Assignment 1: Language Model

Suraj Panwar
MTech DESE
SR: 14644
surajpanwar@iisc.ac.in

1 Introduction

This document provides the methods used for creation of a language model for implementation of a automatic sentence generation system. The results obtained in the process are presented here and the inferences observed from them are also mentioned in the report. The language model implemented here is a Trigram model with Interpolated Kneser Ney smoothing used for unknown words and a simpler bigram model is also constructed based on the Kneser Net model for comparison.

The datasets used here are D1 (Gutenberg Dataset) and D2 (Brown Dataset) each consisting of multiple documents from various different authors. The datasets are separated into adequate training, development and testing subsets, and the performance of the model is calculated on combination of these subsets using perplexity as the parameter to compare the performance of the models created.

2 Separation of Datasets

The corpus used in this assignment are D1 (Brown Corpus) and D2 (Gutenberg Corpus) the separation of dataset is done as follows:

2.1 Brown Corpus

The Brown Corpus was first separated into the in built categories :

- Adventure
- Editorial
- Fiction
- Government
- Hobbies
- Humor

Each category was separately divided into training, development and test set, in the ratio of 60:20:20 and then the final training, development and test set were constructed by combining all the previous sets from each respective category.

2.2 Gutenberg Corpus

Gutenberg Corpus contains works of many renowned authors as separate files in the corpus. The division for this corpus has been done by separating the files in the ratio of 60:20:20 for training, development and test set and then appending the documents in each category to create a final training, development and test set.

The final sets thus are a combination of various documents randomly picked from the corpus and then merged into their respective sets.

3 Generation of N-Grams

As the language model constructed is a trigram model, the training, development, and datasets for both corpus have been separated into a unigram, bigram and a trigram dictionaries using the NLTK(Natural Language Tool Kit) library in python. The final generated N-Gram model s have been named as:

- unigram_guten_train
- bigram_guten_train
- trigram_guten_train
- unigram_brown_train
- bigram_brown_train
- trigram_brown_train

Similar dictionaries have also been made for development and test set for both the datasets following the same syntax shown above.

Perplexity	Bigram Model	Trigram Model
S1	3169.20	395.03
S2	3688.23	452.03
S3	4026.12	368.26
S4	4142.32	443.91

Table 1: Perplexity for various cases.

4 Language Model

The language model constructed for the this task is a trigram model with interpolated Kneser Ney smoothing for unknown words.

The model was selected on the basis of performance of the model (perplexity) on a given corpus. The performance of the Trigram model is compared in Table 1, and it can be seen that the performance of the model is much better than the corresponding Kneser Ney smoothing based Bigram model in all situations.

However, still a simpler Interpolated Kneser Ney smoothing based model has also been constructed to compare the performance with the more complex model using the perplexity parameter as can be seen in Table 1.

5 Results obtained

The perplexity was calculated for the test set for both the Trigram and the Bigram interpolated Kneser Ney smoothing based model. The perplexity was calculated for the following combinations:

5.1 S1: Train: D1-Train, Test: D1-Test

The brown corpus is a general collection news articles, sports articles, etc. and thus the vocabulary is quite common.

Table 1 shows that the vocabulary used in the brown corpus being common have higher counts for N-grams and hence, give a lower perplexity compared to a corpus such as Gutenberg corpus.

5.2 S2: Train: D2-Train, Test: D2-Test

The Gutenberg corpus is a collection of novels from classical English literature. As the writing styles of different authors are very different and due to rich and diverse vocabulary used in each document as can be seen in Table 1 that the perplexity is higher compared to a more general corpus such as Brown corpus.

5.3 S3: Train: D1-Train + D2-Train, Test: D1-Test

The dataset for training has been taken as the combined training set of D1 and D2 and the test is conducted on Brown corpus test set.

5.4 S4: Train: D1-Train + D2-Train, Test: D2-Test

The dataset for training has been taken as the combined training set of D1 and D2 and the test is conducted on Gutenberg corpus test set.

6 Task 2: Sentence Generation

The sentence generation has been conducted by using both Brown and Gutenberg corpus and the separate results are shown, the sentence naturalness increases with increase in the test corpus which holds the possible continuations for the given words.

6.1 Limitations

Due to the large corpus, the test complexity can not be increased by increasing the pool of possible words for the continuation of the previous words after a certain value as the computation time is increasing many folds for any increase in the pool of words.

Also, by implementing a bound on the probability of the trigram continuation selected the accuracy can be further increased but that would result in increase in computation time and hence, have been avoided here.

6.2 Results

The random sentence generator will generate a random sentence of token length 10 and will try to make it a complete stand alone sentence. The sentence can either be generated completely randomly or a seed bigram can be given in which the algorithm will try to generate the sentences in that context. A few example of sentences generated from the trigram model with Kneser Ney smoothing are given below:

- this is not to be made of wood and stone.
- this is not only the average per capita income.

The accuracy and naturalness of the sentences can be increased further if bounds on probability of the trigram selected were implemented, However, that results in a multi fold increase in the computation time and hence has been avoided here.

7 Git hub Resource

The code for both Perplexity calculator and the Sentence generator has been uploaded on git hub with a read me file containing the instructions.

The files have been uploaded at:
https://github.com/Suraj-Panwar/NLU_ASSIGNMENT_1.git