# Part-of-Speech (POS) Tagging Using Conditional Random Fields

Wuyeh Jobe
Carnegie Mellon University Africa
Kigali Innovation Village, Kigali Rwanda
jwuyeh@andrew.cmu.edu

## Abstract

*Part-of-speech tagging is a paramount NLP task that involves assigning a class to a word based on its use in a sentence. While POS tagging on its own does not provide any interesting application of NLP, it is nevertheless an important preliminary step to many NLP problems. For instance, POS helps provide context and meaning to words when doing sentiment analysis using Continuous Bag of Words (CBOW) or even when reconstructing sentences (word unjumbling). Various approaches have been used to do POS tagging. In this paper, we provide an explanation of creating a POS tagger using Conditional Random Fields.*

## 1. Introduction

Part-of-speech tagging is a crucial step in some NLP tasks such sentiment analysis and sentence reconstruction. It aims to assign a part of speech to a word based on its location in sentence, the context in which it is applied, etc.

Many approaches have been used to create POS taggers. This range from simple algorithms such as N-gram taggers (which applies statistics on the words in isolation – e.g. unigram tagger, or look at their immediate surroundings – e.g. bigram and trigram taggers) [1] to sophisticated algorithm that apply complex machine learning models (e.g. Hidden Markov Model, Conditional Random Fields, and Long Short Term Memory Recurrent Network)[2].

The different approaches, for the most parts, stems from the need to build POS taggers that generalize well to new corpuses and address the nuances in language.

In this exercise, we implement a POS tagger that uses Conditional Random Fields (CRF) to give the part of speech for a word in a sentence. Conditional Random Fields are discriminative models (models that model conditional probability distribution [2]) used for predicting sequences by using information learned from previous labels and getting more information to aid the model in making proper predictions [2][3].

For the rest of the paper, we will discuss the processes of data extraction and creating a tag list, experimenting with simple statistical models, training, testing and evaluating the CRF model and then end with a conclusion.

## 2. Data Extraction and creating a tag list

Because the train data was huge (over 500,000 sentences), it was paramount that the data extraction and cleaning process apply best practices for better performance. Thus, the Pandas package in Python was used alongside dictionaries and lists for efficient data extraction through indexing.

For grouping the words based on sentences (this is essential for getting the data for training and also when generating the tags for new words) and getting the tags associated with words, the index of the Pandas data frame with the loaded data was set to two indices with the sentence_id and token_id combined. This makes it possible to extract the tags associated with the words easily, since they are in different files.

The word-tag tuple is stored in a dictionary with the sentence id as the key and the word-tag list for a sentence as the value. Thus, when the values for the dictionary are extracted, the result is a two-dimensional array with each list in the main list representing a sentence.

When extracting the data to be tagged, the same grouping technique is used, although for this one, the set of words associated with a sentence can be extracted by using Pandas indexing.

There was minimal data cleaning carried out to remove NaN values from the datasets.

## 3. Experimenting with N-gram Taggers

After the tag list was created, it was easy to fit the data into the N-gram taggers explained in [1]. Using the unigram, bigram, and trigram tagger models individually produced very poor results on the testing sample of the tagged list. However, when these three taggers are combined through a process called "backoff" (when a tagger fails to assign a tag for a word, use another tagger to assign the specific tag [1]), they performed really well with a score of **92.8%** on the test sample (last 10% of the tag list) from the dataset. However, the performance was bad (**48.9%**) when tested on 1000 sentences of the **brown** corpus' news category. This warranted a search for a better methodology, and hence a better algorithm.

## 4. Training with Conditional Random Fields

Rather than just depending on statistical analysis of the words and their tags, Conditional Random Fields as noted in [2] and [3] provide a way for training a part of speech model that takes into consideration the features of the word and its surrounding words. This gives more context to the model and hence able to generalize well to new corpuses.

There is a Python package [4] called ***sklearn_crfsuite*** that implements the Conditional Random Field algorithm. This is the package used in this implementation.

### 4.1. Feature Engineering

As insinuated above, feature engineering is an essential task in this process. As explained in [2], "in CRF, a set of features are defined to extract features for each word…and…these set of features are called ***state features***". The features that depict the relationship of a word with other words are called ***transition features*** and "CRF will try to determine the weights of different feature functions that will maximize the likelihood of the labels in the training data" [2].

For getting the state and the transition features for words, the features explained in [2] (first letter in the word capitalized?, first word in the sentence?, last word in the sentence?, is the word alphanumeric, etc.) were tried. All the words were passed into the feature function to generate a dataset of features as **X** and the tags as **y**.

### 4.2. Training

The CRF model was trained on 90% of the data using Limited-memory Broyden-Fletecher-Goldfarb-Shanno (L-BFGS) training algorithm as an optimization algorithm and for 100 iterations. The regularization parameters for the Elastic Net (L1+L2) regularization were set to 0.01 and 0.1 respectively. Additionally, CRF was set to generate all possible transitions features, even ones that do not even occur in the training data.

It took about three hours to complete on a 64bit Windows machine with 8gb ram and 200GB of storage space.

### 4.3. Testing

The testing was done on 10% of the corpus data as well as 50% of the brown corpus' news category.

### 4.4. Evaluation

*The result improved significantly with this model. The performance on the 10% test data was about **94%**, while on the brown corpus was about 74%.*

## 5. Ideas for Improving Performance

While this was a great leap (from 48% to 74%), there is more room for improvement. Although several ideas come to mind, they could not be experimented due to challenges outlined in the 'Challenges' section. What follows is a series of steps that have been taken to improve performance.

The model could be improved by doing more feature engineering and parameter optimization

We worked on removing some features used in [2] and adding new ones from [5] based on work done in [6]. From [6], the most important features for predicting the tags are from checking *Does the word have a title case? Is it all in caps?, Is it a number? Its suffixes?* Firstly, the features used in [5] were tried as they are with the same optimization parameters as in 4.1.

Another feature engineering technique applied was adding the punctuations and other 'words' that were not marked "PLAIN" and had no tags in the **pos_tag_train**. For maintaining consistency in the features and the labels, however, 'words' that were digits were marked 'CARDINAL' and the rest of the non- "PLAIN" words were marked '.' (dot) indicating punctuation. This, along with making all the regularization parameters 0.1, lead to an additional 3% increase on the brown corpus. On the testing sample, it was 0.98%. However, this model was not used because it was predicting determinants and other tags as punctuations because of the conversion of most the non- "PLAIN" words into punctuations. A quick fix to this is to give each of the non-"PLAIN" words specific tags. However, this will require another time to train again.

To reduce the time for training, the iteration was limited to 40. This number was decided based on using a small sample to train and after each training increase or decrease to get the number of iterations that gives the best result, and lower than 100.

***After all these activities, the final score on the brown corpus was 75.99% and 95.1 on the 10% test sample.***

## 6. Challenges

More improvements could have been made when computing resources were not limited. Waiting for the model to train the data for three hours was really not a good experience and even though more parameters could have been added, different optimization algorithms explored, and more feature engineering done, it was simply not feasible. Notwithstanding, this was a worthwhile adventure.

## 7. Conclusion

Part of speech tagging is an essential preliminary step in many NLP tasks. So many algorithms are applied to achieve this goal. Nevertheless, the CRF model have proven to be an effective algorithm albeit the effectiveness depends on how well the features are engineered, parameters are optimized and how much training data is available.

In the future, rather than optimizing the parameters by trial and error, it would be good to automate the process or mathematically derive them for better analyses.

## References

[1] "Categorizing and Tagging Words", [Online]. Retrieved April 27, 2020 from http://www.nltk.org/book/ch05.html

[2] A. Ramachandran, "NLP Guide: Identifying Part of Speech Tags using Conditional Random Fields", October 5, 2018. [Online]. Retrieved April 27, 2020 from: https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31

[3] R. Chawla, "Overview of Conditional Random Fields", August 7, 2017. [Online] Retrieved April 26, 2020 from : https://medium.com/ml2vec/overview-of-conditional-random-fields-68a2a20fa541.

[4] "sklearn-crfsuite", [Online]. Retrieved April 26, 2020 from https://sklearn-crfsuite.readthedocs.io/en/latest/index.html

[5] "Tutorial", [Online]. Retrieved April 27, 2020 from : https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html#features

[6] "crf-pos-tagger", [Online]. Retrieved April 27, 2020 from: https://github.com/abcdw/crf-pos-tagger