CS5012 - P1
(Small) Language Models
200007413

1 Implementation

1.1 HMM Model

The HMM model training begins through counting the number of times each POS tag is followed by another POS tag, and the number of times each POS tag emits a particular word. This is done through iteration over each sentence from left to right, adding start and end markers to sentences.

The training completes by calculating the probabilities for each transition and emission. The transition probabilities as the number of times a particular tag is followed by a particular tag, by dividing the transition count by the total transitions for a tag. Where combinations of transitions are not found, <unk> transitions are calculated using Witten-Bell smoothing for unseen values, making unknown probabilities less likely where there are more potential other tags that have been seen following the previous tag.

1.2 Viterbi Algorithm

To calculate the most likely sequence of POS tags for a given sentence, the Viterbi algorithm utilizes the probabilities calculated from a HMM model.

The Viterbi algorithm initially constructs a 2-dimensional table to store the probabilities of each potential tag at each position in the sentence, given the previous tags. The initial column is filled with the probability of the start tag transitioning to each tag. The algorithm then iterates over each following word in the sentence and fills in the probabilities of each tag that follows the immediate previous tag, multiplied by the emission probability of the word given the tag. The maximum probability at each position is then selected, starting with the final word and working backwards to get the most likely sequence of tags.

Log space is used to prevent probability underflow as probabilities are multiplied ogether the number of times equal to the length of the sentence, which can lead to very small probabilities, which may underflow to zero in normal space.

1.3 Accuracy Calculation

Accuracy calculation is simply calculated as the proportion of the tags outputted by the Viterbi algorithm that match the tags in the test set. Through looping over each sentence in the test set, we can total the number of tags that match the test set, and divide by the total number of tags.

1.4 Bigram Model

The bigram model is implemented by counting the number of times each word is followed by another word in the training data using a similar 2-dimensional array as the transitions in the HMM model, and 1-dimensional array as the emissions. This is used to calculate the bigrams and unigrams which are both used to calculate the bigram probabilities.

2 Results

Lang	HMM Accuracy	HMM Perplexity	Bigram Perplexity
en	0.98252	95.00279	13.99254
orv	0.82102	8045.24735	146.68261
tr	0.97091	428.37705	21.10778

Table 1: Accuracy of the HMM model, with perplexity values for the HMM and Bigram models using the English, Old East Slavic, and Turkish test corpuses. Given to 5 decimal places.

3 Reflection

The HMM model had high accuracy for English and Turkish test sets. This is consistent with the low perplexity observed on these test sets. The model had a lower accuracy on the Old East Slavic test set, which is consistent with the higher perplexity observed. The dramatic increase in perplexity on the Old East Slavic test set is likely due to the

The HMM model had high accuracy on the English and Turkish test sets. A significant performance drop was observed on the Old East Slavic test set. This could be due to the small size of the training set for Old East Slavic. This may be due to the relatively small training set for Old East Slavic. The model may not have been able to learn the language model effectively due to the small size of the training set. The model may have overfitted the training data, leading to poor generalization on the test set.

The HMM model additionally had low perplexity on the English and Turkish test sets - performing best with the English test set. The model had a higher perplexity on the Old East Slavic test set, which is consistent with the lower accuracy observed on this test set.

Overall, the HMM model by and large performed well. The complexity of the Old East Slavic language may have contributed to the lower accuracy and higher perplexity observed throughout these experiments, suggesting a larger training corpus for more complex languages may be necessary to achieve better results.

The overall implementation is efficient, completing training and testing in a reasonable amount of time. The Bigram model is significantly slower than the HMM model, as it requires more calculations to determine the probabilities of each word given the previous word, and there are more words than token types in most languages. This also results in the HMM model having a lower memory footprint than the Bigram model.