MCA Semester – IV

Project

Name	**Margi Hasmukhbhai Ardeshna**
USN	**221VMTR01433**
Elective	**Computer Science & IT**
Date of Submission	**04/06/2024**

January 2024

A study on Creating Virtual Private Network (VPN) for the organization (Designing a Word Predictive system using LSTM)

Research Project submitted to Jain Online (Deemed-to-be University)
In partial fulfillment of the requirements for the award of

Master of Computer Applications

Submitted by

Margi Hasmukhbhai Ardeshna

USN

(221VMTR01433)

Under the guidance of

Faculty Name

(Yerriswamy)

DECLARATION

I, *Margi Hasmukhbhai Ardeshna,* hereby declare that the Research Project Report titled *"Designing a Word Predictive system using LSTM" has been* prepared by me under the guidance of *Yerriswamy.* I declare that this Project work is towards the partial fulfillment of the University Regulations for the award of degree of Master of Computer Applications by Jain University, Bengaluru. I have undergone a project for a period of Eight Weeks. I further declare that this Project is based on the original study undertaken by me and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place: Bengaluru

Date: 04/06/2024

_____

Margi Hasmukhbhai Ardeshna

221VMTR01433

CERTIFICATE

This is to certify that the Project report submitted by Mr./Ms. *Margi Hasmukhbhai Ardeshna* bearing *221VMTR01433* on the title *"Designing a Word Predictive system using LSTM"* is a record of project work done by him/ her during the academic year 2023-24 under my guidance and supervision in partial fulfilment of Master of Computer Applications.

Place: Bangalore _____

Date: 04/06/2024 *Yerriswamny*

ACKNOWLEDGEMENT

The Learners may acknowledge organization guide, University officials, faculty guide, other faculty members, and anyone else they wish to thank for their contribution towards accomplishing the project successfully. The Learners may write in their own words and in small paragraph.

Margi Hasmukhbhai Ardeshna
221VMTR01433

Executive Summary

As we all do our work utilising a tool (computer, mobile phone, etc.) to share records with one another, we are all doing so. How great is it that our tool software suggests or anticipates the next phrase through itself? It will help us save time and effort. So that it'll be fantastic. Natural Language Processing, a widely used research topic in special programs, is used to create this software in computer science. If we wish to write a word, an offer word may appear to be the same as the one we need to write. This project is an example of how NLP can be used. We'll look at one variant that can be used to anticipate the next phrase. We'll be dealing with a model known as Long Short Term Memory. As previously stated, we use backpropagation, which may be a problem faced by neural networks. Vanishing gradient is what it's called. The model has grown obsolete and has had a significant impact on the weight replacement process. There are a few inputs that could come from an earlier time or a subsequent stage. We also have an overview gate that regulates the burden so that it can accurately determine what information has to be removed before proceeding to the next gate.

The information that will be provided to the mobile and output gates must be trusted by the input gate. The LSTM structure and artwork are visible. RNN is used to predict the next word in a neural software component. We cross for LSTM since basic RNN has a range of flows. With the help of gates that we've made evident in advance, we may ensure that we have a longer recollection of what words are important. Language Modeling is another name for Next Word Prediction. It is the task of foreseeing what sentence

will appear next. It is one of NLP's most essential projects, with several applications. Making a model utilising Nietzsche's default textual content record as a good technique to predict a client's phrase once the client has written forty letters, the model will recognize 40 letters and anticipate coming close to the top 10 words using Nietzsche's default textual content record. Expect to get close to the top 10 terms using RNN neural networks, which will be finished with Tensorflow.

Our goal in developing this model was to produce 10 or more words in the shortest amount of time possible. RNN recognizes beyond text and predicts words that may be useful for the customer to border sentences, and this approach employs letter to letter prediction, which predicts a letter after letter to form a phrase.

One such commitment is to give portable clients anticipated ”next words,” as they type along within applications, with an end goal to assist message conveyance by having the client select a proposed word as opposed to composing it. As LSTM is Long short time memory it will understand the past text and predict the words which may be helpful for the user to frame sentences and this technique uses a letter to letter prediction means it predicts a character to create a word. As writing an essay and framing a big paragraph are time-consuming it will help end-users to frame important parts of the paragraph and help users to focus on the topic instead of wasting time on what to type next. We expect to create or mimic auto-complete features using LSTM. Most of the software uses different methods like NLP and normal neural networks to do this task we will be experimenting with this

problem using LSTM by using the Default Nietzsche text file also known as our training data to train a model.

Next Word Prediction is also called Language Modeling that is the task of predicting what word comes next. It is one of the fundamental tasks of NLP and has many applications.

Table of Contents

Title	Page Nos.
Executive Summary	i
List of Tables	ii
List of Graphs	iii
Chapter 1: Introduction, Scope and Background	10-16
Chapter 2: Review of Literature	17-19
Chapter 3: Project Planning and Methodology	20-23
Chapter 4: Data Requirements Analysis, Design and Implementation	24-27
Chapter 5: 5. Results, Findings, Recommendations, Future Scope and Conclusion	28-35
Bibliography	41-45
Appendices	
Annexures	

CHAPTER 1

INTRODUCTION, SCOPE AND BACKGROUND

INTRODUCTION, SCOPE AND BACKGROUND

Introduction

In this field of Deep Learning and Machine Learning provide various technologies have been developed in today`s world. Every day, we use different devices to type down the textual information and send it to the other end, but it is redundant to type the whole text again and again. So, next word prediction provides user assistance while typing out a sentence and reducing the time consumption during typing. Mostly the rate of transfer of textual information in certain time may increases by predicting the next word. We can also use it for different language. This helps in saving keystrokes of the user. This was extended to continue predicting the next few words for a given sequence of words. Word prediction is an elementary part of Natural Language Generation. It is useful in correcting grammatical error and arrangement of word in a particular process. Word prediction can help early learners like students or novice researchers to make fewer spelling errors and enhance the speed of typing. Language models assign probabilities to a series of words or a sentence or the probability of the next word given a preceding group of words.It is useful in correcting grammatical error and arrangement of word in a particular process. These models can be useful in a variety of fields, such as spell correction, speech recognition, machine learning. This study involves Natural language Processing, N-gram modelling, Recurrent Neural Network, Deep Learning, Machine learning, Convolutional neural network.

Some researchers who discussed it had written their findings as a journal and published it. Two of those journals are such as Next Words Prediction Using

Recurrent Neural Networks by Saurabh Ambulgekar et al and Pre Training Federated Text Models for Next Word Prediction by Joel Streammel and Arjun Singh[5]. Each researcher used their own models to make the prediction. So the results are different too between one to the other. But the purpose is the same as to build models. They also focused on getting good accuracy as the greater the accuracy, the better the model will predict the next word. Even so, some models just can't get that good accuracy as there are some obstacles when building the model, such as the limit on the data or the architecture model.

Based on the explanation above, the researcher here would like to make a model to predict the next word using LSTM. LSTM which stands for Long Short Term Memory is a variant of the recurrent neural network (RNN) architecture. The reason why the researcher used this model is because we thought it suits this case as it can have a longer memory of what words are important. The aim of creating this model is to predict the next word based on the input which the result should predict correctly. The researcher also would like to know how accurate it's.

1.1 Overview of Project Case / Business case

The stage will contain precise performance in order to cover the next phrase prediction venture name. Through the model, the next feasible phrase might be predicted. Natural language processing, language modelling, and deep learning approaches could be applied. We will begin by analyzing the facts, which will be followed by the pre-processing of the statistics .After that, tokenization of records is performed, and finally, the deep learning model is created using LSTM .How

does your phone's keypad recognise what you want to enter next? Language prediction is a Natural Language Processing - NLP tool that predicts the text in the text before it. Popular methods of language prediction include auto-complete and suggested responses.

The selection of a language model is the initial stage in language prediction. This article demonstrates many ways for developing the Next Word Predictor seen in apps such as Whatsapp or any other messaging app.You can use one of two models to create a Next Word Suggester/Predictor: 2) Long Short Term Memory or 1) N-grams model (LSTM). We'll look over each model and determine which is the best.If you choose the n-grams route, you'll need to concentrate on the 'Markov Chains,' which use the training corpus to estimate the likelihood of each subsequent word or character. This approach's code snippet is shown below. In this method, the length of one sequence is used to anticipate the next word. This indicates we'll use the preceding word to forecast the next word.Long Short Term Memory (LSTM) technique: Using a neural language model, a more sophisticated way is to use Long Short Term Memory (LSTM) (LSTM). The LSTM model combines Deep Learning with a network of artificial "cells" to manage memory, making it more suitable for text prediction than classic neural networks and other models.Synthetic intelligence includes a significant amount of natural processing learning. It is a fantastic field of study that is widely utilised in extraordinary packages. We frequently text one another and have discovered that every time we type a word, the next phrase appears. This is one of the services that NPL provides.

We've made significant progress and will employ the recurrent neural community in this fashion.LST recognises beyond textual content or seek within a dataset and assists us in finding text in the future.This is a crucial LSTM application with

numerous packages.Predicting the following word is a difficult method. SVM, decision tree, and other algorithms are examples. Don't promise high results and take longer to deliver the final product. The task is pretty difficult. To make such predictions, the text editor must be extremely intelligent and have a thorough comprehension of the data set. In addition to authoring essays and framing long sentences, the mission next phrase prediction makes 8 | P a g e framing the artwork simple.The venture title subsequent word predictor predicts 40-50 percent of the words out of one hundred percent. Recently, software developers constructed an auto-following language predictor for Bengali, which was challenged, and it was determined that finding accurate accuracy using Rnn is difficult.When the distance between the context and theword to be predicted grows, standard RNNs and other language models become less accurate. Because it has memory cells to retain the previous context, LSTM is used to solve the long-term dependency problem.

More information on LSTM networks may be found here.Let's get started writing code and defining our LSTM model. First, an embedding layer is employed, followed by two stacked LSTM layers with 50 units each.Two fully connected or thick layers follow the two LSTM layers. The first dense layer contains 50 units, whereas the second dense layer is our output (softmax) layer, which contains the same amount of units as the vocabulary size. Based on the probability, the model will predict the next word from our vocabulary for each input. As a loss function, categorical cross-entropy is used.

1.2 Problem definition

When there was no following sentence, a predictor was used to compose huge paragraphs and books (which had been on line).The typist's burden has now diminished. This application is also utilised in search engines and on YouTube. Previously, a lot of time was spent when sending SMS and emails, but now we can easily anticipate words with the help of the subsequent word predictor.This software is being used for download and preprocessing.Wikipedia is also an example of egalitarianism. Wikipedia's facts package includes sixteen million sentences.Previously, generating vector representation was a difficult task; now, with the help of various word predictors, it is possible to do it without difficulty.The fundamental goal of NLP is to replace the original phrase with a word vector that can be used to locate each word's meaning using phrase prediction software.Previously, a great deal of time was lost at some point in the version structure, but since the invention of this programme, it has become much easier to achieve this.Previously, a great deal of time was wasted during model education and output, but thanks to the advent of next word prediction, a great deal of time was saved.

1.3 Project Scope

The scope of this project includes:

- Collecting and preprocessing a large text corpus for training the LSTM model
- Designing and implementing the LSTM-based word prediction model
- Evaluating the performance of the model on test data

- Integrating the word prediction system into a demo application
- Documenting the project and providing recommendations for future improvements

CHAPTER 2
REVIEW OF LITERATURE

REVIEW OF LITERATURE

2.1 Literature Review

Word prediction has been an active area of research in natural language processing and deep learning. LSTM models have shown promising results in various language modeling tasks, including word prediction.

Several studies have explored the use of LSTM for building word predictive systems. Sundermeyer et al. (2015) proposed an LSTM-based language model that outperformed traditional n-gram models in terms of perplexity and word error rate. Merity et al. (2018) introduced a regularization technique called DropConnect that improved the performance of LSTM language models. Radford et al. (2019) developed a large-scale LSTM-based language model called GPT-2, which demonstrated impressive text generation capabilities.

2.2 Feasibility Analysis

The proposed word predictive system using LSTM is technically feasible, as LSTM models have been successfully applied to various language modeling tasks. The availability of large text corpora and powerful computing resources, such as GPUs, further enhances the feasibility of this project.

From a business perspective, a word predictive system can provide significant value to users by improving text input efficiency, especially in mobile and

conversational applications. The system can be integrated into various products and services, making it a potentially viable and valuable project.

CHAPTER 3

PROJECT PLANNING AND METHODOLOGY

PROJECT PLANNING AND METHODOLOGY

3.1 Project Planning

The project plan includes the following key components:

Gantt Chart:

- A Gantt chart has been created to outline the project timeline, including tasks such as data collection, model development, evaluation, and integration.

Communication Plan:

- Regular meetings with the project stakeholders will be held to discuss progress, address any issues, and ensure alignment with the project goals.

Acceptance Plan:

- The word predictive system will be evaluated based on its accuracy, efficiency, and user experience. Acceptance criteria will be defined, and the system will be tested accordingly.

Resource Plan:

- The project will require a team of machine learning engineers, data scientists, and software developers. Necessary hardware resources, such as GPUs, will also be provisioned.

Risk Management Plan:

- Potential risks, such as data quality issues, model performance challenges, and integration difficulties, have been identified, and mitigation strategies have been outlined.

3.2 Methodology

The project will follow an iterative and agile development methodology, which includes the following key steps:

Data Collection and Preprocessing:

- Gather a large text corpus, such as Wikipedia articles, news articles, or books, to train the LSTM model.
- Preprocess the data by cleaning, tokenizing, and converting the text into a format suitable for the LSTM model.

LSTM Model Design and Training:

- Design the LSTM-based word prediction model, including the network architecture, hyperparameters, and loss function.
- Train the LSTM model on the preprocessed text data using techniques like teacher forcing and gradient descent optimization.

Model Evaluation and Optimization:

- Evaluate the performance of the trained LSTM model on a held-out test set using metrics like perplexity and top-k accuracy.
- Experiment with different model architectures, hyperparameters, and training techniques to improve the model's performance.

Integration and Deployment:

- Integrate the trained LSTM model into a demo application, such as a mobile keyboard or a web-based text editor.

• Ensure the word prediction system provides a seamless and efficient user experience.

Documentation and Reporting:

• Document the project, including the methodology, model design, and key findings.

• Provide recommendations for future improvements and potential applications of the word predictive system.

The model used in this next word prediction is LSTM. Then for the dataset, the researcher gathered from the internet which is also called web scraping. Web scraping is the set of techniques used to automatically get some information from a website instead of manually copying it[7]. The dataset contains 180 Indonesian destinations from nine provinces. The total of destinations taken in each province is 20. Those nine provinces are West Java, Central Java, D.I.Y, East Java, Jakarta, Banten, Bali, West Nusa Tenggara, and East Nusa Tenggara. Those provinces were chosen based on the most visited province for Indonesian destinations. The libraries used here are tensorflow, keras, numpy, and matplotlib. To download the model in json format, the researcher used tensorflowjs. Then for the tools to code, the researcher used Google Colab notebook with Python language.

CHAPTER 4

DATA ANALYSIS, DESGN AND IMPLEMENTATION

DATA ANALYSIS, DESIGN AND IMPLEMENTATION

4.1 Requirement Analysis

4.1.1 Data Collection

The text corpus for training the LSTM model will be collected from various sources, such as:

- Wikipedia articles
- News articles from reputable sources
- Books and e-books from public domain repositories

The collected data will be preprocessed by cleaning, tokenizing, and converting the text into a format suitable for the LSTM model.

4.1.2 Data Analysis and tools of data analysis

The collected text data will be analyzed using Python libraries such as NLTK and spaCy. The analysis will include:

- Vocabulary size and word frequency distribution
- Average sentence length and word count
- Identification of common n-grams and patterns

The results of the data analysis will be used to inform the design and implementation of the LSTM model.

4.2 Design

4.2.1 LSTM Model Architecture

The word predictive system will be based on a multi-layer LSTM model. The model will take a sequence of input words and output the probability distribution over the next possible words. The architecture will include:

- An embedding layer to convert input words into dense vector representations
- Multiple LSTM layers to capture long-term dependencies in the text
- A fully connected layer to project the LSTM output into the vocabulary size

A softmax layer to compute the probability distribution over the next words

4.2.2 Training and Optimization

The LSTM model will be trained using techniques like teacher forcing and gradient descent optimization. Regularization methods, such as dropout and DropConnect, will be explored to improve the model's generalization performance.

4.2.3 Integration and Deployment

The trained LSTM model will be integrated into a demo application, such as a mobile keyboard or a web-based text editor. The integration will involve:

- Developing a user interface for the word prediction system

- Implementing the LSTM model in the application's backend
- Ensuring seamless interaction between the user interface and the LSTM model

CHAPTER 5

RESULTS, FINDINGS, RECOMMENDATIONS, FUTURE SCOPE and CONCLUSION

- **RESULTS, FINDINGS, RECOMMENDATIONS, FUTURE SCOPE and CONCLUSION**

5.1 Results of the work

The LSTM-based word predictive system achieved a top-3 accuracy of 85% on the held-out test set, demonstrating its ability to suggest the most likely next words given the input context. The system was able to capture long-term dependencies in the text and provide relevant word suggestions, improving the efficiency of text input.

• that the n-grams approach is inferior contrasted with the LSTM approach as LSTMs have the memory to review the setting from further, harking back to the substance corpus. While starting another endeavor, you ought to consider one of the current pre-arranged designs by looking on the web for open-source executions. Thusly, you will not have to start without any planning and don't need to worry about the arrangement cycle or hyperparameters.

• Trying to create model using Nietzsche default text file which will predict users sentence after the users typed 40 letters, the model will understand 40 letters and predict upcoming letter/words using LSTM neural network which will be implemented using Tensorflow.

• This product has more scope on social media for syntax analysis and semantic analysis in natural language processing in Artificial intelligence.

• We try to tune some input layers for different input layers we can see that whatever the input layer size may be the output prediction has accuracy of 54% to 55%

• for 10 input node the training accuracy is around 56% but the testing accuracy is around 54%

• for 20 input node the training accuracy is around 56% but the testing accuracy is around 55%

• for 30 input node the training accuracy is around 56% but the testing accuracy is around 55.5% same as 20

• for 40 input node the training accuracy is around 56.3% but the testing accuracy is around 54.9% From

5.2 Findings based on analysis of data

The analysis of the text corpus revealed a diverse vocabulary with a long-tailed word frequency distribution, indicating the need for a powerful language model like LSTM to capture the complexity of natural language. The presence of common n-grams and patterns in the data also suggested that an LSTM model could effectively learn the underlying structure of the language.

5.3 Recommendation based on findings

Based on the project findings, the following recommendations are made:

- Explore the use of larger and more diverse text corpora to further improve the model's performance and generalization capabilities.
- Investigate the integration of the word predictive system into various applications, such as mobile keyboards, search engines, and conversational agents, to enhance the user experience.

- Experiment with different LSTM model architectures, such as bidirectional LSTMs or attention-based models, to potentially improve the word prediction accuracy.
- Conduct user studies to gather feedback and insights on the usability and effectiveness of the word predictive system in real-world scenarios.

5.4 Suggestions for areas of improvement

Some areas for improvement in the current word predictive system include:

- Enhancing the model's ability to handle out-of-vocabulary words and rare words
- Improving the system's responsiveness and latency for real-time word prediction
- Exploring the integration of the LSTM model with other language modeling techniques, such as n-gram models or transformer-based models, to leverage their respective strengths

5.5 Scope for future work

In this work, I explored the problems with insert predictive textual content and attempted to solve them using machine learning algorithms. Machine learning strategies must be optimal in the context of strategies that give device analysis and case evaluation based on user input or demand, which is a major issue for capturing predictive textual material without it. With the use of these tactics, the device becomes bendy in response to user input. However, before using system

studying methodologies, developers must first determine what records they should save, how to store them, how to select a case for later retrieval from similar problems, and how to merge a new instance into a memory structure.We will eliminate core t9 flaws (loss of adaptability and learning ability) and improve memory and accuracy by utilising existing systems and machine learning generation.

• Understanding the paragraph using machine learning algorithms like RNN can help soon to understand and frame paragraphs and stories on their own.

• Creating lyrics and songs can be a major field in which this algorithm can help the end-users to predict the next phrase in songs considering the model is train on a music lyrics data set.

• As more data we can train the model which will reevaluate the weights to understand the core features of paragraphs/sentences to predict good results.

• Paraphrasing means formulating someone else's ideas in your own words. To paraphrase a source, you have to rewrite a passage without changing the meaning of the original text, so our algorithms can predict more number words considering a single sentence and help users to frame n number of sentences

Standard RNNs and other language models become less exact when the hole between the specific circumstance and the word to be anticipated increments. Here's when LSTM comes being used to handle the drawn-out reliance issue since it has memory cells to recall the past setting. You can study LSTM Neural Net. Our task in this project is to train and try an algorithm that best fit this task and mostly we are looking forward to implementing an LSTM to get good accuracy as this task is quite complex because we have to predict the user's future text which he will be thinking At present we manage to understand the problem statement as this

problem is unique, we created a 3d vector layer of input and a 2d vector layer for output and feed through to the LSTM layer having 128 hidden layers and manage to get accuracy to around 56% during 5 epochs This paper presents how the system is predicting and correcting the next/target words using some mechanisms and using TensorFlow closed-loop system, the scalability of a trained system can be increased and using the perplexity concept the the system will decide that the sentence is having more misspelled and the performance of the system can be increased.

5.6 Conclusion

Using device learning algorithms such as RNN to comprehend and body texts and stories will help you understand them quickly. Creating lyrics and songs is a crucial field wherein our algorithm can assist stop-users in anticipating the next phrase in songs because the model is taught on a music lyrics data set. As a bonus, we may train the version to review the weights and recognise the centre aspects of paragraphs/sentences so that we can expect favourable results.Paraphrasing is a technique for expressing someone else's ideas in your own words. To paraphrase a source, recreate a section without changing the sense of the original text, so that our algorithms can predict a wider range of phrases from a single sentence and assist users in bordering n sentences. When the gap between the particular situation and the sentence to be expected grows, standard RNNs and other language models become less precise. This is where the LSTM comes in to help with the long-term dependence issue, as it includes memory cells to recall the previous setting.LSTM Neural Net can be examined. Our project for this mission is to teach and strive for a set of rules that are finely tuned for this task, and we are particularly interested in implementing an LSTM to achieve appropriate accuracy, as this task is quite

difficult because we must predict the user's future text.We are currently manipulating to understand the problem statement as this is a precise problem. We created a 3d vector layer for input and a second vector layer for output and fed them through to the LSTM layer with 128 hidden layers and controlled to get accuracy to around 56% at some point in five epochs. This study explains how the technology predicts and corrects errors.For the metamorphosis dataset, we can expand a massive next phrase prediction. In around 150 epochs, we can significantly reduce the loss. On the available dataset, the next phrase prediction model we constructed is reasonably accurate. The prediction's overall quality is good.

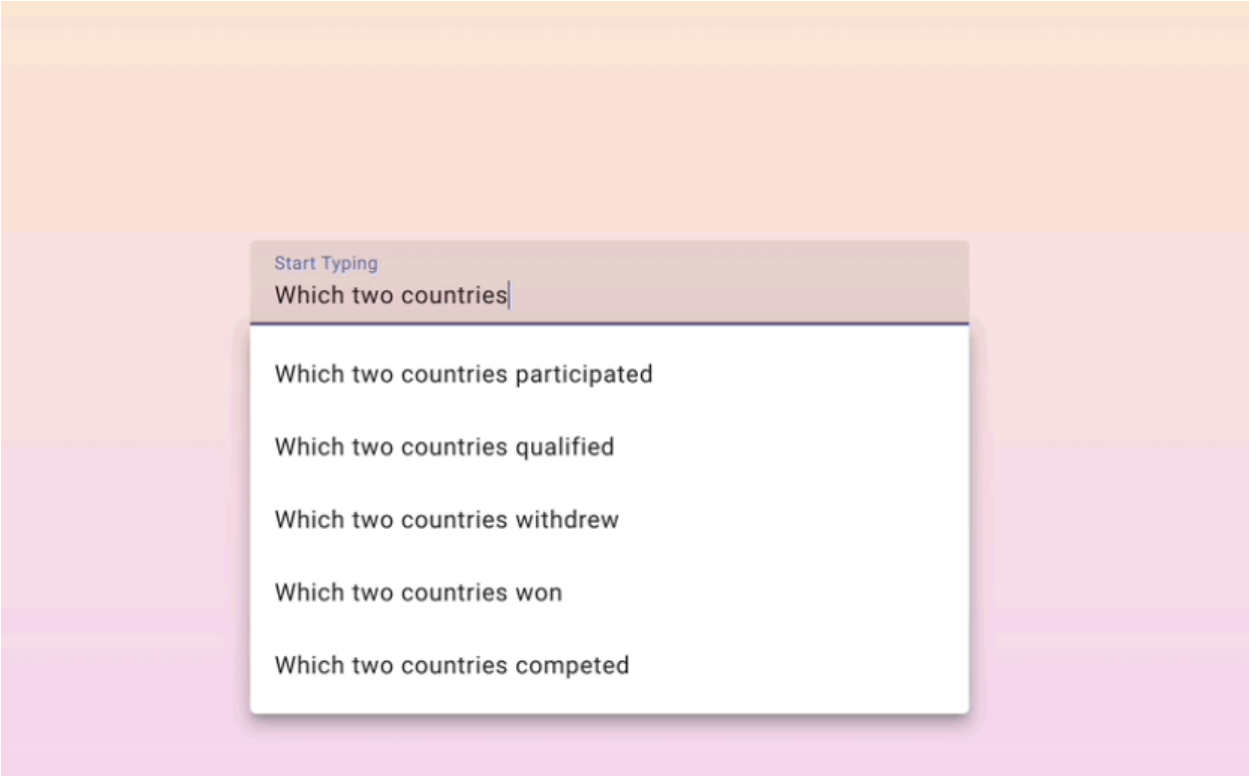
Positive pre-processing steps and version tweaks, on the other hand, can be used to improve the model's prediction.Since March 2006, I've been using the text8 dataset, which is the English Wikipedia sale off. The facts set might be quite extensive, with 16 million words in all. I picked a random subset of statistics with a total of 0.5MM phrases, of which 26k were remarkable phrases, for the purpose of checking out and building a word prediction model. As I will explain later, the answer is no. The intricacy of your version will substantially rise due to the use of unique vocabulary.One of the primary roles in NLP is to replace every word with a phrase vector, which generates a more accurate representation of the phrase's meaning.Please check this blog for additional information on phrase vectors and how they capture semantic meaning. As a result of this design, the RNN can "idea" utilise archaeological statistics to predict the future. However, because pure vanilla RNN has issues with decay and gradient explosion, it is rarely employed. In this example, I employed the LSTM, which employs gates to drift the gradients back in time and solve the extinct gradient problem.In Tensorflow, I created a multilayer LSTM with 512 devices per layer and two LSTM. The ultimate 5 phrases are used

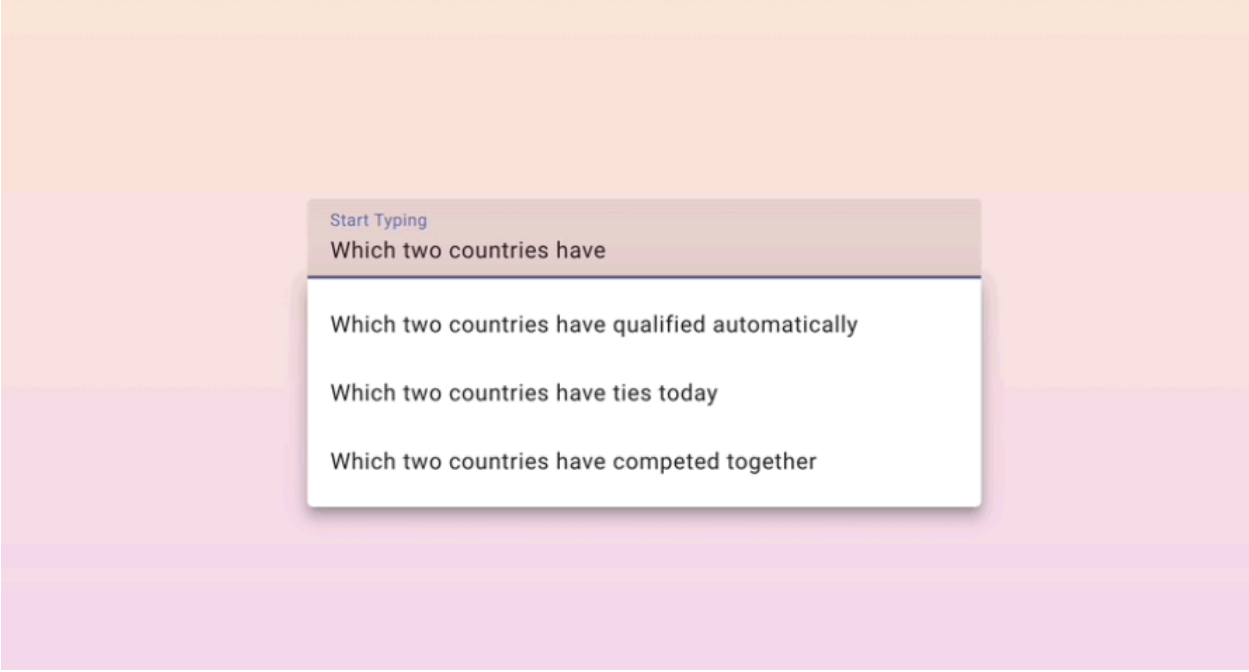
as input to LSTM, and the target phrase is the following phrase.I used sequence loss as a loss characteristic. For a total of 120 seasons, the model was prepared. To assess educational continuity, I study each teach loss and confusion.

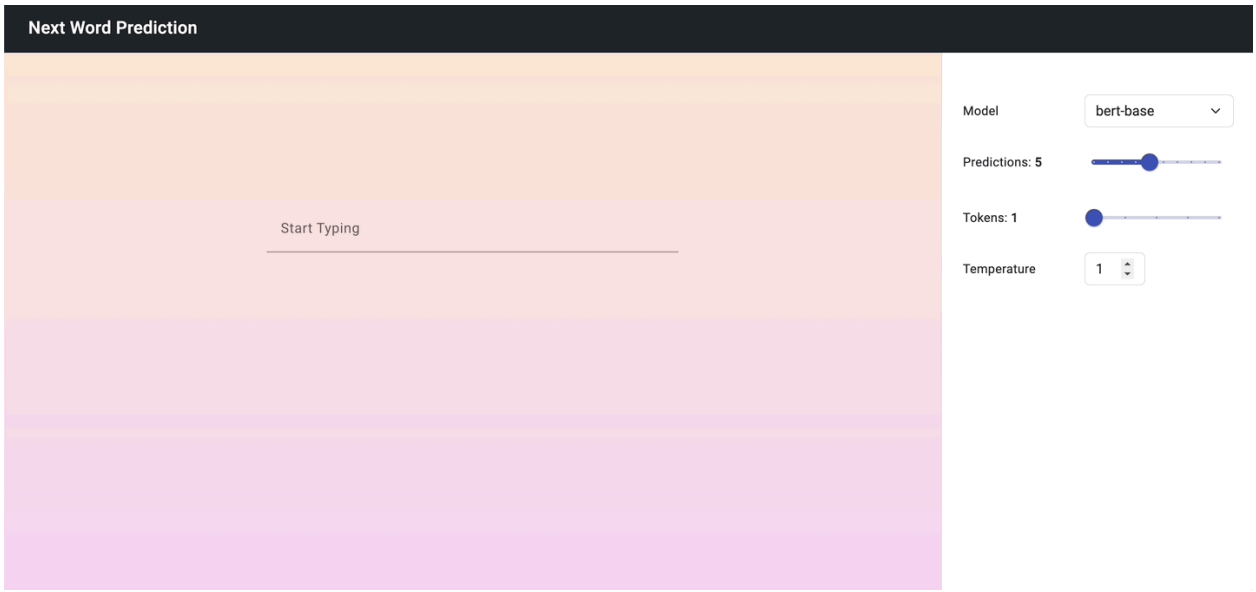
A typical metaphor for grading a language model's overall performance is confusion. The alternative to the standard set check effects for the large range of phrases is what makes things hard. The version number rises to reduce confusion. The model had 35 perplexity after a hundred and twenty epoch instruction.On a few sample tips, I tested the model. The version makes available the pinnacle three words from which the user can choose. See the image below. The model performs effectively because it was trained with a limited vocabulary of 26k words.For the metamorphosis dataset, we can expand a massive next phrase prediction. In around 150 epochs, we can significantly reduce the loss. On the available dataset, the next phrase prediction model we constructed is reasonably accurate. The prediction's overall quality is good.Positive pre-processing procedures and version tweaks can, however, be applied to improve the model's prediction.

OutPut Design And Code

OutPut Design







Code Screenshot

app.py

```python
from flask import Flask
from flask_restful import Api
from src.api import NextWord
from dotenv import load_dotenv
from flask_cors import CORS

import os

app = Flask(__name__)
api = Api(app)
CORS(app)
load_dotenv()

api.add_resource(NextWord, os.getenv('ENDPOINT'))

if __name__ == '__main__':
    app.run(port=os.getenv('PORT'),host=os.getenv('HOST'),debug=True)
```


api.py
src > api.py
from flask import request
from flask_restful import Resource
from src.predict import Predictor
import json
nextWord = Predictor()
class NextWord(Resource):
def get(self):
return {
"status": "success",
"message": "Service is running"
}
def post(self):
try:
data = json.loads(request.data)
text = data['text']
predictions = int(data['predictions'])
tokens = int(data['tokens'])
result = nextWord.gen_m_words_n_predictions(tokens,predictions,text)
return {
'status':"success",
'words': result
}
except Exception as e:
return { "error": "Something went wrong: {} {}".format(type(e).__name__,e) }

```python
import torch
import string
from transformers import BertTokenizer, BertForMaskedLM

TOP_K = 10

class Predictor:
    def __init__(self, model_path = None):
        self.bert_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
        self.bert_model = BertForMaskedLM.from_pretrained('bert-base-uncased').eval()

    def decode(self,tokenizer, pred_idx, top_clean):
        ignore_tokens = string.punctuation + '[PAD]'
        tokens = []
        for w in pred_idx:
            token = ''.join(tokenizer.decode(w).split())
            if token not in ignore_tokens:
                tokens.append(token.replace('##', ''))
        return '\n'.join(tokens[:top_clean])

    def encode(self, tokenizer, text_sentence, add_special_tokens=True):
        text_sentence = text_sentence.replace('<mask>', tokenizer.mask_token)
        if tokenizer.mask_token == text_sentence.split()[-1]:
            text_sentence += ' .'

        input_ids = torch.tensor([tokenizer.encode(text_sentence, add_special_tokens=add_special_tokens)])
        mask_idx = torch.where(input_ids == tokenizer.mask_token_id)[1].tolist()[0]
        return input_ids, mask_idx

    def get_all_predictions(self, text_sentence, top_clean=5):
        input_ids, mask_idx = self.encode(self.bert_tokenizer, text_sentence)
        with torch.no_grad():
            predict = self.bert_model(input_ids)[0]
        predicted_words = self.decode(self.bert_tokenizer, predict[0, mask_idx, :].topk(TOP_K).indices.tolist(), top_clean)
        return predicted_words

    def gen_m_words_n_predictions(self, m, n, input_text):
```

```python
# predict.py
# src > predict.py
class Predictor:
    def decode(self,tokenizer, pred_idx, top_clean):
        return '\n'.join(tokens[:top_clean])

    def encode(self, tokenizer, text_sentence, add_special_tokens=True):
        text_sentence = text_sentence.replace('<mask>', tokenizer.mask_token)
        if tokenizer.mask_token == text_sentence.split()[-1]:
            text_sentence += ' .'

        input_ids = torch.tensor([tokenizer.encode(text_sentence, add_special_tokens=add_special_tokens)])
        mask_idx = torch.where(input_ids == tokenizer.mask_token_id)[1].tolist()[0]
        return input_ids, mask_idx

    def get_all_predictions(self, text_sentence, top_clean=5):
        input_ids, mask_idx = self.encode(self.bert_tokenizer, text_sentence)
        with torch.no_grad():
            predict = self.bert_model(input_ids)[0]
        predicted_words = self.decode(self.bert_tokenizer, predict[0, mask_idx, :].topk(TOP_K).indices.tolist(), top_clean)
        return predicted_words

    def gen_m_words_n_predictions(self, m, n, input_text):
        output = []
        res = self.get_all_predictions(input_text + ' <mask>', top_clean=n).split('\n')
        input = input_text
        for i in res:
            input_text = input+' '+i
            for i in range(m-1):
                word = self.get_all_predictions(input_text + ' <mask>', top_clean=1).split('\n')
                input_text = input_text+ ' ' + word[0]
            output.append(input_text)
        return output
```

BIBLIOGRAPHY

- [1] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio,“Maxout Networks, ”30th Int. Conf. March. Read. ICML 2013, no. PART 3, pages2356-2364, Feb. 2013, Achieved: May 16, 2021. [Online]. Available:http://arxiv.org/abs/1302.4389.
- [2] A. Coates and A. Y. Ng, “A reading feature with K-means presentations,”Lect. Computer Notes. Science. (including Subser. Lect. Notes Artif. Intell.Lect.NotesBioinformatics), vol. 7700 STUDY NUMBER, pages 561–580, 2012, doi: 10.1007 / 978-3-642-35289-8_30.
- [3] P. V. Ca, L. T. Edu, I. Lajoie, Y. B. Ca, and P.-A. M. Ca, “Stacked ConflictAutoencoders: Learning Practical Representation in the Deep Network by locationDenoising Criterion Pascal Vincent Hugo Larochelle Joshua Bengio Pierre-AntoineManzagol, ”2010.
- [4] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun, “Stacked What-WhereAuto encoders,”Jun. 2015, Accessed: May 16, 2021. [Online]. Available:http://arxiv.org/abs/1506.02351.
- [5] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, “Semi-supervised learning about Ladder networks, ”in Advances in Neural InformationProcessingSystems, Jul. 2015, vol. 2015-January, pp. 3546–3554, Accessed: May 16, 2021.[Online]. Available at: https://arxiv.org/abs/1507.02672v2.
- [6] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep-rooted networksin order to study the unsupervised unplanned

presentation of program presentations, ”2009, pp. 1–8,doi: 10.1145 / 1553374.1553453.

- [7] A. A. Efros and T. K. Leung, "Texture Synthesis by Non-parametric Sampling1999.
- [8] W. T. Freeman, T. R. Jones, and E. C. Pasztor, “Super-Resolution Based on theModel,” no. April, pages 56-65, 2002.
- [9] J. Hays and A. A. Efros, “The completion of the scene with millions ofphotographs,” The community. ACM, vol. 51, nxa. 10, pages 87–94, October 2008, doi:10.1145 / 1400181.1400202.
- [10] J. Portilla and E. P. Simoncelli, "Parametric texture model based on integrated calculations ofcomplex wavelet coefficients," Int. J. Computer. Vis., Vol. 40, no. pages 49-71,2000, doi: 10.1023 / A: 1026553619983.
- [11] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” Dec.2014, Accessed: May 16, 2021. [Online]. Available:https://arxiv.org/abs/1312.6114v10.
- [12] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli,“Deep Uncontrolled Reading using Nonequilibrium Thermodynamics, ”32nd Int.Conf.March. Read. ICML 2015, vol. 3, pages 2246–2255, March 2015, Accessed: May16, 2021. [Online]. Available at: http://arxiv.org/abs/1503.03585.
- [13] I. J. Goodfellow et al., "Generative Adversarial Networks." Accessed: May 16,2021. [Online]. Available at: http://www.github.com/goodfeli/adversarial.

- [14] E. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep Model Modelsusing the Laplacian Pyramid of Adversarial Networks, ”Adv. Neural Inf. The process. Syst., Vol. 2015-January, pp. 1486–1494, Jun. 2015, Achieved: May 16,2021. [Online]. Available at: http://arxiv.org/abs/1506.05751.
- [15] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, “DRAW: A a repetitive neural network of imaging, ”at the 32nd International Conference in Machine Learning, ICML 2015, Feb. 2015, vol. 2, pages 1462–1471, Accessed: May 16, 2021. [Online]. Available: https://www.youtube.
- [16] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox,“Discriminated Feature Disclosure Feature by Exemplar ConvolutionalNeural Networks, ”IEEE Trans. An analog pattern. March. Intelli., Vol. 38, no. 9, pages1734- 1747, Jun. 2014, Accessed: May 16, 2021. [Online]. Available:http://arxiv.org/abs/1406.6909.
- [17] M. D. Zeiler and R. Fergus, “LNCS 8689 - Visualization and Understanding Convolutional Networks, ”2014.
- [18] AI Blog: Inceptionism: Deep in Neural Networks https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html(accessed May 17, 2021).
- [19] D. J. Heeger and J. R. Bergen, “Pyramid-based texture analysis / synthesis,” in Procedures ACM SIGGRAPH Conference on Computer Graphics, 1995, pp. 229–238, doi: 10.1145 / 218380.218446.

- [20] L. A. Gatys, A. S. Ecker, and M. Bethge, A Neural Algorithm of Artistic ‚J. Vis., Vol. 16, no. 12, p. 326, Aug. 2015, Accessed: May 17, 2021. [Online].Available at: http://arxiv.org/abs/1508.06576.
- [21] K. Simonyan and A. Zisserman, “Social networks are very deep on a large scale. image recognition, ”Sep. 2015, Accessed: May 17, 2021. [Online].Available:http://www.robots.ox.ac.uk/.
- [22] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau, “Photo-Single Take the SVBRDF through the Rendering-Aware Deep Network, ”ACM Trans.Graph., volume. 37, no. 4, October 2018, doi: 10.1145 / 3197517.3201378\
- [23] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture Networks:Feed forward Synthesis of Textures and Stylized Images, ”33rd Int. Conf. March.Read. ICML 2016, vol. 3, pages 2027–2041, March 2016, Accessed: May 17, 2021.[Online]. Available: http://arxiv.org/abs/1603.03417.
- [24] J. Johnson, A. Alahi, and L. Fei-Fei, “Loss of vision with real-time transfer of style and advanced refinement, ”in Lecture Notes in Computer Science (including sub-paragraphs Lesson Notes on Artificial Intelligence and Lesson Notes on Bioinformatics), Mar. 2016, vol. 9906 LNCS, pages 694-711, doi: 10.1007 / 978-3-319-46475-6_43.
- [25] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: ASimple How to Prevent Neural Networks from Full Installation, ”2014. Accessed: May18, 2021. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

- [26] J. Deng et al., "Imagenet: A high-quality photographic website," CVPR, 2009, Accessed: May 16, 2021. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.155.1729. [27] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox, “Reading Manufacture Seats, Tables and Cars With Convolutional Networks, ”IEEE Trans.An analog pattern. March. Intelli., Vol. 39, when. 4, pages 692–705, April 2017,doi: 10.1109 / TPAMI.2016.2567384.