Improving Part-of-Speech Tagging with Features
How Quality Matters More than Quantity

Hanna Winter **Miriam Rieker**
[winterha|riekermm]@ims.uni-stuttgart.de
Institute of Natural Language Processing
University of Stuttgart

Abstract

Part-of-Speech Tagging is a computational method used to assign word classes to words in a corpus. Can Part-of-Speech Tagging be improved by changing, adding, and combining features? This paper proposes an answer to this question by experimenting with eight different word-related features with the goal to reach a better performance for the implemented stochastic Part-of-Speech tagger. The results conclude that the quality of a feature, meaning how well it describes a word, contributes more to the model's improvement than the amount of features added.

1 Introduction

Many applications of Natural Language Processing, such as Constituency Parsing, Statistical Machine Translation and Word Sense Disambiguation require preprocessing of text to extract relevant information (Brill, 2000:403). In most cases, Part-of-Speech (POS) tagging is one of those preprocessing steps. It assigns a word category to every word in the corpus.
There are various approaches to POS-tagging like rule-based taggers that follow handwritten rules, and stochastic models like the Markov Model which enable the user to use larger corpora and different languages. Stochastic models are more robust compared to rule-based taggers since they do not rely on handwritten rules and are able to adapt to changes in language or domain. POS-taggers make their predictions based on the word's context or morphological structure and a set of possible tags (Voultilainen, 2003:224-227).
The goal of this study is to find out whether the prediction of the tagger can be improved by changing the combinations and number of features. After introducing the implemented baseline POS-tagger, the paper continues with feature experiments and their results followed by a discussion and short conclusion.

2 The Baseline

The baseline POS classifier consists of an evaluation method and a basic POS-Tagger working with three features. The Part-of-Speech tagger was implemented as a stochastic model using a multi-class perceptron.

2.1 Method

A machine learning algorithm like the perceptron assigns the input, which is represented as numbers, to classes. In this case, the classes were a total of 41 Part-of-Speech tags according to the PennTreebank [1]. To receive numbers with which the perceptron can predict a class, features were extracted for each word, their occurrences counted and finally assigned to the according POS-tag. For the baseline, three features were used: the current, previous, and following word-string. While the current word-string represents the word's own tag, the previous and following word-strings can help to disambiguate word classes. For example, if a given word can be both noun and verb, the model gives more weight to the tag *noun* if the previous word is classified as determiner.

2.2 Implementation

The POS-Tagger was implemented with *python* in an object-oriented way [2]. The code can be split in four major parts: preprocessing of the corpus, extracting initial weights, assigning classes to the input, and training the weights.
First, in the preprocessing step, words and POS-tags were extracted and then used to extract each word's features. A second feature-related function

[1] see https://www.clips.uantwerpen.be/pages/mbsp-tags
[2] The code is available on bitbucket.org/hannwin/pos-tagger/src

extracted the features for a single word. The feature extraction was implemented in a way to enable the choice of specific single features. This method allowed to experiment with diverse combinations and amount of features later on.
In the second part, the feature weights were initialized by iterating over the POS-tags and features and saving all given features for a single tag. This step concludes the model's training which was done on a news corpus consisting of almost 762,000 words.
The next process assigns classes to the input, meaning POS-tags to a second corpus that contained almost 28,000 words. The features for each word were extracted and then counted for each tag. The tag with the highest count on a word's features was then returned as prediction.
Finally, the predicted POS-tags were compared to the correct ones. For every incorrect tag the model assigned, the weights were decreased for the features of the word whose tag was incorrect, and then updated.

2.3 Evaluation

To analyze the output of the model, Precision, Recall, and F-Score were calculated. It was decided to use micro-, as well as macro-averaged scores. While micro measures are used for datasets with variable sizes, macro calculations can be used when the model's performance on different datasets needs to be investigated.
To evaluate the model's output, its predictions were compared to the gold standard. The script counted true positives, false positives, and false negatives and finally calculated the evaluation scores.

3 Improving the POS-Tagger with Features

In an attempt to improve the baseline model, more features were added to the classifier and different numbers and combinations of these features were then tested and evaluated.

3.1 Performance of the Baseline

The baseline model's performance was measured without tuning the weights first, then with 1, 5, and 10 iterations. Without weight tuning, the baseline achieved a micro F-Score of .62 and a macro F-Score of .48 (see Table 1).

There are two possibilities on how to change feature weights: decreasing the weights for features that contributed to a wrong prediction or increasing the feature weights which lead to the correct class prediction. Moreover, the gradient descent can be changed alongside with the number of iterations. In this case, a gradient descent of 10 was used. It was chosen by manually changing its value and observing the results. The gradient descents with which the model was run were 1, 2, 5, 10, and 100. The lowest values did not change the evaluation results, whereas 100 seemed to be too high as the feature weights contain many low counts.
Overall, the model's evaluation scores tend to fall when both decreasing and increasing the according weights, but the output receives better scores when only the weights leading to the incorrect prediction are decreased. However, the results do not change much with different numbers of iterations, and changes in the model's evaluation scores could only be seen at the third decimal place. Therefore, it was decided to put more weight on feature experiments than weight tuning.

	Precision	Recall	F-Score
Micro	.62	.62	.62
Macro	.56	.42	.48

Table 1: Micro and macro Precision, Recall and F-Score for the baseline POS-tagger.

3.2 Additional Features

To improve the model, five new features were added to the baseline POS-tagger. They were chosen by comparing own ideas with other implementations such as Garrette and Baldrige's (2013) or Ratnaparkhi's (1996).
Recall that the baseline features were the word itself (W), the previous (W-1), and the following word-string (W+1). As new features the second previous (W-2) and second following word-strings (W+2) were taken into consideration. Additionally, it was investigated if the word was capitalized (CAP) or consisted of only digits (NUM), and if it ended in one of the suffixes *-ing, -ed, -ness, -tion, -tional* (SUF).

3.3 Plus-One Model

First, the new features were added to the baseline, one after another, to see if it alone can improve the performance (plus-one model). Three features

could indeed improve the tagger's output, namely capitalization, suffixes, and digits. As Table 2 shows, the micro F-Score could be increased by 1 pp with NUM, by 2 pp with the SUF-feature and by 5 pp with CAP. However, the macro F-Score stayed the same for NUM and CAP and even decreased when considering SUF.

	Micro F-Score	Macro F-Score
Baseline	.62	.48
Baseline + NUM	.63	.48
Baseline + SUF	.64	.45
Baseline + CAP	.67	.48

Table 2: Micro and macro F-Score for the plus-one model considering the features *digit*, *suffixes*, and *capitalization*.

In the case of the models plus-CAP and plus-NUM, the feature applies to few words, yet leads to the assignment of their correct POS which slightly improves the model. As there are more capitalized words than digits, considering capitalization leads to better results. The suffixes, on the other hand, are more ambiguous. The suffix *-ing*, for example, exists for nouns and verbs. As a result, the improvement is smaller than when considering capitalization.

3.4 All-Features Model

As a second step, all eight features were applied (all-features model). Compared to the baseline implementation, micro Precision, Recall, and F-Score improved by 4 pp, whereas all macro scores were lower (see Table 3). Macro Precision shrank the most with a loss of 12 pp.

	Precision	Recall	F-Score
Micro	.66	.66	.66
Macro	.48	.37	.42

Table 3: Micro and macro Precision, Recall and F-Score for the all-features model.

It seems that the baseline cannot be improved by simply adding features, meaning the kind of feature matters more. This hypothesis is confirmed by comparing the evaluation scores of the plus-one model to the all-features model's: the micro F-Score for the feature CAP, as well as the macro F-Scores of CAP, NUM and SUF are higher than when using all eight features.

3.5 Feature Combinations Improving the Baseline

In the next experiment, the model was run with all possible combinations of the eight features. This approach gives a more detailed insight in how much the amount of features versus the kind of feature contribute to approving the POS-tagger.

3.5.1 Procedure

The model was tested without tuning the weights, as the undertaken weight tuning approach did not seem to improve the output. Testing the POS-tagger with all feature combinations can give various insights. First, the feature combinations receiving the best scores can show if there are specific features that occur in all or many of them and therefore contribute more to the model's performance. Second, it will demonstrate whether the amount of features have an impact, meaning if more features lead to a higher improvement. Third, the results might show if there is a specific number of features which works best for the model.
For this purpose, the ten best performing feature combinations were taken for the analysis. All evaluation scores, i.e. micro and macro Precision, Recall and F-Score, were compared and displayed as three decimal numbers as rounding up would lead to many same scores. The results were compared to the baseline classifier and the all-features model.

3.5.2 Results of Feature Combinations

The first observation when looking at the results (see Table 4) comes from the kind of feature. The *current word-string* (W=) seems to play the biggest role, as it occurs in each of the ten best results and even defines the second best result alone. The features *capitalization* (CAP), *suffixes* (SUF) and *digit* (NUM) are each part of six different combinations and therefore contribute to the high results. Recall that these were also the three features which improved the baseline in the plus-one model.
Further, the features *previous* (W-1) and *following word-string* (W+1) which together with the *current word-string* define the baseline POS-tagger, seem not to be relevant for the model's performance. While W+1 occurs in only one combination, W-1 does not appear at all.
Considering the number of features which work best for the POS-tagger, there is no evident result.

Features	micro Precision	micro Recall	micro F-Score	macro Precision	macro Recall	macro F-Score
Baseline	.62	.62	.62	.56	.42	.48
All-Features Model	.66	.66	.66	.48	.37	.42
1. W=, NUM	.911	.911	.911	.864	.825	.844
2. W=	.909	.909	.909	.862	.823	.844
3. W=, NUM, SUF	.90	.90	.90	.861	.824	.842
4. W=, SUF	.899	.899	.899	.859	.824	.841
5. W=, NUM, CAP	.891	.891	.891	.857	.782	.818
6. W=, CAP	.89	.89	.89	.855	.781	.817
7. W=, NUM, CAP, SUF	.885	.885	.885	.854	.784	.817
8. W=, CAP, SUF	.884	.884	.884	.852	.782	.816
9. W=, NUM, CAP, SUF, W-2	.727	.727	.727	.711	.549	.62
10. W=, NUM, CAP, SUF, W+1	.726	.726	.726	.687	.562	.618

Table 4: Results for all possible feature combinations, showing ten best scores in decreasing order.

However, the number of features for the eight best scores ranges from one to four. The ninth and tenth best results, on the other hand, are achieved by using five features. These last two ranks are different in two ways: first, they contain one additional feature each which does not appear in the eight best results, namely the *second previous word-string* for rank nine (model W-2) and the *following word-string* for rank ten. Second, there is a gap between the scores of rank eight (W=, CAP, SUF) and nine (model W-2): the micro F-Score drops from .884 to .727 (difference of 15.7 pp) and the macro F-Score decreases from .816 to .62 (difference of 19.6 pp). In fact, there is a high correlation between the rank and amount of features used (ρ = .81), meaning models with fewer features receive a higher rank.

Lastly, while fewer features work better for the classifier, there is no exact number of features giving better results.

4 Discussion

To summarize, there are four features which lead to a high performance: *current word-string*, *capitalization*, *suffixes* and *digit*. Moreover, the two features *previous and following word-string* that are part of the baseline do not contribute to the model's improvement. Lastly, few features form a better model, but there is no specific number of features which works best.

Few features might work better for POS-tagging as each class receives fewer feature values and can be distinguished more clearly from the other classes. The model makes predictions by counting these values, and the more similar these counts are for different tags, the harder it becomes to clearly assign a class.

The kind of feature, however, seems to matter most. The *current word-string* contributes most to the model's performance, because words that were seen in the training data and also appear in the test data can be assigned the correct tag. The features *capitalization* and *digit* comprehend a small class of words which do not occur often but in contrast are easier to be classified. Lastly, the suffixes of a word can help recognize the POS as many suffixes are assigned to one word class, for example *-tional* for adjectives. Some suffixes are ambiguous, such as *-ing* that includes nouns and verbs, but still limit the choices.

5 Conclusion

The undertaken feature experiments show that the kind of feature is more relevant than the amount of features, and that a smaller number of features improves the model more than a high amount. The better a feature represents a word's context or its morphological structure, the more useful it is to the classification process.

For future work, further insights might be gained by exchanging the features resulting in low scores with new features. These might be features for infrequent yet easily distinguishable word classes, as seen with *capitalization* and *digit*. Moreover, morphological features like *suffixes* could be expanded. Lastly, comparing the computational cost of every model to the results it achieves could show if it is useful for greater applications.

References

Baldridge, Jason and **Garrette**, Dan (2013): Learning a Part-Of-Speech Tagger from Two Hours of Annotation. In: *Proceedings of NAACL-HLT 2013*, pp. 138-147. Atlanta, Georgia, Association for Computational Linguistics.

Brill, Eric (2000): Part-of-Speech Tagging. In *Handbook of Natural Language Processing*, eds. Dale, Robert, Mosil, Hermann and Somers, Harold, pp. 403-415, New York, USA, Basel, Switzerland. Marcel Dekker, Inc.

Güngör, Tunga (2010): Part-of-Speech Tagging. In *Handbook of Natural Language Processing*, eds. Indurkhya, Nitin and Damerau, Fred, pp. 205-235, Boca Raton, USA. CRC Press.

Ratnaparkhi, Adwait (1996): A Maximum Entropy Model for Part-Of-Speech Tagging. EMNLP.

Voultilainen, Atro (2003): Part-of-Speech Tagging. In *The Oxford Handbook of Computational Linguistics*, eds. Mitkov, Ruslan, pp. 219-232, New York, USA. Oxford University Press.