Unigram Backoff *vs.* TnT
Evaluating Part of Speech Taggers

Introduction to Computational Linguistics

Igor Boehm

Franckstrasse 7c
4020 Linz Austria
igor@bytelabs.org

Abstract

Automated statistical part-of-speech (*POS*) tagging has been a very active research area for many years and is the foundation of natural language processing systems. This paper introduces and analyzes the performance of two part-of-speech taggers, namely the NLTK unigram backoff tagger and the TnT tagger, a trigram tagger. Experimental results show that the TnT tagger outperforms the NLTK unigram backoff tagger. However, the tagging accuracy of both taggers decreases almost by the same number when the taggers are tested on a corpus which differs from the one they were trained on.

1. Introduction

The aim of this paper is to compare and evaluate two *POS* taggers, namely a simple NLTK (*Natural Language Toolkit*) backoff tagger and a TnT (*Trigrams'n'Tags*) tagger. The NLTK Backoff Tagger is a simple unigram tagger combined with a simple default tagger, whereas the TnT tagger is based on trigram statistics coupled with sophisticated smoothing approaches.

There is a variety of related works in the area of analysis, evaluation and refinement of already known natural language processing methods in order to improve part of speech taggers. For example [1] analyses and evaluates several part of speech taggers in order to be able to combine them in a very efficient way to perform POS tagging of a multilingual parallel corpus.

[3] deals with the problem of analysing and evaluating the performance of various POS taggers and combination systems on input text which is not error-free.

These examples show that in many areas of computational linguistics which are concerned with the process of automated tagging of various input sources, the task of evaluating different tagging strategies is inherent.

2. Approach

In our approach we distinguish between a simple 1-gram tagger, namely our unigram tagger, and a N-gram tagger, represented by the TnT tagger. Our tagging approaches heavily rely on statistical methods where the underlying statistical model for the N-gram tagger is a Markov model.

Markov models represent a class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past [5]. Thus the probability of a N-gram tagger to assign a word a specific part of speech token depends solely on the previous $N-1$ words.

The unigram tagger is based on a much simpler statistical model where it assigns a tag for each token which is most likely. In order to calculate probabilities for the most likely tag for a token, the relative frequencies of tags assigned to tokens from gold standard data, which is tagged by humans, is taken into account.

2.1. Unigram Backoff Tagger

As stated in the previous section, the unigram tagger class used in this experiment implements a simple statistical tagging method based on relative frequencies of tag to token assignments:

$$Unigrams: \hat{P}(t) = \frac{f(t)}{N}$$

In this equation $f(t)$ represents the frequency of tag t and N represents the total number of tokens in the corpus. The only problem with the unigram tagger occurs if it encounters a token which has not been seen yet. In that case it assigns the default tag *None* to the token. Thus a regular expression tagger is used as a backoff tagger, assigning the tag **CD** to cardinal numbers, and **NN** to everything else. So after the tagging process every token has a valid and meaningful tag.

2.2. TnT Tagger

The TnT Tagger, a N-Gram tagger, is statistically more sophisticated than the unigram tagger. As mentioned earlier, the underlying architecture of the TnT Tagger is based on second order Markov models, thus looking two words into the past. The states of the model represent tags, outputs represent the words [4], and transition probabilities depend on the states which consist of pairs of tags in this case. The TnT tagger is a trigram tagger where the probability of a tag depends on the previous two tags:

$$Trigrams : \hat{P}(t_3|t_1, t_2) = \frac{f(t_1, t_2, t_3)}{f(t_1, t_2)}$$

One major deficiency of standard N-gram models is the sparse data problem. That is, the bigram matrix for any given training corpus is sparse; it is bound to have a very large number of cases of putative “zero probability bigrams” that should really have some non-zero probability [5].

Since the zero probabilities cause complete sequences to be set to zero, if for example a new unseen and valid sequence has been encountered, some means of smoothing must be applied in order to make it possible to rank different sequences which would contain a zero probability, because they just never occurred in the training corpus.

TnT uses linear interpolation of unigram, bigram and trigram maximum likelihood estimates in order to estimate the trigram probability:

$$P(t_3|t_2, t_1) = \lambda_1 \hat{P}(t_3) + \lambda_2 \hat{P}(t_3|t_2) + \lambda_3 \hat{P}(t_3|t_2, t_1)$$

$\lambda 1 + \lambda 2 + \lambda 3 = 1$, so P represents a valid probability distribution. The λs are estimated by deleted interpolation where each trigram is successively removed from the training corpus and best values for the λs are estimated from all other N-grams in the corpus [4].

Unknown words are handled by a method called suffix analysis. This method works well on inflected languages and is described in [6]. Instead of assigning each unknown token a default value (e.g. back-off strategy in our unigram tagger), morphological and syntactic information is extracted in order to determine the most likely tag for an unknown word. For example, in English, any multi-syllable word ending in ’-able’ is almost certainly an adjective [6]. Thus the suffix is a strong indicator for a specific word class when dealing with unknown words.

3. Data Sets

Both taggers, the unigram tagger and the TnT tagger have been trained on the same data set consisting of roughly one million words from the Wall Street Journal newspaper. The part of speech tags used in the training sets were hand annotated during the construction of the Penn Treebank. The Penn Treebank Project annotates naturally-occurring text for linguistic structure using a tag set which consists of 36 part of speech tags.

The taggers in these experiment used two sources of test data in order to make a quantitative comparison feasible:

- Wall Street Journal Newspaper texts.
- Transcripts of radio news broadcasts from the Boston University Radio News Corpus.

The Wall Street Journal Newspaper is an influential international daily newspaper published in New York, primarily covering U.S. and international business and financial news and issues. According to the Wikipedia, the *Journal* features several distinct sections about:

- U.S. and international corporate news, as well as political and economic reporting
- coverage of health, technology, media, and marketing industries
- international financial markets
- personal investments, careers and cultural pursuits
- personal interests of business readers, including real estate, travel, and sports

The Boston University Radio News Corpus consists of over seven hours of speech recorded from seven radio announcers taken from actual broadcasts.

3.1. Important Data Set Aspects

Since the Boston University Radio News Corpus consists of recorded radio broadcasts, punctuation information is not present. Instead of providing punctuation information each phrase is annotated with a timestamp. Another problem is that not every broadcast starts with a capitalised word and sometimes contains capitalised phrases like “In”, in a context where it should definitely not be capitalised.

An additional interesting fact is that apart from the timing information recorded, the textual representation of the broadcasts does not include any cardinal numbers nor does it contain any special characters like “$”, “$\lambda$”, “#” etc. Numbers and special characters normally used in printed media are all spelled out in the Boston University Radio News Corpus whereas the Wall Street Journal News Paper includes many different special characters.

3.2. Data Set Preprocessing

In order to make the data sets digestible for the automated taggers, various preprocessing steps were necessary. Each tagger needed slightly different preprocessing of untagged corpora data for the automated tagging process.

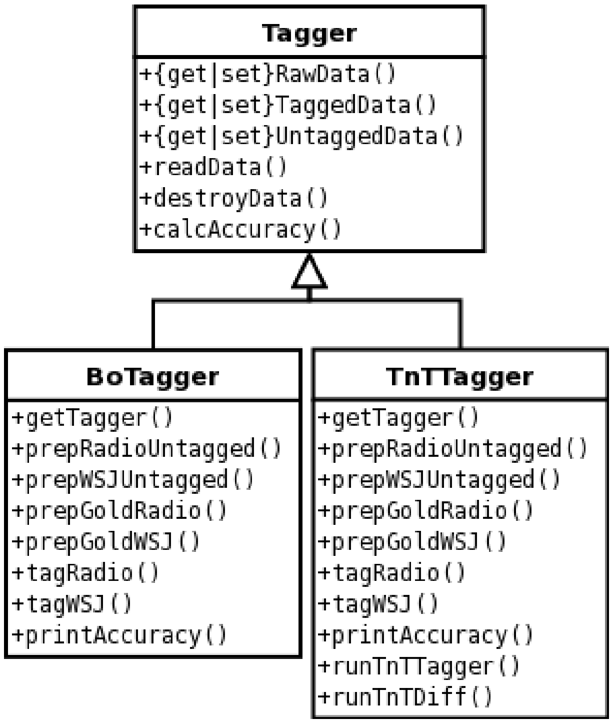

Figure 1: *Class Diagram of Tag Evaluator.*

3.2.1. NLTK Backoff Tagger Preprocessing Steps

Since the NLTK backoff tagger used in this experiment is a 1-gram tagger, and N-gram taggers in general should not consider context that crosses a sentence boundary, all NLTK taggers were designed to work with lists of sentences. Thus both corpora had to be parsed into such a data structure for the NLTK backoff tagger. A tagged corpus is represented by a list of lists of two-tuples, where a tuple consists of the token and its according tag. The list which holds the tuples represents a sentence, sentences themselves are stored in a list structure.

The Boston University Radio Corpus needed some additional preprocessing since the timing information together with special header and footer markers had to be removed. Each utterance which was bounded by header and footer markers was also suffixed with a period.

3.2.2. TnT Tagger Preprocessing Steps

The file format for untagged text files for TnT requires that each token of the text occupies its own line, delimited by a linefeed character. The format of tagged files is similar to that of untagged files. It is extended by adding an additional column per line where the columns are separated by any number of white space characters. Thus the first column represents the tokens and the second column represents the according tag.

4. Results

4.1. Evaluation Approach

In order to automate the process of preprocessing data, executing and evaluating the taggers, a *Python* program has been implemented. The backoff unigram tagger is provided by the NLTK Toolkit which is also implemented in *Python*. The TnT tagger and evaluator is executed from within the *Python* program and only relevant output information is extracted.

Since both taggers share some functionalities and are responsible to carry out the same task, namely automatically tag corpora, an object oriented approach (see figure 1) where common functionality has been extracted into a base class, has been chosen.

A Tagger object provides many methods for preprocessing and tagging corpora as well as calculating the accuracy of the taggers. Thus the process of tagging an untagged corpus with the NLTK tagger requires only the following four lines of *Python* code:

```
>>> from Tagger import Tagger, BoTagger
>>> bo = BoTagger()
>>> bo.setPathRadioUntagged('/somepath')
>>> bo.tagRadio()
```

The function *printAccuracy()* can be used to compute the accuracy of the tagger. This is done by comparing the taggers work with a tagged gold standard.

4.2. Results in Detail

Since the training sets for both taggers consisted of data from the Wall Street Journal Newspaper, one might expect the taggers to produce a better result than on the Boston University Radio News Corpus.

Running both taggers on untagged corpora and then comparing the tagged results with the golden standards yields the results displayed in table 1.

Tagger	Corpus	Accuracy
Backoff	Boston Radio	72.1%
TnT	Boston Radio	84.04%
Backoff	Wall Street Journal	83.4%
TnT	Wall Street Journal	93.54%

Table 1: *Tagger Accuracy.*

The accuracy represents the percentage of words the taggers have tagged correctly. Thus if a tagger has correctly tagged 14 out of 16 words, its accuracy would be equal to:

$$Accuracy = \frac{14}{16} = 87.5\%$$

Table 2 displays the total amount of tokens in each corpora used for this experiment and so we are able to derive the fact that the TnT tagger has tagged 53094 tokens correctly, and made wrong assumptions on 3666 tokens when it processed the Wall Street Journal corpora.

Corpus	Amount of Tokens
Boston Radio	5872
Wall Street Journal	56760

Table 2: *Tokens used in Experiment.*

5. Conclusions and Future Work

These results clearly show that our initial assumption that the taggers would perform better if the test sentences and the train sentences are from the same genre were correct. The fact that the tagger accuracy is roughly 10% higher if it is tested with corpora it was trained on backs this assumption up.

The TnT tagger tags more tokens correctly, roughly 10% percent more than the NLTK backoff tagger. Thus the TnT tagger which is based on Markov models combined with some sophisticated smoothing techniques yields state of the art results [4] and easily outperforms the simple NLTK backoff tagger.

One interesting observation is that the accuracy of both taggers decreases by roughly the same amount if they are tested with other corpora than the ones they were trained on. The NLTK backoff tagger accuracy decreases by 11.3% and the TnT tagger accuracy decreases by 9.5% when compared to the accuracy of tagging the Wall Street Journal.

This evaluation could be continued by making some more fine grained tagging error analysis. In this experiment tagging accuracy has been used in order to evaluate taggers. While the accuracy score is useful for this task, it does not give us any hints about where the taggers make systematic errors which could be eliminated. Thus it would be interesting to make some further error analysis by constructing confusion matrices for both taggers. Another approach would be to analyse the context in which tagging errors occur in order to improve tagging performance.

This paper shows that a sophisticated statistical tagger as the TnT tagger, shows a good performance if it is trained well and operates on related corpora.

6. Source Code

The *Python* code implemented for this paper can be found in:

```
$ /home/s0565052/icl
```

7. References

[1] Lars, R. “Enhancing tagging performance by combining knowledge sources”, Paper accepted for the symposium Korpusar i forskning och undervisning (KORFU 99), Vxj University, Sweden, November 11–12, 1999.

[2] Lars, R. “Something Borrowed, Something Blue: Rule-Based Combination of POS Taggers”, Second International Conference on Language Resources and Evaluation, Athens 31 May - 2 June, 2000. 21-26.

[3] Lin, Xiaofan. “Impact of imperfect OCR on part-of-speech tagging”, HP Labs Tech Report: HPL-2002-7R1 , 2002.

[4] Brants, T., “TnT - A Statistical Part-of-Speech Tagger”, In Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000, April 29 – May 3, 2000 Seattle, WA.

[5] Jurafsky, D., Martin, H. M., “Speech and Language Processing”, Prentice-Hall, 2003.

[6] Samuelsson, C. “Morphological Tagging Based Entirely on Bayesian Inference”, Proceedings of the 9th Nordiska Datalingvistikdagarna (NODALIDA 1993), Stockholm University, Stockholm, Sweden.