ANLP ASSIGNMENT 1

Preprocessing

- Removed trailing white spaces and converted the text to lowercase.
- Expanded contractions using the `contractions` library.
- Tokenized the corpus using the NLTK sentence tokenizer and word tokenizer.
- Retained only alphanumeric characters.
- Removed sentences containing fewer than five words.
- Replaced words not present in the embeddings or with a frequency below the cutoff with `<unk>`.
- Divided the sentences into training, testing, and validation sets in a 70:20:10 ratio.

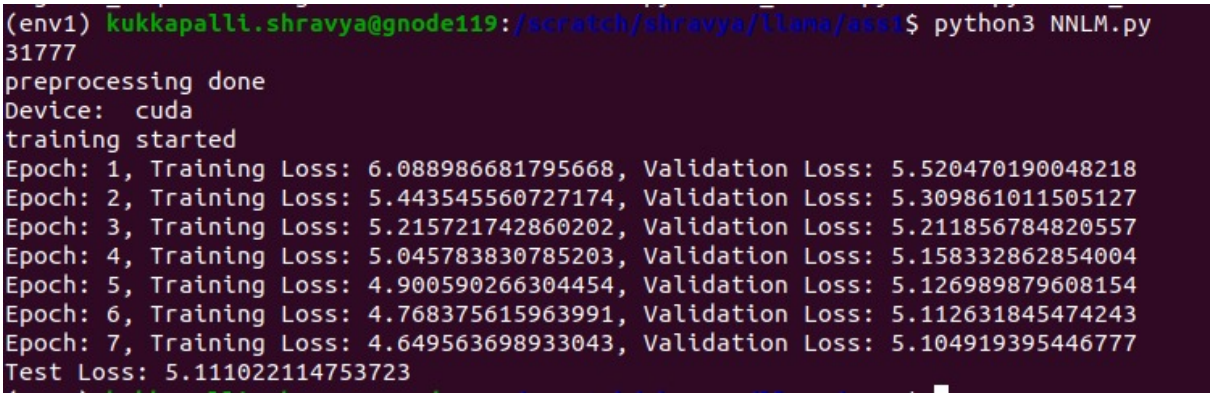
After preprocessing, the total number of sentences is 31,777.

GloVe embeddings (100 - dimensional) were used for all models.

NNLM

1. Input dimension: 500
2. Hidden dimension: 300
3. Output dimension: `vocab_size`
4. Number of epochs: 7
5. Batch size: 64
6. Learning rate: 0.001
7. loss function: Cross entropy
8. Optimizer: adam

The batch size of 64 means that 64 sentences are processed per batch, with each sentence being converted into possible 5-grams.



The perplexity of a sentence is calculated by passing its corresponding 5 - grams as a batch to the model and computing the exponential of the cross-entropy loss.

Average Perplexity of model

1. Train set - 99.44213699256913
2. Test set - 228.9587905199925
3. validation set - 227.11006899701

LSTM

The sentences are padded to the maximum length within their respective batch.

1. Input dimension: 100
2. Hidden dimension: 300
3. Output dimension: `vocab_size`
4. Number of epochs: 10
5. Batch size: 32
6. Learning rate: 0.001
7. No.of LSTM layers: 1
8. Dropout: 0.1
9. loss function: Cross entropy
10. Optimizer: adam

```
Epoch: 1, Training Loss: 2.102381602957331, Validation Loss: 1.8506214249134063
Epoch: 2, Training Loss: 1.7478826848426083, Validation Loss: 1.727338193655014
Epoch: 3, Training Loss: 1.648518237745625, Validation Loss: 1.6630254518985748
Epoch: 4, Training Loss: 1.5848082801562615, Validation Loss: 1.629693752527237
Epoch: 5, Training Loss: 1.5261144389280643, Validation Loss: 1.6084444552659989
Epoch: 6, Training Loss: 1.4878612908309903, Validation Loss: 1.5947236490249634
Epoch: 7, Training Loss: 1.4551771371785938, Validation Loss: 1.58422292470932
Epoch: 8, Training Loss: 1.4276682144437713, Validation Loss: 1.5798578649759292
Epoch: 9, Training Loss: 1.3920235877492646, Validation Loss: 1.578206957578659
Epoch: 10, Training Loss: 1.3579695477396592, Validation Loss: 1.5795817124843596
Test Loss: 1.5131720508163298
(env1) kukkapalli.shravya@gnode119:/scratch/shravya/llama/ass1$
```

The perplexity of a sentence is calculated by passing the sentence to the model and computing the exponential of the cross-entropy loss.

Average Perplexity of model

1. Train set - 79.3295580515977
2. Test set - 177.24482398132906
3. validation set - 186.54219259658925

Transformer Decoder

The sentences are padded to the maximum length of all the sentences in the corpus.

1. Input dimension: 100
2. Feedforward dimension: 300
3. No.of heads = 4
4. No.of decoder layers = 1
5. Output dimension: `vocab_size`
6. Number of epochs: 5
7. Batch size: 32
8. Learning rate: 0.001
9. No.of LSTM layers: 1
10. Dropout: 0.1
11. loss function: Cross entropy
12. Optimizer: adam

```
Epoch 0 | Train Loss: 0.6374029702581894 | Val Loss: 0.4132879051566124
Epoch 1 | Train Loss: 0.41036158804407064 | Val Loss: 0.37825586527585986
Epoch 2 | Train Loss: 0.38504257798194885 | Val Loss: 0.3621618239581585
Epoch 3 | Train Loss: 0.3693247403007472 | Val Loss: 0.3520828613638878
Epoch 4 | Train Loss: 0.35766209032515 | Val Loss: 0.3450618512928486
Test Loss: 0.3501290245421568
(env1) kukkapalli.shravya@gnode119:/scratch/shravya/llama/ass1$
```

The perplexity of a sentence is calculated by passing the sentence to the model and computing the exponential of the cross-entropy loss.

Average Perplexity of model

1. Train set - 1.4792980118492003
2. Test set - 1.5331180445030648
3. validation set - 1.5132878369267422

Analysis

Model	Train perplexity	Test perplexity	Val perplexity
NNLM	99.44213699256913	228.9587905199925	227.11006899701
LSTM	79.3295580515977	177.24482398132906	186.54219259658925
Transformer decoder	1.4792980118492003	1.5331180445030648	1.5132878369267422

Performance: NNLM < LSTM << Transformer decoder

NNLM
The NNLM relies on a fixed context window of 5 words to predict the next word in a sequence. This limited context window restricts the model's ability to capture dependencies that span beyond the 5-word context. As a result, while it can handle short-range dependencies reasonably well, it fails to model long-term dependencies effectively.

The model memorizes the short-term patterns in the training data but fails to generalize well because it cannot handle longer-range dependencies beyond the 5-word window.

LSTM
LSTM networks are specifically designed to handle long-term dependencies by incorporating gates (input, forget, and output gates) that regulate the flow of information through the network over time. This "infinite" context window allows the LSTM to theoretically remember information over arbitrary time lags, providing a significant advantage over the fixed context window of the NNLM.

The **LSTM** performs better than the NNLM because of its capacity to model longer-term dependencies. However, it still suffers from challenges in effectively capturing very long-range dependencies, leading to a significant gap between training and test/validation perplexities.

Transformer decoder

The **Transformer Decoder** outperforms both NNLM and LSTM by a large margin due to its self-attention mechanism, which enables it to handle both short-term and long-term dependencies with equal efficiency. This results in consistently low perplexity scores across all datasets, highlighting its superior capability in the modeling language.