HW1: N-gram Language Models

Pratyush Kar (UT EID: pk8498)

February 21, 2018

Abstract

Language models are probabilistic models that assign a probability to a sentence of being part of that language. N-grams are a kind of language models that give probability $P(w_n|w_1^{n-1})$ for a word w_n given the context $w_1...w_{n-1}$. Traditionally, n-gram models are computed in the forward direction i.e. the left $n-1$ words are used to assign the probability of w_n. In this experiment we compare forward bigram models to backward and bidirectional bigram models on 3 different datasets (of varying complexities)– Atis, WSJ and Brown.

1 Implementation

Based on the given `BigramModel [BG]` class we can implement the `BackwardBigramModel [BBG]` class with just a few minor modifications to the code. Each sentence that is part of the dataset is reversed (word by word) and passed to the backward bigram model for both training and testing purposes. So now the backward bigram model assigns probability $P(w_n|w_{n+1})$ to the word w_n based on the word that comes after it in the sentence (w_{n+1}). We interchange the `<S>` with `</S>` tokens and vice-versa because the backward model generates the probability of the sentence right-to-left. So, the model starts with the token `</S>` and keeps generating words (w_n) till it generates the token `<S>` which marks the beginning of the sentence. We use the same policy of assigning the first occurrence of a token with `<UNK>` to handle out-of-vocabulary (OOV) words and the same linear interpolation method as the forward bigram model.

For the `BidirectionalBigramModel [BDBG]` we create a class with a `private` instance of both `BigramModel` and `BackwardBigramModel`. During the train process both these models are trained independently. Both classes `BG` and `BBG` have a function called `sentenceTokenProbs` which returns a log probability of each word in the sentence along with `<S>` or `</S>` as a `double` array. We linearly interpolate these values based on parameters λ_1 and λ_2, where $\lambda_1 + \lambda_2 = 1$, to generate the final probability of the word w_n based on equation **??**. The tokens `<S>` and `</S>` are skipped for this computation. We use a value of 0.5 for both λ_1 and λ_2, which basically means that equal weights are given to both `BG` and `BBG` for probability calculation.

$$P_{\texttt{BDBG}}(w_n|w_{n-1}, w_{n+1}) = \lambda_1 * P_{\texttt{BG}}(w_n|w_{n-1}) + \lambda_2 * P_{\texttt{BBG}}(w_n|w_{n+1}) \tag{1}$$

2 Evaluation Metrics and Dataset

Perplexity is a measure of the probability that the models assigns to the corpus normalised for the number of words in the corpus. It is important to note that `BG` and `BBG` have differing definition for perplexity, one considers `<S>` token while the other considers `</S>` token while computing perplexity. For this reason perplexity cannot be used to compare the models directly. To account for this a new measure is introduced called word perplexity which is the same as perplexity but ignores the `<S>` and `</S>` tokens.

Each of the 3 models `BG`, `BBG` and `BDBG` are run on three different datasets – Atis, WSJ and Brown. For all three cases the dataset is split into a training set and a test set based on

a user-defined parameter $split_{frac}$ which denotes the percentage of data to be considered for testing. For our implementation we set 0.1 as the value for $split_{frac}$.

3 Results

3.1 Forward vs. Backward Bigram Model

English is considered to be a strongly head-initial language[1] which means that in each syntactic phrase the head of the phrase precedes its complements. Given this result, our initial hypothesis was that `BBG` model would perform better than `BG` model since, it is easier to predict the head of the phrase if we are given its complements. The word perplexity results (given in Table **??**) for WSJ and Brown datasets agree with this hypothesis. But, the same measure for the Atis dataset show that `BG` performs slightly better than the `BBG` model.

An example from Atis dataset:
`[ The return flight ] should leave at around seven.`

An example from WSJ dataset:
`[ Commonwealth Edison ] now faces [ an additional court-ordered refund ] on [ its summer/winter rate differential collections ] that [ the Illinois Appellate Court ] has estimated at [ 140 million ].`

Penn Treebank dataset consists of both POS tagged and syntactically bracketed versions of the text[2]. If we look closely at the sentence structure of Atis dataset and compare that to the sentence structure of the WSJ or Brown dataset we can easily see that the syntactic phrase structure of WSJ (or Brown) is more complex than Atis. For example, predicting `<Illinois>` after `<the>` is much harder for `BG` than predicting `<the>` given `<Illinois>` for `BBG`. Hence, we can utilise the richer syntactic structure in WSJ (or Brown) while training the `BBG` model but, not in the case of Atis. This explains the observed discrepancy in the results.

3.2 Forward & Backward vs. Bidirectional Bigram Model

Based on the results (in Table **??**) it is evident that the `BDBG` model performs significantly better than both `BG` and `BBG`. This was expected because the `BDBG` models a "sort-of" *trigram* model without the inherent drawbacks of a trigram model. Trigram models generally suffer from extremely sparse data and need lots of data to be useful. By combining both `BG` and `BBG` models we are able to utilise the context on both the sides of the word (w_n) and hence, get a better word perplexity value across the board.

Table 1: Perplexity and word perplexity results for `BigramModel [BG]`, `BackwardBigramModel [BBG]` and `BidirectionalBigramModel [BDBG]`

Dataset $split_{frac} = 0.1$		Perplexity		Word Perplexity		
		`BG`	`BBG`	`BG`	`BBG`	`BDBG`
Atis	train	9.043e+00	9.013e+00	1.059e+01	1.164e+01	**7.235e+00**
	test	1.934e+01	1.936e+01	2.405e+01	2.716e+01	**1.270e+01**
WSJ	train	7.427e+01	7.427e+01	8.889e+01	8.666e+01	**4.651e+01**
	test	2.197e+02	2.195e+02	2.751e+02	2.664e+02	**1.261e+02**
Brown	train	9.352e+01	9.351e+01	1.134e+02	1.108e+02	**6.147e+01**
	test	2.313e+02	2.312e+02	3.107e+02	2.997e+02	**1.675e+02**

[1] `https://en.wikipedia.org/wiki/Head-directionality_parameter`.
[2] Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn Treebank: An Overview.