ASSIGNMENT 1
CS60075

**Language Modelling**

**Submitted by:**
Name: Shruti Shreyasi
Roll No: 19EC10086

## PART A:: N-Gram Language Model:

### OVERVIEW OF FUNCTIONS AND STAGES IN THE CODE

- DATA PRE PROCESSING
  - Each sentence was padded with N-1 start tokens and 1 end token for ease of handling the tokens at the beginning of the sentence
  - For the case of unigrams, start and end tokens are not added because only the probability of the tokens matter
  - The words in the corpus occurring only once are treated as UNK or unknown tokens

- N-GRAM MODEL
  - For every sentence in the train corpus, frequency of the n-grams and its corresponding n-1 grams are noted and stored

- PERPLEXITY
  - Perplexity of each sentence is calculated and laplace smoothing is applied while considering the counts of the n-grams and the n-1 grams

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

  - The perplexity of the entire corpus is determined as :: perplexity of a corpus = (product of probabilities of each sentence) ^ (1 / corpus size)

- GENERATE SENTENCE
  - A function is created to generate sentence based on the most probable n-gram
  - It takes a partial sequence and generates a sequence with that start
  - A max_length parameter can be passed which limits the length of the generated sequence in case end token is not encountered

- The most probable sequence is returned from this function. It considers the most probable next word at every point of time.
- This function is useless for unigrams because it will return the most frequent word in the train set for all probable words.

## OBSERVATIONS AND RESULTS OBTAINED

- PERPLEXITY ON TEST SET

```
perplexity of 1-gram model is:: 777.7048283969436
perplexity of 2-gram model is:: 580.9471390295671
perplexity of 3-gram model is:: 1291.848782371178
```

- GENERATION OF SENTENCES

seed = 'the sale'
max_length = 20

```
sentence generation by 2-gram model::
the sale of the company said

sentence generation by 3-gram model::
the sale of its common stock

sentence generation by 4-gram model::
the sale is part of a restructuring plan it expects pre tax operating loss of two cts a

sentence generation by 5-gram model::
the sale is part of a major program to divest several of its businesses representing about 200 mln
```

seed = 'by'
max_length = 20

```
sentence generation by 2-gram model::
by the company said

sentence generation by 3-gram model::
by contrast will see a 15 pct of the total outstanding common stock

sentence generation by 4-gram model::
by contrast the personnel and fixed asset expenses increased

sentence generation by 5-gram model::
by contrast the personnel and fixed asset expenses increased
```

## DISCUSSION

PERPLEXITY

- The Perplexity score of the Bigram model is less than the Unigram model because the Unigram model does not look at the previous words and the probability is independent of the previous words. Hence the probability of the test set as per the Bigram model is more as compared to the Unigram model.
- The Perplexity score of the Trigram model is more than the Bigram model. This can be because the train set is not long enough. The trigrams seen in the test set are not present in the training set and thus their count is taken as 1 due to smoothing. Due to the absence of many trigrams enountered, the perplexity increases as compared to the Bigram model.
- It can be noted that there can be more variations while calculating the perplexity. One might ignore the end of sentence tokens or take the entire corpus as a single string with 1 start and 1 end token for every sentence. The order of the results however, will remain the same because perplexity takes the $N^{th}$ square root while computation so the effect is minimised

SENTENCE GENERATION

- The generate function is useless for unigrams because it will return the most frequent word in the train set for all probable words.
- The sentence generated by the bigram model is very trivial because it considers just one previous token while prediction, the trigram model captures language more nicely.
- It can be noted that as the value of N increases, the complexity of the sentences also increases. This is because more previous words are considered while prediction and the continuity increases.
- It can be noted that if N is increased too much then the sentence predicted will be almost identical to the train corpus because the number of unique N-1 grams are limited as compared to the total possible N grams.

---

## PART B:: LSTM Based Language Model:

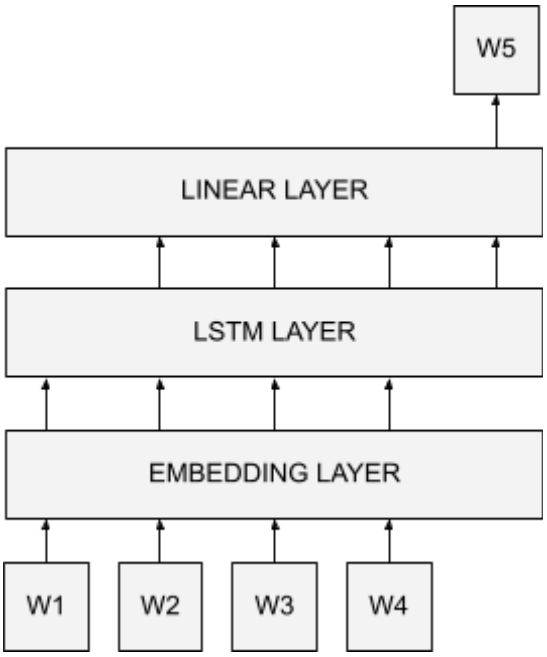


FIG: Schematic of the layers used in the model

LAYERS:

- GloVe embedding layer
- LSTM Layer
- Linear layer

LOSS FUNCTION

- Perplexity score

### OVERVIEW OF MODEL AND LAYERS

- PRETRAINED GloVE EMBEDDING
    - GloVe 6B, d = 50 is used for word embeddings
    - The words from corpus not present in GloVe are marked as UNK and encoded with a random sequence
    - The padding tokens are also defined

- CLASS TO HANDLE DATASET
    - All tokens present in the train dataset are assigned a number which is to be compared to the output of the linear layer
    - All the sentences of the dataset are concatenated into a single string with unknown tokens replaced by <UNK>
    - For using dataloader, the <input, label> pair is defined for every index as a sequence starting at index i and its shifted sequence.

- MODEL CLASS AND TRAINING
    - An embedding layer of dimension 200 (50*4) is taken.
    - The output is passed to a LSTM layer which has 2 layers, input dimensions 200 and hidden size of 200.
    - The result obtained is passed to a linear layer of output size equal to the vocabulary size
    - For training, batch size is set to 256 and 5 epochs are taken. Length of sequence is set to 4.
    - Adam optimizer with learning rate 0.005 is used.
    - Cross entropy loss is taken as the loss function and perplexity is computed as PP = e ^ cross entropy loss
    - Perplexity is minimised and the model is trained. The trained model is saved.

- VALIDATION AND TESTING
    - For validation, the batch size, learning rate and learning rate are experimented with. Hyperparameter tuning is done.
    - For testing the model, the fine tuned model is selected and loss function is calculated on it.

## OBSERVATIONS AND RESULTS OBTAINED

Perplexity while training in each epoch

```
tensor(6.3254, device='cuda:0', grad_fn=<DivBackward0>)
(558.5544974114267+0j)
tensor(5.4461, device='cuda:0', grad_fn=<DivBackward0>)
(231.85919648686715+0j)
tensor(5.1069, device='cuda:0', grad_fn=<DivBackward0>)
(165.1524886470643+0j)
tensor(4.8953, device='cuda:0', grad_fn=<DivBackward0>)
(133.65454204964826+0j)
tensor(4.7557, device='cuda:0', grad_fn=<DivBackward0>)
(116.24068568188125+0j)
```

Perplexity while training

```
tensor(4.7557, device='cuda:0', grad_fn=<DivBackward0>)
(116.24068568188125+0j)
```

Perplexity while validation

```
tensor(5.7632, device='cuda:0')
(318.35159416356606+0j)
```

Perplexity while testing

```
tensor(5.7660, device='cuda:0')
(319.26934236103034+0j)
```

## LINK TO TRAINED MODEL:

[model.txt]

## DISCUSSION:

- The perplexity score obtained using neural models is better as compared to N-grams. The perplexity score of the LSTM model is 319 whereas the perplexity score of the bigram model is 581.
- The performance of neural models is better compared to probabilistic models. The result could have been more differentiating if all the words were present in GloVe embeddings, which was not the case.

## PROBLEMS FACED:

- A large number of words had frequency 1 and the performance of N-gram model was not upto the mark. To tackle this, all such words were replaced by UNK.
- A lot of words in the data set were not present in GloVe. To tackle this, they were replaced by random embedding and replaced by UNK.

## CONCLUSION:

Neural models are better at language modelling as compared to probabilistic models like N-gram