Lab 4: Viterbi and Beam Search

Registration Number: 180128251

March 30, 2019

1 Implementation

The application is developed with Python 3 and was tested on a computer with a 8GB RAM and an Intel i5 - 8th Gen at 1.60Hz. It was developed as an extension to provide functionality for viterbi and beam search options to the already NER program. It therefore accepts two optional arguments in addition to the positional arguments of the training data and testing data, **-v** to indicate the use of viterbi and **-b** to indicate the use of the beam search algorithm, which takes an integer argument to indicate the size of k when finding the *top-k*. The functions added to the already existing application are: **viterbi** that returns a list of predicted tags given a sentence. The function (similar to the **argmax** function) accepts the current weights, input sentence and feature type(in this case, the *current word-current label* features). Each word in the sentence has its tag predicted by selecting the tag with the maximum dot product value of the weights and its feature count in addition to the previous viterbi score. Likewise, a **beam_search** function was also implemented to predict tags using the beam search algorithm. It however has an optional parameter of *top_k* that indicates what value of k to be used when obtaining the *top-k* previous tags in the sequence to be used for predicting the next word. The algorithm is similar to that of the **viterbi** algorithm in that, it performs the prediction for every word in the sentence at a time, but prediction is done using only the *top-k* values.

2 Evaluation

Table 1: Algorithm metric comparison

		Time	Accuracy
Argmax		48.769s	0.66031
Viterbi		1.837s	0.66031
	k = 2	1.647s	0.66031
Beam search	**k = 3**	1.697s	0.66031
	k = 4	1.744s	0.66031

As expected, the **viterbi** algorithm runs much faster than the standard structured perceptron with the **argmax** function. This however does not change the results or cause a reduction in the accuracy value since the **viterbi** algorithm is an exact search algorithm. Although **viterbi** is an exact search algorithm and performs the same functionality in returning the same results as the standard structured perceptron with the **argmax** function would, it does this in much less time of 1.837s as compared to the 48.769s run by our **argmax** function that produces the same results.

Applying the **beam search** algorithm for training our model, also results in no change in the accuracy of our model. However, the time it takes for the script to execute is a bit lower than that of the **viterbi** algorithm and that of the standard structured perceptron with the **argmax** algorithm by extension. As our value of k increases, the algorithm takes longer(although not by much) to execute as shown in the table above. When run with a $k = 2$, it took 1.647s to completely run, likewise $k = 3$ took 1.697s and $k = 4$ completed in 1.744s. The **beam search** algorithm being an inexact search algorithm may have yielded different accuracy values, albeit not in our case. It can safely be said that using beam search for this task did not affect the accuracy of our model.

3 Conclusion

It can be observed that, a better method for solving a problem that uses the Hidden Markov Model (HMM) is to use the viterbi algorithm when juxtaposed with the argmax algorithm. The advantage of this method is that, it's an exact search algorithm so the same solution is expected to be obtained if the argmax algorithm is used instead. However, the beam search algorithm can help improve the speed of our viterbi algorithm if the labels to be predicted are too many. Although, the beam search doesn't check our entire search space, it is still likely to produce better results when the value of k is carefully chosen.