N-Gram Language Model

Nayan Singhal (ns28844)
University of Texas at Austin
Austin, TX 78712
nayans@utexas.edu

Abstract

Language model is the conditional distribution model that determines the probability of the word using all the previous words in the sentence. Generating a phrase or some combination of words is very important in Natural Language Processing especially while generating the text as output. It has application in various fields like Speech recognition, Optical Character Recognition, Machine Translation, Part of Speech Tagging , etc. N gram models are statistical model based on N-1 order Markov model i.e. it depends on N-1 previous words. In this work, we implement the backward bigram model and bidirectional bigram model using forward bigram model API and compare their performance on 3 dataset: axis, wsj, brown.

1. Introduction

The Language model is the probability distribution model that computes the probability of a sentence or sequence of words or the probability of the next word. To estimate the probability of the next word using all the previous words, we have to use the chain rule. But it's very hard to apply the chain rule because we will never see enough data for estimating the sentence. To simplify it, we have to use Markov assumption, which states that the current distribution of the current state is independent of all non-parents. With that assumption, the probability of the word depends only on the previous n words which is known as n-gram model.

In unigram model, the probability of the word depends only on its own, whereas in bigram model, the probability of the word depends on the previous word. Instead of developing a bigram model which depends on the previous word, it can also depend on the next word. In this work, we are trying to implement the backward model in which we calculate the probability of the word depends on the next word. And, in the end, we would like to implement the bidirectional model which is a combination of both models.

2. Methodology

We build our backward and bidirectional bigram model on top of the forward bigram model. To understand the implementation of both models, first we will explain the implementation of the forward bigram model.

2.1. Forward Bigram Model

The forward bigram model is implemented in two parts: training and testing. For training the forward model, it accumulates the unigram and bigram count for all the sentences in the training data. And, if the token seen for the first time, then we count it as an unknown token ($<$UNK $>$) to handle out-of-vocabulary items. It also accounts for the start and end of the sentence unigram and bigram. Then we compute the unigram and bigram probabilities from unigram and bigram counts.

For the testing of the forward model, sentences in the test data is used to evaluate the model. It accumulate the log probability of all the sentences in order to avoid underflow. And, it maintains the total sentence log probability as the sum of individual token log probabilities since adding logs is same as multiplying probabilities. To compute the log probability of token, we are using interpolated probability of unigram and bigram by giving equal weights to both models.

2.2. Backward Bigram Model

We didnt implement the backward bigram model, instead we built the model on top of the forward bigram model. We generalize the forward bigram model for both forward and backward model. For the backward model, sentences are reversed and startToken and endToken are reversed in order to calculate the bigram and unigram probability for both training and testing time.

2.3. Bidirectional Bigram Model

For the Bidirectional Bigram Model, we have taken an instance of the forward and backward model. We calculate the log probability for both forward and backward model

independently and calculate the result for the bidirectional bigram model by interpolating the values calculated using forward and backward bigram model. We tried a lot of weight for both the model, but we got good accuracy by giving equal weights to both the model. And it makes sense too, because both models have almost the same results.

3. Experiments

We train and test our model on LDC tagged files i.e. atis (airline booking query), wsj (Wall Street Journal text) and brown (Brown corpus of mixed-genre text) corpora using the first 90% of the sentences for training and the last 10% for testing. We calculate the perplexity and Word perplexity which exclude the prediction for the start or the end of the sentence.

3.1. Perplexity

For perplexity, both forward bigram and backward bigram model have same values as can be shown from Table 1 and Table 2 . This is happening because both are trying to approximate the actual probabilities. The actual probabilities is calculated using the chain rule without using Markov substitution.

Dataset	Forward	Backward
atis	9.0432	9.0129
wsj	74.2680	74.2678
brown	93.5193	93.5091

Table 1. Perplexity for both forward and backward model on training data.

Dataset	Forward	Backward
atis	19.3414	19.3644
wsj	219.7152	219.5198
brown	231.3024	231.2055

Table 2. Perplexity for both forward and backward model on testing data..

3.2. Word Perplexity

For Word perplexity, forward bigram model is slightly better than the backward bigram model for the smaller dataset atis. But, backward bigram model is better for the larger dataset, i.e. wsj and brown. This is happening because both are trying to approximate the actual probabilities. The actual probabilities are calculated using the chain rule without using Markov substitution.

Both forward and backward model have almost similar word perplexity but Bidirectional Bigram Model have values 50% lesser values than both the model as shown in Table 3 and Table 4. It is because though the model both have equal strength but they predicts the complementary results. The forward bigram model is trying to predict the next word using the the previous word and the backward bigram model is trying to predict the word using the word after the current word. Because of their combination effect, the bidirectional bigram model average out the results.

Dataset	Forward	Backward	Bi-Directional
atis	10.5919	11.6362	7.2352
wsj	88.8901	86.6603	46.5144
brown	113.3595	110.7829	61.4689

Table 3. Word Perplexity for forward, backward and Bidirectional model on training data.

Dataset	Forward	Backward	Bi-Directional
atis	24.0540	27.1614	12.7002
wsj	275.1178	266.3516	126.1131
brown	310.6674	299.6857	167.4871

Table 4. Word Perplexity for forward, backward and Bidirectional model on testing data.

From the word perplexity and perplexity values, we can see that perplexity values are lesser than the later one. The reason I think that it's very easy to predict the start and end of the sentence as compared to the normal words in the sentence.

There is one important point which we observe is that in all the three models, the word perplexity for the testing data is almost 3 times higher than the training data. It seems that though we have applied a smoothing method for bigram model by interpolating with the unigram model, but still the model is over-fit for the training data.

4. Conclusion

In this work, we explored the task of language modeling using the bigram model. We generalized the forward bigram model for both forward and backward model. We also implemented the bidirectional bigram model by keeping the instance of both forward and backward bigram model and calculated the word perplexity by giving equal weight to the word perplexity obtained from both models. The results of our experiments show that though we didnt get good Word perplexity for backward model which is expected, but we get good Word perplexity for the bidirectional bigram model because of the ensemble effect of both model.

To Future work, we would like to try some smoothing methods so that we can make our model robust from overfitting.