Task 2: WORDS PREDICTION LSTM*

*LSTM for next word or words prediction Shakespeare's play's lines

Areeba Fatah
AI (artificial intelligence)
FAST, NUCES)
Islamabad, Pakistan
i210349@nu.edu.pk

I. Introduction

In this task we were asked to use LSTM for next word prediction on the given dataset. The next step was to locally deploy and have real time next word prediction.Following is the detail of

II. Methodology

The dataset given was shakespeare's data having over 111000+ lines from well known plays. The large quantity of dataset makes it difficult to run on a local machine. The dataset has three files i.e. an image, a csv and a txt. For the sake of simplicity and faster implementation i used alllines.txt file. The following is the elaborated detail of implementation done.

A. Text cleaning and preprocessing

- **Text Cleaning** In this first step i read the text file, converted it to lower string, removed special characters, non ASCII characters, roman numbers and the words scene and act. Then removed stop words.
- **Text Conversion** Then to convert to integers i used three different approaches i,e, max word frequency as integer, one hot encoding and direct conversion to integers.
- **One hot encoding** It is not feasible for such a wide corpus I had used it but got encountered with memory issues soon enough i realized either i had to cut down data or use another technique. I had cut down the data like done in the reference notebook but the their was not great context due to smaller data. [1]
- **Max Frequency word** The next approach i used for replacing the word with its frequency count like counting the number of occurrences and replacing with it.
- **Direct Conversion** The last approach was to convert the word of vocab to it's index which helped the model train faster.
- **LSTM sequence generation** Here the target will be the next word in a sequence.

B. Model Architectures and Training

- **LSTM Architecture** Many to one architecture of LSTM is followed. Here there is one embedding layer or input layer followed by LSTM layer then linear layer for output.

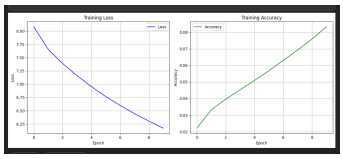

Fig. 1. Max Count embedding

- **Grid Search for best parameters** Here based on training accuracy i implemented Grid Search for finding possibly good parameters for LSTM.
- **RNN architecture** Here i have implemented a simple RNN with embedding then RNN layer then the output layer. This was to compare and understand the performance of both of them.
- **Pytorch Use** The tensorflow i have installed have many issues with flask that is why i had to use pytorch so accuracy calculation and everything is done manually.

C. Deployment

- **Flask** For deploying the app locally i have used Flask since i am familiar with it.

III. Results

The following are the results or findings of my experimentations.

A. Words to integers methods

- **One hot encoding** It is not suitable for this problem as it will create a sparse matrix . Tried to use glove embedding but it also resulted in crashed session. I followed [1] for One hot encoding but it was on only some of the data.
- **Max Occurrence count as integer** It was very biased and slow, it resulted in under-fitting as the model was unable to learn anything useful even after a long time has passed. Training was very slow and the accuracy was very low. The accuracy over 10 epochs is 8% which took almost an hour, hence poor performance as shown in 1 and 2
- **Direct Conversion** It means that vocab has unique words assign each word its index to convert to integers. This approach is very fast and gives good results. Training

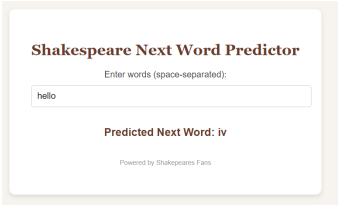

Fig. 2. Max count approach gives repetitive words back since accuracy = 9%

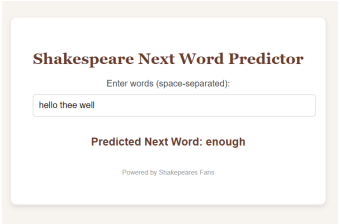

Fig. 3. Direct Conversion Flask app result

accuracy is up to 99% an Testing is 80 %. Although the testing accuracy decreases bit it outperforms RNN's validation accuracy. This is by far the best case as shown in **??**. , 4 and 3

B. Grid Search

for LSTM training using direct conversion preprocessing method.

- embedding dim : 32, 64
- hidden dim: 64, 128
- num epochs: 10,20
- Best Parameters:
  - embedding dim : 64
  - hidden dim: 128
  - num epochs: 20

  having Accuracy: 0.9927

C. RNN and LSTM comparison

- Over 10 epochs ,LSTM=67% training accuracy and RNN =69% accuracy

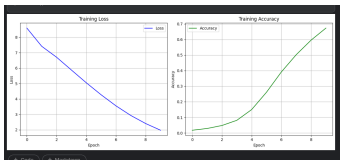

Fig. 4. LSTM training curve

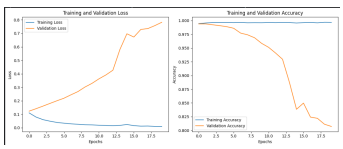

Fig. 5. Direct conversion method validation and loss

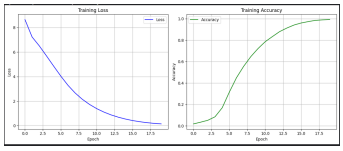

Fig. 6. RNN training

- Over 20 epochs , LSTM=99% training accuracy and RNN =98% accuracy.
- RNN validation = Validation Accuracy: 0.0341 , while lstm had over 80% validation thus RNN seems even more overfitting then LSTM. This implies since RNN has shorter memory so it's accuracy is less than LSTM over longer spans of time/epochs.

IV. Discussion

- The first issue is the requirement of assignment that is stop words removal. With the removal of stop words our context of words is lost making the models give output which doesn't make much sense to us. for instance if i type "hello" it will predict "thy" and so on.
- The second issue is lack of powerful machines with strong enough memory to be able to use one hot like encoding and so on.
- Direct Mapping of text to index is by far the fastest way to train and get accuracy above 50%.
- We can map this problem on many to many architecture of LSTM as well.
- Over time LSTM has better memory than RNN as mentioned in results.

Conclusion

LSTM models are good for remembering longer sequences of data but the data should also be related in such a way that its embeddings makes sense. Like this problem, models could have performed much better if stop words were not removed. Furthermore, Direct conversion mappings of text to integer is the fastest way to deal with huge amount of Data.Lastly, LSTM has better memory and power to store context for a longer time than RNN.

Prompts

- "Give me code to read a text file and clean it"
- " What LSTM architecture suit this problem of next word prediction?"
- " Basic LSTM with pytorch code"
- " Why is my tensorflow not working"
- "Alternate of tensorflow"
- "How to add grid search fot this?"
- "How to plot the results?"
- " Add validation code "
- " Flask code for this "
- " Suitable html design"

REFERENCES

[1] A. Chaudhary, "Text Generation using LSTM," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/code/apoorvchaudhary/text-generation-nlp-lstm. [Accessed: Sep. 29, 2024].