Assignment 2

Introduction To Natural Language Processing

Tejasvi Chebrolu

2019114005

Introduction

The assignment aims to compare the performances of a *Neural Language* Model versus two *Statistical Models - Kneser-Ney and Witten-Bell*. The models have been evaluated on the *Brown Corpus* which has been cleaned and contains **48946** sentences. The training data consisted of **18946** sentences, the validation data consisted of **20000** sentences, and the test data consisted of **10000** sentences. The perplexity of each sentence in all the sets was calculated over the course of this assignment.

Tasks

- [x] ~~Cleaning The Data~~
- [x] ~~Splitting The Corpus~~
- [x] ~~Implementing The Neural Model~~
    - [x] ~~Train The Model~~
    - [x] ~~Test The Model~~
- [x] ~~Testing The Statistical Models~~
- [x] ~~Analysis Of The Results~~

Cleaning The Data

The following steps were taken in Cleaning:

- Stripped unnecessary extra lines.

- Text inside brackets was removed.
- Heading were removed.
- Special characters were removed.
- The text was converted into lowercase so that a better model could be developed.
- Each line contained only one sentence.

Splitting The Corpus

As seen before the corpus was split into 18496, 20000, 10000 sentences for the training, validation, and testing sets respectively. This was done using *splicing* in Python.

Implementing The Neural Model

The Hyperparameters used for building the Neural Model are as follows:

- **Number of Epochs - 5**
- **Batch Size - 100**
- **Learning Rate - 0.001**
- **Number of Units in each *LSTM* Layer - 512**
- **Number of *LSTM* layers - 2**

Testing The Statistical Models

Witten Bell and *Kneser-Ney* are two algorithms that are used in the smoothing of language models.

Formulae:

The formula for *Kneser-Ney* is:

$$P_{KN}(w_i|w_{i-n+1:i-1}) = \frac{\max(c_{KN}(w_{i-n+1:i}) - d, 0)}{\sum_v c_{KN}(w_{i-n+1:i-1}\ v)} + \lambda(w_{i-n+1:i-1}) P_{KN}(w_i|w_{i-n+2:i-1})$$

The formula for *Witten Bell* is:

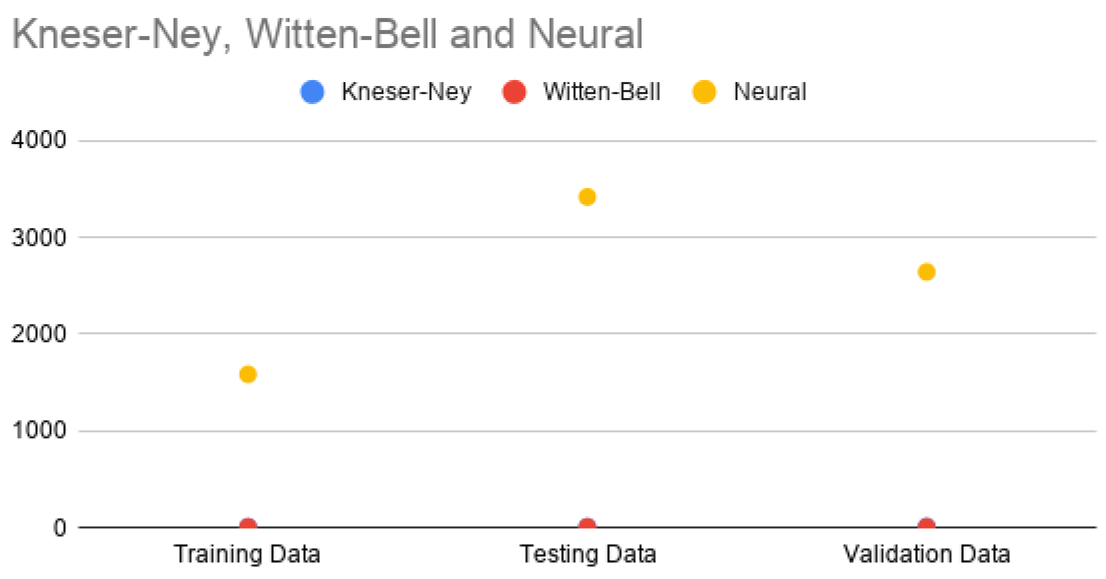
$$P_{\mathrm{KN}}(w_i|w_{i-n+1:i-1}) = \frac{\max(c_{KN}(w_{i-n+1:i}) - d, 0)}{\sum_v c_{KN}(w_{i-n+1:i-1}\ v)} + \lambda(w_{i-n+1:i-1}) P_{KN}(w_i|w_{i-n+2:i-1})$$

These models were then trained, tested, and validated on the Brown corpus.

Analysis

The following table shows the **average perplexities** of the various models across the different sets of data:

	Kneser-Ney	Witten-Bell	Neural
Training Data	8.857679874726093	4.829861168200131	1583.463230799964
Testing Data	9.119074074503942	5.671836855582933	3417.214761795408
Validation Data	11.810126395728332	5.351977054921513	2641.784142543386



From the table, we can conclude that the best performing model is the *Witten-Bell* model as it has the least overall perplexities. We can also conclude that in all, the statistical models are performing better than the neural models which is not what was expected. The Neural Model is almost 1000 times worse than the statistical models which could be because of multiple issues. Also, for all the models, the model performed best on the training data which could indicate a level of overfitting but the fact that the test data was not all that off from the validation data indicates that there was no major overfitting present in the models.

Reasons for Neural Model performing poorly -

- Reluctance to make changes to base algorithm because of high time taken to train a neural model.
- Low number of epochs which meant that the model could not be trained well enough to have a better performance compared to the statistical models.
- The batch size had to be altered keeping in mind the performance of Google Collab's GPU and it was not possible to max out all the hyperparameters to ensure the best performance of the model.