ANLP – Assignment 1

Course Coordinator: Manish Srivatsava
Name : Lakshmipathi Balaji
Roll No: 2021114007
Mail : lakshmipathi.balaji@gmail.com

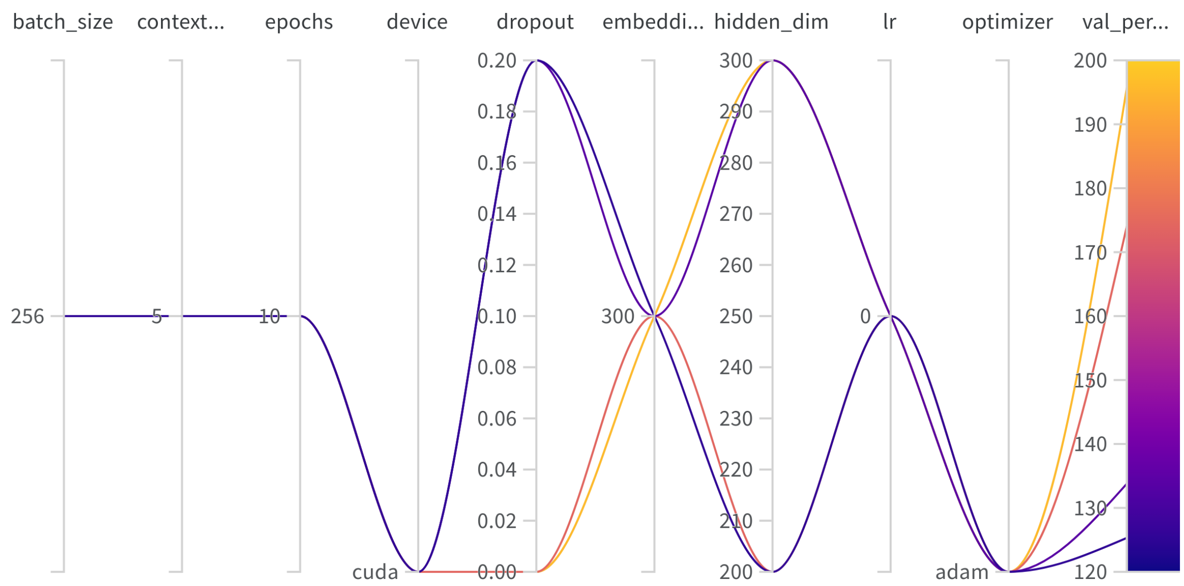
Assignment details:

- Implementation of Language Models with the arhcitectres of NNLM, LSTM, TransformerDecoder.
- The code and the perplexity score files are attached with the submission.
- WandB is used for better analysis on performance of models with various hyperparameters.
- Here is the link for the wandb project of this assignment. - https://wandb.ai/lakshmipathi-balaji/anlp?workspace=user-lakshmipathi-balaji

Things learnt:

- Implementation of NNLM, LSTM and Transformer Decoder and finegrained understanding of how it works.
- The notion of perplexity.

Observations of model performances:



I have tried with a few parameters and the parameters tried are mentioned in the below config file,

```yaml
method: grid
metric:
  goal: minimize
  name: val_perplexity
parameters:
  batch_size:
    values:
      - 256
  context_size:
    values:
      - 5
  device:
    values:
      - cuda
  dropout:
    values:
      - 0
      - 0.2
  embedding_dim:
    values:
      - 300
  epochs:
    values:
      - 10
  hidden_dim:
    values:
      - 300
      - 200
  lr:
    values:
      - 0.01
  optimizer:
    values:
      - adam
program: nnlm.py
```

NNLM config

```yaml
method: grid
metric:
  goal: minimize
  name: val_perplexity
parameters:
  batch_size:
    values:
      - 128
  device:
    values:
      - cuda
  embedding_dim:
    values:
      - 300
  epochs:
    values:
      - 20
  hidden_dim:
    values:
      - 128
      - 256
  lr:
    values:
      - 0.001
  num_layers:
    values:
      - 2
      - 1
  optimizer:
    values:
      - adam
  seq_len:
    values:
      - 25
program: lstm.py
```

LSTM config

```yaml
method: grid
metric:
  goal: minimize
  name: val_perple
parameters:
  batch_size:
    values:
      - 256
  device:
    values:
      - cuda
  embedding_dim:
    values:
      - 300
  epochs:
    values:
      - 10
  hidden_dim:
    values:
      - 128
      - 256
  lr:
    values:
      - 0.001
  num_heads:
    values:
      - 6
      - 9
  num_layers:
    values:
      - 1
      - 2
  optimizer:
    values:
      - adam
  seq_len:
    values:
      - 25
program: lstm.py
```

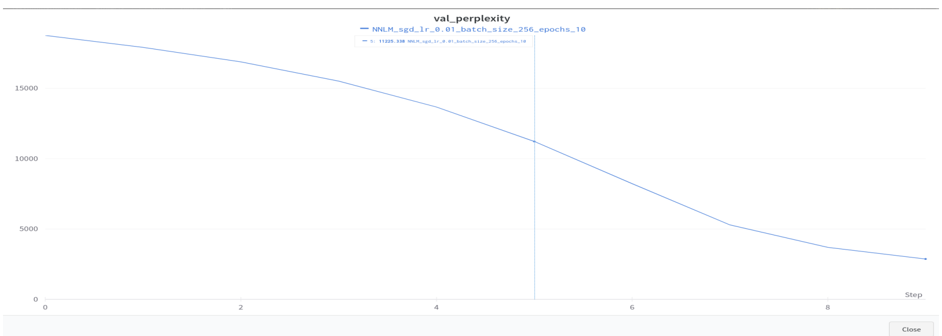
TransformerDecoder Config

The best mode in the case of NNLM is observed to the one with the hyperparameters -

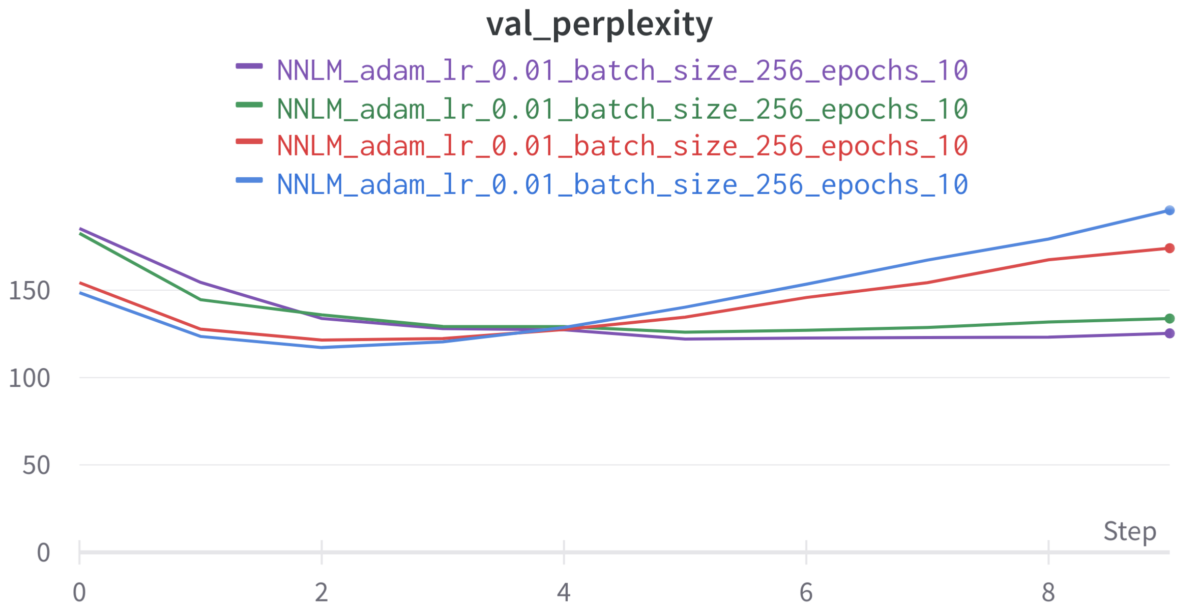
Key	Value
batch_size	256
context_size	5
device	"cuda"
dropout	0
embedding_dim	300
epochs	10
hidden_dim	300
lr	0.01
optimizer	"adam"

Though there has been an experiment with the optimizer SGD the performance of the model is not say great with respect to val_perplexity, so the experiments are only continued with adam optimizer.

- The result for SGD optimizer is in the below image.



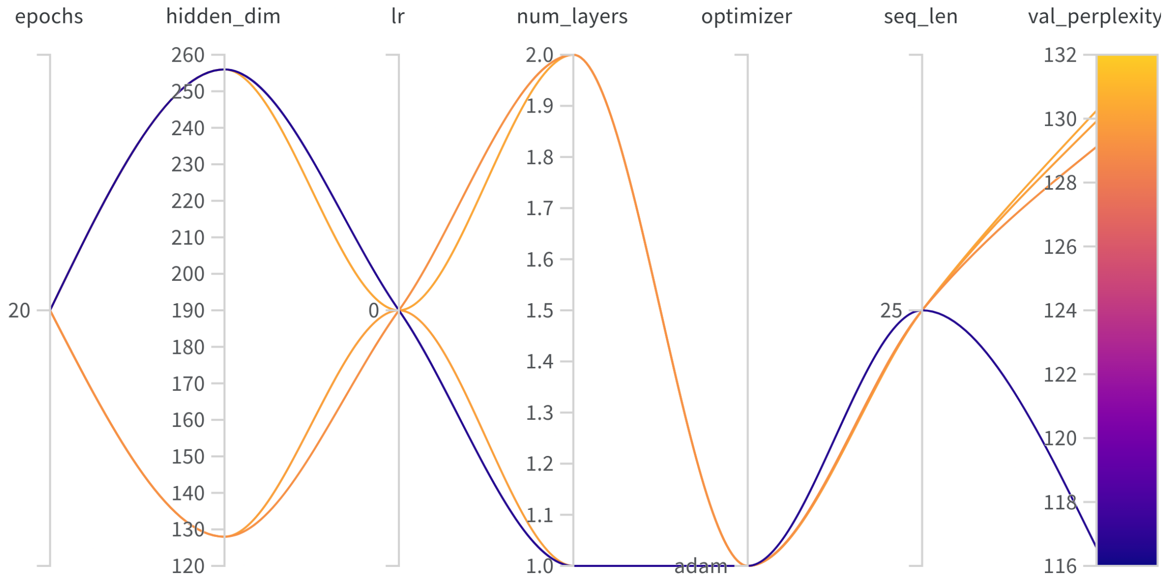
Other Observations on NNLM experiments:



The models with no dropout 2 converged much sooner when compared to models with dropout 0.2, this is possibly because some neurons are deactivated when dropout is done thus leading to slower convergence. And the perplexity on validation set increased which shows that the model started to overfit the traning data.

Observations on LSTM model performance:

The hyperparameters used for LSTM architecture experiments are mentioned before, and the best model can be observed through the below panel.



So the best performing model on the given dataset with different splits is with hyperparameters given below -

batch_size	128
device	"cuda"
embedding_dim	300
epochs	20
hidden_dim	256
lr	0.001
num_layers	1
optimizer	"adam"
seq_len	25

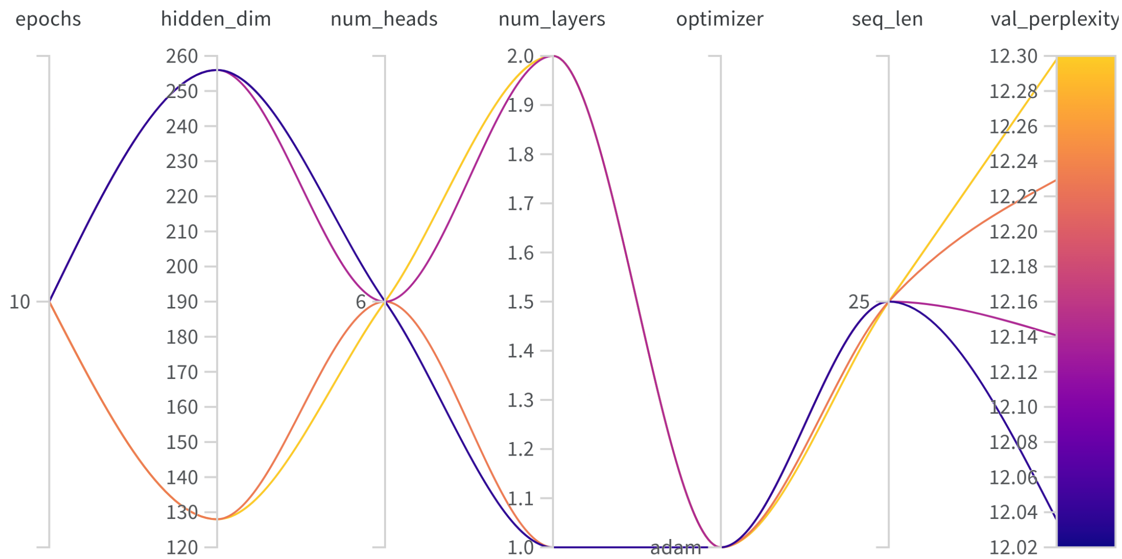
test_perplexity	116.83374521958146
val_perplexity	116.56633417895716

Results

It can be noted from the above plot that the model with these hyperparameters showed a significant performance when compared to others.

Observations on TransformerDecoder model performance:

- The transformer decoder model showed a significantly better performance when compared to the other ones which can be observed through the below perplexity scores.



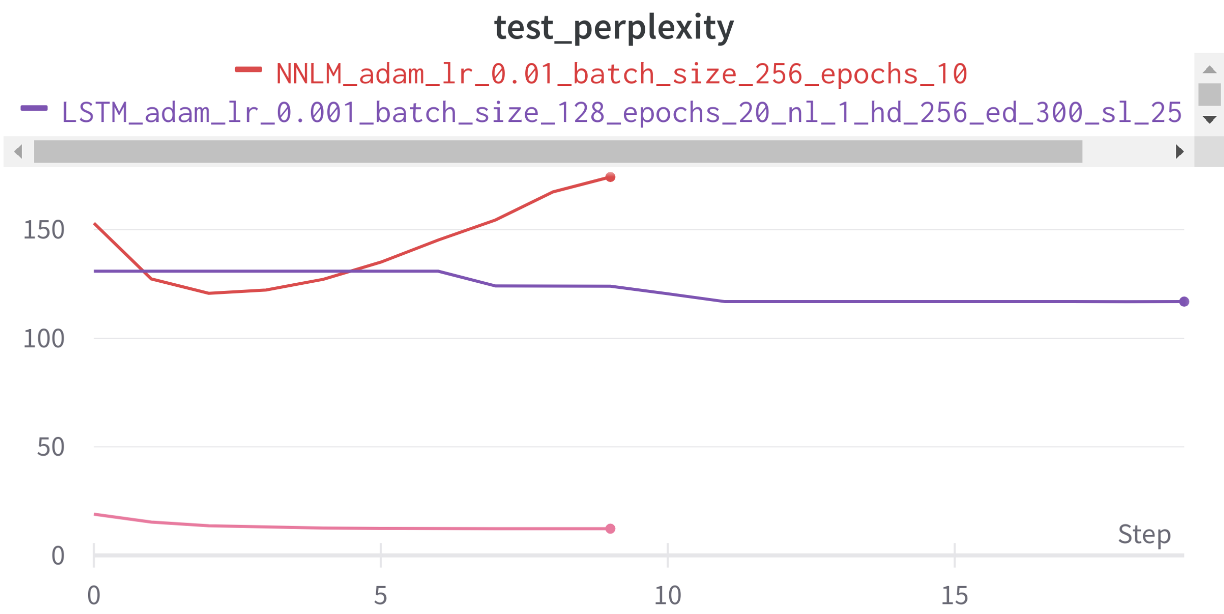
- Though they show a similar perfomance in comparision the best perfoming model on the given datset is -

Key	Value
batch_size	256
device	"cuda"
embedding_dim	300
epochs	10
hidden_dim	256
lr	0.001
num_heads	6
num_layers	1
optimizer	"adam"
seq_len	25

Key	Value
test_perplexity	12.3013386237461 7
val_perplexity	12.035595460852088

Results

Observations across different architectures:



- The above graph shows how better the Transformerdecoder architecture based model performed better on test set. This is because of significant improvements in the architecture design.
- We can observe that at the initail epochs the NNLM model performs better than LSTM this is possibly because in the NNLM model there a lot more parameters when compared to lstm.
- Eg: For a embedding dim. 300 and hidden dim of 300 the LSTM model would have (500*300) + (300*vocab_size(around 20,000)) which is a lot more when compared to lstm.

Key Note:

- The formula for perplexity used is

$$H(W) = -\frac{1}{N}\log P(w_1 w_2 \ldots w_N)$$

$$\begin{aligned}
\text{Perplexity}(W) &= 2^{H(W)} \\
&= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \\
&= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}
\end{aligned}$$

From *Speech and Language Processing* book.

- The pretrained embeddings used are from gensim library from the model - *fasttext-wiki-news-subwords-300.* Though *glove-wiki-gigaword-200* not many experiments are done with that embeddings.
- Though there are many things that can be observed and explained in the experiments and the architectures of these models only some key observations are mentioned in this report.

The link to the models that are saved after training are given in this onedrive link. Ass1.