NLU first Assignment

Sonu Dixit
sonudixit2k@gmail.com

1 Language Model

I have implemented n-gram model to calculate perplexity of test data. For handling unseen word in test data, i reserved 10 percent probability for unknown words. I selected randomly 10 percent words and converted them to unknown word(a special token to model unseen words probability). In all the files i hae used 80-20% test train split. this split was made randomly(out of all the senteces of the corpus i selected 20 percent of them)

2 First file firstPart.py

Brown Corpus has been used for training and testing. Out of all sentenes, 80-20% random split has been used for testing. For modelling unsen words, i tried using Laplace smmoothening and kneser kney, but the best result i got was on just reserving 10 percent probablity for unseen words.Out of all words in train data, i selected 10% of them randomly and converted them to unkown token. Perplexity results are as follows: using Bigram Model(53.58), using Trigram Model: (90.68), surprisingly perplexity increased from bigram to trigram model. it should have decreased.

3 Second File secondPart.py

GutenBerg Corpus was used.Similar method as explained above has been used.Perplexity Results are: using Bigram=67, using Trigram=47.As expected, In this case perplexity decreased in moving from bigram to trigram model.

4 Third File thirdPart.py

Train Data= 80% brown corpus and 100% gutenBerg corpus. Test Data= 20% brownCorpus. I used same method for calculating perplexity as explained above. Result obtained are: using Bigram(77.71), using trigram(113.29)

5 Fourth File fourthPart.py

Train Data= 100% brown corpus and 80% gutenBerg corpus. Test Data= 20% gutenbergCorpus. I used same method for calculating perplexity as explained above. Result obtained are: using Bigram(85.64), using trigram(118.75)

6 Fifth File for Sentence Generation

I have used combination of ngrams(n=2,3,4) for generating sentences. For nGram Model(n=2,3,4) i prefixed every sentence with n-1 start tokens, and suffixed every sentence with n-1 end tokes. For every given trigram, 5 most probable words are genertaed using 4gram model. If no trigram is quadGram model does not return any word, then next word is generated using trigram model. if trigram model also fails to generate the word, bigram model is referred to generate the next word. For every word, 5 most possible next words are selected. out of these 5 words, a word is selected at random for making the sentece.