Gaurav Nanda	Homework #1
Natural Language Processing	February 16, 2014

1 Backgram Bigram Model

1.1 Implementation

Backward Training Approaches

There are two approaches that could have been used to train the Backgram Bigram Model.

1. Train sentences in the same order as the Forward Bigram Model, i.e. pick the first sentence and keep on moving forward till the end. However, while training individual sentence, read tokens in the reverse order.

2. Train sentences in the reverse order, i.e. pick the last sentence first and keep on going back till the first sentence. Also while training individual sentence, read tokens in the reverse order.

Intuitively, both of these approaches should yield the same result. However, there is a difference due to the approach we follow in handling the Out of Vocabulary words. Whenever, we encounter a new word, we replace that with <UNK>. Going left to right (L2R) or R2L, would not impact unigram probabilities, but would make a difference to the bigram probabilities.

I ran some tests and found that difference in word perplexity calculated by these two approaches in very less (<0.5%). **Therefore, I just decided to go with the simpler, first approach.**

Probability Calculation

To compute probabilities in the backward model, we simply reverse each sentence and calculate the sentence log probability using similar scheme to the forward model.

1.2 Results - Word Perplexity

While interpolating, we are using 10% unigram and 90% bigram contributions for both forward and backward models. For backward model, I have only printed "Word perplexity" (in trace files and report both) because that is the fairest way to compare these two models and also reduces noise information.

	ATIS		WSJ		BROWN	
	Train	Test	Train	Test	Train	Test
Forward Model	10.59	24.05	88.89	275.11	113.35	310.66
Backward Model	11.63	27.16	86.66	266.35	110.78	299.68

Table 1: Word perplexity for Forward and Backward Bigram Models.

1.3 Analysis

Our results show that Backward model is slightly better for WSJ and Brown datasets. However, forward model is slightly better for ATIS dataset.

These results were contrary to my expectations. I read this[1] paper, which talks about Bangla language giving better results with the backward model, because of the language structure. This led me thinking that English is syntactically ordered from left to right, therefore forward model should be atleast as good as the backward model. (Though, I am not sure that this statement is always correct.)

Investigating furhter, I calculated the count of unique tokens in the beginning ($< S >$ Token) and end (Token $< /S >$) of each sentence for all three datasets and found very interesting results.

	ATIS	WSJ	BROWN
Initial Unique Tokens	48	2899	2801
Termination Unique Tokens	120	256	8

Table 2: Count of unique start and termination tokens in each corpus.

Hypothesis: Datasets where count of unique termination tokens is lesser than unique starting tokens, Backward model seems to perform better than Forward Bigram model and vice-versa.

We also observe that for **Brown** corpus, all the sentence ending tokens seem to be one among some 8 tokens (Code that I have submitted still calculates number of termination tokens, however I don't print them to ensure cleaner trace files). Here is table of those tokens with their associated bigram probabilities $p(W| < /S >)$:

Word	Probability
$< /S >< UNK >$	6.35E-5
$< /S > Governor$	2.11E-5
$< /S > .$	0.94
$< /S > 3$	2.11E-5
$< /S >?$	0.04
$< /S >!$	0.0125
$< /S > ..$	0.0015
$< /S > insulated$	2.11E-5

Table 3: All the sentence termination tokens in Brown corpus with their bigram probabilities.

This table suggests that, most of the sentences in the corpus are ending with "." and we are considering DOT as a separate token. Therefore, when we test our backward Bigram model on

the test data, which also has DOT as the termination token for most of the sentences, it tends to assign high probability for DOT and overall probability of the sentence goes high.

On the other hand, in the forward model, DOT comes in the end of the sentence and DOT can follow a number of tokens, therfore its prediction probability would be lesser.

To test my hypothesis, I removed dots from the training and test data and results were as follows:

	ATIS		WSJ		BROWN	
	Train	Test	Train	Test	Train	Test
Forward Model	10.59	24.05	98.11	311.38	126.26	363.22
Backward Model	11.63	27.16	104.01	331.80	134.75	396.78

Table 4: Word perplexity for Forward and Backward Bigram Models after removing dots.

	ATIS	WSJ	BROWN
Initial Unique Tokens	48	2899	2801
Termination Unique Tokens	120	7085	9490

Table 5: Count of unique start and termination tokens in each corpus after removing dots.

For ATIS dataset, results are unchanged because there are no dots present at the end of sentences. However, for BROWN and WSJ corpus, there is drastic change in numbers. Now, the number of unique termination tokens have increased to a large extent. Therefore, the word perplexity of the backward model is higher (performance is worse) in comparison to forward model. These results support our initial hypothesis.

1.4 Code Execution

I have added another option to the code to remove dots "." from sentences.

Sample Execution:

// If you do not want to remove dots.
java nlp.lm.BackwardBigramModel ./pos/atis/ 0.1 false

// If you want to remove "." (dots).
java nlp.lm.BackwardBigramModel ./pos/atis/ 0.1 true

1.5 Conclusion

If we want to treat tokens like ".", "?", "!" etc as separate tokens, backward model seem to be behaving little better than the forward model. However, if we remove the sentence termination tokens, then forward model seems to be little better than the backward bigram model.

Overall, performance difference in forward and backward models is not huge.

2 Bidirectional Model

2.1 Implementation

This is implemented using already existing Forward and Backward Bigram models. Initially, both forward and backward models are trained individually. While testing, for each sentence we compute the bigram probabilities of every token using both forward and backward model. Then we find the final bigram probabilities by using forward and backward numbers in a fixed ratio.

We have to be little careful in this implementation because array of probabilities returned by Backward Bigram model is in the reverse order. Therefore, we have to properly handle array indices to ensure that we are picking up the corresponding numbers from two arrays.

2.2 Results - Word Perplexity

In the basic approach, we have taken contribution of both forward and backward model to be same 50

	ATIS		WSJ		BROWN	
	Train	Test	Train	Test	Train	Test
Forward Model	10.59	24.05	88.89	275.11	113.35	310.66
Backward Model	11.63	27.16	86.66	266.35	110.78	299.68
Bidirectional Model	7.23	12.7	46.51	126.11	61.46	167.48

Table 6: Word perplexity for Forward and Backward Bigram Models.

2.3 Analysis

Using bidirectional model, we see a huge increase in the performance. Primary reason for the increase is that we are looking both forward and backward of a token before assigning it a probability. Forward and backward information gives more context about the token and hence higher probability of predicting the token in a similar context. This is somehat simile to training your model over higher order models like trigram or 4-gram models.

Currently we are using (token-1 and token+1) to calculate probability of a token. I surmise a model using (token-2, token-1, token+1, token+2) would give even better performance. However, there is also a chance of overfitting with increasing order of bidirectional n-gram models.

Another issue is to find the optimal ratio of forward and backward models to be used. Word perplexity numbers computed by Forward and Backward are almost similar, therefore optimal

ratio is also very near to 50%. However, results are little skewed towards the model which has lesser value of perplexity.

For example, for Brown corpus,

A) If we don't remove Dots from the model, we expect the optimal number to little baised towards backward.

Forward Ratio	Word Perplexity (Training)	Word Perplexity (Test)
0	110.78	299.68
0.2	67.01	182.24
0.4	61.85	168.31
0.48	**61.45**	**167.39**
0.5	61.46	167.48
0.52	61.52	167.7
0.6	62.22	169.86
0.8	68.26	187.5
1	113.35	310.66

Table 7: Word perplexity for different ratios of Forward and Backward model

B) If we do remove Dots from the model, we expect the optimal number to little baised towards forward.

Forward Ratio	Word Perplexity (Training)	Word Perplexity (Test)
0	134.74	396.78
0.2	79.15	233.88
0.4	72.11	212.56
0.48	71.3	210.12
0.5	71.23	209.91
0.52	**71.21**	**209.86**
0.6	71.66	211.2
0.8	77.52	228.73
1	126.26	363.22

Table 8: Word perplexity for different ratios of Forward and Backward model

2.4 Code Execution

I have added another option to the code to remove dots "." from sentences and an option to give ratio of forward to backward model.

Sample Execution:

// If you do not want to remove dots and want to give 0.48 probability to forward model
java nlp.lm.BidirectionalBigramModel ./pos/atis/ 0.1 false 0.48

// If you want to remove "." (dots) and want to give 0.52 probability to forward model
java nlp.lm.BidirectionalBigramModel ./pos/atis/ 0.1 true 0.52

2.5 Conclusion

Bidirectional model performs better than forward or backward model. We get optimal results when the contribution of both of the models is nearly same or slightly skewed towards the one having lower "Word perplexity".

References

[1] Naira Khan, MdṪarek Habib. "History (Forward N-Gram) or Future (Backward N-Gram)? Which model to consider for N-gram Analysis in Bangla?" http://www.bracuniversity.net/research/crblp/papers/BAN04.pdf