 1}{|q_i| + |V|}$$

where $|V|$ is the number of symbols and therefore the number of different words forms encountered in the training text.

We will also try some other approaches to estimating the emission probabilities for unknown words. In all of these approaches the emission probabilities for the known words are estimated using maximum-likelihood estimation and the emission probabilities for the unknown words are estimated separately. In the following we will describe the different approaches.

Most probable POS-tag

Looking at the whole training corpus we can find the most frequent POS-tag and assume that unknown words always have this POS-tag. This implies that for all unknown words $b_i(o_t) = 1$ for that single tag and $b_i(o_t) = 0$ for all other tags.

Overall POS distribution

The above-mentioned method is obviously flawed in that it assumes that all unknown words have the same POS-tag. A way of estimating the probabilities that does not suffer from this flaw is to use the overall distribution of the POS-tags from the known words. I.e. the probability of an unknown word having tag q_j is the same as the probability of a known word having the tag q_j.

Open word-classes POS distribution

Some word classes are considered *closed*, which means that new words cannot (easily) be added to these. The opposite is called *open* word-classes. Prepositions is an example of a closed word-class and substantives an example of an open word-class. It is pretty safe to assume that the probability that an unknown word is a member of an open word class is much higher that is being a member of a closed word class. Therefore we can expect a better estimate of the POS-tag of unknown words if we instead of looking at the distribution of POS-tags on known words only looks at the distribution of POS-tags on words from the open word-classes.

It should be noted that although this will probably lead to a better estimate than looking at all word-classes this will most likely remove the possibility of reaching a perfect assignment of POS-tags. This is because if just one word from a closed word-class is not present in the training material, this word can never be tagged correctly.

Hapax legomena

Jurafsky and Martin (2008, chap. 6) mentions that different people have suggested that the distribution of POS-tags over unknown words is similar to the distribution over words that occur only once in the training set. These words are known as **hapax legonema**. So we also try using this distribution for the unknown words.

Hapax legomena in open word-classes

Here we combine the two approaches described above and use the distribution of hapax legonema in open word-classes to estimate the probabilities for the unknown words.

Maximum-entropy classification
The last approach we have tested is significantly more complex than the ones described above. Here we train a maximum-entropy classifier to predict the emission probabilities for the unknown words.

Maximum-entropy models has been used for POS-tagging by different people with great success (Ratnaparkhi, 1996; Toutanova et al., 2003). Here we only use the maximum-entropy models for estimation the emission probabilities for unknown words.

Following Ratnaparkhi (1996) and Toutanova et al. (2003)[2] we use the following features[3].

word contains a number
word contains an upper-case letter
word contains a hyphen
word is all upper-case
word contains a particular prefix (up to length 4)
word contains a particular suffix (up to length 4)
word is upper-case and has a digit and and a dash

Toutanova et al. (2003) also has the following feature.

word is upper-case and followed within 3 words by Co, Inc., etc

This feature is reported to be very useful but because it is very language-specific we leave it out.

A training data-set in extracted from the training text, and a model is trained with this data. This model in then used when predicting the POS-tags for the unseen words during the tagging.

We use the Megam-classifier (Daumé III, 2004) as our maximum-entropy classifier.

4 Experiments and results

In our experiments with the implemented HMM POS-tagger we have used the two files `WSDtraining.txt` and `WSDtest.txt` downloaded from `http://w3.msi.vxu.se/~nivre/teaching/statnlp/projT.html`[4].

We have used the data in two different ways. In one experiment we have used `WSDtraining.txt` for training and `WSDtest.txt` for testing. In the other we have merged the two files and used all of the data in a ten-fold cross validation

[2]Although we do not only use features that occur than 10 times as Ratnaparkhi (1996) and we do only use suffixes and prefixes up to length 4. Toutanova et al. (2003) uses up to length 10.

[3]The description of the features are taken directly from Jurafsky and Martin (2008, chap. 6) who summarizes the features used by Ratnaparkhi (1996) and Toutanova et al. (2003)

[4]Downloaded on March 14th, 2009

experiment. This means that the data has been split into ten equally big parts. Then ten experiments have been performed. Each of them using one of the parts as test data and the other as training data. The reported result for such an experiment is the average of the then results achieved.

The data-files has the following format:

```
("<Per>"
(PM NOM "Per"))
("<och>"
(KN "och"))
("<Nina>"
(PM NOM "Nina"))
("<om>"
(PP "om"))
("<uppfostran>"
(NN UTR SIN IND NOM "uppfostran"))
("<:>"
(DL MAD ":"))
```

We see that some words have several tags, for instance *Nina* that has both *PM* and *NOM*. Of these only the first has been used.

We have also created a baseline parser that works by for each known word assigning to the word the most frequent POS-tag for this word in the training data. Unknown words are assigned the most frequent POS-tag altogether.

We have also compared the implemented tagger with the TnT-tagger (Brants, 2000), which is a HMM-tagger providing state-of-the art performance.

The taggers are evaluated using *accuracy*, i.e. the number of correct POS-tags assigned relative to the total number of words in the text, so:

$$\text{accuracy} = \frac{|\text{correctly tagged words}|}{|\text{words in text}|}$$

The results for all taggers are shown in table 1. In the cross-validation experiments we show results for sentences containing only known words, sentences that contain unknown words and for all all sentences.

As expected the TnT-tagger performs better than the tagger implemented here. The maximum-entropy approach to estimation the probabilities for the unknown words outperforms the other approaches quite significantly. For the sentences with unknown words in them is scores 95.09% whereas the second best scores 93.85%.

It is worth noticing that the baseline tagger is better than the tagger that uses Laplace-smoothing for the emission probabilities. This indicates that this smoothing method moves too much probability mass to unseen events.

	All Known	Unknown	All	Orig. split
Sentences	267.1	424.5	691.6	1,341
Baseline	95.06%	90.04%	91.43%	89.78%
LaPlace smoothing	92.66%	86.62%	88.39%	88.21%
Most prob. POS-tag	97.07%	91.80%	93.52%	92.33%
Overal POS dist	97.07%	91.58%	93.46%	92.23%
Open word-class POS dist	97.07%	92.37%	93.85%	92.71%
Hapax legomena	97.07%	92.00%	93.78%	92.44%
Hapax open word-classes	97.07%	92.57%	94.08%	92.68%
Maximum entropy model	97.07%	**95.09%**	**95.75%**	**95.39%**
TnT bigram	96.86%	96.26%	96.43%	96.23%
TnT trigram	97.13%	96.57%	96.72%	96.37%

Table 1: Accuracies for the different taggers on the different approaches to unknown words. The three first columns are results from the 10-fold cross-validation experiment. The fourth column uses the original split of the data.

5 Conclusion

We have implemented a HMM POS-tagger and looked as different strategies for dealing with unknown words. The most advanced approach, estimating the probabilities using with a log-linear model, turned out to be the best.

The accuracy of the tagger is near that of the TnT-tagger. The tagger has only been tested on a small Swedish corpus, with a very small tag set, and the latter may be the reason that the tagger does so well despite its simplicity.

5.1 Future work

The obvious extension would be using a higher-order HMM-model in the tagger. Either just for the transitions but possibly also for the emission-probabilities. The latter has been shown to improve parsing accuracy by Thede and Harper (1999), which calls this a *full second-order* HMM model.

Another possibility is to experiment with the features used in the maximum entropy model to see if accuracy on unknown words can be improved this way.

References

Brants, Thorsten. 2000. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231, Morristown, NJ, USA. Association for Computational Linguistics.

Daumé III, Hal. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August.

Jurafsky, Daniel and James H. Martin. 2008. *Speech and Language Processing, An Introduction to Natural Language Processing,Computational Linguistics, and Speech Recognition.* Prentice-Hall, 2nd edition.

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

Thede, Scott M. and Mary P. Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 175–182, Morristown, NJ, USA. Association for Computational Linguistics.

Toutanova, Kristina, 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, Morristown, NJ, USA. Association for Computational Linguistics.