ABSTRACT

Natural Language Processing (NLP) is a subfield of Artificial Intelligence and is used in various applications like speech recognition, spam mail classification, text analytics and so on. One such application is smartly composing emails, which works based on the concept of next word prediction. Next Word Prediction, also called Language Modelling is used for predicting what word comes next. Various machine learning and deep learning algorithms are available for next-word prediction.

Most of the projects use algorithms and methods like Recurrent Neural Networks (RNN), Attention models, Long Short Time Memory (LSTM) and Encoder-Decoder. This project uses Bidirectional Encoder Representations from Transformers (BERT) which is a new, pre-trained, transformer-based deep learning technique for Natural language processing developed by Google to predict the next word.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYMS	ABBREVIATIONS
BAS	BERT Answer Selection system
BERT	Bidirectional Encoder Representations form Transformers
BLEU	Bilingual Evaluation Understudy
BSD	Berkeley Software Distribution
CSS	Cascade Style Sheets
CWI	Centrum Wiskunde & Informatica
GLUE	General Language Understanding Evaluation
GPT-2	Generative Pre-trained Transformer 2
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
LSTM	Long-Short Term Memory
METEOR	Metric for Evaluation of Translation with Explicit Ordering
MLM	Masked Language Modelling
Multi-NLI	Multi-Genre Natural Language Inference
NLP	Natural Language Processing
NP	Noun Phrase
NWP	Next Word Prediction
PP	Prepositional Phrase
QA	Question Answering

RNN	Recurrent Neural Network
SPICE	Semantic Propositional Image Caption Evaluation
SQuAD	Stanford Question Answering Dataset
T9	Text on nine keys
ULMFiT	Universal Language Model Fine-Tuning
VP	Verb Phrase

CHAPTER 1

INTRODUCTION

Natural Language Processing (NLP) is a subfield of Linguistics, Computer Science and Artificial Intelligence. It uses machine learning and deep learning algorithms to analyse, understand human language and speech to perform certain actions. It is used in various applications and one such application is smart composing of emails which uses the concept next word prediction.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. Bidirectional training allows the model to learn the context of a word based on all of its surroundings (left and right of the word). This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. BERT uses Masked LM (MLM), which allows this bidirectional training in models.

1.1 PROBLEM STATEMENT

Employees and business people type lots of emails and documents in a day but their time is wasted searching for appropriate words, and typing words in documents. Next Word Prediction automates this work by suggesting the best suitable words to the user based on previous text.

1.2 OBJECTIVES

- To get input text and number of words to be predicted from user.
- To predict list of words using bert
- To create a webpage in streamlit
- To deploy next word prediction using streamlit

CHAPTER 2

LITERATURE SURVEY

2.1 BERT: PRE-TRAINING OF DEEP BIDIRECTIONAL TRANSFORMERS FOR LANGUAGE UNDERSTANDING

A new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers was introduced. BERT was designed to pre-train deep, bi-directional representations form unlabelled text by jointly conditioning on both left and right context in all layers. As a result, BERT model can be fine tuned with just one additional output layer to create state-of-the-art models for wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications. Thus, BERT obtained new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5% point absolute improvement) and SQuAD v 2.0 Test F1 to 83.1 (5.1% point absolute improvement).[1]

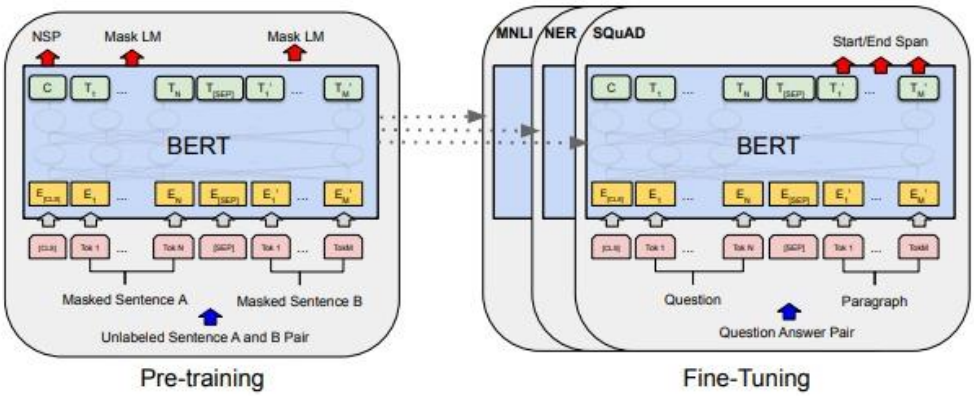


Fig 2.1

2.2 A COMPARATIVE STUDY OF TEXT CLASSIFICATION AND MISSING WORD PREDICTION USING BERT AND ULMFiT

For predicting the melanoma type skin cancer, first pre-processing of input skin image is made. A comparative study was performed on the two types of emerging NLP models, ULMFiT and BERT by using them in Text classification and Missing word prediction and emphasize how these two tasks can cover most of the prime industry use cases. Performance of above two models is systematically framed by using selective metrics and train them with various configurations and inputs. This paper was intended to assist the industry researchers on the pros and cons of fine-tuning the industry data with these two pre-trained language models for obtaining the best possible state-of-the-art results.[2]

2.3 NEXT WORD PREDICTION USING RECURRENT NEURAL NETWORKS

Next Word Prediction was implemented by making the model utilize a default text record which foresees clients' sentences after the client composed forty letters. Then the model comprehends forty letters and anticipates impending top 10 words using RNN neural organization and TensorFlow. As RNN 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. This technique uses letter-to-letter prediction means it predicts letter after letter to create a word. [3]

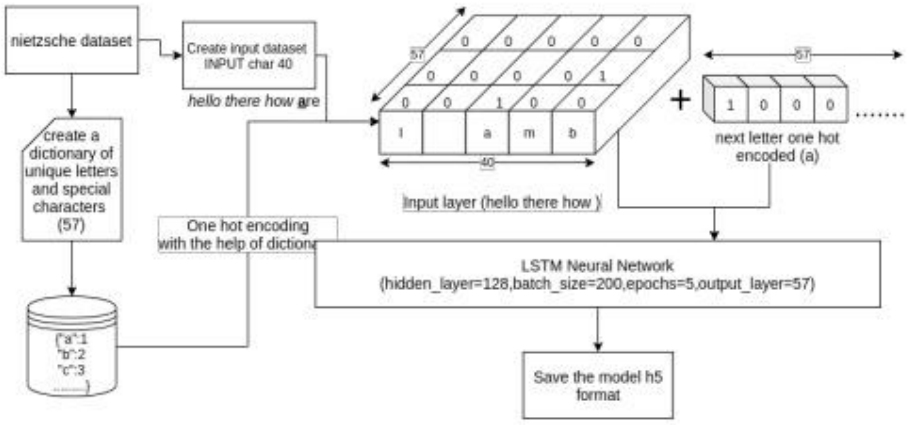


Fig 2.2

2.4 PREDICTING NEXT WORD USING RNN AND LSTM CELLS: STATISTICAL LANGUAGE MODELLING

The assorted potentialities for the efficient utilization of language models in structured document retrieval were mentioned. A tree-based generative language model for ranking documents and parts had been used. Nodes within the tree correspond to different document parts like titles, paragraphs and sections. At every node within the document tree, there's a well-defined language model. The language model for a leaf node was predictable directly from the text within the document part related to the node. Inner nodes within the tree were predictably employing a linear interpolation among the various youngster nodes. The paper additionally described how some common structural queries would be satisfactorily described inside this model. [4]

2.5 PRETRAINING FEDERATED TEXT MODELS FOR NEXT WORD PREDICTION

Federated learning is a decentralized approach for training models on distributed devices, by summarizing local changes and sending aggregate parameters from local models to the cloud rather than the data itself. The idea

of transfer of learning was employed in federated training for Next Word Prediction (NWP) and conducted a number of experiments demonstrating enhancements to current baselines for which federated NWP models have been successful. Specifically, federated training baselines from randomly initialized models were compared with various combinations of pretraining approaches including pre-trained word embeddings and whole model pretraining followed by federated fine-tuning for NWP on a dataset of Stack Overflow posts. A lift in performance was seen using pre-trained embeddings without exacerbating the number of required training rounds or memory footprint.

The notable differences were observed using centrally pre-trained networks, especially depending on the datasets used. This research offered effective, yet inexpensive, improvements to federated NWP and paved the way for more rigorous experimentation of transfer learning techniques for federated learning. [5]

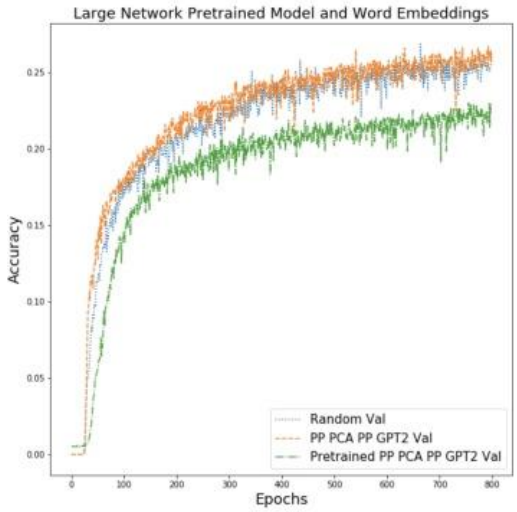


Fig 2.3

Model	Accuracy	Accuracy No OOV No EOS	Parameters	Weights(MB)
Small Random*	**0.2246**	**0.1821**	2.4M	9.6
Small GloVe	0.2269	0.1838	2.4M	9.6
Small PCA FastText	0.2250	0.1823	2.4M	9.6
Small PP_PCA_PP FastText	0.2285	0.1852	2.4M	9.6
Small PCA GPT2	0.2293	0.1859	2.4M	9.6
Small PP_PCA_PP GPT2	0.2262	0.1834	2.4M	9.6
Large Random*	**0.2485**	**0.2086**	7.8M	31.3
Large GloVe	0.2557	0.2162	7.8M	31.3
Large FastText	0.2548	0.2137	7.8M	31.3
Large PCA GPT2	0.2522	0.2118	7.8M	31.3
Large PP_PCA_PP GPT2**	**0.2569**	**0.2169**	7.8M	31.3

Fig: 2.4

2.6 TOKENIZATION AS THE INITIAL PHASE IN NLP

The significance and complexity of tokenization, the beginning step of NLP were addressed. Notions of word and token are discussed and defined from the viewpoints of lexicography and pragmatic implementation, respectively. Automatic segmentation of Chinese words is presented as an illustration of tokenization. Practical approaches to the identification of compound tokens in English, such as idioms, phrasal verbs and fixed expressions, were developed.[6]

2.7 AN APPROACH FOR A NEXT-WORD PREDICTION FOR THE UKRAINIAN LANGUAGE

LSTM and Markov chains and their hybrid were chosen for next-word prediction in the Ukrainian language. The sequential nature of these models helps to successfully cope with the next-word prediction task. The Markov chains presented the fastest and most adequate results. The hybrid model presents adequate results but it works slowly. Using the model, the user can generate not only one word but also a few or a sentence or several sentences, unlike T9.[7]

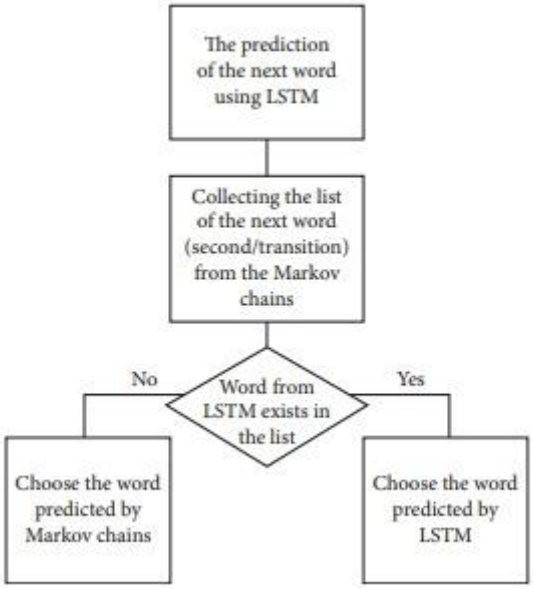


Fig 2.5

2.8 NEXT WORD PREDICTION

The long Short Time Memory formula was used to perceive past text and predict the words. The accuracy is enhanced by pre-processing the data set, finding term frequencies using the term-document matrix and performing tokenization and padding sequences [8]

Fig 2.6

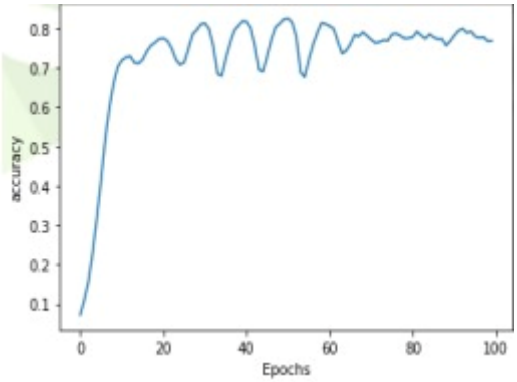


Fig 2.7

2.9 BAS: AN ANSWER SELECTION METHOD USING BERT LANGUAGE MODEL

Question Answering system consisting of several parts were made in which one of the main components was the Answer Selection component. This component detected the most relevant answer from a list of candidate answers. Instead of applying the language model as an independent model, it was implemented in the answer selection part. This action enabled the model to have a better understanding of the language in order to understand questions and answers better than previous works. This system used the BERT language model to comprehend the language. Thus, this system was named as BERT Answer Selection (BAS) system. The empirical results of applying the model on the TrecQA Raw, TrecQA Clean, and WikiQA datasets demonstrated that using a robust language model such as BERT can enhance the performance. Using a more robust classifier can also enhance the effect of the language model on the answer selection component. [9]

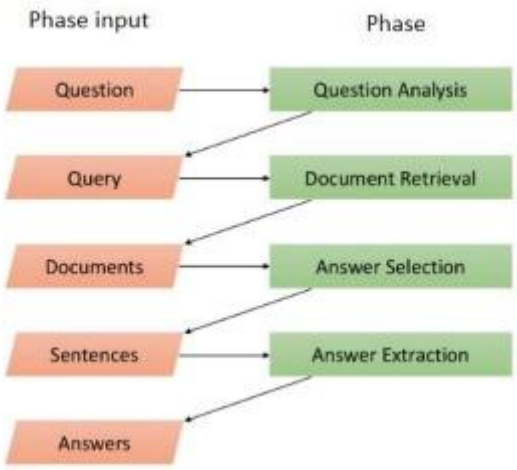


Fig:2.8

2.10 IMPROVING TEXT AUTO-COMPLETION WITH NEXT PHRASE PREDICTION

First, phrases are extracted with constituency parsing using AllenNLP to retrieve qualitative phrases and were categorized as Noun Phrase(NP), Verb Phrase(VP) and Prepositional Phrase(PP). Then, the pre-trained GPT-2 model and T5 model were used for phrase prediction and these models were fine-tuned based on some conditions. This technique outperformed Next Sentence Prediction with a margin from 0.2 to 0.3 BLEU/METEOR/SPICE score. [10]

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Existing systems to predict the next word, either use Recurrent Neural Networks (RNN) or Long Short Time Memory (LSTM). Recurrent Neural Networks have more flaws than Long Short Time Memory. LSTM involves eleven phases: (i) The required libraries are imported (ii) Dataset is read (iii) Tokenizers are used (iv) Unique words are obtained using a dictionary (v)Feature Engineering also known as Word Embeddings are performed to convert words into a numerical representation (vi) variables X and Y are used to store features and labels (vii) LSTM model is created (viii) Model is being trained and saved (ix) The saved model is evaluated (x) The model is tested by giving an input text (xi) The next word is predicted based on the input. This gives possible next words based on the input text.

3.1.1 Drawbacks of Existing System

1) RNNs and LSTMs are directional

2) RNNs and LSTMs fail in case of broader inputs as proximity increases, the importance of words decreases

3) LSTMs require more memory to train

4) LSTMs are easy to overfit

5) LSTMs are sensitive to different random weights initializations

CHAPTER 4
PROPOSED WORK

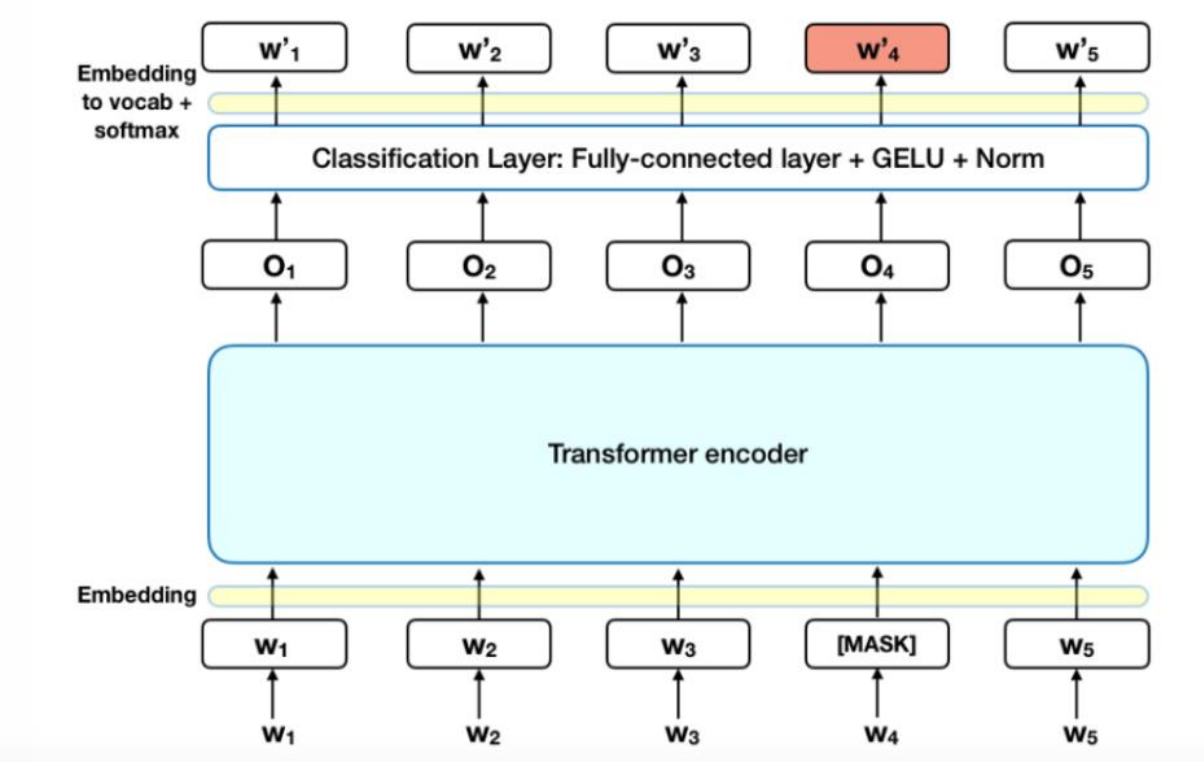
4.1 METHODOLOGY

In my proposed system, the drawbacks in the existing system are reduced by using Bi-directional Encoder Representations from Transformers (BERT) model for predicting the next word based on an input text. BERT is bi-directional in nature and it works in both ways from left to right and right to left. Thus, BERT doesn't follow sequential ordering. Bert uses two training mechanisms namely Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) to overcome the dependency challenge. In addition to the usage of BERT, to get as many possible next words based on the requirement, users are provided with an option to provide the number of possible words to be predicted and listed.

4.1.1 Advantages

- Prediction accuracy value is around 0.91
- Proximity doesn't influence the importance of word
- Case insensitive
- A list of possible words can be predicted

4.2 BLOCK DIAGRAM



4.3 ADMIN MODULE

4.3.1 Data pre-processing

The first phase is to pre-process the input text entered by the user. In BERT, pre-processing techniques like stop words removal, stemming and so on are not needed as it uses multi-head self-attention mechanism. Therefore, tokenization of words is performed. Tokenization tokenises and categorizes text into word tokenizers, punctuation tokenizers, white space tokenizers, regular expression tokenizers and so on.

4.3.2 Masking

BERT masks 15% of the words in input text to run the model to predict the possible next words.

4.3.3 Getting the number of words to be predicted from the user

The number of possible next words to be predicted for the input text is obtained from the user by providing a slider to select a value from 1-25.

4.3.4 Prediction

The tokens are encoded, decoded and punctuations are ignored. If <mask> is the last token, then it is appended with '.' to avoid prediction of punctuation. Then, list of possible next words are predicted and are displayed to the user.

4.3.5 Deploy this project on Streamlit

First, required python libraries are installed through command prompt using pip command. After installation, this project is deployed using a web app framework called Streamlit.

Streamlit turns data scripts into shareable web apps in minutes. Deployment of scripts in streamlit is purely on Python. No explicit front-end implementation is required. After this, a python file app.py is created, and the required libraries are imported.

4.4 USER MODULE

4.4.1 Enterring Input Text and value

Input text for which the next word to be predicted is entered by the user in the text box available in the web app. Then, user selects value (say x) from 1-

25 using the slider to get a list of x number of possible words predicted for the input text.

4.4.2 View Prediction

Finally, prediction is performed and possible next words are displayed to the user as a list of words.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 HARDWARE SPECIFICATION

PROCESSOR	:	DUAL CORE
HARD DISK CAPACITY	:	400 GB
MONITOR	:	14 “LG MONITOR
INTERNAL MEMORY CA	:	4 GB
KEYBOARD	:	LOGITECH OF 104 KEYS
CPU CLOCK	:	1.08 GHz
MOUSE	:	LOGITECH MOUSE
HARD DISK	:	1TB HDD / 512GB SSD

5.2 SOFTWARE SPECIFICATION

OPERATING SYSTEM	:	WINDOWS 10
LANGUAGE	:	PYTHON
TOOLS USED	:	STREAMLIT

5.3 SOFTWARE DESCRIPTION

5.3.1 STREAMLIT

Streamlit is an open-source app framework written in Python. Adrien Treuille, Thiago Teixeira, and Amanda Kelly developed streamlit to avoid pre-requiste knowledge of CSS, HTML and javascript for data scientistis, machine learning engineers to develop a web app. Other frameworks like Flask, Django

can also be used to deploy a project in web app but it requires the user to build the web app manually. This involves the knowledge of HTML, CSS and javascript by users to build web app. Streamlit on the other hand, can deploy any machine learning model or python project on web server at ease without requiring user to build the web app manually.

Streamlit is free to use and can be installed easily using command "pip install streamlit". Then, the python project can be executed using the command "streamlit run filename.py". This creates a local server with web app in URL "http://localhost/8501"

Why Streamlit?

- User-friendly
- Built in development server and debugger
- Open source
- Supports multi-page apps
- Supports theming
- Supports Command line features
- Caching option is available

5.3.2 PYNPUT

Pynput is a python library used to control and monitor input devices. In current version mouse and keyboard input and monitoring are supported. It can be used in a custom software framework and supports Python 2.6, 2.7 and 3.3 versions.

5.3.3 JINJA (TEMPLATE ENGINE)

Jinja, developed by Armin Ronacher, is a template engine for the Python programming language and is licensed under a BSD License. Similar to the Django web framework, it provides that templates are evaluated in a sandbox.

5.3.4 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. It is free, open-source and is designed to be highly readable. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python can be used on a server to create web applications. It can work in different platforms (Windows, Mac, Linux, Raspberry Pi, etc). It can also connect to different database systems. It is also used in implementation of machine learning, deep learning models for a particular domain.

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Currently, python has 2.6x, 2.7x and 3.x versions.

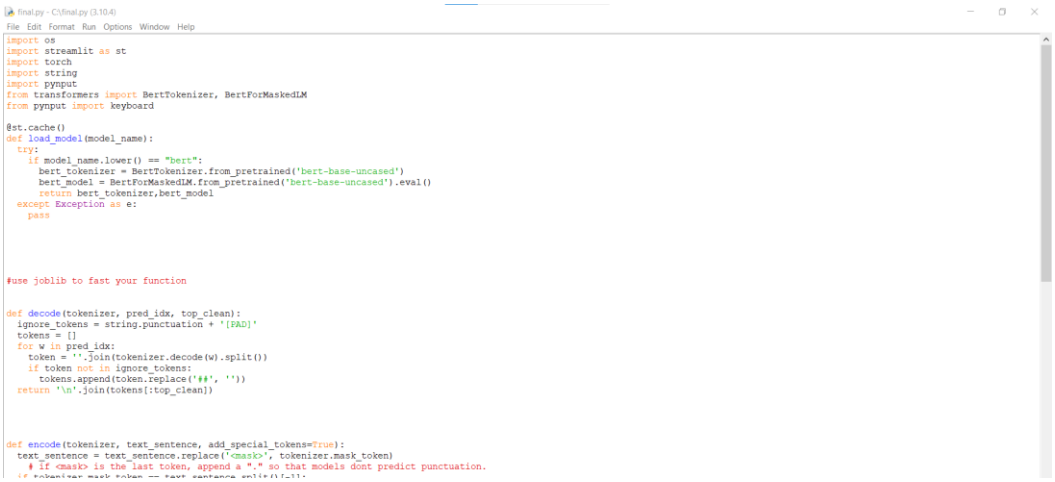


Fig 5.1: Python IDE

CHAPTER 6

RESULTS AND CONCLUSION

The pre-trained deep learning model BERT is used. Users upon entering input text and number of possible words to predicted, BERT function is invoked, tokenisation is performed, Model predicts the words and output consisting of list of predicted words is displayed to the user.

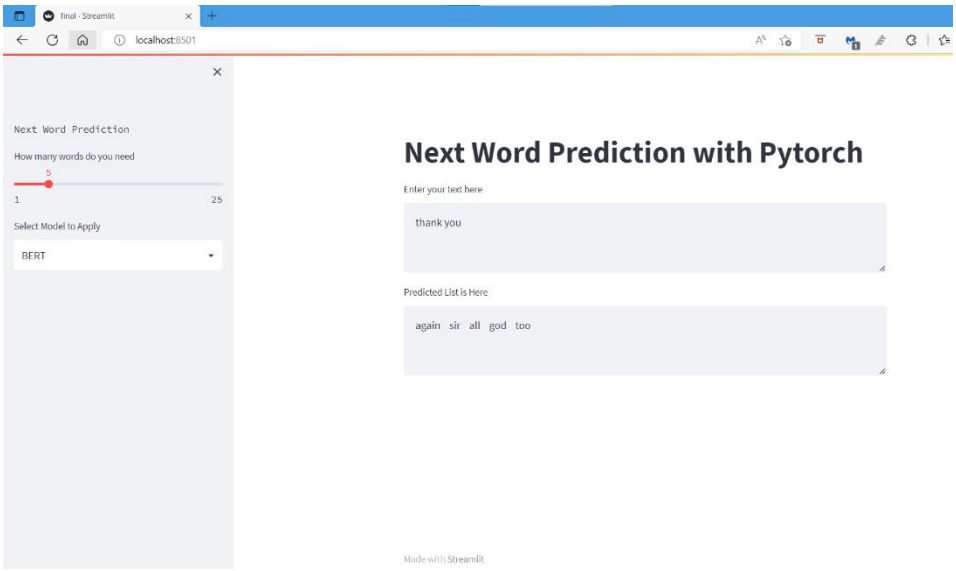


Fig 5.1

The above image displays the possible words “again, sir, all, god, too” predicted for input text “thank you”. The user has chosen five possible words to be predicted. This value can be varied by moving the slider point.

CHAPTER 7

FUTURE WORK

In future, this project can be improved by performing next sentence prediction and combining next word prediction and next sentence prediction for smart composing of emails, documents and this can be used by employees, business people. This saves time and humans can work on other complex operations.

APPENDICES

A.1 SOURCE CODE

final.py

```
import os
import streamlit as st
import torch
import string
import pynput
from transformers import BertTokenizer, BertForMaskedLM
from pynput import keyboard
@st.cache()
def load_model(model_name):
  try:
    if model_name.lower() == "bert":
      bert_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
      bert_model=BertForMaskedLM.from_pretrained('bert-base uncased')
.eval()
      return bert_tokenizer,bert_model
  except Exception as e:
   pass
```

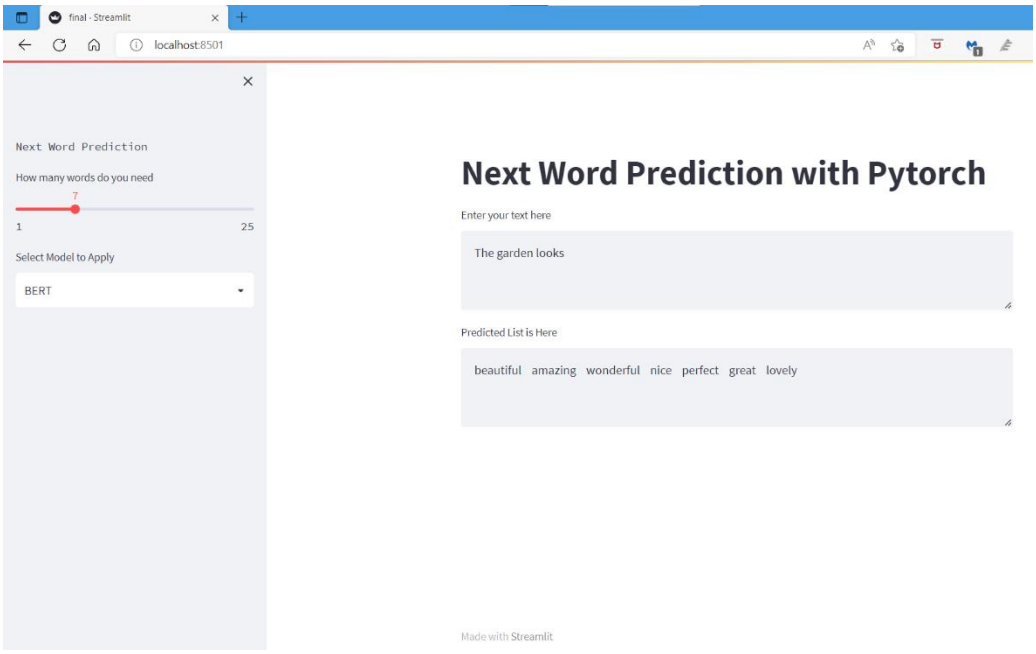
```
#use joblib to fast your function
def decode(tokenizer, pred_idx, top_clean):
  ignore_tokens = string.punctuation + '[PAD]'
  tokens = []
  for w in pred_idx:
    token = ''.join(tokenizer.decode(w).split())
    if token not in ignore_tokens:
      tokens.append(token.replace('##', ''))
  return '\n'.join(tokens[:top_clean])
def encode(tokenizer, text_sentence, add_special_tokens=True):
  text_sentence = text_sentence.replace('<mask>', tokenizer.mask_token)
    # if <mask> is the last token, append a "." so that models dont predict punctuation.
  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(text_sentence, top_clean=5):
  #=====================BERT =====================
  input_ids, mask_idx = encode(bert_tokenizer, text_sentence) with torch.no_grad():
  predict = bert_model(input_ids)[0]
  bert=decode(bert_tokenizer,predict[0,mask_idx,:].topk(top_k) .indices.tolist(),  top_clean)
  return {'bert': bert}
def get_prediction_eos(input_text):
  try:
  input_text += ' <mask>'
  res = get_all_predictions(input_text, top_clean=int(top_k))
  return res
  except Exception as error:
  pass
try:
  st.title("Next Word Prediction with Pytorch")
  st.sidebar.text("Next Word Prediction")
  top_k = st.sidebar.slider("How many words do you need", 1 , 25, 1) #some times it is possible to have less words
  print(top_k)
```

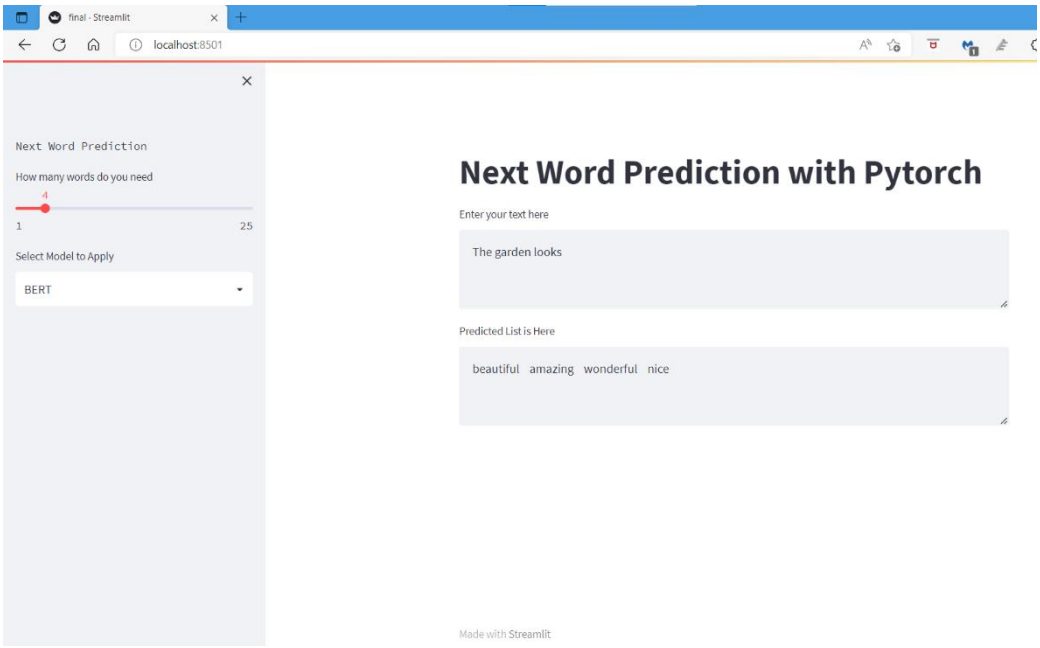
```
        model_name = st.sidebar.selectbox(label='Select Model to Apply', options=['BERT', 'XLNET'], index=0,  key = "model_name")

        bert_tokenizer, bert_model  = load_model(model_name)

        input_text = st.text_area("Enter your text here")

        #click outside box of input text to get result

        res = get_prediction_eos(input_text)

        answer = []

        print(res['bert'].split("\n"))

        for i in res['bert'].split("\n"):

            answer.append(i)

        answer_as_string = "    ".join(answer)

        st.text_area("Predicted List is Here",answer_as_string,key="predicted_list")

    except Exception as e:

        print("SOME PROBLEM OCCURED")
```

A.2 SCREEN SHOTS

INPUT VALUES



PREDICTED VALUES



REFERENCES

1. Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” arXiv:1810.04805v2 [cs.CL] 24 May 2019
2. Praveenkumar Katwe, Aditya Khamparia, Kali Prasad Vittala & Ojas Srivastava “A Comparative Study of Text Classification and Missing Word Prediction Using BERT and ULMFiT” ISBN: 978-981-15-5258-8 01 August 2020
3. Sourabh Ambulgekar, Sanket Malewadikar, Raju Garan and Dr. Bharti Joshi “Next Words Prediction Using Recurrent Neural Networks” ITM Web of Conferences 40:03034 January 2021
4. Aejaz Farooq Ganai, Farida Khursheed “Predicting next Word using RNN and LSTM cells: Statistical Language Modeling” DOI: 10.1109/ICIIP47207.2019.8985885
5. Joel Stremmel, Arjun Singh “Pretraining Federated Text Models for Next Word Prediction” arXiv:2005.04828v3 [cs.LG] 17 Aug 2020
6. Jonathan J. Webster & Chunyu Kit “Tokenization as the initial phase in NLP” Jonathan J. Webster & Chunyu Kit
7. Khrystyna Shakhovska, Iryna Dumyn, Natalia Kryvinska and Mohan Krishna “An Approach for a Next-Word Prediction for Ukrainian Language” Wiley, 15 August 2021
8. Keerthana N, Harikrishnan S, Konsaha Buji M, Jona JB “Next Word Prediction” ISSN: 2320-2882, IJCRT Volume 9, Issue 1 December 2021
9. Jamshid Mozafari, Afsaneh Fatemi, Mohammad Ali Nematbakhsh “BAS: An Answer Selection Method using BERT Language Model”

10. Dong-Ho Lee, Zhiqiang Hu, Roy Ka-Wei Lee “Improving Text Auto-Completion with Next Phrase Prediction” EMNLP 2021 pages: 4434-4438