Language Models
NLP Report 1

Harshal Priyadarshi
UTEid - hp7325

February 18, 2016

1 Language Models

N-gram model is one of the conventional techniques to create a model that best identifies if the given sentence belongs to the corpus trained upon. A general N-gram model invovlves calculating the probability of occurance of sentance by finding product of conditional probabilities of all the words in the sentence conditioned on the previous $N-1$ words. This, more specifically is called a Forward N-gram model. If it is conditioned on the next $N-1$ words, then it is called Backward N-gram model. However if it is conditioned on the mix of previous and next words(total adding to $N-1$), it is called Bidirectional N-gram model.
As higher the N, in the N-gram, higher the number of data (exponential in N) required to obtain an efficient model. Since our data is relatively small, a bigram model, would be the optimal model. Thus 3 models chosen for proof of concept were:

- Forward Bigram Model
- Backward Bigram Model
- Bidirectional Bigram Model

2 Forward vs Backward Bigram Model

Perplexity is an implicit criterion to check the fitting of the data to the model. Here 2 notions of perplexity is used - **perplexity** and **word perplexity**. Perplexity includes conditioning over end-sentence(or start-sentence) markers for a forward(or backward) bigram model, while Word perplexity does not. Thus Word perplexity, is a more robust for both forward and backward case, as its definition remains the same for both traversal.
The backward bigram model was implemented using the same forward bigram model, with the reverted sentence. This essentially reverses the conditioning on the bigram probabilities from previous to next tokens. The smoothing factor of $1:9$ on unigram and bigram probabilities remains the same as Forward model. However, the $< UNK >$ parameter was now created using the last occurance of word, due to this reversal. This is the only point of difference, which led to slight improvement in the **Perplexity** for test data as seen in **Table 2** (log scale). But this improvement is negligible, of the order of $0.1\% - 0.0001\%$. This justifies the fact, that perplexity for both model execution is essentially the same.

However, the **Word Perplexity** showed a significant improvement (in log scale), ranging from $2-4\%$ for test data, as seen in Table 1. In English grammar, a right-branching sentence is a sentence in which the main subject of the sentence is described first, and is followed by a sequence of modifiers that provide additional information about the subject. For instance,in 2 sentences, **He eats**, and **I eats**, given, the modifier **eats** the 1^{st} sentence is the preferred usage. Thus the succeeding modifier provides useful information about the preceeding word. This information is captured by backward bigram model, hence the better performance of backward model implies that the corpus has more right-branching sentences. Since this is the case with huge corpus like Brown and WSJ, we can state a general fact that English has more right-branching.

3 Bidirectional Model

On incorporating the above two models into a single bidirectional model, where present token's probability depends on the previous and next token, we should get a better **Per-**

Atis	Forward	Backward
train	10.5919	11.6362
test	24.0539	27.1613
Brown	Forward	Backward
train	113.3595	110.7828
test	310.6673	299.6857
WSJ	Forward	Backward
train	88.8900	86.6602
test	275.1178	266.3515

Table 1: Word Perplexity Results

Atis	Forward	Backward
train	9.0431	9.0129
test	19.3413	19.3643
Brown	Forward	Backward
train	93.5192	93.5091
test	231.3024	231.2055
WSJ	Forward	Backward
train	74.2679	74.2677
test	219.7151	219.5198

Table 2: Perplexity Results

plexity and **Word Perplexity**, as now we have more information from modifiers that occur before/after the modified token. This is indeed what we get as shown in the **Table 3**. In first look this might look similar to a trigram model, as conditioning is done on 2 words. However, there is a subtle, but important difference. Trigram model, even if we assume that the conditioning is obtained on the center token, calculates the probability of exact occurance of the trigram $< prev, present, next >$, which can be sparse. However, the Bidirectional bigram model interpolates the occurance of the tokens $< prev, present >$ and $< present, next >$. Thus the sparse nature of occurace of exact 3 desired tokens in trigram is prevented by using the two bigram models which are less constrained and thus less sparse. The perplexities of bidirectional model is calculated using the previous two trained model's bigram and unigram using the following equation, for $\beta_1 = 0.1$, $\beta_2 = 0.9$ and $\lambda = 0.5$

$$p_{bidirectional} = \beta_1 p_{unigram} + \beta_2(\lambda p_{bigram\ \ forward} + (1-\lambda)p_{bigram\ backward}) \tag{1}$$

Atis	Forward	Backward	Bidirectiona
train	10.5919	11.6362	7.2351
test	24.0539	27.1613	12.7002
Brown	Forward	Backward	Bidirectiona
train	113.3595	110.7828	61.4688
test	310.6673	299.6857	167.4871
WSJ	Forward	Backward	Bidirectiona
train	88.8900	86.6602	46.5144
test	275.1178	266.3515	126.1131

Table 3: Word Perplexity Comparison

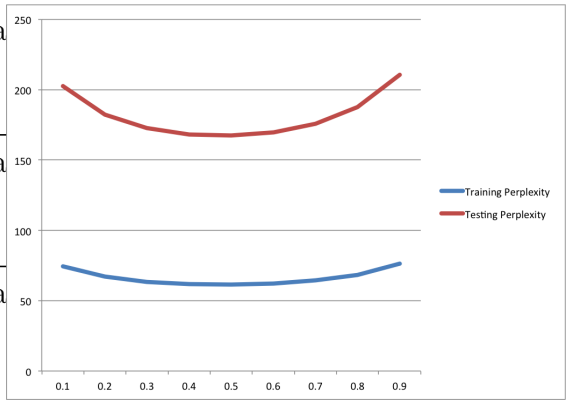

Table 4: Word Perplexity vs Forward Ratio

As expected, due to more information for a given token, the bidirectional model has achieved a higher fitting, and thus significantly lower perplexity in comparison to its parent models. An **interesting observation** though is that, the model performs best when both models are equally weighted, as can be seen in **Table 4**. This is justified, as English, is not prominently right-branching or left-branching. It is empirically verified by almost similar perplexity for both forward and backward models. Thus weighing one model more than other, will lead to loss of overall information stored in the modifiers captured by the parent models.