CS 388 — Homework 1 — Due: Feb 17th 2014

Madhura Parikh
mparikh@cs.utexas.edu

1 Problem statement

In this assignment we explore two other variations of the conventional *n-gram language model* and analyze how they compare with each other. Specifically we shall be using an fixed-weight interpolated bigram language model, with a unigram model, in our experiments. Thus in this discussion we will be looking at the following three variations of the classic n-gram model:

Forward model: This is the standard n-gram language model. As mentioned previously, in our case this uses a fixed weight interpolation of a bigram model with a unigram model.

Backward model: This is identical to the forward model, except that it models sentence generation from right to left.

Bidirectional model: This is simply a combination of the forward and backward models.

2 Methodology

2.1 Data

For training and testing our language models, we use sentence segmented data from the Linguistic Data Consortium's Penn Treebank collection. We use three different datasets - airline booking queries data *(atis)*, Wall Street Journal text*(wsj)* and the Brown corpus *(brown)* - having sizes as summarized in the table 1 below

2.2 Training and evaluation

We used a training-test split of 9:1 for each of the above datasets. For evaluating the performance on the test-data, we used two metrics: *perplexity* and *word-perplexity*. To make the modeling easier, every sentence begins with a special *start-of-sentence* token (<S>) and ends with a special *end-of-sentence* token (</S>). The forward and backward model differ in that one predicts the sentence-end and the other the sentence-start, and thus the perplexity score may not be a fair comparison of the two models. We therefore introduce the word-perplexity metric that ignores sentence boundary prediction, affording a better comparison between the models. For the bidirectional model, we only use the word-perplexity metric for evaluation.

	atis	wsj	brown
#sentences	577	48,689	52,452
#words	4,353	1,107,344	1,172,970

Table 1: Summary of the data used.

2.3 Some implementation details

The bigram model is interpolated with a unigram model for smoothing with a weight = 0.9 for the bigram model and 0.1 for the unigram model. Standard techniques such as a special token <UNK> to deal with OOV words and computation of probabilities and perplexities in the log-space to avoid underflow are also utilized. The code is written in Java. Given the forward model class, we create the backward model class simply by inheriting from the forward model class. For any method in the parent class that takes the sentence data as an argument, we call the superclass methods, only this time passing the reversed sentence data as the argument. Similarly to create the bidirectional model, we design a new bidirectional class, that has the forward and backward model class objects as its members. The only methods we change significantly here are the `sentenceLogProb2` and `interpolatedProb`, to implement the combination of both the forward and the backward model. More details can be gleaned from the source code.

2.4 Choosing an interpolation weight for the Bidirectional model

To decide what weight to assign the forward(λ_f) and the backward model($\lambda_b = 1-\lambda_f$), we performed an experiment that computed the word-perplexities at different values of λ_f, starting from $\lambda_f = 0.1$ upto $\lambda_f = 0.9$ incremented in steps of 0.1. For these experiments we split the 90% training set further into a 70% training set and a 20% development set - with which to tune the weights. The test-data was thus held out and not used in this experiment. As the figure 1 shows, the development set gave the best results when both the models were weighted equally, i.e($\lambda_f = \lambda_b = 0.5$).

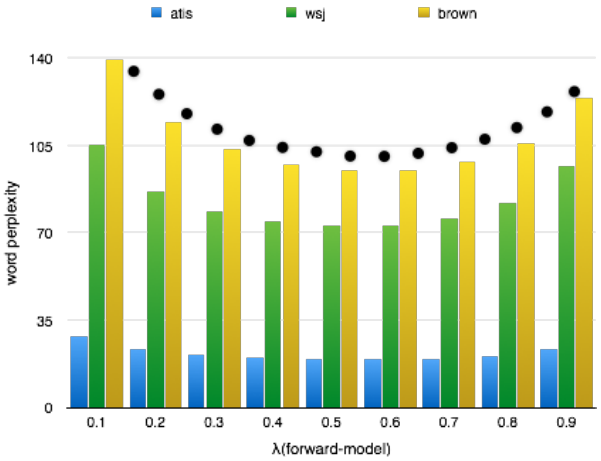

Figure 1: Determining the best weights for the forward and backward model in the bidirectional model.

3 Results and analysis

In the tables 2 and 3 below, we report the perplexity(where applicable) and word-perplexity for all the three models on both the training and test set. We also depict these results graphically in figure 2.

	forward model		backward model		bidirectional model	
	perplexity	word-perplexity	perplexity	word-perplexity	perplexity	word-perplexity
atis	9.04	10.59	9.01	11.63	—	7.01
wsj	74.26	88.89	74.26	86.66	—	35.19
brown	93.51	113.35	93.50	110.78	—	41.36

Table 2: Performance on training set

	forward model		backward model		bidirectional model	
	perplexity	word-perplexity	perplexity	word-perplexity	perplexity	word-perplexity
atis	19.34	24.05	19.36	27.16	—	9.85
wsj	219.71	275.11	219.51	266.35	—	69.01
brown	231.30	310.66	231.20	299.68	—	81.38

Table 3: Performance on test set

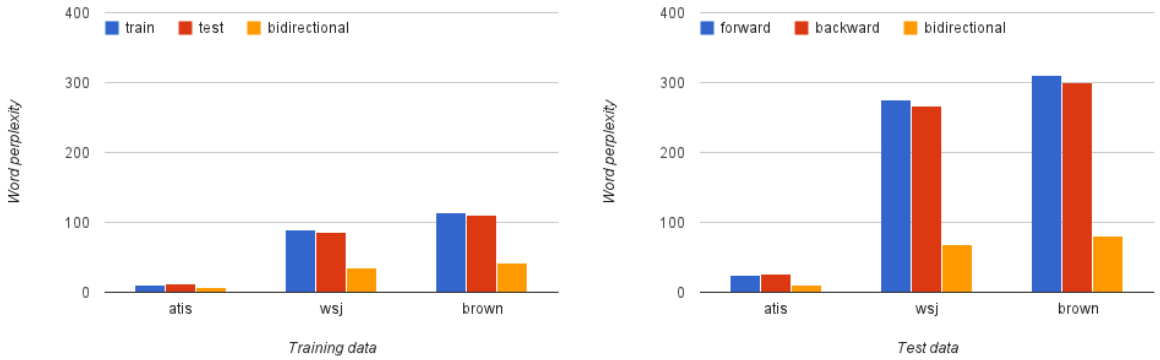

Figure 2: Performance of different models on training and test set

From just a cursory glance, it is clear that while both the forward model and the backward model perform nearly the same on both the test and training data, the bidirectional model yields dramatic improvement over both the models taken individually. Now we provide some concrete results and discuss them, we mainly use the word-perplexity metric for these discussions.

3.1 Forward model vs. Backward model

From the tables above, we see that for the `atis` data the forward model is about 10% better than the backward model for the training set, and around 13% better on the test set. For the `wsj` data, the backward model has a slight improvement of around 2.5% on the training data and around 3% on the test data, when compared to the forward model. Similarly on the brown corpus, the backward model has a minor improvement of 2.2% on the training data and 3.5% on the test data.

One reason that the forward model does better on the `atis` data is probably because of the nature of the data. Here we are dealing with airline queries - thus the domain has a very limited vocabulary and very similar style of sentences that is already captured very well by the forward model. For instance many sentences contain phrases of the form `eleven a.m` or `seven p.m.` Here given a numerical word like `seven`, the probability of `p.m.` or `a.m.` in the forward bigram model is very large(since these two are the most likely possibilities) whereas in the backward bigram, given `a.m.`, there would be 12 possibilities for the following word. By contrast in a more flexible domains such as `brown` and `wsj` with a larger vocabulary, the models seem to do nearly the same.

3.2 Bidirectional model vs Forward and Backward model

We see that the bidirectional model does significantly better than both the forward and backward models, outperforming each by about the same amount. Below we tabulate the exact figures (table 4) Again we observe that for the `atis` data, the gains are not as significant, because of the controlled nature of the data and the small vocabulary, which is already modelled fairly well by the individual models. On the other hand we get a significant improvement in the range of 60%-70% for the other datasets. The performance improvement on the training set is slightly

	bidirectional vs. forward		bidirectional vs. backward	
	training set	test set	training set	test set
atis	33.8%	59%	39.7%	63.7%
wsj	60.4%	74.9%	59.3%	74%
brown	63.5%	73.8%	62.6%	72.8%

Table 4: Percentage improvement of the bidirectional model compared to forward model and backward model

less as compared to the test set for both the forward and the backward models, because the models are already doing well on the training data. The most interesting point though is - why is there such a significant benefit when the two models are combined, far overshooting their individual performance? This can be explained if we think about the effect of interpolating the forward and backward model. Intuitively, when we combine the two models in this way, we are essentially increasing the size of the language model for the same amount of data - in the best case (when the bigrams for the forward and the backward model are disjoint) - we would be effectively doubling the number of bigrams for the same quantity of data. This should obviously mean that the bidirectional model would be much 'less perplexed' whenever it encounters a new sentence, thus explaining the large improvement.

4 Conclusion

From our experiments, we conclude that the forward and backward models do nearly the same in more flexible and large vocabulary domains, while in a controlled and small vocabulary domain, the forward model is slightly better off. The bidirectional model shows an impressive performance gain when compared to the individual models, because it effectively increases the language model size for the same amount of data and the same n (for bigrams $n = 2$).