NEXT WORD PREDICTION MODEL

Vinod Gaitonde

Manchinasetti Sai Srihari

Sai Kiran Velishala

Hardik Kamaleshkumar Parekh

Prudhvi Sanaboyina

Piyush Jain

Deebak V S Vijay

PROBLEM STATEMENT

Next Word Prediction is a widely used feature in smartphone keyboards, messaging applications, and search engines such as Google. It enhances user experience by predicting and suggesting the next word in a sentence based on the previous input.

Use Case: A Next Word Prediction Model trained based on an organization's domain, emails & chat messages can be eventually integrated in respective applications e.g. chat, email text editor, which can help with efficiency.

This project presents a deep learning approach to building a **Next Word Prediction** model using Python, leveraging the TensorFlow and Keras libraries. We train a **Recurrent Neural Network (RNN),** specifically a **Long Short-Term Memory (LSTM)** network, which is well-suited for handling sequential data and long-term dependencies, making it ideal for this task.

DATA SET INFO

SOURCE:

https://www.gutenberg.org/

The Adventures of Sherlock Holmes, by Arthur Conan Doyle

* Due to Compute & RAM limitations, we could only use one book for training.

DATA CLEAN UP & PROCESSING

- UTF-8 encoded ".txt" file of the novel is used.
- Tokenize & create unique word dictionary.
- Create n-gram.
- Pad sequences to make them of same length.
- Prepare features(n-gram) against label.
- Convert output array into suitable format for training

```
tokenizer.word_index:  7358
total_words:  7359
word_index :  {'the': 1, 'and': 2, 'of': 3, 'to': 4, 'a': 5, 'i': 6,
```

VOCABULARY FROM THE BOOK

```
input_sequences count 69880
0 input sequence [1676, 3769]
1 input sequence [1676, 3769, 1]
2 input sequence [1676, 3769, 1, 955]
3 input sequence [1676, 3769, 1, 955, 3]
4 input sequence [1676, 3769, 1, 955, 3, 109]
```

N-GRAM PREPARATION

N-GRAM SEGREGATED IN FEATURES(X) & LABEL(Y)

ML METHODOLOGIES USED

Recurrent Neural Network (RNN)

RNN model is built using Keras library.

The Sequential model is created which is a linear stack of layers in Keras.

Layer	Nodes	Dropout	
Embedding	100	0.2	Converts words from their raw categorical representation into dense vectors that capture semantic relationships between words, making it significantly easier for the LSTM to learn complex
LSTM	150	0.2	Capture long-term dependencies between words and phrases in a sentence
Dense			Transforms the complex, high-dimensional representation produced by the LSTM into a final output probability distribution over the vocabulary

Other Hyperparameters	
Activation Function	SoftMax
Loss Function	Categorical Cross Entropy
Optimizer	Adam
Epochs	60
Batch Size	128

DOCKER CONTAINER

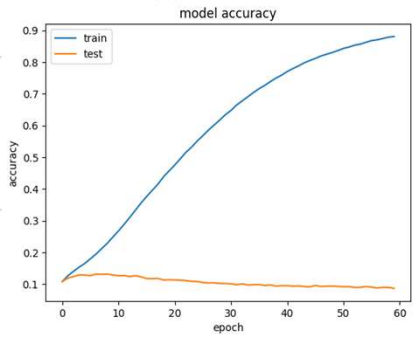
Docker

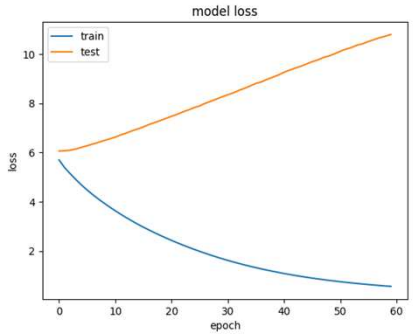
- A Docker container image is a standalone, executable package of software that includes everything needed to run an application.
- In this case, includes the pre-trained model and the python program
- Can be ported to another computer easily
- Docker Container uploaded at: https://drive.google.com/file/d/1TH4YPALstGicy1AeoPx8uY5L0s08j1_4/view?usp=sharing

```
# Use a base Python image
FROM python:3.10-slim

# Create a working directory
WORKDIR /app

# Copy requirements file
COPY requirements.txt ./requirements.txt
COPY Next_Word_Pred_V1.py ./Next_Word_Pred_V1.py
COPY next_Word_16_Nov_v1.keras ./next_Word_16_Nov_v1.keras

# Install the required packages
RUN pip install --no-cache-dir -r requirements.txt


# Specify the command to run your app
CMD ["python", "./Next_Word_Pred_V1.py"]
```

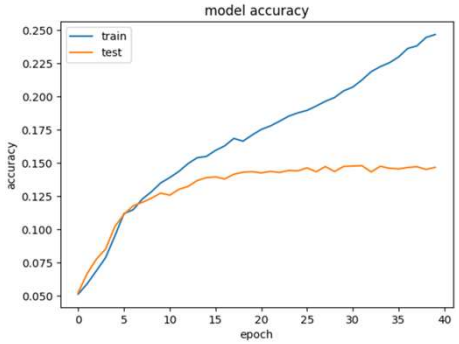
MODEL TUNING

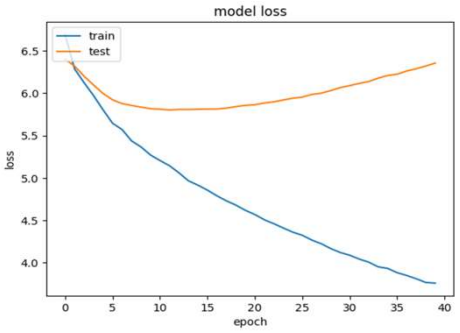
Without cross-validation
Embedding: 100
LSTM: 150
Dense
activation='softmax'
optimizer='adam'
validation_split=0.2
batch_size=128
epochs=60



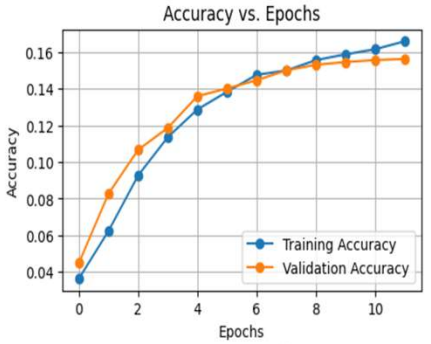


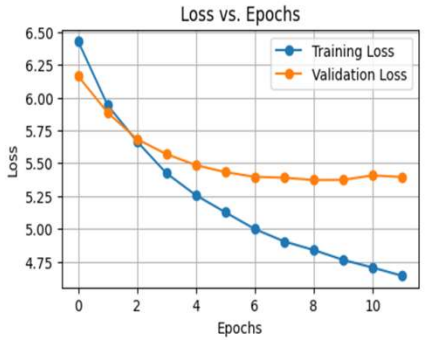
With 5x cross-validation
Embedding: 100
LSTM: 150
Dense
activation='softmax'
optimizer='adam'
validation_split=0.2
batch_size=128
epochs=60



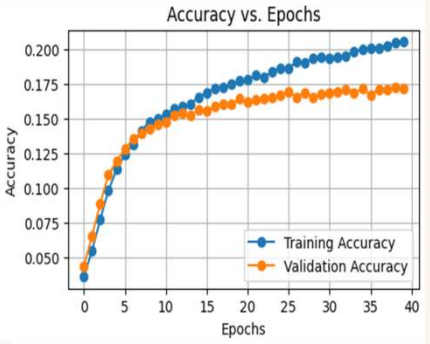


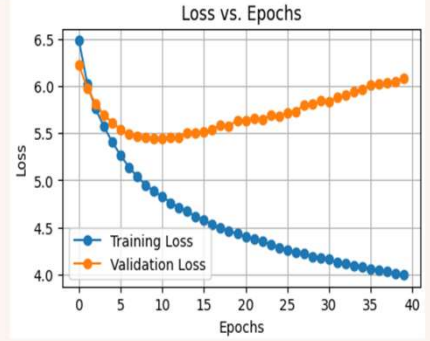
With Early Stopping
Embedding: 100
LSTM: 256, Dropout: 0.2
LSTM: 128, Dropout: 0.2
Dense
activation='softmax'
optimizer='adam'
validation_split=0.2
batch_size=128
epochs=40





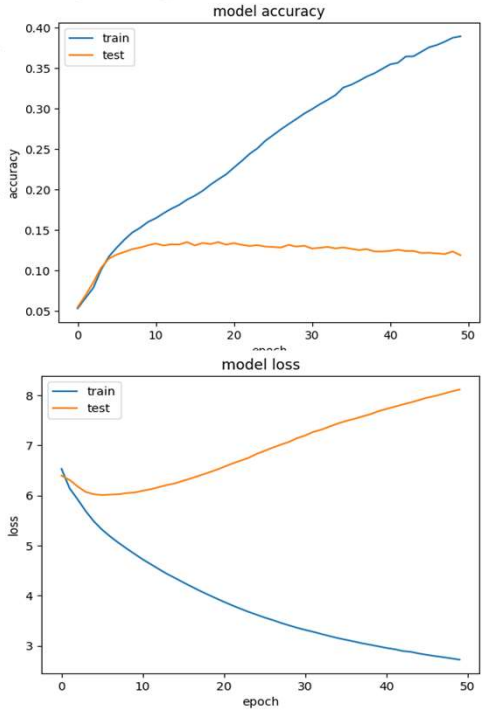
Without Early Stopping
Embedding: 100
LSTM: 256, Dropout: 0.2
LSTM: 128, Dropout: 0.2
Dense
activation='softmax'
optimizer='adam'
validation_split=0.2
batch_size=128
epochs=40



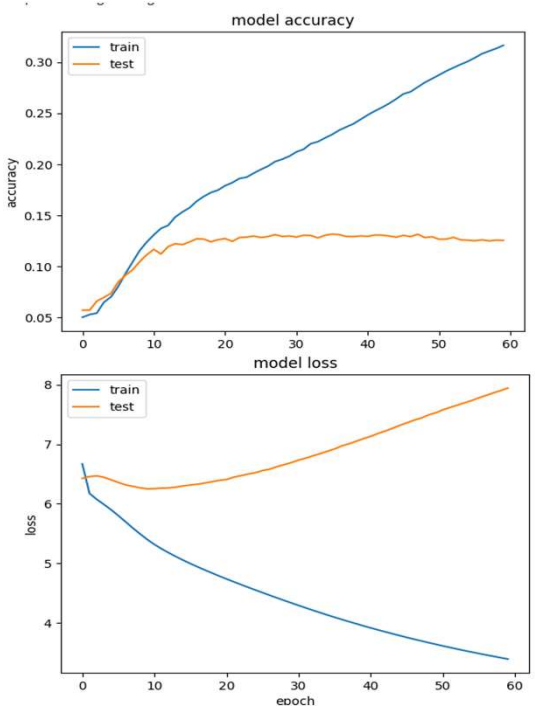


MODEL TUNING

```
Embedding: 100, Dropout: 0.2
LSTM: 150      , Dropout: 0.2
Dense
activation='softmax'
optimizer='adam'
validation_split=0.2
batch_size=128
epochs=60
```




```
Embedding: 100
LSTM: 50
Dense
activation='softmax'
optimizer='adam'
validation_split=0.2
batch_size=128
epochs=60
```



PREDICTION RESULTS


```
Enter your line: Hello What is your
Predicted Next Words: ['majesty', 'fiver', 'own', 'case', 'statement']

Enter your line: Why did you kill
Predicted Next Words: ['it', 'you', 'the', 'your', 'this']

Enter your line: just where the firelight strikes
Predicted Next Words: ['it', 'the', 'in', 'that', 'you']

Enter your line: I could not help laughing at the ease with which he
Predicted Next Words: ['had', 'was', 'explained', 'would', 'could']

Enter your line: A man entered who could hardly have been less than six
Predicted Next Words: ['months', 'years', 'feet', 'but', 'days']

Enter your line: 0
Execution completed.....
```

CONTRIBUTION

	Vinod Gaitonde	Manchinasetti Sai Srihari	Sai Kiran Velishala	Hardik K Parekh	Prudhvi Sanaboyina	Piyush Jain	Deebak V S Vijay
Data Cleanup & acquisition	25			25		25	25
ML Model Selection Training		20	20	20	20	20	
Hyper Parameter Tuning		25	25		25	25	
Metrics	25	20	20		20	15	
Presentation	10	10	10	35	10		25
Documentation	25	10	10	10	10		35

CONCLUSION

- Fair prediction of the Next Word with selected Model [Accuracy: 40%]
  - We achieved accuracy of ~90%; however, observed the model to have overfitting and did not generalize well beyond the data set.
- RNN/LSTM is well suited for use cases of complex sequential data like language
- Embedding layer helps translate the vocabulary in a dense vector and very helpful for language
- Following parameters are helpful to avoid “overfitting” of the model
  - Drop out (20%) helped very well with achieving better prediction & validation accuracy
  - In this particular case, L2 regularization did not help as accuracy didn’t improve
  - Cross validation did not show improvement in the training & validation accuracy balance
  - Early stop was tried however, considering limited data, most of the cases, the training stopped after 10 epochs
  - Added two layers of LSTM with drop out which helped with validation accuracy but only up to 20 epochs after which it was observed to be plateaued
  - Similar observation with bi-directional LSTM as well
- Training using “limited amount of data set” leads towards overfitting or lack of accuracy
  - Due to limited compute & RAM resources, we could not try with larger data set to train the model

LINK TO SOURCE CODE & DOCKER

- Project Git hub link: https://github.com/vinodgaitonde/Capstone_Group10
- Docker Container location: https://drive.google.com/file/d/1TH4YPALstGicy1AeoPx8uY5L0s08j1_4/view?usp=sharing

FUTURE ENHANCEMENTS

- ML Ops to be setup for continuous collection of data , retraining & deployment using DVC & ML Flow
- Experiment with Pre-trained tokenizer models like GPT 2.0 for this purpose
- Transfer learning (TL) technique that uses a pre-trained model on required set of data set
- Optimize the size of the docker container
- Integrate the container with a chat application

Q & A

THANK YOU!