Assignment 1 - Language Model Generation

Sourabh Balgi
Sr. No. - 14318
EE Department
M. Tech. (Systems Engineering)
sourabhbalgi@gmail.com

Abstract

Assignment report with the details of the methodology, observations, results and plots. Brown corpus and Gutenberg corpus were used to create the Unigram and Bigram models with different setting. In the next sections, the observations and results are presented. Absolute discounting and Kneser-ney smoothing is used for model generation.

1 Model Implementation Methodology

All the models were implemented in MATLAB without any use of built-in functions. All the preprocessing of the text data was done according to the dataset under consideration for the current setting.

1.1 Model selections and Smoothing

Unigram and Bigram models are selected for the given Corpus and evaluations settings S1-S4. Kneser-Ney smoothing is applied to the distribute the probability mass over non occurring n-grams

1.2 Data splitting and Pre-processing

The given data is split according to the number of sentences in each file. The Train-Dev-Test split in carried according to 7:1:2 proportions. The data was not divided based on the individual files which lead to bias of certain vocabulary words confined only to selected set of files. The Data split carried out here contains the vocabulary spread over all the files, as every file is included in train, dev and test sets.
The Brown corpus has parts-of-speech tags which are to be removed. Gutenberg corpus was preprocessed by removing unwanted patterns which are not really informative related to our language model (e.g. The Chapter/verse numbers in The Bible is not really necessary. All the punctuation were removed because of their common occurrence with large frequencies.

1.3 Tokenization

The processed data is divided into tokens of unigram and bigram for model estimation. The unigram and bigram counts are obtained for the calculation of n-gram probabilities. Initial models were implemented with $<s>$ as start of sentence tag and $</s>$ as end of sentence. Since large number of sentences are present, the counts of these tokens are very large. Since it's always present with a high probability in test data, the actual metric hence calculated will yield a less value even though the model is not predicting the words correctly. $<UNK>$ is added to vocabulary to handle OOV (Out-of-Vocabulary) situations.

2 Model Evaluation Metric

Perplexity: The perplexity (PP) of a language model on a test set is the inverse probability of the test set, normalized by the number of words.
For a test set $W = w_1 w_2 ... w_N$:

$$\text{PP(W)} = \text{P}(w_1 w_2 ... w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

The generated model is tuned for the discounting parameter over the development set and then tested on the test set. The perplexity defines the measure of likely predicting the test set. A low perplexity would imply a very good model, as most information is captured in the model

3 Observations and Results

Notations :
D1: Brown Corpus
D2: Gutenberg Corpus d = absolute discounting parameter

- *Setting S1* :
  Train: D1-Train, Test: D1-Test
  Dev Data Unigram Perplexity is 1308.6423 for d = 0.5
  Test Data Unigram Perplexity is 1287.4157 for d = 0.5
  Dev Data Bigram Perplexity is 706.8972 for d = 0.9
  Test Data Bigram Perplexity is 716.6883 for d = 0.9

- *Setting S2* :
  Train: D2-Train, Test: D2-Test
  Dev Data Unigram Perplexity is 754.8077 for d = 0.5
  Test Data Unigram Perplexity is 827.7814 for d = 0.5
  Dev Data Bigram Perplexity is 330.6528 for d = 0.8
  Test Data Bigram Perplexity is 408.5582 for d = 0.8

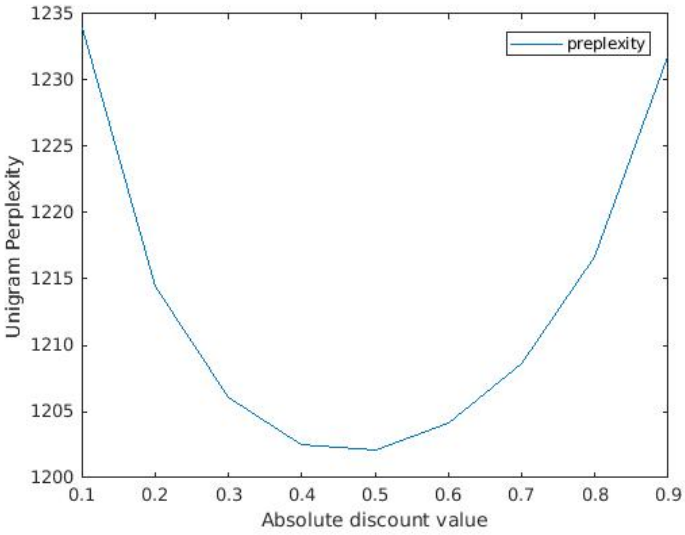
- *Setting S3* :
  Train: D1-Train + D2-Train, Test: D1-Test
  Dev Data Unigram Perplexity is 1014.9308 for d = 0.5
  Test Data Unigram Perplexity is 1531.2429 for d = 0.5
  Dev Data Bigram Perplexity is 449.2117 for d = 0.8
  Test Data Bigram Perplexity is 760.0671 for d = 0.8

- *Setting S4* :
  Dev Data Unigram Perplexity is 1014.9308 for d = 0.5
  Test Data Unigram Perplexity is 888.056 for d = 0.5
  Dev Data Bigram Perplexity is 449.2117 for d = 0.8
  est Data Bigram Perplexity is 422.8754 for d = 0.8

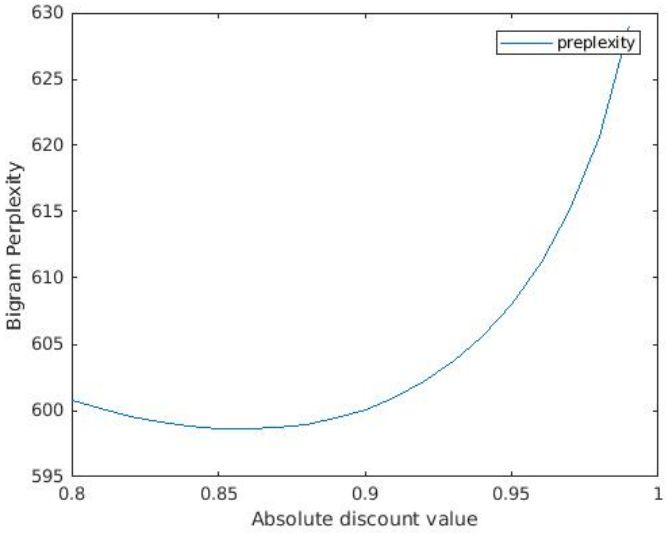
The plots were similar for all the other settings as well.

3.1 Sentence Generation

A sentence of length 10 words is generated from the best bigram model among all the setting. The best in this case being the S2 setting.
Example generated sentence : ”yankee for will them us thee mr for against but ”



Unigram Perplexity optimization



Bigram Perplexity optimization

4 Reference

Kenneth Heafield et al. : Scalable Modified Kneser-Ney Language Model Estimation