Assignment Report

Anshuman Mourya
Computer Science and Automation
Indian Institute of Science , Bangalore
anshumanm@iisc.ac.in

Abstract

To designed a language model on **Brown**(D1) corpus and **Gutenberg**(D2) corpus using ngrams. Following are two tasks that we performed -

- **Task 1:** Divide both dataset into train, dev, and test and build the best LM in the following settings and evaluate.
  - **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
- **Task 2:** Generate few sentences of 10 tokens.

1 Dataset Split

Dataset Split is performed in ration 80:20 (train:test).The Test data is further divided into 50:50 (test:dev) split.

2 Preprocessing

- Replacing unigrams in train data with frequency less than two by UNK in order to handle out-of-vocabulary words in test data.
- Appending two START marker at beginnning of every line and one STOP marker at end of every line of both train and test data.
- Converting every word to lowercase for language model task.
- Replacing out-of-vocabulary words in test data with UNK.

3 Implementation

3.1 Language Model Used

I tried implementing two language models as listed :

- **trigram with stupid backoff** : Tried this model on all four datasets. Whenever Trigram is absent ,backoff to bigram and whenever bigram is absent , backoff to unigram.
- **ngram with parameter tuned interpolation** : This is the main model implemented. Development set is used here for validation. This is implemented on all four datasets. Interpolation is done on the basis of trigram, bigram and unigram. Hyperparamter are tuned in range [0.1,0.9].

3.2 Sentence Generation

Implemented using trigram model. Used two START marker to predict the the next word. Used the second START marker and generated word to predict the next word and so on. Sentence of ten tokens are generated in this way.

4 Result

4.1 Task 1

For Simple trigram-backoff - Train: 80per Test: 20per
For Interpolation with tuned parameters - Train: 60per Dev:20per Test: 20per

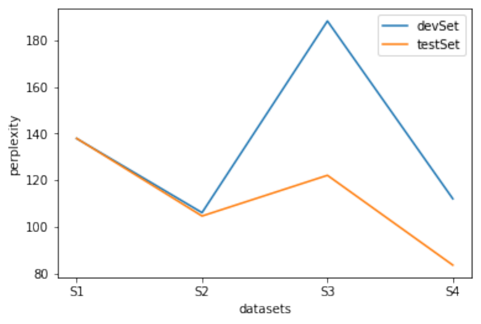

Figure 1: Perplexity for Interpolation

S1 : train = D1 train and test = D1 test	
Simple trigram backoff	368.12
Interpolation	137.37
S2 : train = D2 train and test = D2 test	
Simple trigram backoff	241.22
Interpolation	104.59 (lamb=0.6)
S3 : train = D1 train+D2 train and test = D1 test	
Simple trigram backoff	451.52
Interpolation	122.28
S4 : train = D1 train+D2 train and test = D2 test	
Simple trigram backoff	249.81
Interpolation	83.56

Above results are consistence with literature i.e. Interpolation gives better results than simple(stupid!) backoff.(all lambda are obtain after tuning and code includes hyperparameter tuning also) Fig1 shows Perplexity on different datasets for test and dev set using Interpolation.

4.2 Task 2

Some examples of token generated from Language Model:

- What is this Small Print !" Matsuo puzzled and concerned .
- Kate had not cast out of the generation to generation .
- They were obliged to volunteer for six hundred thousand and

sectionAcuuracy/Measures

4.3 Task 1

Perpexility is used as the measure for this task. (All values in Result section table contain perpexility value)

4.4 Task 2

Human Evaluation.