Part of Speech Tagging using Hidden Markov Models, the Viterbi Algorithm, and Baum-Welch Expectation Maximization

Tyler Wengert
UCCS
twengert@uccs.edu

Abstract

Part of Speech Tagging (POS Tagging) is when each word in a corpus is labeled with its part of speech. This problem has many challenges due to the ambiguous nature of language; many times, a word can take on different parts of speech in different contexts, or a sentence can be interpreted to have various meanings that change how that words function. POS Tagging is extremely useful in evaluating language and in preparing data for other Natural Language Processing (NLP) tasks.

1 Introduction

When it comes to accomplishing POS Tagging tasks successfully, statistical language models have yielded very strong results. The Hidden Markov Model (HMM) is based off of many statistical language modelling principals as well as the Markov Assumption, which states that a system's next state is dependent only on its current state, not any of the previous states that the machine has visited. This is a memoryless scheme. To construct an HMM, you need to build a set of emission and transition probabilities from an already tagged data set. This certainly limits the HMM approach to POS Tagging, since you are required to have a pre-tagged corpus to train you model.

This weakness is addressed by using the Baum-Welch Algorithm for Expectation Maximization. Using a standard HMM model as the base, the model is initialized with random transition and emission probabilities and run over a series of steps, updating the probabilities at each step, to hopefully learn proper probabilities over time. This type of approach was utilized by (Gao and Johnson, 2008).

Once these models are trained, they of course need a way to predict tags for a given sequence of un-tagged words. For both of these model types, the Viterbi algorithm is useful for the final predictions. The Viterbi algorithm is a Dynamic Programming (DP) algorithm, and therefore improves the computation time for predictions over its non-DP counterparts. The prediction it outputs is the Viterbi path, or the most likely tagging path through the given sequence of words.

2 Related Works

In the early 20th century, a Russian mathematician, Andrey Markov, and his younger brother Vladimir pioneered work in probabilistic models for random sequences. Working under the assumption that the next state of the system is determined entirely by the current state and is not effected by the history of the system. Markov realized that his system showed strong results in modelling the vowels and consonants in the Russian language.

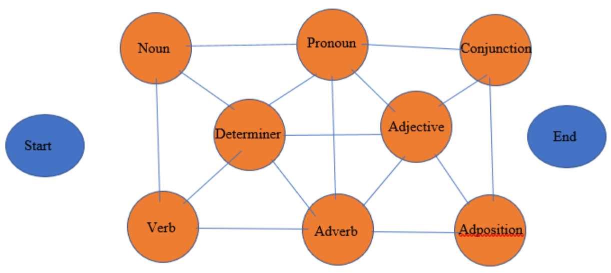
The Hidden Markov Model is a Markov Model where the states in the network are hidden from observation. Blunsom (2004) describes in great detail popular HMM implementations for POS tagging using the Viterbi Algorithm. Researchers such as (Alva and Hegde, 2016) have produced a variety of transition and emission probability tables from corpora of different types and lengths. Paired with the Viterbi Algorithm, which is a DP algorithm for finding the most probable path through the states in an HMM, A trained HMM can produce excellent results, as in (Al Shamsi and Guessoum, 2006) and (Nambiar et al., 2019).

To take the approach even further, researchers like (Gao and Johnson, 2008) combined the HMM with the Expectation Maximization algorithm to train a model without having a pre-tagged training corpus. The HMM is initialized with random transition and emission probabilities and run through a set of iterations to eventually learn effective probabilities.

3 Implementation

This program was built using Python 3.7.6 64-bit using the IDLE IDE. The program uses the NumPy package (Oliphant, 2006) for handling the data arrays, NLTK (Loper and Bird, 2002) for pre-processing of the corpora, and Pandas (Wes McKinney, 2010) for further organization of the NumPY arrays.

3.1 Hidden Markov Model



The key parts of the HMM are the transition and emission probabilities calculated from the corpus. The transition probabilities represent the likely that one tag will follow another. For example, if the current word is a noun, what is the likelihood that the next word is a verb or an adjective. Emission probabilities represent the likelihood that a given word will be a particular tag type. For example, the chance that the word book will be a verb or a noun.

The formula for calculating the emission probabilities is as follows:

$$\frac{occurrences(word, tag\_type)}{occurrences(tag\_type)}$$

The formula for calculating the transition probabilities is as follows:

$$\frac{co-occurrences(tag\_type\_1, tag\_type\_2)}{all-co-occurrences(tag\_type\_1)}$$

3.2 Viterbi Algorithm

The Viterbi Algorithm is a DP algorithm that was created in 1967 by Andrew Viterbi, as a decoding algorithm for digital communication. The algorithm is also commonly used to find the most probable path through a branching field of probabilities. This best path is known as the Viterbi Path, and is the sequence of observations that the model will output as its predicted tags for a given sequence of words.

3.3 Baum-Welch Expectation Maximization

The Baum-Welch Algorithm is an adaptation of traditional Expectation Maximization algorithms specifically tuned to maximizing the probability tables in a Markov Model. It was first described by (Baum et al., 1970), who showed that the probability tables of any Markov Chain could be iteratively be maximized. In NLP, this allows for the training of an HMM for POS Tagging without the use of pre-tagged data.

4 Results

4.1 HMM with Viterbi Algorithm

Test Run	Test Accuracy
1	83%
2	87%
3	86%
4	82%
5	81%
6	92%
7	81%
8	93%
9	84%
10	87%
Average Accuracy	85.6%

The model was run on a random 90:10 training data split 10 separate times. Sentences from the input file were mixed up, but word order within the sentences was maintained.

The final HMM model and Viterbi implementation performed very well, despite the small size of the available training data. The model showed an overall average accuracy of nearly 86%.

4.2 Baum-Welch with Viterbi Algorithm

Test Run	Test Accuracy
1	11%
2	30%
3	8%
4	17%
5	16%
6	14%
7	10%
8	17%
9	21%
10	14%
Average Accuracy	15.8%

In analyzing the predictions this model gave, it seemed clear that it was starting to pick up on some of the occurrence patterns in the text, but didn't have any reference as to which tags belonged to which types of relationships. For example many times, nouns and pronouns were both labeled as nouns, or the majority of pronouns were labeled as nouns while the majority of nouns were labeled as pronouns. Perhaps the learning could be improved if instead of starting from just an unlabelled set, the machine was also given an ultra-small (no more than a handful of sentences) labelled sequence to use as a sort of anchor to gauge what tags belong to which types of relationships.

5 Further Work and Improvements

Our data set was fairly small and had a small vocabulary, and so the problem was largely eliminated here, but one challenge that becomes more exaggerated with larger data sets is the issue of out-of-vocabulary (OOV) words. One popular approach is to break words down into their constituent parts, and use a rule based algorithm to aid the statistical model in recognizing the tenses of unfamiliar words from the word endings.

In my implementation on the Expectation Maximization Algorithm, the model's learned probabilities never reached the quality that the standard HMM model's probabilities did. The model was far more sporadic in its success, sometimes receiving scores far higher or lower than the average. Overall, it seems like my EM implementation has a hard time getting to the the best tags, and needs some tweaks to help it learn better probabilities.

6 Conclusion

The standard HMM model performed vary well, with an average accuracy of 85.6% on testing data. The probabilities calculated from the training text gave a good estimation of what the machine would see in the test set. A larger training set would likely allow this model to achieve results in the range of 90% - 95%. The HMM model combined with the EM Algorithm had a hard time learning, and the algorithm needs more optimization to be able to train successfully from untagged data. The statistical approach to modelling language has shown a good deal of success through time, and especially in combination with other methods such as rule sets or n-grams, HMM systems have produced some of the best results in POS Tagging.

References

Fatma Al Shamsi and Ahmed Guessoum. 2006. A hidden markov model-based pos tagger for arabic. In Proceeding of the 8th International Conference on the Statistical Analysis of Textual Data, France, pages 31–42.

Pooja Alva and Vinay Hegde. 2016. Hidden markov model for pos tagging in word sense disambiguation. In 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), pages 279–284. IEEE.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. The annals of mathematical statistics, 41(1):164–171.

Phil Blunsom. 2004. Hidden markov models. Lecture notes, August, 15(18-19):48.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 344–352. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, pages 56 – 61.

Sindhya K Nambiar, Antony Leons, Soniya Jose, et al. 2019. Natural language processing based part of speech tagger using hidden markov model. In 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pages 782–785. IEEE.

Travis Oliphant. 2006. NumPy: A guide to NumPy. USA: Trelgol Publishing.