NGrams Experiment on MSR Data Collection

Mingzhi Yu

February 18, 2016

1 Experiment Overview

The Ngram experiment is designed to examine the Ngram language model through a sentence completion test. In the experiment, we are trying to find the most appropriate word to fill in the blank, which means we are trying to find the best word that will makes a sentence has the best perplexity according to our probability models. The goal of this experiment is to examine the performance of the different ngram model.

2 Data Collection

We are using the Microsoft Research Sentence Completion Challenge data set as our training and testing data. The details of the data set could be found from the challenge webpage. The training data

2.1 Training Data

The training corpus are from the Project Gutenberg, which are mostly classical novels. To have a larger corpus chose, 20 text are chosen.

2.2 Development and Test set

The testing set are 1040 fill in the blank questions. Among them 520 are used as development set and the rest 520 are used as the test set.

2.3 Language Models

In the experiment, the language models are built by off-the-shelf toolkits form SIRLM toolkits. The reason we are using it is because SIRLM provides more various options with smooth and language modeling.

In the experiment, 5 languages models are used. The models include unigram with laplace, bigram with laplace, trigram with laplace, trigram with laplace, interpolated bigram and interpolated trigram. The smooth constant is set to be 1. The lambdas are the threshold for unknown words are set according to the performance of the development file.

3 Experiment Design

3.1 Training data preprocessing

In order to have more accurate results, I preprocessed the training text. A text processing program has been included in the pacakge – sirlmProcessor. This program will take a standard .txt file as input. And the converted text file will be output to ‘output.txt‘.

How the sirlmProcessing processes the text:

1. Reading text by sentence and splitting the sentences: The sentence boundary is based on the punctuations, including: ”.”,”;”,”?”,”!”

2. Cleaning the white space tokenize(including ”/t”,”/s”,”/n”,”r”)

3. Removing unnecessary numeric numbers such as the number tagged for each paragraph.

3.2 Testing

For each of the question, there are 5 possible answers. For each of the answer, there is a corresponding sentence in the test file. The program will test each sentence and produce its perplexity according to the language model. We will choose the sentence with the lowest perplexity and the answer corresponding with that sentence will be regarded as the answer for that question.

3.3 Evaluation

The correct answer for each question has been known. Based on this, I will evaluate the accuracy as the performance of your model on the test set. Accuracy here means out of all the questions, how many did some model answer correctly. Formally, Accuracy= the number of correctly answered questions/ the number of questions

4 Result and Analysis

4.1 Chart

The figure demonstrates the statics of each models after tests.

Model	Accuracy
Unigram	23.2692307692308 %
Bigram	21.1538461538462 %
Trigram	21.3461538461538 %
Interpolated Bigram	23.0769230769231%
Interpolated Trigram	22.5 %

4.2 Analysis

From figure, the best performed model is unigram and then the interpolated 2 bigram. Among them, the bigram model smoothed with laplace has the lowest accuracy. All of the 5 models do not have very obvious differences. However, the performance of the both interpolated models are fair.

5 Conclusion

From the experiment, we see that generally the mixture models work better. However, because the training text has been preprocessed, the accuracy rate is related to the preprocessing rules. Also, because we mainly use classical novels as our training data, our accuracy rate is also related to that. In generally, even though we see a better performance of mixture models here, we still need to consider the factor such as the type and size of the training data and the text preprocessing.