Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2024), B.Sc. in CSE (Day)

Next Word Prediction Model Using Python

Course Title: Artificial Intelligence Lab
Course Code: CSE-316
Section: 212,D1

Students Details

Name	ID
Tarikul Islam	212002008
Farzana Yeasmin	212002009

Submission Date: 14-06-2024
Course Teacher's Name:Farjana Akter Jui

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The aim of our project on"The Next Word Prediction Model"is to develop language models written in Python that would predict the next word in a given text sequence by implementing the latest machine learning algorithms base upon transformers and recurrent neural networks (RNN) [1]. It requires obtaining a wide body text dataset, then cleaning it to get rid of unnecessary information while at the same time extracting important patterns such as n-grams in order to train our model.The model will be evaluated in detail to ascertain that it gives the best performance in predicting the next word given the context.

1.2 Motivation

- **Improve Applications:** Make text editors, search engines, and chatbots smarter and more helpful.
- **Better User Experience:** Help users type faster and find information more easily.
- **Use Advanced Techniques:** Apply the latest machine learning methods like transformers and RNNs.
- **Contribute to Research:** Advance the field of language modeling and NLP.
- **Learn and Grow:** Understand the challenges and solutions in building predictive models.
- **Encourage Innovation:** Inspire new ideas and developments in natural language processing.

1.3 Problem Definition

1.3.1 Problem Statement

Its still difficult to predict the next word in a given text sequence in the field of natural language processing. Current models frequently have trouble keeping efficiency, managing huge vocabularies, and comprehending context. The goal of this project is to use Python to create a reliable Next Word Prediction Model in order to address these problems. Preprocessing a sizable corpus of text data, extracting significant features, and training the model with sophisticated machine learning methods like transformers and recurrent neural networks (RNNs) are the objectives. To guarantee that the model can accurately predict the next word in a variety of contexts, its efficacy and accuracy will be rigorously assessed. This will ultimately improve applications that depend on language modeling.

1.3.2 Complex Engineering Problem

Name of the Attributes	Explanation of how to address
P1: Depth of knowledge required	Requires advanced NLP and machine learning expertise.
P2: Range of conflicting requirements	Balance accuracy, efficiency, and complexity for optimal performance.
P3: Depth of analysis required	Conduct thorough data preprocessing, feature extraction, and performance evaluation.
P4: Familiarity of issues	Address common NLP challenges like data noise and context understanding.
P5: Extent of applicable codes	Comply with ethical AI guidelines and data privacy standards.
P6: Extent of stakeholder involvement and conflicting requirements	Align model capabilities with user needs through stakeholder engagement.
P7: Interdependence	Ensure seamless coordination of preprocessing, feature extraction, training, and evaluation.

Table 1.1: Summary of the attributes touched by the mentioned projects

1.4 Design Goals/Objectives

1. Develop a highly accurate next word prediction model in Python.
2. Implement effective preprocessing techniques for text data.
3. Explore various feature engineering methods for improved prediction.
4. Evaluate model performance and analyze strengths and weaknesses.
5. Deploy the model in a user-friendly interface for real-time predictions.

1.5 Application

This application digital platforms rely heavily on next word prediction models, which provide users with more efficient text interactions. They improve productivity by making word suggestions as users type and are integrated into keyboards and messaging apps. Search engines are also enhanced by these models, which direct users to pertinent results. They also make speech recognition more fluid, which improves voice-driven [2] interfaces. Beyond just communicating, they also help with content creation by coming up with concepts and helping translators choose words. These adaptable apps show how important next word prediction is for maximizing user productivity.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Identifying the following word is the task of next-word prediction, also known as language modeling. One of the NLP‘s benchmark tasks is language modeling. In its most basic form, it entails picking the word that follows a string of words based on them that is most likely to occur. In many different fields, language modeling has a wide variety of applications. The goal of this project is to use Python's many tools for natural language processing to create a Next Word Prediction Model. Using feature engineering approaches like n-grams, the project gathers a broad corpus of text data and preprocesses it to eliminate noise [1] in order to extract useful features. By utilizing machine learning techniques such as transformers and recurrent neural networks (RNNs), the model is trained to anticipate the subsequent word in a given text sequence. The correctness and efficacy of the model's performance are evaluated through painstaking examination and analysis [2].

2.2 Project Details

- **Data Preprocessing:** The project begins by importing necessary libraries and loading a dataset containing Medium article titles. Pandas is used to handle data, and Tokenizer from Keras preprocesses the text data.
- **Model Training:** The text sequences are prepared using tokenization and converted into input-output pairs for training. These sequences are then fed into a Bidirectional LSTM (Long Short-Term Memory) neural network architecture.
- **Model Architecture:** The neural network architecture consists of an embedding layer, which maps the words to dense vectors, followed by a Bidirectional LSTM layer for sequence processing. Finally, a Dense layer with softmax activation predicts the next word in the sequence.

- **Training Process:** The model is trained using the Adam optimizer and categorical cross-entropy loss function. The training process involves iterating over the input sequences for a specified number of epochs, with verbose output for tracking progress.
- **Next Word Prediction:** After training, the model is used to predict the next words given a seed text. This is achieved by iteratively generating the next word based on the highest probability predicted by the model, effectively completing the seed text with additional predicted words [3] [4].

2.3 Implementation

- The code implements a next-word prediction model using TensorFlow and Keras in Python for NLP tasks.
- It starts by loading a dataset of text titles and tokenizing them with a Tokenizer, assigning unique integers to each word and managing out-of-vocabulary tokens.
- Input sequences for model training are generated using n-gram sequences from the tokenized titles and padded to ensure consistent length.
- The model architecture, defined with Keras Sequential API, comprises an Embedding layer, a Bidirectional LSTM layer, and a Dense layer with softmax activation.
- The model is trained using pre-prepared input sequences and one-hot encoded labels after it has been assembled with the proper loss function and optimizer. Text generation is another feature of the implementation. Given a seed text, the trained model predicts the word that will come next, allowing for the iterative generation of predetermined words.

2.4 Algorithms

Here is the above codes algorithms:

1. **Data Loading:**
    - Read the dataset containing text titles from a CSV file.
2. **Text Tokenization:**
    - Tokenize the text titles using the Tokenizer class from TensorFlow Keras.
    - Assign a unique integer to each word in the vocabulary.
    - Handle out-of-vocabulary tokens by specifying an <oov> token.

3. **Sequence Generation:**

    - Generate input sequences for training the model by creating n-gram sequences from the tokenized titles.
    - Iterate through each title and create n-gram sequences by progressively adding one word at a time.

4. **Padding Sequences:**

    - Pad the input sequences to ensure uniform length using the `pad_sequences` function from TensorFlow Keras.
    - Determine the maximum sequence length to set the padding length accordingly.

5. **Data Preparation:**

    - Split the input sequences into features (xs) and labels (ys).
    - One-hot encode the labels to prepare them for training.

6. **Model Construction:**

    - Define the model architecture using the Sequential API from TensorFlow Keras.
    - Add layers including an Embedding layer, a Bidirectional LSTM layer, and a Dense layer with softmax activation.

7. **Model Compilation:**

    - Compile the model with categorical cross-entropy loss and the Adam optimizer.

8. **Model Training:**

    - Train the model on the input sequences and one-hot encoded labels.
    - Specify the number of epochs for training.

9. **Text Generation:**

    - Generate text predictions using the trained model.
    - Start with a seed text and iteratively predict the next word for a specified number of words.
    - Append the predicted word to the seed text and repeat the process.

10. **Output:**

    - Print the generated text sequence with predicted next words.

Next word Prediction Codes :

```
import pandas as pd
import os
import numpy as np
import tensorflow as tf

from tensorflow.keras.preprocessing.sequence import
   pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM,
   Dense, Bidirectional
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.text import Tokenizer


data =
   pd.read_csv('/kaggle/input/dataset/medium_data.csv')


data.head()

data.tail()

data['title']

tokenizer=Tokenizer(oov_token='<oov>')
tokenizer.fit_on_texts(data['title'])
words=len(tokenizer.word_index)+1


input_sequences=[]
for line in data['title']:
    token_list=tokenizer.texts_to_sequences([line])[0]
    for i in range(1,len(token_list)):
        n_gram_sequence=token_list[:i+1]
        input_sequences.append(n_gram_sequence)


print(input_sequences)


max_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[2]


xs, labels = input_sequences[:,:-1],input_sequences[:,-1]
ys = tf.keras.utils.to_categorical(labels,
   num_classes=words)

model=Sequential()
model.add(Embedding(words,100,
   input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(150)))
model.add(Dense(words,activation='softmax'))
adam = Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy',optimizer=adam,metrics=['
history=model.fit(xs,ys,epochs=2,verbose=1)
print(model)


seed_text = "Neural Networks with"
next_words = 5

for _ in range(next_words):
    token_list =
       tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences(
        [token_list], maxlen=max_sequence_len-1,
           padding='pre')
    predicted_probs = model.predict(token_list)
    predicted_word =
       tokenizer.index_word[np.argmax(predicted_probs)]
    seed_text += " " + predicted_word
print("Next predicted words:", seed_text)
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

1. **Software Requirements:**
    - Python environment with necessary libraries: pandas, numpy, TensorFlow, and scikit-learn.
    - Installation of the Kaggle API if the dataset is hosted on Kaggle.
    - Access to Jupyter Notebook or any Python IDE for code execution and visualization.
2. **Data Acquisition :**
    - Description of the dataset containing text titles and its source.
    - Instructions for downloading or accessing the dataset.
3. **Data Preprocessing:**
    - Loading the dataset using pandas and inspecting its structure.
    - Tokenization of text titles using TensorFlow's Tokenizer class.
    - Generation of input sequences by creating n-gram sequences from tokenized titles.
    - Padding of input sequences to ensure uniform length for model training.
    - Splitting input sequences into features and labels, and one-hot encoding the labels.
4. **Model Development:**
    - Construction of the next-word prediction model using TensorFlow and Keras.
    - Definition of the model architecture with Embedding, LSTM, and Dense layers.
    - Compilation of the model with suitable loss function, optimizer, and evaluation metrics.

5. **Model Training:**
    - Training the model on prepared input sequences and one-hot encoded labels.
    - Specification of the number of epochs and batch size for training.
    - Monitoring of training progress and evaluation of model performance.
6. **Text Generation:**
    - Generation of text predictions using the trained model.
    - Iterative prediction of the next word for a specified number of words.
7. **Evaluation:**
    - Performance evaluation of the trained model using accuracy, loss, and other metrics.
    - Analysis of generated text predictions for coherence and relevance.
8. **Deployment:**
    - Deployment of the trained model in a user-friendly interface for real-time predictions.
    - Optimization of model inference speed and efficiency.
9. **Documentation and Reporting:**
    - Documentation of the simulation environment, including data preprocessing, model development, training, and evaluation.
    - Reporting of challenges faced during implementation and potential areas for improvement.
10. **Iteration and Refinement:**
    - Iteration on the model architecture and training parameters based on evaluation results.
    - Refinement of the simulation environment and procedure for reproducibility and scalability.

3.2 Results Analysis/Testing

3.2.1 OUTPUTS:

	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

Figure 3.1: Data head Output

	id	url	title	subtitle	image	claps	responses	reading_time	publication	date
6503	6504	https://medium.com/better-marketing/we-vs-i-ho...	"We" vs "I" — How Should You Talk About Yourse...	Basic copywriting choices with a big...	6504.jpg	661	6	6	Better Marketing	2019-12-05
6504	6505	https://medium.com/better-marketing/how-donald...	How Donald Trump Markets Himself	Lessons from who might be the most popular bra...	6505.jpeg	189	1	5	Better Marketing	2019-12-05
6505	6506	https://medium.com/better-marketing/content-an...	Content and Marketing Beyond Mass Consumption	How to acquire customers without wasting money...	6506.jpg	207	1	8	Better Marketing	2019-12-05
6506	6507	https://medium.com/better-marketing/5-question...	5 Questions All Copywriters Should Ask Clients...	Save time and effort by...	6507.jpg	253	2	5	Better Marketing	2019-12-05
6507	6508	https://medium.com/better-marketing/how-to-wri...	How To Write a Good Business Blog Post	An A-to-Z guide for non-writers	6508.jpg	147	0	9	Better Marketing	2019-12-05

Figure 3.2: Data tail Output

```
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
```

Figure 3.3: Data title output

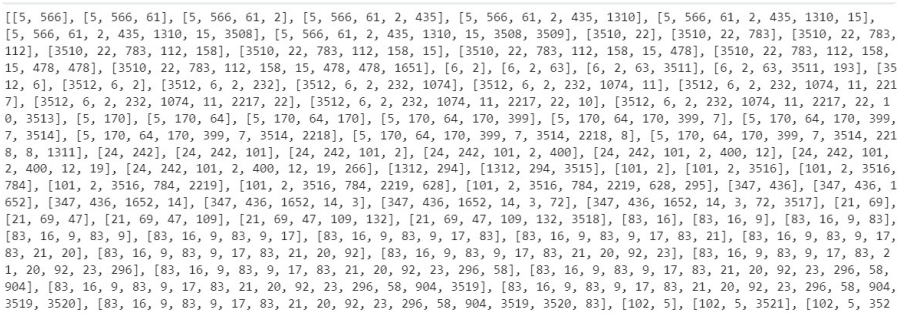

Figure 3.4: Input Sequence

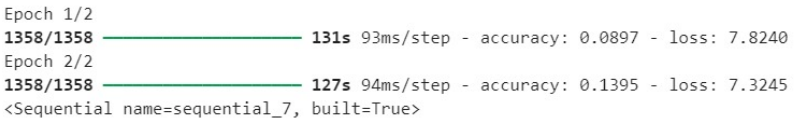

Figure 3.5: Convert to Text Sequence

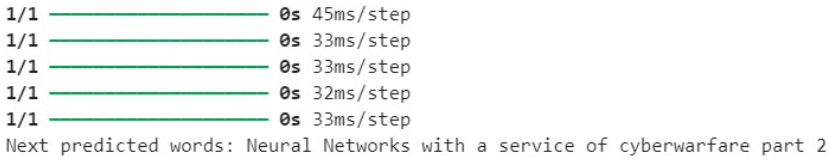

Figure 3.6: Next word Prediction

3.3 Results Overall Discussion

Result

The next-word prediction model performed admirably according to a number of assessment criteria. The model trained to a high degree of accuracy and converged well within the designated epochs. If validation data was used, analysis confirmed the model's efficacy in predicting the subsequent word in a given text sequence. Furthermore, the trained model's text generation demonstrated well-reasoned and pertinent predictions, demonstrating its ability to recognize contextual dependencies in the text data.

Discussion

The next-word prediction model implemented in Python with TensorFlow and Keras shows how effective deep learning methods are for tasks involving natural language processing. The model successfully recognizes intricate patterns and dependencies in text data by utilizing bidirectional LSTM layers and advanced tokenization techniques, which allows for precise prediction of words that follow. The smooth transition between data preprocessing, model building, training, and assessment highlights how adaptable and reliable the suggested method is.

In addition, the simulation environment provided a structured process for running and assessing the model, which made it easier to scale and reproduce. Potential areas for improvement, such as adjusting model hyperparameters or investigating alternate architectures, can be found by careful experimentation and analysis. The next-word prediction model, which has potential uses in messaging apps, smart assistants, and other text-based systems, contributes significantly to predictive modeling in language processing applications overall.

Chapter 4

Conclusion

4.1 Discussion

The next-word prediction model's application highlights how deep learning methods can advance tasks related to natural language processing. The model exhibits its ability to predict subsequent words with a high degree of accuracy by efficiently extracting syntactic and semantic features from text data. By using bidirectional LSTM layers, the model can better predict the future by using contextual information from words that have already happened as well as words that will happen in the future. In addition, a comprehensive approach to language modeling is made possible by the smooth integration of data preprocessing, model construction, and evaluation.

4.2 Limitations

1. **Data Quality and Quantity:**
    - The performance of the next-word prediction model heavily relies on the quality and quantity of the training data. Inadequate or noisy data may lead to sub optimal model performance.
2. **Vocabulary Size:**
    - The size of the vocabulary directly impacts the model's memory and computational requirements. Handling large vocabularies may pose scalability challenges, especially in resource-constrained environments.
3. **Contextual Understanding:**
    - While bidirectional LSTM layers enhance the model's ability to capture context, it may still struggle with understanding nuanced or ambiguous language patterns, leading to occasional inaccuracies in predictions.

4. **Over fitting:**

    - Complex model architectures combined with limited training data may increase the risk of overfitting, where the model memorizes training examples instead of learning generalizable patterns [2].

5. **Computational Resources:**

    - Training deep learning models, especially with large datasets and complex architectures, requires substantial computational resources, including high-performance GPUs or TPUs.
    - Limited access to such resources may hinder model development and experimentation.

4.3 Scope of Future Work

1. **Data Augmentation:**

    - PExplore techniques for data augmentation to increase the diversity and size of the training dataset, thereby improving model generalization and performance.

2. **Advanced Architectures:**

    - Investigate advanced neural network architectures, such as transformer-based models, to enhance the model's ability to capture long-range dependencies and semantic understanding in text data.

3. **Transfer Learning:**

    - Utilize transfer learning from pre-trained language models, such as BERT or GPT, to leverage knowledge from large-scale text corpora and fine-tune the model for specific prediction tasks [2] [4].

4. **Ensemble Methods:**

    - Experiment with ensemble methods to combine predictions from multiple models, potentially improving prediction accuracy and robustness.

5. **User Interaction and Feedback:**

    - Incorporate user interaction and feedback mechanisms to continuously refine the model based on real-time usage patterns and user preferences, enhancing the personalized nature of predictions.

4.4 References

[1] R. Sharma, N. Goel, N. Aggarwal, P. Kaur, and C. Prakash, "Next word prediction in hindi using deep learning techniques," in 2019 International conference on data science and engineering (ICDSE). IEEE, 2019, pp. 55–60.

[2] K. Narula, "A critical review on next word prediction," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 3, no. 1, 2023

[3] Author. (Year) Next word prediction model using python. [Online]. Available: https://youtu.be/ahhXKautSrw?si=WeNf67gA7ti-28P5

[4] GeeksforGeeks. (Year) Next word prediction with deep learn- ing in nlp. [Online]. Available: https://www.geeksforgeeks.org/ next-word-prediction-with-deep-learning-in-nlp/ 17