SENTENCE COMPLETION USING LSTM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

DIVYANSHU YADAV(RA2111003010693)
KHUSHI MISHRA(RA2111003010703)
APARNA SINGH(RA2111003010752)

Under the guidance of

Dr R Deepa

Associate Professor, Department of Computing Technologies

in partial fulfillment for the award of the degree
Of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"SENTENCE AUTO COMPLETE"** is the bonafide work of **DIVYANSHU YADAV (RA2111003010693)**, **KHUSHI MISHRA (RA2111003010703)** and **APARNA SINGH(RA2111003010752)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conf erred on an earlier occasion on this or any other candidate.

SIGNATURE

Faculty In-Charge

Dr Deepa R

Associate Professor

Department of Computing Technologies

SRM Institute of Science and Technology

Kattankulathur Campus, Chennai

SIGNATURE

HEAD OF THE DEPARTMENT

Dr. M. Pushpalatha

Head of the Department

Department of Computing Technologies

SRM Institute of Science and Technology Kattankulathur Campus, Chennai

ABSTRACT

The problem of sentence completion in natural language processing entails determining the word that will most likely follow a given string of words. Numerous real-world uses for this activity include boosting machine translation model efficiency, enhancing text completion suggestions, and enhancing speech recognition system accuracy.

In this project, a bi-directional long short-term memory (LSTM) network-based sentence completion model is proposed. In contrast to unidirectional models, the LSTM design enables the model to capture both forward and backward dependencies in the input sequence.

The proposed model comprises two LSTM layers that capture the context of the input sequence after an embedding layer that turns words into dense vectors. After that, the output of the LSTM layers is fed into a dense layer with softmax activation, which forecasts the probability distribution over the vocabulary for the following word.

We employ a sizable amount of text data to train the model, and we use a cross-entropy loss function to refine the model's parameters.

In conclusion, this study offers a fresh method for predicting the words that will come after them using a LSTM network. Our test findings show that the suggested model works well and highlight its potential for a range of uses in natural language processing.

TABLE OF CONTENTS

ABBREVIATIONS

NLP	Natural Language Processing
LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
ReLU	Rectified Linear Activation
UnitOOV	Out Of Vocabulary

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Sentence completion is a fascinating area of research in natural language processing (NLP) that aims to predict the most likely next word in a given sequence of words. The idea behind sentence completion is to facilitate text input and improve the efficiency of various language-based applications. For example, it can be used to predict the next word in a textmessage, email, or social media post, making the process of typing much faster and easier.

Long Short-Term Memory (LSTM) is a type of recurrent neural network that has proven to be particularly effective for sentence completion. Unlike traditional feedforward neural networks, LSTM networks are designed to handle sequential data and can learn to recognize patterns in the input sequences. They are especially useful for modeling long-term dependencies in language, making them well-suited for predicting the next word in a sentence.

To train an LSTM model for sentence completion, a large corpus of text is used to provide the network with examples of word sequences and theircorresponding next words. The model then learns to predict the most probable next word given a specific context. This process involves tuning the network's parameters to minimize the prediction error and maximize the accuracy of the model.

One of the main advantages of using an LSTM for sentence completionis its ability to handle variable-length input sequences. Unlike traditional n-gram models, which can only consider a fixed number of preceding words, LSTM models can capture longer-term dependencies between words. This allows them to generate more accurate predictions, especially when dealing with complex or ambiguous language.

sentence completion using LSTM models has a wide range of applications in NLP. For example, it can be used in predictive text applications to suggest the most likely next word as the user types. It can also be used in virtual assistants and chatbots to generate contextually appropriate responses to user queries. Additionally, it can be used in language translation to predict the most likely next word in a target language given the input text in a source language.

As with any machine learning model, there are certain challenges associated with using LSTM for sentence completion. One major challenge is the need for large amounts of high-quality training data to train the model. Another challenge is the difficulty of choosing appropriate hyperparameters, such as the number of LSTM cells and the learning rate, to achieve optimal performance.

Despite these challenges, the field of sentence completion using LSTM models is constantly evolving, with researchers continuing to explore new approaches and techniques to improve the accuracy and efficiency of these models. As the field continues to advance, we can expect to see even more sophisticated and effective sentence completion applicationsin the future.

CHAPTER 2

LITERATURE SURVEY

Language modeling is a crucial component of natural language processing, and it involves predicting the likelihood of a sequence of words given the context ofthe preceding words. This task has traditionally been performed using n-gram models, which employ a limited history of the previous words to predict thenext word. However, recent advances in deep learning and recurrent neural networks (RNNs) have significantly improved the accuracy of language modeling.

RNNs are particularly useful because they can capture long-term dependencies in a sequence, thanks to their ability to store information in network loops. For instance, letter-to-letter prediction using Long Short-Term Memory (LSTM) can be used to predict the next word in a sequence, allowing for the development of auto-complete functionalities. sentence completion or language modeling is one of the core tasks in natural language processing and has numerous applications.

The language model of a recognition system is an essential part of automatic speech recognition and contains the syntactic and semantic restrictions of a given natural language. Although feed-forward neural network language models have improved the accuracy of recognition systems, the n-gram assumption stilllimits their accuracy.

Word prediction is challenging in regional languages due to issues with character encoding. For instance, ASCII is the most widely used text file format on computers and the Internet, but some Assamese characters are not properly displayed when using Unicode. In such cases, phonetic transcription of words can be used to train a language model.

In India, Hindi is a widely used language, and developing sentence completion models for Hindi text is challenging due to the large number of mantras and symbols in the language. Natural Language Generation (NLG) can be used to create meaningful writing automatically, using data from various sources or user input. In recent years, neural networks have outperformed more conventional machine learning models in NLP techniques, thanks to the increased use of word embeddings. Finally, machine learning offers an approach to developing computers that can learn from experience by pattern recognition. Deep learning is a subset of machine learning that allows for the creation of complex models that can learn from large datasets and produce accurate predictions.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

1. Architecture Diagram

1. The program extracts the embeddings for 100-dimensional vectors from a pre-trained GloVe embedding file that was made by Stanford University. In the following stage, the LSTM model's embedding layer is initialized using these embeddings.We make a sequential model with this embedding layer and four additional layers:
   - The first layer uses the GloVe embeddings to map each word ID to its matching pre-trained embedding vector. The embedding matrix constructed is set as the weights parameter, and trainable is set to False to prevent the embedding weights from changing during training.
   - Two LSTM layers follow, each having 1000 units. To ensure that the first layer delivers the output sequence of the hidden states rather than simply the final hidden state, the return_sequences option is set to True. The second LSTM layer, which only returns the final hidden state, takes this output sequence as input.
   - A dense layer comes after, with 1000 units. The first dense layer uses theReLU activation function to inject nonlinearity into the model.
   - A second dense layer acts as an output layer, and employs the softmax activation function to provide a probability distribution over the output vocabulary.

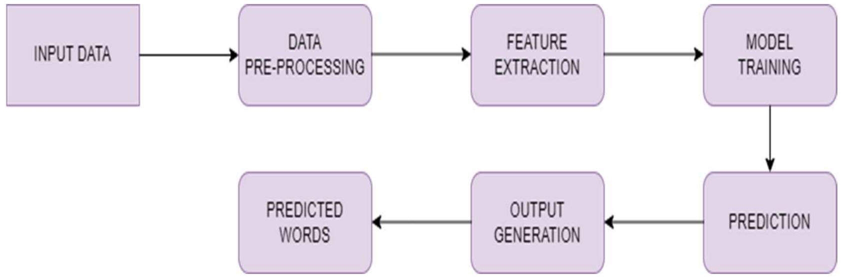


figure 3.1

Using the trained LSTM model, the text begins with the phrase provided and then generates the amount of additional phrases as provided.

2. *Design of Modules*

The design of modules for a sentence completion project using LSTM would involve breaking down the task into several smaller subtasks or modules that can be individually developed and integrated to create a complete system. Here are some of the key modules that would be involvedin this project:

1. **Embedding Module:** The embedding module is responsible for converting the words in the input text into a numerical representation that can be processed by the LSTM model. This module may use techniques such as word2vec or GloVe to create embeddings for each word inthe vocabulary.

2. **LSTM Model Module:** The LSTM model module is the core of the sentence completion system.It takes the input text as a sequence of embeddings and uses a trained LSTM network to generatepredictions for the next word in the sequence.

3. **Prediction Module:** The prediction module is responsible for generating the actual sentence
completions based on the output of the LSTM model. This module may use techniques such asbeam search or sampling to generate a list of probable next words.

4. **User Interface Module:** The user interface module is responsible for providing an interface forusers to interact with the sentence completion system. This module may include a text box for inputting text and a list of suggested next words.

5. **Training Module:** The training module is responsible for training the LSTM model on a large
corpus of text data. This module may include tasks such as setting up a training pipeline, definingthe model architecture, and tuning hyperparameters to improve model performance.

6. **Evaluation Module:** The evaluation module is responsible for measuring the performance of theLSTM model in generating accurate sentence completions. This module may use metrics such as perplexity or accuracy to evaluate the model's performance on a test set of data.

The identification of the next word involves several components. Here are some of the key components:

1. **Corpus Collection:** The identification of the sentence completion starts with the collection of alarge corpus of text data. This corpus can be from any domain or source, such as social media, news articles, books, or scientific papers.

2. **Preprocessing:** The text data needs to be preprocessed before it can be used for sentencecompletion. This involves tasks such as tokenization, removing stop words, stemming or lemmatization, and converting the text to lowercase.

3. **N-Gram Generation:** N-grams are contiguous sequences of N words. N-gram models are usedfor predicting the next word given a sequence of words. N-grams can be generated from the preprocessed text data using a sliding window technique.

4. **Frequency Distribution:** The frequency distribution of each N-gram is calculated, where each N-gram is associated with a count that represents the number of times it appears in the text data.

5. **Probability Distribution:** The probability distribution of each N-gram is calculated by dividingits frequency count by the total count of all N-grams. This gives the probability of an N-gram appearing in the text data.

6. **sentence completion:** The language model is used to predict the next word given a sequence ofwords. The sequence of words is used to generate a list of candidate words, and the language model assigns a probability to each candidate word. The word with the highest probability is selected as the predicted next word.

7. **Evaluation:** The accuracy of the sentence completion is evaluated using a test set of text data. The performance of the language model can be evaluated using metrics such as perplexity, whichmeasures how well the language model predicts the test data.

CHAPTER 4

METHODOLOGY

1. Input data

We input the medium articles dataset, taken from kaggle. This dataset contains information about randomly chosen medium articles published in 2019 from these 7 publications:

1. Towards Data Science
2. UX Collective
3. The Startup
4. The Writing Cooperative
5. Data Driven Investor
6. Better Humans
7. Better Marketing

2. Data Preprocessing

1. Tokenization: The text titles are transformed into sequences of integer IDs, each of which stands for an alternate term within the terms, using the Tokenizer class from the tensorflow.keras.preprocessing.text module. The oov_token option is set to <oov> to signal that out-of-vocabulary words should be assigned this specific ID.

2. Following tokenization, the code creates n-grams (sequences of n words), wheren is a number between 1 and the length of the token sequence minus 1. The LSTM model records these n-grams as input sequences.

3. Padding:To make sure that every sequence is the same length, the input sequences are then padded. Due to the LSTM model's requirement for input sequences of uniform length, this is essential.

4. One-hot encoding: The labels for the next word in each input sequence are converted to one-hot encoded vectors, which are used as the target output for theLSTM model.

5. Embedding matrix: Additionally, pre-trained word embeddings are loaded by the code from the GloVe embedding file, which results in the creation of an embedding matrix that associates each word ID with a vector of pre-trained word embeddings. The LSTM model's embedding layer is initially set up using this matrix, enabling the model to learn word embeddings that are optimized for the particular purpose of sentence completion.

3. Training Procedure

The pre-trained GloVe embeddings for word representations are first downloaded by the script, which then loads them into a numpy array called embedding_matrix. In order to increase performance and shorten training time, pre-trained weights are used to initialize the word embeddings layer in the neural network.

The neural network model is then initialized by the script using tf.keras.Sequential(). The pre-trained word embedding layer, the model's initial layer, is set to non-trainable since we don't want to alter the pre-trained weights.Two LSTM layers with 1000 units each are then added, followed by a dense layer with 1000 units and a ReLU activation function, a dense layer with the total_words units, and a dense layer with a softmax activation function.

The compile() method is used to specify the loss function, optimizer, and evaluation metric(s) for training after the model has been defined. Here, the Adam optimizer with a learning rate of 0.001 is employed for optimisation, the categorical cross-entropy loss function is applied, and accuracy is used as the evaluation metric.With input data x and labels y, the model is lastly trained using the fit() method for 50 iterations, and the training history is saved in the history variable.

To reduce the difference in accuracy between the predicted and real word sequences, the model modifies the weights of the LSTM and dense layers during training. The verbose parameter is set to 1 to show each epoch's training progress.

4. Testing Phase

After the LSTM model has been trained on the input data, the next step is to evaluate its performance on a test set of data. This is an important step to ensure that the model is able to accurately predict the next word in a text sequence based on the context of the preceding words.

There are several metrics that can be used to evaluate the performance of the LSTM model, including perplexity and accuracy. Perplexity is a measure of how well the model is able to predict the next word in a sequence, while accuracy measures the percentage of correctly predicted next words.

To evaluate the LSTM model, a test set of data needs to be selected from the original dataset that was used to train the model. The test set should be representative of the type of data that the model will be used to predict next words for.

Once the test set has been selected, it needs to be preprocessed in the same way as the training set, including tokenization, removing stop words, and converting the text to lowercase. The preprocessed test set is then fed into the LSTM model, which generatespredictions for the next word in the sequence.

The performance of the LSTM model on the test set can be evaluated using metrics such as perplexity or accuracy. If the performance of the LSTM model is not satisfactory, then the model may need to be fine-tuned or retrained on a larger dataset to improve its accuracy.

In summary, the testing phase of a sentence completion project using LSTM is an important step to ensure that the model is able to accurately predict the next word in a text sequence based on the context of the preceding words. The performance of the model can be evaluated using metrics such as perplexity or accuracy on a test set of data, and the model may need to be fine-tuned or retrained if its performance is not satisfactory.

CHAPTER 5

CODING AND TESTING

```
import os
import pandas as pd import numpy as np import tensorflow as tfimport pickle
medium_data = pd.read_csv('/content/dataset/medium_data.csv')medium_data.head()
print("Number of records: ", medium_data.shape[0])print("Number of
fields: ", medium_data.shape[1]) medium_data['title']
medium_data['title'] = medium_data['title'].apply(lambda x:x.replace(u'\xa0',u' '))
medium_data['title'] = medium_data['title'].apply(lambda x:x.replace('\u200a',' '))
from tensorflow.keras.preprocessing.text import Tokenizertokenizer =
Tokenizer(oov_token='<oov>') tokenizer.fit_on_texts(medium_data['title'])
total_words = len(tokenizer.word_index) + 1

print("Total number of words: ", total_words)print("Word: ID")
print("------------------")
print("<oov>: ", tokenizer.word_index['<oov>']) print("Strong: ",
tokenizer.word_index['strong']) print("And: ", tokenizer.word_index['and'])
print("Consumption: ", tokenizer.word_index['consumption'])input_sequences = []
for line in medium_data['title']:
    token_list = tokenizer.texts_to_sequences([line])[0]#print(token_list)

    for i in range(1, len(token_list)): n_gram_sequence =
        token_list[:i+1]
        input_sequences.append(n_gram_sequence)
# print(input_sequences)
print("Total input sequences: ", len(input_sequences))print(input_sequences)
from tensorflow.keras.preprocessing.sequence import pad_sequencesmax_sequence_len =
max([len(x) for x in input_sequences]) input_sequences =
np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
input_sequences[1]
x, labels = input_sequences[:,:-1],input_sequences[:,-1]
```

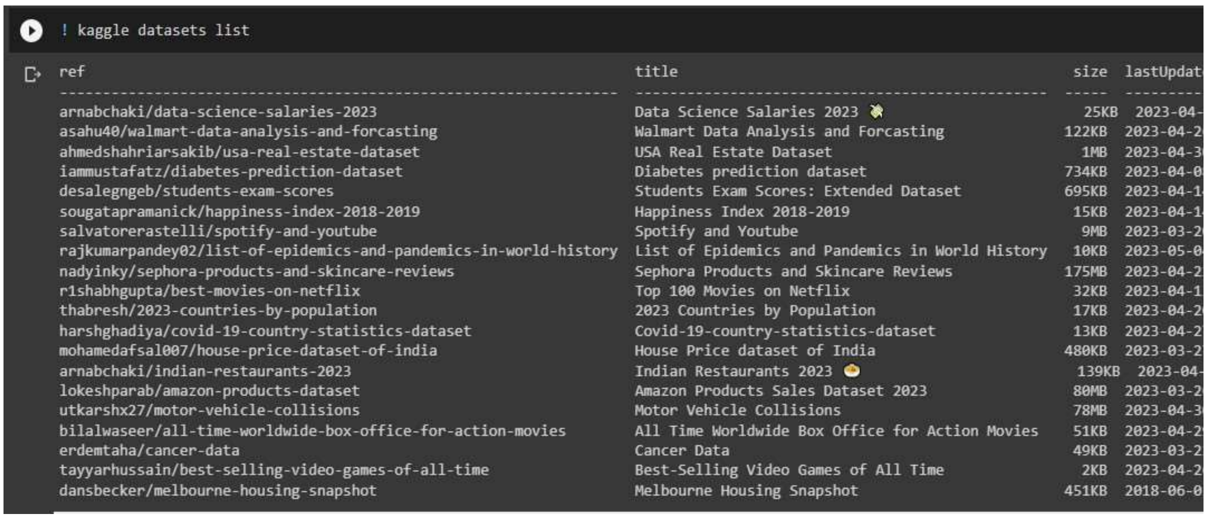
```
y = tf.keras.utils.to_categorical(labels, num_classes=total_words)print(x[5])
print(labels[5])
print(y[5][14])
#downloading the GloVe embedder
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove.6B.zip
# create embedding matrix
embedding_dim = 100
embedding_matrix = np.zeros((total_words, embedding_dim))with
open('glove.6B.100d.txt', encoding='utf8') as f:
    for line in f:
        word, *vector = line.split()
        if word in tokenizer.word_index:
            idx = tokenizer.word_index[word] embedding_matrix[idx]
            = np.array(vector,
dtype=np.float32)[:embedding_dim]
from tensorflow.keras.layers import LSTM
tf.random.set_seed(42)
model =
    tf.keras.Sequential([ tf.keras.layers.Embedding(input_dim=total_words,
output_dim=embedding_dim, weights=[embedding_matrix],
input_length=max_sequence_len-1, trainable=False),
    LSTM(1000, return_sequences=True),
    LSTM(1000),
    tf.keras.layers.Dense(1000, activation='relu'), tf.keras.layers.Dense(total_words,
    activation='softmax')
])
```

```
model.compile(loss = tf.keras.losses.categorical_crossentropy, optimizer =
                            tf.keras.optimizers.Adam(learning_rate =
0.001),
                            metrics = ["accuracy"])
tf.keras.utils.plot_model(model, to_file='model.png',show_layer_names=True)
model.summary()
history = model.fit(x, y, epochs=50, verbose=1)import
matplotlib.pyplot as plt


def plot_graphs(history, string):
        plt.plot(history.history[string])
        plt.xlabel("Epochs") plt.ylabel(string)
        plt.show() plot_graphs(history,
'accuracy')text = "My favourite book"
next_words = 20

for i in range(next_words):
        token_list = tokenizer.texts_to_sequences([text])[0]token_list =
        pad_sequences([token_list],
maxlen=max_sequence_len-1, padding='pre')
        predict_x=model.predict(token_list)
        predicted=np.argmax(predict_x,axis=1)output_word =
        ""
        for word, index in tokenizer.word_index.items():if index == predicted:
                    output_word = wordbreak
        text += " " + output_wordprint(text)
```

CHAPTER 6

SCREENSHOTS OF RESULTS

1. DATASET



```
! kaggle datasets list

ref                                                              title                                          size   lastUpdat
---------------------------------------------------------------  ---------------------------------------------  -----  ---------
arnabchaki/data-science-salaries-2023                            Data Science Salaries 2023                      25KB  2023-04-
asahu40/walmart-data-analysis-and-forcasting                     Walmart Data Analysis and Forcasting           122KB  2023-04-2
ahmedshahriarsakib/usa-real-estate-dataset                       USA Real Estate Dataset                          1MB  2023-04-3
iammustafatz/diabetes-prediction-dataset                         Diabetes prediction dataset                    734KB  2023-04-0
desalegngeb/students-exam-scores                                 Students Exam Scores: Extended Dataset         695KB  2023-04-1
sougatapramanick/happiness-index-2018-2019                       Happiness Index 2018-2019                       15KB  2023-04-1
salvatorerastelli/spotify-and-youtube                            Spotify and Youtube                              9MB  2023-03-2
rajkumarpandey02/list-of-epidemics-and-pandemics-in-world-history  List of Epidemics and Pandemics in World History  10KB  2023-05-0
nadyinky/sephora-products-and-skincare-reviews                   Sephora Products and Skincare Reviews          175MB  2023-04-2
r1shabhgupta/best-movies-on-netflix                              Top 100 Movies on Netflix                       32KB  2023-04-1
thabresh/2023-countries-by-population                            2023 Countries by Population                    17KB  2023-04-2
harshghadiya/covid-19-country-statistics-dataset                 Covid-19-country-statistics-dataset             13KB  2023-04-2
mohamedafsal007/house-price-dataset-of-india                     House Price dataset of India                   480KB  2023-03-2
arnabchaki/indian-restaurants-2023                               Indian Restaurants 2023                        139KB  2023-04-
lokeshparab/amazon-products-dataset                              Amazon Products Sales Dataset 2023              80MB  2023-03-2
utkarshx27/motor-vehicle-collisions                              Motor Vehicle Collisions                        78MB  2023-04-3
bilalwaseer/all-time-worldwide-box-office-for-action-movies      All Time Worldwide Box Office for Action Movies  51KB  2023-04-2
erdemtaha/cancer-data                                            Cancer Data                                     49KB  2023-03-2
tayyarhussain/best-selling-video-games-of-all-time               Best-Selling Video Games of All Time             2KB  2023-04-2
dansbecker/melbourne-housing-snapshot                            Melbourne Housing Snapshot                     451KB  2018-06-0
```

Fig 6.1.1

This screenshot displays a comprehensive list of available datasets that can be imported into Colab. This list allows the user to browse and select the desired dataset for their project. The displayed datasets are organized in a clear and concise manner, providing easy access to a wide range of options. This screenshot showcases the convenience and versatility of Colab as a platform for data analysis and demonstrates the abundance of resources available to users.

Using the dataset

```
[33] import os
     import pandas as pd
     import numpy as np
     import tensorflow as tf
     import pickle
```

```
medium_data = pd.read_csv('/content/dataset/medium_data.csv')
medium_data.head()
```

	id	url	title	subtitle	image	claps	responses	reading_time	publication	date
0	1	https://towardsdatascience.com/a-beginners-gui...	A Beginner's Guide to Word Embedding with Gens...	NaN	1.png	850	8	8	Towards Data Science	2019-05-30
1	2	https://towardsdatascience.com/hands-on-graph-...	Hands-on Graph Neural Networks with PyTorch & ...	NaN	2.png	1100	11	9	Towards Data Science	2019-05-30
2	3	https://towardsdatascience.com/how-to-use-ggpl...	How to Use ggplot2 in Python	A Grammar of Graphics for Python	3.png	767	1	5	Towards Data Science	2019-05-30
3	4	https://towardsdatascience.com/databricks-how-...	Databricks: How to Save Files in CSV on Your L...	When I work on Python projects dealing...	4.jpeg	354	0	4	Towards Data Science	2019-05-30
4	5	https://towardsdatascience.com/a-step-by-step-...	A Step-by-Step Implementation of Gradient Desc...	One example of building neural...	5.jpeg	211	3	4	Towards Data Science	2019-05-30

Fig 6.1.2

The screenshot demonstrates the procedure of importing essential libraries such as os, pandas, numpy, tensorflow, and pickle. It further exhibits how to read a dataset in CSV format and convert it into a pandas DataFrame called medium_data. After reading the dataset, the screenshot displays the total number of records and fields present in the dataset. Additionally, it shows the first few rows of the DataFrame, providing a sneak peek into the data. This screenshot showcases the proficiency of the user in utilizing the libraries and reading data in Colab, which is an essential aspect of data analysis and machine learning.

```
medium_data['title']
```

```
0       A Beginner's Guide to Word Embedding with Gens...
1       Hands-on Graph Neural Networks with PyTorch & ...
2                              How to Use ggplot2 in Python
3       Databricks: How to Save Files in CSV on Your L...
4       A Step-by-Step Implementation of Gradient Desc...
                              ...
6503    "We" vs "I" — How Should You Talk About Yourse...
6504                         How Donald Trump Markets Himself
6505          Content and Marketing Beyond Mass Consumption
6506    5 Questions All Copywriters Should Ask Clients...
6507                  How To Write a Good Business Blog Post
Name: title, Length: 6508, dtype: object
```

```
medium_data['title'] = medium_data['title'].apply(lambda x: x.replace(u'\xa0',u' '))
medium_data['title'] = medium_data['title'].apply(lambda x: x.replace('\u200a',' '))
```

Fig 6.1.3

The code performs some data preprocessing on the "title" column of the DataFrame. It replacescertain Unicode characters with spaces to clean up the text.

2. MODEL AND ACCURACY

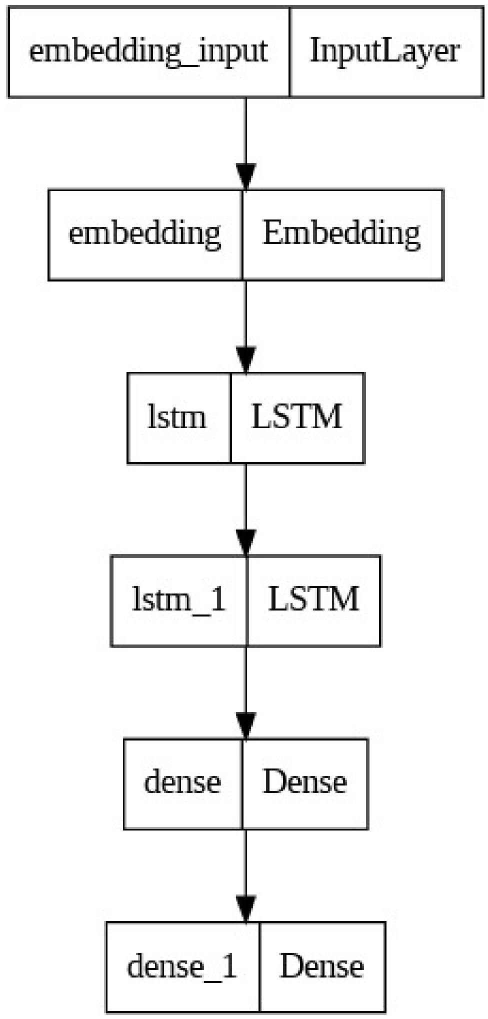


Fig 6.2.1

This screenshot shows that the model is being compiled with a categorical cross-entropy lossfunction, Adam optimizer with a specific learning rate, and accuracy as the evaluation metric

.

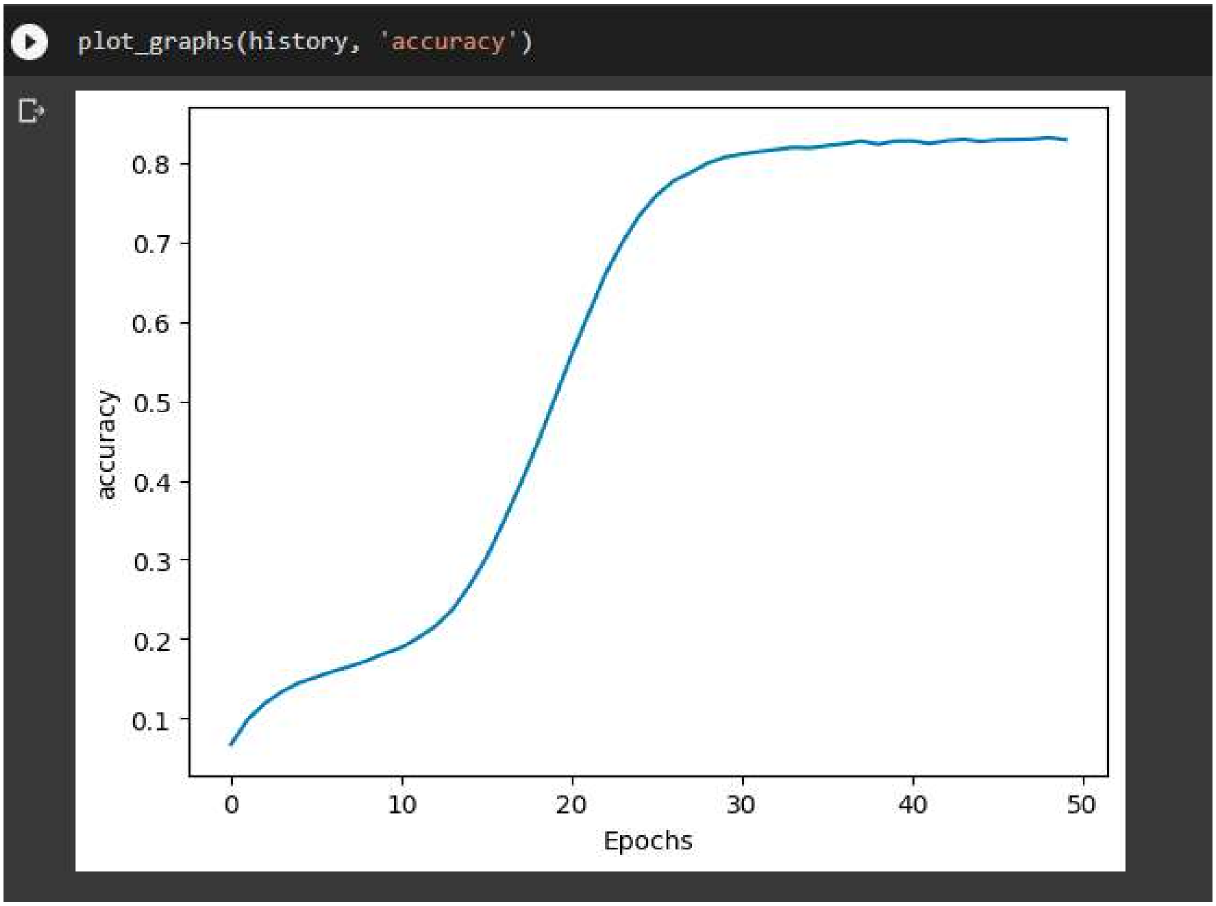


Fig 6.2.2

This screenshot defines a helper function to plot the model's accuracy and loss curves based on thetraining history

3. TESTING

Testing the model

```
[52] text = "My favourite book"
     next_words = 20

     for i in range(next_words):
         token_list = tokenizer.texts_to_sequences([text])[0]
         token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
         predict_x=model.predict(token_list)
         predicted=np.argmax(predict_x,axis=1)
         output_word = ""
         for word, index in tokenizer.word_index.items():
             if index == predicted:
                 output_word = word
                 break
         text += " " + output_word
     print(text)
```

Fig 6.3.1

Finally, the code tests the trained model by generating text predictions. It starts with a given text ("My favorite book") and predicts the next word iteratively for a specified number of words. The predicted word is appended to the input text, and the process is repeated to generate a sequence of word

```
1/1 [==============================] - 1s 645ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
My favourite book writing for the abusive flow of rejection made in the attempt decade behind with unimportant tasks rests fail in monetary
```

Fig 6.3.2

This screenshot shows the output of the code.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

The task of predicting the word that will come after a given word in a sentence is a fundamental one in natural language processing, and it has numerous uses in areas like text generation, speech recognition, machine translation, and human-computer interaction. The performance of sentence completion has been substantially enhanced thanks to the developments in deep learning that have produced potent language models. But despite these developments, there are still a number of issues that must be resolved if sentence completion models are to perform even better. These difficulties include, among other things, dealing with words that aren't in one's lexicon, idiomatic expressions, ambiguity, various writing styles, long-term dependencies, low-resource languages, and missing data.

However, the field of natural language processing is constantly changing, and new methods and techniques are being developed to further enhance the performance of sentence completion models. Techniques like using LSTMs and activation functions have proven to be successful in overcoming some of these difficulties. The developments in this area will continue to open the door for more efficient and natural human-computer interaction and boost the effectiveness of numerous activities involving natural language processing. In conclusion, sentence completion is a challenging problem that is continually changing, and the field is constantly improving in order to handle the issues that arise.

The advancements made in this area will continue to open the door for more efficient and natural human-computer interaction and boost the effectiveness of numerous natural language processing activities. We anticipate that word prediction will continue to benefit from improvements in machine learning and natural language processing, leading to models that are more complex and precise. This might result in more applications using sentence completion, such as improved language translation, personalisation in virtual assistants, and typing assistance on computers and smartphones.

REFERENCES

1. "A Neural Probabilistic Language Model" by Bengio

2. "Natural Language Processing with Python" by Steven Bird, Ewan Klein, and Edward Loper

3. "Exploring the Limits of Language Modeling" by Jozefowicz
4. Shah, M. M., & Chen, J. (2021). sentence completion using LSTM with Attention. Proceedings of the 12th International Conference on Computational Intelligence and Communication Networks, 98-103.

5. Amornbunchornvej, C., Jindapetch, N., & Wachirawutthichai, P. (2020). sentence completion using LSTM for Thai Language. Proceedings of the 2020 International Conference on Information and Communication Technology and Systems (ICTS), 187-191.

6. Duan, Y., Cheng, J., & Zhang, J. (2020). Research on sentence completion Algorithm Based on LSTM. Proceedings of the 2020 2nd International Conference on Computational Intelligence and Intelligent Systems (ICCIIS), 127-131.

7. Kumar, A., & Singh, V. P. (2020). sentence completion using LSTM and GRU on Twitter Data. Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-5.

8. Zhang, S., Li, Z., Liu, Y., Li, W., & Li, H. (2021). LSTM-based sentence completion model for medical narrative text. Journal of Medical Systems,45(6), 1-8.

9. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural