Part of Speech Tagging Using Structural Models

Nil Beserler
University of California San Diego
nbeserle@ucsd.edu

Abstract

Natural Language Processing has been a steadily growing and popular field in Machine Learning, it is concerned with giving computers the ability to understand language similar to human understanding. A tool within NLP is part of speech tagging, also called grammatical tagging, which tags each word in a sentence to a corresponding part of speech tag. This paper will attempt part of speech tagging using various deep learning methods particularly structural models to handle sequential data. The models that will be explored, and compared are; Structural Support Vector Machines, Hidden Markov Model, and Conditional Random Fields.

1 Introduction

Sequential data refers to data that has an inherent order such as sentences in a paragraph or biological sequences of proteins. This paper will aim to accomplish part of the speech tagging task for the sentences in the Treebank dataset. To perform such tasks the models are specifically designed to handle variable length sequences, track dependencies, maintain order information, and share the parameters across the sequences. The most familiar approaches are rule-based, artificial neural network, stochastic and hybrid approaches(Chiche). This paper will explore three of the artificial neural network approaches including Structural Support Vector Machines, Hidden Markov Model, and Conditional Random Fields. These models are all well suited for the task given they can capture sequential dependencies, generalize over unseen data, and can incorporate linguistic features.

2 Methods

This paper examines methods to achieve state-of-the-art results for part of the speech tagging task by using different models including; Structural Support Vector Machines, Hidden Markov Model, and Conditional Random Fields for varying window sizes. To compare the models their test and training accuracies as well as their test and training times will be compared.

3 Dataset description

The dataset used in this paper is the Treebank corpus from NLTK (Natural Language Toolkit). NLTK is a widely used text-processing library for natural language processing. Treebank is a linguistic corpus from the Penn Treebank which includes sentences that are grammatically parsed into a syntax tree. It provides information about sentence syntax and part of speech tagging making it an appropriate corpus for this paper. It consists of 3914 sentences, 100676 words and tags, 12408 unique words, and 46 unique tags. To reduce complexity due to computational resources the experiment will preprocess the data to only explore part of speech tagging for 3 tags; verbs, nouns, and adjectives. Other tags will be labeled as "other". Additionally, only a sample of 1000 will be used from the dataset. An example of a sentence from the dataset can be displayed as:

```
Example 1:
Sentence: ['pierre', 'vinken', ',', '61', 'years', 'old', ',', 'wil
l', 'join', 'the', 'board', 'as', 'a', 'nonexecutive', 'director',
'nov.', '29', '.']
Labels: ['NOUN', 'NOUN', 'OTHER', 'OTHER', 'NOUN', 'ADJ', 'OTHER',
'OTHER', 'VERB', 'OTHER', 'NOUN', 'OTHER', 'OTHER', 'ADJ', 'NOUN',
'NOUN', 'OTHER', 'OTHER']
```

Figure 1: Words and their POS tags in the dataset

4 Experiment

4.1 Model Descriptions

4.1.1 Structural Support Vector Machine

Support vector machines are a type of supervised machine learning model that separates classes by maximizing the margin between the data points, it is mainly used for binary and multiclass classification. Structural SVMs are specifically designed

to handle sequential or structured output, similarly, they work by maximizing the margin between the classes but they also take into account the structural nature of the dataset. They consider contextual information and dependencies among variables.

In the experiment the Treebank dataset is preprocessed, a Word2Vec model is trained for word embeddings, the features and labels are created then the dataset is split into training and testing data. Next, a class is defined for the SSVM problem using the dlib library which is a common library for machine learning algorithms, it contains the functions make psi; which creates feature vector representations, get truth joint vector; which returns the feature vector for a particular example, and separation oracle; which is used to find the most violating label and it's feature vector. After the creation of the problem it is solved and the model's accuracy is evaluated on the test set.

The model is evaluated in various window sizes, a sliding window strategy is used for window sizes 3,5 and 7. The fixed window size is moved along the input sentences learning dependencies between the neighboring words allowing the model to consider the inter-textual context and capture the relevant patterns for the part of the speech tagging task.

4.1.2 Hidden Markov Model:

Hidden Markov Models are commonly used for part of speech tagging due to their principles. It is a probabilistic model that assumes underlying hidden states that generate observable models. These models define the joint probability of a sequence of symbols and their labels (state transitions) as the product of the starting state probability, the probability of each state transition, and the probability of each observation being generated from each state (NLTK Theme). HMM assumes that the probability of transitioning to the next word is only dependent on the current state and not on another previous state, which allows it to model sequential/structural data.

In the experiment, the hidden states could be considered as the part of speech tag, while the observable events are the words themselves. Therefore the HMM model calculates the state transition probabilities, meaning the probabilistic transitions between the part of speech tags.

The experiment makes use of nltk's HiddenMarkovModelTagger model and is evaluated by using the hmm tagger object. The preprocessing is done similarly to the structural support vector machine and a 80/20 split is used for training and testing data.

4.1.3 Conditional Random Field

Conditional Random Fields are another type of probabilistic model used for predictive tasks on structural data, it models the conditional probability distribution between input and output. In this specific task, it assigns the most probable POS tag to words based on neighboring words and other contextual features. More specifically it learns the conditional probabilities of transitioning from one word to another and the probabilities of observing a specific tag, then the probabilities are used to calculate the likelihood of observing the given sequence in the sentence. Unlike Hidden Markov Model Conditional Random Fields directly model the dependencies between input and output and do not use hidden states. Yet it can have complex feature representations and capture a wider range of dependencies due to its ability to include a wide variety of feature representations, modeling label dependencies, and the discriminative training process.

In the experiment scikit-learn's crfsuite package is used for the implementation of the CRF model, similarly to other models the dataset is preprocessed and split into training and testing data. However due to CRF's ability to capture complex relationships and features more features are extracted for this model including the following: word, is first, is last, is capitalized, is all caps, is all lower, prefix, suffix, previous 2 words, next 2 words. Then the model is trained and evaluated.

4.2 Model Comparisons

4.2.1 Structural Support Vector Machines with Different Window Sizes

Window Size:	Train Time	Test Time	Accuracy
3	178.09s	2.90s	40%
5	541.88s	5.09s	52%
7	732.20s	5.94s	53%

The window sizes experimented with for SSVM were 3,5,7. The performance improved as the window size increased which is likely due to increased context to learn from which directly increases the accuracy by capturing more complex relationships. However as the complexity increases so does the dimensionality of the data and the computation power

required, therefore longer testing and training times are required for the model to make predictions.

Even though there is a leap in accuracy from window sizes 3 to 5, there's only a 1% difference between window sizes 5 and 7. A window size of around 5 could be the optimal parameter since an increased window size also runs the risk of overfitting and unnecessarily increasing complexity.

4.3 SSVM, HMM and CRF Comparision

	Train Time	Test Time	Accuracy
SSVM (window s. 7)	732.20s	5.94s	53%
HMM	0.04s	0.20s	90%
CRF	2.34s	0.01s	95%

Out of the three models Hidden Markov model had the fastest training time of 0.04 seconds HMM was the fastest due to its training algorithm that involves efficient computations over the hidden states as well as the transition probabilities. The CRF model also had a short training time, whereas the SSVM model with the window size had the longest training time at 732.20 seconds due to the complex optimization problems that are computationally intensive.

The Conditional Random Field model had the fastest testing time at 0.01 seconds because of the dynamic programming algorithm, in this case, the Viterbi algorithm. The Viterbi algorithm (VA) is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process observed in memoryless noise (Forney). The HMM also had a relatively faster testing time at 0.04 seconds using the Viterbi algorithm for inference. The SSVM had the longest testing time, likely due to the added sliding window approach, which considers a large amount of context for prediction, increasing the model complexity.

The CRF model achieved the highest accuracy score out of the three models by capturing the complex dependencies and utilizing the added extracted features. Although a simpler model HMM also achieved a high accuracy score of 90%, the difference is likely due to HMM's inability to capture more complex relationships and dependencies. Whereas SSVM performed the poorest suggesting there is more room for hyper-tuning the model specifically for this task, with more computational resources the model could also be trained on additional data to improve its performance.

5 Conclusion and Future Directions

In conclusion, the CRF model was the best-performing model overall for the part of the speech tagging task, with a short training time of 2.34 seconds, the best testing time of 0.01 seconds, and the best accuracy score of 95%. The rich feature representations and explicit modeling of dependencies were responsible for the state-of-the-art results. The difference in performance between the three models can be attributed to their underlying principles, feature representations, computational complexities, and training algorithms.

For future research, it is important to consider the specific characteristics of the dataset and task at hand while picking the model. This study could be extended by increasing the possible labels for part of speech tagging, using a larger sized dataset, further fine-tuning the parameters, and introducing additional models such as Long Short Term Memory networks or Transformers.

6 Acknowledgments and References

6.1 Acknowledgments

The models were implemented using dlib's SSVM, nltk's hmm module, and sklearn's crfsuite.

6.2 References

References

[1] Chiche, Alebachew, and Betselot Yitagesu. *Part of Speech Tagging: A Systematic Review of Deep Learning and Machine Learning Approaches*. *Journal of Big Data*, vol. 9, no. 1, 2022. https://doi.org/10.1186/s40537-022-00561-y.

[2] NLTK Theme, and Sphinix. *NLTK Documentation*. *NLTK Project*, 2 Jan. 2023. www.nltk.org/api/nltk.tag.hmm.html.

[3] Forney, G.D. *The Viterbi Algorithm*. *Proceedings of the IEEE*, vol. 61, no. 3, 1973, pp. 268–278. https://doi.org/10.1109/proc.1973.9030.