Language Model

1 Introduction

I have build a Language Model which predicts the accuracy of a sentence. I have implemented the stupid back-off method to build the model. I have used perplexity as an evaluation metric. This model was applied on 2 corpus, brown and gutenberg. I have taken 1) 70% of each document to test set and 30% for training. 2) 85% of each document to test set and 15% for training Implementation details

1.1 preprocessing

: I have extracted all the words(tokens) from the corpus. Then I have removed unwanted tokens like spacial characters,tabs,newlines etc. Then I have used inbuilt lemmatiser to lemmatize the words.

1.2 Training

I am using dictionary data structure to keep track of the n gram counts. I have taken 1,2,3 and 4 gram.Given a sentence I am taking n continuous words, combining them and keeping track of there count.Finally I am creating a list of all these 4 dictionary.

1.3 Testing

For each sentence I am calculating the probability of each word w_n using the formula

$$p(w_n|w_1 \dots w_{n-1}) = \frac{count(w_1, w_2, \dots, w_n)}{count(w_1, w_2, \dots, w_{n-1})} \tag{1}$$

If the count of $w_1 \dots w_n$ is zero then I have to back track to see if $w_2 \dots w_n$ is not zero. Hence in general the new formula is given by

$$p(w_n|w_1 \dots w_{n-1}) = \frac{count(w_i, w_{i+1}, \dots, w_n)}{count(w_i, w_{i+1}, \dots, w_{n-1})} \tag{2}$$

where,$count(w_i, w_{i+1}, \dots, w_n)$ is the largest gram for which the count is not zero. Finally I calculate the probability of the sentence by using the formula

$$p(w_1 \dots w_n) = log(p(w_1)) + \dots + log(p(w_n)) \tag{3}$$

The perplexity is given by

$$p = \exp\left(-\frac{log(p(w_1)) + \dots + log(p(w_n))}{n}\right) \tag{4}$$

where n is the number of words in the sentence. I am adding the perplexity for all the sentences over all the documents and averaging it over the number of documents.

2 Sentence Generation

First I have sorted my 4 gram dictionary according to there counts. Now I randomly pick a 4 gram from the top ten 4 grams. Then I predict the next word using the same stupid back off strategy. After generating $w_1, ..w_4$ I will pick the 4 gram count of all the words whose prefixes are $w_2..w_4$. I will sort this new list of all 4 grams and pick one randomly form top 10.If I don't find any such 4 gram then I will back track to trigram,bigram and uni gram. Finally I get a gram $w_i...w_4, w_5$ so w_5 is my next predicted word. Similarly I iterate over these steps 6 times to get a sentence of 10 words.

3 Results

3.1 Brown Data set

	perplexity	
train-data	85%	70%
2-gram	443.77	437.72
3-gram	407.56	403.187
4-gram	393.45	389.73

3.2 gutenberg Data Set

	perplexity	
train-data	70%	85%
2-gram	174.26	228.61
3-gram	159.08	140.26
4-gram	152.71	113.79

Some of the sentences generated by my model

1)the word of the faith forget the face of jesus 2)on the basis of their aesthetic creed and the will 3)in the united states by the leading saline water processes 4)at the end of falling and closeup and then shoot