Barış Coşkun
N18137709
CMP670
Assignment 1

For the first assignment of Natural Language Processing course, N-gram language models should be implemented. For the first part n-gram models are implemented where n is 1, 2 and 3. Also some additional functions are required for measuring the quality of models. MLE probability and smoothed probability should be calculated for a given sentence. Also, 5 sentences should be generated using these 3 models. For the second part, it is asked to use linear interpolation algorithm and for the third part discounting algorithm should be imported to probability calculations.

For the first part, ngram(n) function implemented which takes the n value and creates a language model. Basically it returns the 2 dimensional array that has unique word combinations of b word in training file as a row and corresponding number of word combinations as a second row. Then, prob(exampleSentence,n) function implemented. That function basically calculates the overall probability of given exampleSentence according to n-gram model. Then sprob(exampleSentence,n) function is implemented. The difference of sprob function is that uses add-one algorithm that handles the 0 probability error by adding 1 to each number of word combinations. By adding one to all combinations, the unseen ones in training data also have some few probability. Then ppl function is implemented that calculates the perplexity score of a given sentences according to the selected probability function(prob, sprob, linearInterpolation or discounting). In order to generate sentence using language models, nextWord(words,wordPos,n) function is implemented. Basically, it generates a possible word by using n-gram language model that can be concatenated to words. This process is limited to 20 words which is controlled by wordPos variable. Also when a stopping condition or end-of-sentence is generated by next function, the process is stopped directly. Besides, in order to find a suitable words for sentence beginning, a function called findBeginning() is implemented. That function returns a list of sentence beginning and a random one is selected among them. As it can be seen from results, as model get higher level (means higher n value), the generated sentences become more meaningful. Also as expected, perplexity scores are increased.

For the second part, linearInterpolation function is implemented. This algorithm calculates the probability by giving some weight to lower level of language models and calculates the weighted average as a result. The weights are calculated according the development dataset. The development set is used for calculating the perplexity score for different weight combinations. The combination that maximizes the perplexity score is used for test set.

For the third part, discounting function is implemented. This algorithm subtracts a “beta” amount of probability for each word combination and distribute this probability according to the result of one lower level of language model. The same approach that we used for linear interpolation is used for optimizing the “beta” value. A set of “beta” values are used for calculating the development set and the one that maximizes the perplexity score is used for calculating the result of test data.

The perplexity results from second and third parts are higher than first part. However there is no significant difference between second and third part.