Analyzing Hidden Markov Models and Bidirectional Long Short-Term Memory Networks for POS Tagging

Kevin Wen
Pratt School of Engineering
Electrical and Computer Engineering
Duke University
sw386@duke.edu

Abstract

In this project, two methods of part-of-speech (POS) tagging were implemented and compared. The first utilized a Hidden Markov Model (HMM) while the second was a Bidirectional Long Short Term Memory (LSTM) neural network. Both models were trained and validated on a synthetic data and then the Brown Corpus. The HMM performed well, with 90% validation accuracy while the LSTM achieved state of the art performance, surpassing 97% accuracy.

1 Introduction

Part-of-speech (POS) tagging is the task of assigning part of speech tags to words in a text. It is a sequential labeling task, a type of pattern recognition task that involves the assignment of a categorical label to each member of a sequence of observed values. POS tagging is important as it builds the foundations for much more complex natural language processing applications [1]. POS tagging builds the foundations for named entity recognition (NER) tasks. Additionally, POS tagging can be used to build lemmatizers which are widely used in data preprocessing. Therefore it is vital to build a POS tagger that is both robust and accurate. This project will explore the use of two models capable of POS tagging with high accuracy. The first is a generative probabilistic language model known as a Hidden Markov Model (HMM). The second is a discriminative recurrent neural network known as a Bidirection Long Short-Term Memory (BiLSTM) neural network. Both models will be trained on two sets of data: synthetic data generated by the HMM and the Brown Corpus. For this project, only labeled data was considered and both models were trained under supervised training.

2 Methods

2.1 Markov Model

The historical method for assigning POS-tags to the words in a text is using a sequence classifier. The standard Hidden Markov Model is a sequence classifier contains the following components [2]:

$$Q = [q_1 \quad \dots \quad q_n] \; O = [o_1 \quad \dots \quad o_T]$$

$$\Pi = [\pi_1 \quad \dots \quad \pi_n] \; V = [v_1 \quad \dots \quad v_m]$$

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n} \end{bmatrix}$$

ECE 684 (2021 Fall) final report.

$$B = b_j(o_t) = \begin{bmatrix} b_{1,1} & \dots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \dots & b_{m,n} \end{bmatrix}$$

In a POS tagging context, Q is the set of states representing available tags. O is a set of observations, which represent the words of a sentence. This set of observations are drawn from V, a set of all vocabulary words. Π is the initial state probability distribution, meaning the probability the Markov Chain will start in a specific state q_i. A is the state transition probability matrix, where $a_{i,j}$ is the probability of transitioning from state q_i to state q_j. Finally, B is a set of emission probabilities, indicating the probability of an observation o_t being in state q_j. In addition to these components, the following components were included so that the model would be able to act as a Bigram model for synthetic data generation:

$$C = \begin{bmatrix} c_{1,1} & \dots & c_{1,m} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \dots & c_{m,m} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & \sigma_m \end{bmatrix}$$

In this context, C is a transition matrix indicating the likelihood of transition from vocab word v_i to vocab word v_j while Σ is the initial vocab probability distribution, meaning the probability the Markov chain will start with vocab word v_i.

Training a Markov Model involves computing the probabilities for A, B, C, Π, and Σ. However, an issue that arises when dealing with unknown words is that they have a probability of 0 in each matrix. To address this, Laplace (plus-one) smoothing is applied in the form of:

$$a_{i,j} = \frac{count(q_j|q_i)+1}{\Sigma_k count(q_k|q_i)+1} \; b_{i,j} = \frac{count(v_j|q_i)+1}{\Sigma_k count(v_k|q_i)+1} \; c_{i,j} = \frac{count(v_j|v_i)+1}{\Sigma_k count(v_k|v_i)+1}$$

The maximally probable tags for a sentence can be obtained using the Viterbi algorithm which computes and stores the most probable sequence of states from time 1 to time T [3]. The Viterbi algorithm is a dynamic programming algorithm with the following transition function:

$$v_0(j) = \pi_j b_j(o_0)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1} a_{ij} b_j(o_t)$$

One weakness of the Viterbi algorithm is that for an unknown word, $b_j(o_t)$ is not representative of the true tag for the unknown word. Therefore, addition out of distribution (OOV) smoothing was added by replacing $b_j(o_t)$ with the overall tag distribution. In other words, we assume the probability of an unknown word having tag q_j is the percentage of tags in the training corpus being q_j. Previous literature has shown that this addition improves HMM performance and is more representative of the true $b_j(o_t)$ distribution of an unknown word [2].

2.2 BiLSTM

A Bidirectional Long Short Term Memory (BiLSTM) neural network is a recurrent neural network that can incorporate contextual information from long inputs and have been shown to a powerful model for sequential labeling tasks. BiLSTM is a modified version of the original LSTM model by having both a forward and a backward LSTM which contribute to the overall output [4].

The standard LSTM model is composed of several LSTM cells. Each LSTM cell contains several layers that allow it to remember contextual information. Inputs to the LSTM cell at time t are x_t, the current data, h_{t-1} the previous hidden state, and C_{t-1}, the previous cell state. When data is passed to an LSTM cell, the first layer it reaches is a forget gate layer:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

In this equation, σ is the sigmoid function, W is the weights, b is the biases, h is the previous hidden state, and x is the current input. This layer determines what proportion of data in each cell state the neural network should forget.

Next, the output of the forget layer is fed into the input gate layer which is constructed from both a $\tanh$ and σ function:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \sigma(W_C * [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

In these functions, i_t indicates which values to update in a cell state while $\tilde{C}_t$ indicates the candidate values to update with. These two, alongside the previous cell state, compute the new cell state C_t.

Finally, the output layer takes the previous hidden state and the current cell state to compute both the output and the new hidden state using another sigmoid and tanh function:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

The outputs of the LSTM cell are then o_t, which is used for categorization, as well as h_t and C_t which are fed into the next LSTM cell.

The custom BiLSTM model implemented for POS tagging contains a word embedding layer, a BiLSTM layer, a fully connected linear layer, and a dropout layer. Word embedding is a powerful neural network technique to deal with large vocabulary sizes. Unlike one hot encoding, which converts a word into a vector the size of the vocabulary length, which can result in the curse of dimensionality. In word embedding, a tokenizer converts each vocabulary word into a unique index. Then word embedding layer is trained alongside the LSTM neural network to give a low-dimensional vector representation of a word. Following the word embedding layer is a dropout layer, which masks part of the input data to prevent overfitting. Next is the core BiLSTM layer which is trained for POS tagging. Finally, the last layer of the neural network is the fully connected linear layer, which takes the output and applies a linear transformation to classify each word according to their POS. Additionally, the BiLSTM was trained using the Cross-Entropy Loss Function:

$$\mathcal{L} = \tfrac{-1}{N} \textstyle\sum_j y_j * \log(\hat{y}_j)$$

2.3 Training Data

Both the HMM and BiLSTM models were trained on synthetic data generated from the HMM model as well as the Brown Corpus. The Brown Corpus is composed of 57,340 tagged sentences of variable length. The synthetic data is composed 30,000 tagged sentences of length 10. Each dataset was partitioned into training and validation sets, and variations of the amounts partitioned into each were tested. The BiLSTM model was trained for 5 epochs using a batch size of 1.

3 Results

3.1 Synthetic Data

From Table 1 it was found that BiLSTM performed much better than HMM across the board. Utilizing 80% of the synthetic data allowed the BiLSTM to achieve 92% accuracy while the HMM model achieved 76.2%. As the percent of total data allocated to training decreased, the accuracy decrease dramatically. In the end, using only 20% of the synthetic data meant the HMM accuracy became 0.485 while BiLSTM became 0.615.

3.2 Brown Corpus

From Table 2 it was found that BiLSTM performed much better than HMM across the board, achieving state of the art performance. Utilizing 80% of the synthetic data allowed the BiLSTM to achieve 97% accuracy while the HMM model achieved 89%. Furthermore, for BiLSTM it was found that as the percentage of data partitioned for training decreased, the overall effect on accuracy was small. For HMM, it was found that the best accuracy was when either a small amount of the Brown Corpus was partitioned for training or when a large amount was partitioned for training.

Table 1: Best Accuracy Synthetic Data

Percent Training Data	HMM Accuracy	BiLSTM Accuracy
20%	0.485	0.615
40%	0.566	0.807
60%	0.575	0.912
80%	0.762	0.923

Table 2: Best Accuracy Brown Corpus

Percent Training Data	HMM Accuracy	BiLSTM Accuracy
20%	0.860	0.949
40%	0.776	0.958
60%	0.882	0.968
80%	0.905	0.972

4 Conclusions

The results obtained were partially surprising and partially expected. For the synthetic data, it was clearly seen that as more data was available to train the HMM and BiLSTM, the performance of both models were superior. One explanation for this is that the HMM performs worse with a smaller dataset as more words are excluded from the learned vocabulary, so the out of vocab (OOV) introduced in the Viterbi algorithm was used more often, which may result in reduced validation accuracy. For the BiLSTM model, a similar reduction in validation accuray was observed. It is likely that with such a small dataset, BiLSTM began to overfit from the training data, leading to reduced accuracy. There is support for this hypothesis as the training accuracy for synthetic 20% training data is 0.922, while the validation accuracy is 0.615. It was even more surprising to see that despite changing the data size for BiLSTM, the validation accuracy did not chnage much, and remained between 95% and 97% accuracy. It may be the case for BiLSTM that true data which has meaningful context may not need a large dataset to train for it to learn the necessary features that determine the POS for words in a sentence. This effect was also replicated with the HMM which is surprising to see. For HMM, we see that as the fraction of training data was reduced from 80% to 40%, HMM accuracy started decreasing as well. However, at 20% training data, the accuracy increased and achieved 90%. This may be because with that small a dataset, only the most commonly used words become part of the HMM vocabulary. The most commonly used words may be enough of a predictor to achieve high classification accuracy with OOV viberti. Another explanation is that if the vocab size becomes too large, then the computed probabilites in the matrices become small and hard to distinguish from one another, reducing the viability of the viberti algorithm.

Analyzing the performance of the two models shows that HMM is an order of magnitude faster. Training an HMM model takes a couple minutes on a CPU while training a BiLSTM model takes around 8 minutes per epoch on a Google Colab GPU. For 5 epochs, that is around 40 minutes of training time. Interestingly, 5 epochs seems to be enough to tune BiLSTM to its optimal accuracy as during the last epoch, the training accuracy begins to exceed the validation accuracy which is representative of overfitting to the train dataset. BiLSTM works well if the hardware and the time constraints are without issue, whereas HMM may be a viable alternative if these resources are constrained. Additionally, it was found that for an HMM model, training on a larger dataset reduces both the training time and the validation time. This makes sense as with a larger vocabulary, there are more intermediate steps to compute in the Viterbi dynamic programming algorithm. Therefore, if validation time is a constraint, then it may be worthwhile to use a small dataset with limited vocab to train the HMM model.

Future investigations include research into why exactly using small percentages of the original training data for HMM and BiLSTM enable the two models to retain their high validation accuracy.

References

[1] Milan Straka, Jana Straková, and Jan Hajic. Czech text processing with contextual embeddings: POS tagging, lemmatization, parsing and NER. *CoRR*, abs/1909.03544, 2019.

[2] Martin Haulrich. Different approaches to unknown words in a hidden markov model part-of-speech tagger. 2009.

[3] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000.

[4] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network, 2015.