Natural Language Understanding
Assignment-1
Language Model Implementation

Karan Malhotra
M.Tech (Systems Engineering)
kmkaran212@gmail.com

1 Introduction

Language Models are the models that assigns probability to sequence of words by assigning probabilities to each possible next word.These models have variety of applications such as in speech recognition, Machine Translation , Handwriting recognition etc.

For a Given corpus with a number of sentences with vocabulary as **V**, the probability for sequence of words $(w_1, w_2, ..., w_n)$ can be estimated as

$$P(w_1, w_2, ..., w_n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

where $w_1^{k-1} = (w_1, w_2, ..., w_{k-1})$ the last $K-1$ history of words.

The approach of N-gram is that the probability of word given its entire history can be replaced by probability with the history of last few words. When last one word is used as history then the model is termed as Bigram and when last 2 words are used as history then model is Trigram. In general a N-gram model uses N-1 last words as history for a particular word.

2 DataSets

The two different Text Corpus provided were:

- D1:Brown Corpus
- D2:Gutenberg Corpus

D1-Brown Corpus consists of 500 texts, each consisting of just over 2,000 words. The texts were sampled from 15 different text categories.

D2-Gutenberg Corpus is a collection of 3,036 English books written by 142 authors which contains of around 2621613 of tokens and around 50k unique words.

According to the given guidelines for the task the given corpus data is divided in the manner as follows-

- S1:Train: D1-Train, Test: D1-Test
- S2:Train: D2-Train, Test: D2-Test
- S3:Train: D1-Train + D2-Train, Test: D1-Test
- S4:Train: D1-Train + D2-Train, Test: D2-Test

For Both of the text Corpus, the train test split was done in ratio of 9:1(train:test). Moreover the train set obtained was again split into ratio of 9:1(train:held) to obtain a held out data set for tuning certain hyper-parameters for the built language models.

3 Language Model Implementation

The three types of Language models implemented in this task are :
1. **Bigram Model with Kneser Ney smoothing.**
2. **Bigram Model with Katz Backoff .**
3. **Trigram Model with Stupid Backoff.**

The various steps included in implementation of these models are:
1.Insertion of beginning sentence marker and end sentence marker in all sentences of the the train and test data.
2.Splitting of train set to obtain held out dataset and then tokenizing the effective train set.
3.Now finding unigrams and their respective counts which will also give the Vocabulary **V**
4.5500 unigrams with count as 1 are replaced with UNK to overcome the close vocabulary problem.
5.Fnding all the possible bigrams and trigrams with their respective counts.
6.All the models are being trained according to their probability formulation with the bigrams and trigrams respectively.
7.The discount factor in case of Kneser Ney

smoothing and Katz backoff is tuned by using held out data. The value of discount which gives lowest perplexity on held out data is chosen from a subsets of values provided.
8.After Training steps the test data is tokenized and then the perplexity of the tokenized test data is calculated for all the cases.
9. The three models built are now compared with respect to their perplexity on test data for all 4 cases of train and test split and the observations are being recorded in a table provided in the later section of this report.

4 Random Sentence Generation

The **Trigram Model** is used to generate Random sentence as it gives the lowest perplexity values for the provided data corpus(results shown in next section).The steps followed to generate random sentecne are:
Step1. Some words have been picked from Vocabulary and randomly one word is selected form these words to start the sentence.
Step2.The trigrams for the the word are seen and a trigram is randomly picked from it.
Step3.A uniform random variable is generated between 0 and 1 and if the probability of that trigram is greater than the random variable then the trigram is selected and otherwise the step 2 is repeated again.
Step4. Now the word is inserted into the sentence and the last word of the selected trigram is chosen as a new word of importance and full procedure from step2 is repeated till the sentence for 10 words is not generated.
Step5. All these above steps are repeated to generate till a sentence is generated such that the last token of the sentence comes out to be a period(.).

Some of Random generated sentences are:-
1. ”His speech shows what is Author of this connotation.”
2. ”This machine will be any change of her profession .”
3. ”An argument with a profound and more distant locations.”
4. ”’An institutionalized liberalism , not apt to follow it .”

5 Evaluation of Language Model

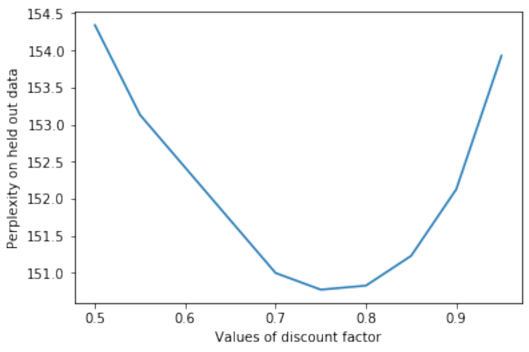
The two types of Evaluation metrics for Language Model are intrinsic and extrinsic evaluation metrics.An intrinsic evaluation metric is one that measures the quality of a model independent of any application whereas extrinsic needs an application so that model could be embedded in it. For this task an intrinsic measure called **Perplexity** (sometimes called PP for short) is chosen. Perplexity of a Language model on test set is the inverse probability of the test set, normalized by the number of words. For a given test set W perplexity is defined as:

PP(W)= P(w1,w2,w3....wN)$^{-1/N}$

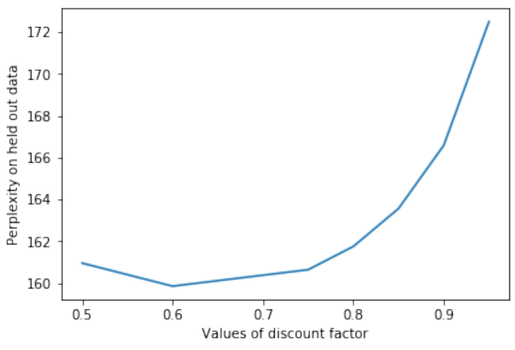
where N is the number of tokens with including end of sentence marker but not beginning of sentence marker.

Perplexity measure on the held out set is utilized to find the optimal discount factor used for Kneyser Ney smoothing and Katz backoff. The following plots shows the variation of perplexity with respect to discount factor for both the smoothing techniques in the case of S2 Dataset in which train and test both are from Gutenberg corpus(D2).

Bigram(Knesyer Ney smoothing):



Bigram(Katz Backoff):



The discounts value which give minimum perplexity on the held out data is selected and the models are trained on the respective discount values selected.

Performance on Test Data :

Model type	**S1**	**S2**	**S3**	**S4**
Bigram (Kneyser Ney)	277	150	364	163
Bigram (Katz Bckoff)	298	159	399	170
Trigram (Stupid Back-off)	202	91	257	99

T1: Table to show Perplexity values

From the above table it is observed that the Trigram model with Stupid back-off obtained the lowest perplexity for the test sets and the reason behind this is the fact that Trigram Model gives more information about a particular word compared to bigram model and that leads to lower perplexity. But as it is known that a lower perplexity value doesn't guarantees an improvement in any specific language processing task, So for saying a model better than the other one, an extrinsic evaluation is required.

6 Conclusion

The N-gram language models were implemented and trained on the given data corpus and the model with lowest perplexity came out to be a Trigram Model witth backoff technique.Various smoothing techniques and the tuning of the discount factor for them was a crucial step in building of the models.

As stated earlier a lower perplexity score of a model never guarantees a better performance on a Language processing task, therefore the extrinsic evaluation of the model is also a important factor in determining the best model among the given . The generation of random sentence from the best model which was Trigram model gave sentences as ouptut which clearly shows the extent of the context info a trigram model represents for a particular word.

7 Github Link

The code for these models are uploaded in the github page:
https://github.com/karanMal/NLU