CS 388 HW1: N-gram Language Models

Mit Shah (UT EID: mks3226)
University of Texas at Austin
Austin, TX 78712
mkshah@utexas.edu

Abstract

The assignment is aimed at learning usage of N-grams like bigrams, trigrams, etc. and their directional variants in language modeling. Most commonly used models are forward n-gram models, in which sentences are parsed from start of the sentence $< S >$ to end of the sentence $< /S >$. Though it works for English, in some languages it might be easier to predict n-grams in a backward manner, starting from $< /S >$ to $< S >$. Working in that direction, results are analyzed for forward and backward bigram models on mainly 3 data sets: Atis, Brown and WSJ. Also, both models are combined to produce a bidirectional bigram model, in which the probability of a word in the sentence is determined by interpolating those in forward and backward models.

1. Introduction

A n-grams model estimates the probability of a word given the prior context. It uses n-1 words of prior context. For e.g., a bigram model uses only 1 previous word to predict the current word, a trigram uses 2 previous words and so on.

Generally, direction of doing so is forward, from starting of the sentence $< S >$ to end of the sentence $< /S >$. However, it can be done in a backward manner also, i.e., from $< /S >$ to $< S >$. Motivation for doing so is that sometimes word dependencies can be better predicted in backward direction than forward direction. For English, in general forward direction performs well, as it is a head-initial specifier-initial language. Some languages might be head-final specifier-initial. For them, backward direction should be better in theory.

Now, if we have both these forward and backward models, we can easily use both of them to predict the probability of the current word: just interpolate probabilities of that word given by both the models. This results in a Bidirectional n-grams model.

2. Method and Implementation

Let's first get an overview of how forward bigram model is implemented. Then we will discuss how on top of that, Backward Bigram model and Bidirectional Bigram models are implemented.

2.1. Forward Bigram Model

Implementation is mainly divided into two parts: Training and Testing.

Training consists of two phases. In first phase, counts for Bigrams and Unigrams are accumulated in HashMaps. To handle out of vocabulary words, first occurrence of a word is replaced by token "UNK". Afterwards, values in Bigram HashMap is divided by Unigram counts of the first token in Bigram, so that we have the probability for Bigrams. Similarly, values in Unigram HashMap is divided by total count of tokens to get their probability.

During testing, probability and perplexity of each sentence is calculated in log order. To calculate sentence probability, it is traversed in forward direction and for each word, its Bigram and Unigram probabilities are interpolated with 0.9 and 0.1 weights respectively. Log probability of each word is added in order to get one for the sentence. Two variants of testing are implemented in which, one of them excludes predicting end of the sentence.

2.2. Backward Bigram Model

It was implemented by mainly applying 2 modifications in the code for forward bigram model. Before sending a sentence for training or testing, its words were reversed in order by using Collections.reverse utility of Java and restored to their original order after their training / testing. Second modification was to use $< S >$ instead of $< /S >$ while accumulating counts for Unigrams and Bigrams (training) and for predicting probability (testing, in both the variants) at the end of the sentences.

2.3. Bidirectional Bigram Model

Here, three HashMaps were used in total: 2 Bigram Hashmaps and 1 Unigram HashMap. One of the Bigram

Table 1. Training Perplexity

Model \Dataset	Atis	WSJ	Brown
Forward	9.0431	74.2679	93.5192
Backward	9.0129	74.2677	93.5091

Table 2. Testing Perplexity

Model \Dataset	Atis	WSJ	Brown
Forward	19.3413	219.7151	231.3024
Backward	19.3643	219.5198	231.2055

Table 3. Training Word Perplexity

Model \Dataset	Atis	WSJ	Brown
Forward	10.5919	88.8900	113.3595
Backward	11.6362	86.6602	110.7828
Bidirectional	6.4728	41.0779	52.6764

Table 4. Testing Word Perplexity

Model \Dataset	Atis	WSJ	Brown
Forward	24.0539	275.1178	310.6673
Backward	27.1613	266.3515	299.6857
Bidirectional	13.2819	128.4626	174.5357

HashMap was used to store the probabilities when traversed in forward manner and the other one was used to store the probabilities when traversed in backward manner.

While testing, log probability of a sentence was calculated in the following manner. For each word in the sentence, its forward probability was calculated by interpolating its Unigram probability and Bigram probability from first hashmap in a weighted manner. Then, its backward probability was calculated in the same manner, but by using second hashmap for Bigram. Average of its forward and backward probability was taken and converted into log scale. Finally, the log probability of the sentence was calculated by adding log probabilities of all of its words.

3. Dataset

Mainly three data sets were used for the evaluation purpose.

1) Atis: Airline Booking Query. Contained 577 sentences.

2) WSJ: Wall Street Journal. Contained 48,689 sentences.

3) Brown: Brown corpus of mixed-genre text. Contained 52452 sentences.

4. Experiments and Results

Datasets were divided into training and testing split by 90% and 10% respectively. For each data set; training perplexity, testing perplexity, training word perplexity and testing word perplexity were calculated.

Results are shown in tables 1 2 3 4.

4.1. Comparison between Forward and Backward Model

If we look at tables 1 2, we can see that there is hardly any difference in Training / Testing perplexities between forward and backward models. For all the three data sets, the difference is of almost of magnitude 0.01. So, we can say that at least for English language, both the models give almost similar results for training and testing perplexity.

Now, if we go through tables 3 4, we see increase in perplexities with respect to those in 1 2. As these tables refer to word perplexities, we can conclude that excluding prediction of the end of the sentences is resulting in increase in perplexity. This may suggest that models were pretty good at predicting end of the sentences.

Also, another notable change is that difference in word perplexities values of Forward and Backward models are also now of order of 1 to 10 instead of 0.01. It seems like for Atis forward models is performing better, while for Brown and WSJ, backward model is performing better. So relative performance of these models might be slightly dependent on the data sets, mainly its writing style and how majority of its sentences are structured.

4.2. Comparison of Bidirectional Model with Forward and Backward Models

Coming to bidirectional model, from tables 3 4 we can see that there is a huge improvement in word perplexities over both Forward and Backward models. Values are reduced to almost half in all the cases - all the data sets and during both training / testing. It suggests that on an average, a word in these data sets have similar amounts of dependency on both its left and right words. Some words might be better predicted from previous word, while some word might be from next word. Proportion of these words seems similar (as forward and backward models give similar values). So taking both the dependencies into account, reduces perplexity drastically.

5. Conclusion

This assignment discussed three different bigrams models: Forward (regular), Backward and Bidirectional. Results on three datasets - Atis, Brown and WSJ - were analyzed for comparison of these models. It was concluded that Forward and Backward models give similar results, but when you combine them and use a Bidirectional model, there is a huge improvement in results.