SVMTool: A general POS tagger generator based on Support Vector Machines

Jesús Giménez and Lluís Màrquez

TALP Research Center, LSI Department
Universitat Politècnica de Catalunya
Jordi Girona Salgado 1–3, E-08034, Barcelona
{jgimenez,lluism}@talp.upc.es

Abstract

This paper presents the SVMTool, a simple, flexible, effective and efficient part–of–speech tagger based on Support Vector Machines. The SVMTool offers a fairly good balance among these properties which make it really practical for current NLP applications. It is very easy to use and easily configurable so as to perfectly fit the needs of a number of different applications. Results are also very competitive, achieving an accuracy of 97.16% for English on the Wall Street Journal corpus. It has been also successfully applied to Spanish exhibiting a similar performance. A first release of the SVMTool Perl prototype is now freely available for public use. A most efficient C++ version is coming very soon.

1. Introduction

Most NLP applications demand at some step some part–of–speech (POS) information, usually at initial stages prior to elaborate further complex data analyses. Generally, POS-tagging is required to be as accurate as possible, and as efficient as possible. But, certainly, there is a trade-off between this two desirable properties. This is so because obtaining a higher accuracy relies on processing more and more information, digging deeper and deeper into it. However, sometimes, depending on the kind of application, a loss in efficiency may be acceptable in order to obtain more precise results. Or the other way around, a slight loss in accuracy may be tolerated if that causes the POS-tagger to run faster.

Moreover, some languages have a richer morphology than others, requiring the POS-tagger to have into acount a bigger set of feature patterns. Also the tagset size and ambiguity rate may vary from language to language. Besides, if few data are available for training, the proportion of unknown words may be huge. Sometimes, morphological analyzers could be utilized to reduce the degree of ambiguity when facing unknown words. Thus, a POS-tagger should be flexible with respect to the amount of information utilized and context shape.

Another very interesting property for POS-taggers is their portability. Multilingual information is a key ingredient in NLP tasks such as Machine Translation, Information Retrieval, Information Extraction, Question Answering and Word Sense Disambiguation. Therefore, having a tagger that works equally well for several languages is crucial for the system robustness.

Besides, quite often for some languages, but also in general, lexical resources are hard to obtain. Therefore, ideally a POS-tagger should be capable for learning with few (or even none) annotated data.

The SVMTool is intended to comply with all the requirements of modern NLP technology, by combining simplicity, flexibility, robustness, portability and efficiency with state–of–the-art accuracy. This is achieved by working in the Support Vector Machines (SVM) learning framework (Cristianini and Shawe-Taylor, 2000), and by offering NLP researchers a highly customizable POS-tagger generator.

The Perl version 1.0 of the SVMTool may be now freely downloaded at `http://www.lsi.upc.es/~nlp/SVMTool/`[1].

The properties the SVMTool is intended to exhibit are:

Simplicty The SVMTool is easy to configure and to train. The learning is controlled by means of a very simple configuration file. There are very few parameters to tune. And the tagger itself is very easy to use, accepting standard input and output pipelining. Embedded usage is also supplied by means of the SVMTool API.

Flexibility The size and shape of the feature context can be adjusted. Also, rich features can be defined, including word and POS n-grams as well as ambiguity classes and "may_be's", apart from lexicalized features for unknown words and sentence general information. The behaviour at tagging time is also very flexible, allowing different strategies.

Robustness The overfitting problem is well addressed via the tunning of the C parameter in the soft margin version of the SVM learning algorithm. Also, a sentence-level analysis may be performed in order to maximize the sentence score. And, for unknown words not to punish so severely on the system effectiveness, several strategies have been implemented and tested.

Portability The SVMTool is language independent. It has been successfully applied to English and Spanish without a priori knowledge other than a supervised corpus. Moreover, thinking of languages for which labeled data is a scarce resource, the SVMTool also may learn from unsupervised data based on the role of non-ambiguous words (Mihalcea, 2003) with the only additional help of a morpho-syntactic dictionary.

Accuracy Compared to state–of–the–art POS taggers reported up to date, it exhibits a very competitive accuracy (over 97.1% for English on the WSJ corpus).

[1] Of course, comments and feedback from all the NLP community members will be very welcome

Clearly, rich sets of features allow to model very precisely most of the information involved. Also the learning paradigm, SVM, is very suitable for working accurately and efficiently with high dimensionality feature spaces.

Efficiency Performance at tagging time depends on the feature set size and the tagging scheme selected. For the default (one-pass left-to-right greedy) tagging scheme, the current Perl prototype exhibits a tagging speed of 1,500 words/second. This has been achieved by working in the primal formulation of SVM. The use of linear kernels causes the tagger to perform more efficiently both at tagging and learning time, but forces the user to define a richer feature space. However, the learning time remains linear with respect to the number of training examples.

A more detailed description of the SVMTool approach to POS-tagging can be found in (Giménez and Màrquez, 2003).

2. The SVMT Tool

The SVMTool software package consists of three main components, namely the learner (SVMTlearn), the tagger (SVMTagger) and the evaluator (SVMTeval), which are described below.

Previous to the tagging, SVM models (weight vectors and biases) are learned from a training corpus using the SVMTlearn component. Different models are learned for the different strategies. Then, at tagging time, using the SVMTagger component, one may choose the tagging strategy that is most suitable for the purpose of the tagging. Finally, given a correctly annotated corpus, and the corresponding SVMTool predicted annotation, the SVMTeval component displays tagging results.

2.1. SVMTlearn

Given a training set of examples (either annotated or unannotated), it is responsible for the training of a set of SVM classifiers. So as to do that, it makes use of SVM–light[2], an implementation of Vapnik's SVMs in C, developed by Thorsten Joachims (Joachims, 1999).

Options. SVMTlearn behaviour is easily adjusted through a configuration file. These are the currently available options:

- *Sliding window:* The size of the sliding window for feature extraction can be adjusted. Also, the core position in which the word to disambiguate is to be located may be selected. By default, window size is 5 and the core position is 2, starting at 0.
- *Feature set:* Three different kinds of feature types can be collected from the sliding window:
  - word features: Word form n-grams. Usually unigrams, bigrams and trigrams suffice. Also, the sentence last word, which corresponds to a punctuation mark ('.', '?', '!'), is important.
  - POS features: Annotated parts–of–speech and ambiguity classes n-grams, and "may_be's". As for words, considering unigrams, bigrams and trigrams is enough. The ambiguity class for a certain word determines which POS are possible. A "may_be" states, for a certain word, that certain POS may be possible, i.e. it belongs to the word ambiguity class.
  - affix and orthographic features: including prefixes and suffixes, capitalization, hyphenization, and similar information related to a word form. They are only used to represent unknown words.

Table 1 shows the rich feature set used in experiments.

word features	$w_{-3}, w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2}, w_{+3}$
POS features	$p_{-3}, p_{-2}, p_{-1}, p_0, p_{+1}, p_{+2}, p_{+3}$
ambiguity classes	a_0, a_1, a_2, a_3
may_be's	m_0, m_1, m_2, m_3
word bigrams	$(w_{-2}, w_{-1}), (w_{-1}, w_{+1}), (w_{-1}, w_0)$ $(w_0, w_{+1}), (w_{+1}, w_{+2})$
POS bigrams	$(p_{-2}, p_{-1}), (p_{-1}, a_{+1}), (a_{+1}, a_{+2})$
word trigrams	$(w_{-2}, w_{-1}, w_0), (w_{-2}, w_{-1}, w_{+1}),$ $(w_{-1}, w_0, w_{+1}), (w_{-1}, w_{+1}, w_{+2}),$ (w_0, w_{+1}, w_{+2})
POS trigrams	$(p_{-2}, p_{-1}, a_{+0}), (p_{-2}, p_{-1}, a_{+1}),$ $(p_{-1}, a_0, a_{+1}), (p_{-1}, a_{+1}, a_{+2})$
sentence_info	punctuation ('.', '?', '!')
prefixes	$s_1, s_1 s_2, s_1 s_2 s_3, s_1 s_2 s_3 s_4$
suffixes	$s_n, s_{n\text{-}1} s_n, s_{n\text{-}2} s_{n\text{-}1} s_n, s_{n\text{-}3} s_{n\text{-}2} s_{n\text{-}1} s_n$
binary word features	intial Upper Case, all Upper Case, no initial Capital Letter(s), all Lower Case, contains a (period / number / hyphen ...)
word length	integer

Table 1: Rich feature pattern set used in experiments.

- *Feature filtering:* The feature space can be kept in a convenient size. Smaller models allow for a higher efficiency. By default, no more than 100,000 dimensions are used. Also, features appearing less than n times can be discarded, which indeed causes the system both to fight against overfitting and to exhibit a higher accuracy. By default, features appearing just once are ignored.
- *SVM model compression:* Weight vector components lower than a given threshold, in the resulting SVM models can be filtered out, thus enhancing efficiency by decreasing de model size but still preserving accuracy level. That is an interesting behaviour of SVM models being currently under study. In fact, discarding up to 70% of the weight components accuracy remains stable, and it is not until 95% of the components are discarded that accuracy falls below the current state–of–the–art (97.0% - 97.2%).
- *C parameter tunning:* In order to deal with noise and outliers in training data, the soft margin version of the SVM learning algorithm allows the misclassification of certain training examples when maximizing the margin. This balance can be automatically adjusted by optimizing the value of the C parameter of SVMs. A local maximum is found exploring accuracy on a validation set for different C values at shorter intervals.

[2]The SVM^{light} software is freely available (for scientific use) at the following URL: `http://svmlight.joachims.org`

- *Dictionary repairing:* The lexicon extracted from the training corpus can be automatically repaired either based on frequency heuristics or on a list of corrections supplied by the user. This makes the tagger robust to corpus errors.

- *Ambiguous classes:* The list of POS presenting ambiguity is, by default, automatically extracted from the corpus but, if available, this knowledge can be made explicit. This acts in favor of the system robustness.

- *Open classes:* The list of POS tags an unknown word may be labeled as is also, by default, automatically determined. As for ambiguous classes, if available, it is well appreciated for the same reason.

- *Backup lexicon:* A morphological lexicon containing words that are not present in the training corpus may be provided. It can be also provided at tagging time.

2.2. SVMTagger

Given a text corpus (one token per line) and the path to a previously learned SVM model (including the automatically generated dictionary), it performs the POS tagging of a sequence of words. The tagging goes on-line based on a sliding window which gives a view of the feature context to be considered at every decision (see Figure 1). Calculated part–of–speech tags feed directly forward next tagging decisions as context features.

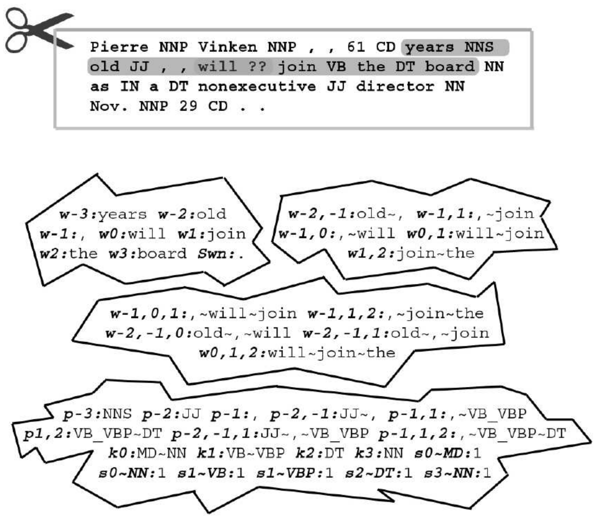

Figure 1: SVMTagger. Feature extraction

Options. SVMTagger is very flexible, and adapts very well to the needs of the user. Thus you may find the several options currently available:

- *Tagging scheme:* Two different tagging schemes may be used.
  - Greedy: Each tagging decision is made based on a reduced context. Later on, decisions are not further reconsidered, except in the case of tagging at two steps or tagging in two directions.
  - Sentence-level: By means of dynamic programming techniques (Viterbi algorithm), the global sentence sum of SVM tagging scores is the function to maximize.

- *Tagging direction:* The tagging direction can be either "left-to-right", "right-to-left", or a combination of both. The tagging order varies results yielding a significant improvement when both are combined. This makes the tagger very robust.

- *One pass / Two passes:* Another way of achieving robustness is by tagging in two passes. At the first pass only POS features related to already disambiguated words are considered. At a second pass disambiguated POS features are available for every word in the feature context, so when revisiting a word tagging errors may be alleviated.

- *SVM Model Compression:* Just as for the learning, weight vector components lower than a certain threshold, can be ignored.

- *Backup lexicon:* Again, a morphological lexicon containing new words that were not available in the training corpus may be provided.

2.3. SVMTeval

Given a SVMTool predicted tagging output and the corresponding gold-standard, SVMTeval evaluates the performance in terms of accuracy. It is a very useful component for the tunning of the system parameters, such as the C parameter, the feature patterns and filtering, the model compression et cetera.

Moreover, based on a given morphological dictionary (e.g. the automatically generated at training time) results may be presented also for different sets of words (known words vs unknown words, ambiguous words vs unambiguous words). A different view of these same results can be seen from the class of ambiguity perspective, too, i.e., words sharing the same kind of ambiguity may be considered together. Also words sharing the same degree of disambiguation complexity, determined by the size of their ambiguity classes, can be grouped.

2.4. SVMT embedded

Embedded usage of the SVMTool is also possible. It is based on the SVMTool API which offers all the capabilities of the SVMTool in an ellegant manner. The user must follow four simple steps:

1. *Loading of SVMTool models* according to the settings selected.

2. *Preparation* of input tokens for SVMTool processing.

3. *POS–tagging* of input tokens.

4. *Collection* of tagging results. Not only the winner POS is available but the losers as well, and the SVM score for all of them. This information could be really helpful in the case of hard decisions, when two or more POS were nearly tied.

3. Evaluation

The SVMTool has been already successfully applied to English and Spanish corpora, exhibiting state–of–the–art performance (97.16% and 96.89%, respectively). In both cases results clearly outperform the HMM–based TnT part–of–speech tagger (Brants, 2000), compared exactly under the same conditions. In our opinion, TnT is an example of a really practical tagger for NLP applications. It is available to anybody, simple and easy to use, reasonably accurate, and extremely efficient, allowing a training from 1 millon word corpora in just a few seconds and tagging thousands of words per second.

As to efficiency, at tagging time, a speed of 1,500 words per second is achieved by the current Perl implemented prototype on a Pentium–IV, 2GHz, 1RAM Gb. Regarding learning time, it strongly depends on the training set size, tagset, feature set, learning options, et cetera. The upper bound for the experiments reported below are about 24 CPU hours machine (Wall Street Journal corpus, 912K words for training, full set of attributes and fine adjusting of the C parameter). See (Giménez and Màrquez, 2003) for further details.

Below you may find a summary of the results obtained by the SVMTool.

3.1. Results for English

Experiments for English used the Wall Street Journal corpus (1,173 Kwords). Sections 0-18 were used for training (912 Kwords), 19-21 for validation (131 Kwords), and 22-24 for test (129 Kwords), respectively. 2.81% of the words in the test set are unknown to the training set. Best other results so far reported on this same test set are (Collins, 2002) (97.11%) and (Toutanova et al., 2003) (97.24%). See results in Table 2.

	known	amb.	unk.	all.
TnT	96.76%	92.16%	85.86%	96.46%
SVMTool	97.39%	93.91%	89.01%	97.16%

Table 2: Accuracy results of the SVMTool (on a one-pass, left-to-right and right-to-left combined, greedy tagging scheme) compared to TnT for English on the WSJ corpus test set. 'known' and 'unk.' refer to the subsets of known and unknown words, respectively, 'amb' to the set of ambiguous known words and 'all' to the overall accuracy.

3.2. Results for Spanish

Experiments for Spanish used the LEXESP corpus (106 Kwords). It was randomly divided into training set (86 Kwords) and test set (20 Kwords). 12.21% of the words in the test set are unknown to the training set. See results in Table 3.

Using additional morpho-syntactic information provided by a morphological analyzer (X.Carreras et al., 2004) in the form of a backup lexicon both tools improve very considerably their performance. Sure it is due to the fact that now there are no unknown words. But notice these words have not been seen among the training data. See results in Table 4.

	known	amb.	unk.	all.
TnT	97.73%	93.70%	87.66%	96.50%
SVMTool	98.08%	95.04%	88.28%	96.89%

Table 3: Accuracy results of the SVMTool (on a one-pass, left-to-right and right-to-left combined, greedy tagging scheme) compared to TnT for Spanish on the LEXESP corpus, 'known' and 'unk.' refer to the subsets of known and unknown words, respectively, 'amb' to the set of ambiguous known words and 'all' to the overall accuracy.

	amb.	all.
TnT	94.05%	98.41%
SVMTool	95.43%	98.86%

Table 4: Accuracy results of the SVMTool (on a one-pass right-to-left greedy tagging scheme) compared to TnT for Spanish on the LEXESP corpus with the aid of a backup morphological lexicon, 'amb' refers to the set of ambiguous known words and and 'all' to the overall accuracy.

4. Ongoing steps

A most efficient C++ version is currently being implemented. It will offer all the current capabilities, including embedded use. It is planned to be available for public release by Summer 2004. The SVMTagger component will be the first to be released. The other two components, SVMTlearn and SVMTeval will be coming soon after.

Finally, new strategies to further increase the system effectiveness while guaranteeing robustness and efficiency, are being studied. These involves better schemes for learning unknown words as well as a new sentence-level approach based on maximizing a product of probabilities instead of a sum of SVM scores.

5. References

Brants, T., 2000. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth ANLP*.

Collins, M., 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 7th EMNLP Conference*.

Cristianini, N. and J. Shawe-Taylor, 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

Giménez, J. and L. Màrquez, 2003. Fast and Accurate Part-of-Speech Tagging: The SVM Approach Revisited. In *Proceedings of the Fourth RANLP*.

Joachims, T., 1999. *Making large-Scale SVM Learning Practical*. MIT-Press.

Mihalcea, Rada, 2003. The Role of Non-Ambiguous Words in Natural Language Disambiguation. In *Proceedings of the Fourth RANLP*.

Toutanova, K., D. Klein, and C. D. Manning, 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL'03*.

X.Carreras, I. Chao, L. Padró, and M. Padró, 2004. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th LREC Conference*.