HunPos – an open source trigram tagger

Péter Halácsy
Budapest U. of Technology
MOKK Media Research
H-1111 Budapest, Stoczek u 2
peter@halacsy.com

András Kornai
MetaCarta Inc.
350 Massachusetts Ave.
Cambridge MA 02139
andras@kornai.com

Csaba Oravecz
Hungarian Academy of Sciences
Institute of Linguistics
H-1068 Budapest, Benczur u. 33.
oravecz@nytud.hu

Abstract

In the world of non-proprietary NLP software the standard, and perhaps the best, HMM-based POS tagger is TnT (Brants, 2000). We argue here that some of the criticism aimed at HMM performance on languages with rich morphology should more properly be directed at TnT's peculiar license, free but not open source, since it is those details of the implementation which are hidden from the user that hold the key for improved POS tagging across a wider variety of languages. We present HunPos[1], a free and open source (LGPL-licensed) alternative, which can be tuned by the user to fully utilize the potential of HMM architectures, offering performance comparable to more complex models, but preserving the ease and speed of the training and tagging process.

0 Introduction

Even without a formal survey it is clear that TnT (Brants, 2000) is used widely in research labs throughout the world: Google Scholar shows over 400 citations. For research purposes TnT is freely available, but only in executable form (closed source). Its greatest advantage is its speed, important both for a fast tuning cycle and when dealing with large corpora, especially when the POS tagger is but one component in a larger information retrieval, information extraction, or question answering system. Though taggers based on dependency networks (Toutanova et al., 2003), SVM (Giménez and Màrquez, 2003), MaxEnt (Ratnaparkhi, 1996), CRF (Smith et al., 2005), and other methods may reach slightly better results, their train/test cycle is orders of magnitude longer.

A ubiquitous problem in HMM tagging originates from the standard way of calculating lexical probabilities by means of a lexicon generated during training. In highly inflecting languages considerably more unseen words will be present in the test data than in more isolating languages, which largely accounts for the drop in the performance of n-gram taggers when moving away from English. To mitigate the effect one needs a morphological dictionary (Hajič et al., 2001) or a morphological analyzer (Hakkani-Tür et al., 2000), but if the implementation source is closed there is no handy way to incorporate morphological knowledge in the tagger.

The paper is structured as follows. In Section 1 we present our own system, HunPos, while in Section 2 we describe some of the implementation details of TnT that we believe influence the performance of a HMM based tagging system. We evaluate the system and compare it to TnT on a variety of tasks in Section 3. We don't necessarily consider HunPos to be significantly better than TnT, but we argue that we could reach better results, *and so could others coming after us*, because the system is open to explore all kinds of fine-tuning strategies. Some concluding remarks close the paper in Section 4.

[1] http://mokk.bme.hu/resources/hunpos/

Proceedings of the ACL 2007 Demo and Poster Sessions, pages 209–212,
Prague, June 2007. ©2007 Association for Computational Linguistics

1 Main features of HunPos

HunPos has been implemented in OCaml, a high-level language which supports a succinct, well-maintainable coding style. OCaml has a high-performance native-code compiler (Doligez et al., 2004) that can produce a C library with the speed of a C/C++ implementation.

On the whole HunPos is a straightforward trigram system estimating the probabilities

$$\underset{t_1 \ldots t_T}{\operatorname{argmax}} P(t_{T+1}|t_T) \prod_{i=1}^{T} P(t_i|t_{i-1}, t_{i-2}) P(w_i|t_{i-1}, t_i)$$

for a given sequence of words $w_1 \ldots w_T$ (the additional tags t_{-1}, t_0, and t_{T+1} are for sentence boundary markers). Notice that unlike traditional HMM models, we estimate emission/lexicon probabilities based on the current tag and the previous tag as well. As we shall see in the next Section, using tag bigrams to condition the emissions can lead to as much as 10% reduction in the error rate. (In fact, HunPos can handle a context window of any size, but on the limited training sets available to us increasing this parameter beyond 2 gives no further improvement.)

As for contextualized lexical probabilities, our extension is very similar to Banko and Moore (2004) who use $P(w_i|t_{i-1}, t_i, t_{i+1})$ lexical probabilities and found, on the Penn Treebank, that "incorporating more context into an HMM when estimating lexical probabilities improved accuracy from 95.87% to 96.59%". One difficulty with their approach, noted by Banko and Moore (2004), is the treatment of unseen words: their method requires a full dictionary that lists what tags are possible for each word. To be sure, for isolating languages such information is generally available from machine readable dictionaries which are often large enough to make the out of vocabulary problem negligible. But in our situation this amounts to idealized morphological analyzers (MA) that have their stem list extended so as to have no OOV on the test set.

The strong side of TnT is its suffix guessing algorithm that is triggered by unseen words. From the training set TnT builds a trie from the endings of words appearing less than n times in the corpus, and memorizes the tag distribution for each suffix.[2] A clear advantage of this approach is the probabilistic weighting of each label, however, under default settings the algorithm proposes a lot more possible tags than a morphological analyzer would. To facilitate the use of MA, HunPos has hooks to work with a morphological analyzer (lexicon), which might still leave some OOV items. As we shall see in Section 3, the key issue is that for unseen words the HMM search space may be narrowed down to the alternatives proposed by this module, which not only speeds up search but also very significantly improves precision. That is, for unseen words the MA will generate the possible labels, to which the weights are assigned by the suffix guessing algorithm.

2 Inside TnT

Here we describe, following the lead of (Jurish, 2003), some non-trivial features of TnT sometimes only hinted at in the user guide, but clearly evident from its behavior on real and experimentally adjusted corpora. For the most part, these features are clever hacks, and it is unfortunate that neither Brants (2000) nor the standard HMM textbooks mention them, especially as they often yield more significant error reduction than the move from HMM to other architectures. Naturally, these features are also available in HunPos.

2.1 Cardinals

For the following regular expressions TnT learns the tag distribution of the training corpus separately to give more reliable estimates for open class items like numbers unseen during training:

```
^[0-9]+$
^[0-9]+\.$
^[0-9.,:-]+[0-9]+$
^[0-9]+[a-zA-Z]{1,3}$
```

(The regexps are only inferred – we haven't attempted to trace the execution.) After this, at test time, if the word is not found in the lexicon (numerals are added to the lexicon like all other items) TnT checks whether the unseen word matches some of the regexps, and uses the distribution learned for this regexp to guess the tag.

[2] The parameter n cannot be externally set — it is documented as 10 but we believe it to be higher.

2.2 Upper- and lowercase

The case of individual words may carry relevant information for tagging, so it is well worth preserving the uppercase feature for items seen as such in training. For unseen words TnT builds two suffix tries: if the word begins with uppercase one trie is used, for lowercase words the other trie is applied. The undocumented trick is to try to lookup the word in sentence initial position from the training lexicon in its lowercase variant, which contributes noticeably to the better performance of the system.

3 Evaluation

English For the English evaluation we used the WSJ data from Penn Treebank II. We extracted sentences from the parse trees. We split data into training and test set in the standard way (Table 1).

Set	Sect'ns	Sent.	Tokens	Unseen
Train	0-18	38,219	912,344	0
Test	22-24	5,462	129,654	2.81%

Table 1: Data set splits used for English

As Table 2 shows HunPos achieves performance comparable to TnT for English. The increase in the emission order clearly improves this performance.

	seen	unseen	overall
TnT	96.77%	85.91%	96.46%
HunPos 1	96.76%	86.90%	96.49%
HunPos 2	96.88%	86.13%	**96.58%**

Table 2: WSJ tagging accuracy, HunPos with first and second order emission/lexicon probabilities

If we follow Banko and Moore (2004) and construct a full (no OOV) morphological lexicon from the tagged version of the test corpus, we obtain 96.95% precision where theirs was 96.59%. For words seen, precision improves by an entirely negligible 0.01%, but for unseen words it improves by 10%, from 86.13% to 98.82%. This surprising result arises from the fact that there are a plenty of unambiguous tokens (especially the proper names that are usually unseen) in the test corpus.

What this shows is not just that morphology matters (this is actually not that visible for English), but that the difference between systems can only be appreciated once the small (and scantily documented) tricks are factored out. The reason why Banko and Moore (2004) get less than HunPos is not because their system is inherently worse, but rather because it lacks the engineering hacks built into TnT and HunPos.

Hungarian We evaluated the different models by tenfold cross-validation on the Szeged Corpus (Csendes et al., 2004), with the relevant data in presented Table 3.

Set	Sent.	Tokens	Unseens	OOV
Train	63,075	1,044,914	0	N.A
Test	7,008	116,101	9.59%	5.64%

Table 3: Data set splits used for Hungarian.

Note that the proportion of unseen words, nearly 10%, is more than three times higher than in English. Most of these words were covered by the morphological analyzer (Trón et al., 2006) but still 28% of unseen words were only guessed. However, this is just 2.86% of the whole corpus, in the magnitude similar to English.

morph	lex order	seen	unseen	overall
no	1	98.34%	88.96%	97.27%
	2	98.58%	87.97%	97.40%
yes	1	98.32%	96.01%	98.03%
	2	98.56%	95.96%	**98.24%**

Table 4: Tagging accuracy for Hungarian of HunPos with and without morphological lexicon and with first and second order emission/lexicon probabilities.

On the same corpus TnT had 97.42% and Halácsy et al. (2006) reached 98.17% with a MaxEnt tagger that used the TnT output as a feature. HunPos gets as good performance *in one minute* as this MaxEnt model which took three hours to go through the train/test cycle.

4 Concluding remarks

Though there can be little doubt that the ruling system of bakeoffs actively encourages a degree of oneupmanship, our paper and our software are not offered in a competitive spirit. As we said at the out-set, we don't necessarily believe HunPos to be in any way better than TnT, and certainly the main ideas have been pioneered by DeRose (1988), Church (1988), and others long before this generation of HMM work. But to improve the results beyond what a basic HMM can achieve one needs to tune the system, and progress can only be made if the experiments are end to end replicable.

There is no doubt many other systems could be tweaked further and improve on our results – what matters is that anybody could now also tweak HunPos without any restriction to improve the state of the art. Such tweaking can bring surprising results, e.g. the conclusion, strongly supported by the results presented here, that HMM tagging is actually quite competitive with, and orders of magnitude faster than, the current generation of learning algorithms including SVM and MaxEnt. No matter how good TnT was to begin with, the closed source has hindered its progress to the point that inherently clumsier, but better tweakable algorithms could overtake HMMs, a situation that HunPos has now hopefully changed at least for languages with more complex morphologies.

Acknowledgement

We thank Thorsten Brants for TnT, and György Gyepesi for constant help and encouragement.

References

Michele Banko and Robert C. Moore. 2004. Part of speech tagging in context. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 556, Morristown, NJ, USA. Association for Computational Linguistics.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA.

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, pages 136–143, Morristown, NJ, USA. Association for Computational Linguistics.

Dóra Csendes, János Csirik, and Tibor Gyimóthy. 2004. The Szeged Corpus: A POS tagged and syntactically annotated Hungarian natural language corpus. In Karel Pala Petr Sojka, Ivan Kopecek, editor, *Text, Speech and Dialogue: 7th International Conference, TSD*, pages 41–47.

Steven J. DeRose. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39.

Damien Doligez, Jacques Garrigue, Didier Rémy, and Jérôme Vouillon, 2004. *The Objective Caml system*. Institut National de Recherche en Informatique et en Automatique.

Jesús Giménez and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of RANLP*, pages 153–163.

Jan Hajič, Pavel Krbec, Karel Oliva, Pavel Květoň, and Vladimír Petkevič. 2001. Serial combination of rules and statistics: A case study in Czech tagging. In *Proceedings of the 39th Association of Computational Linguistics Conference*, pages 260–267, Toulouse, France.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of the 18th conference on Computational linguistics*, pages 285–291, Saarbrücken, Germany.

Péter Halácsy, András Kornai, Csaba Oravecz, Viktor Trón, and Dániel Varga. 2006. Using a morphological analyzer in high precision POS tagging of Hungarian. In *Proceedings of LREC 2006*, pages 2245–2248.

Bryan Jurish. 2003. A hybrid approach to part-of-speech tagging. Technical report, Berlin-Brandenburgische Akademie der Wissenschaften.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Karel Pala Petr Sojka, Ivan Kopecek, editor, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vancouver.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259.

Viktor Trón, Péter Halácsy, Péter Rebrus, András Rung, Péter Vajda, and Eszter Simon. 2006. Morphdb.hu: Hungarian lexical database and morphological grammar. In *Proceedings of LREC 2006*, pages 1670–1673.