NATURAL LANGUAGE WORD PREDICTION MODEL USING DEEP LEARNING

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Sripath Cherukuri (221710302061)

Yelagonda Aniketh Reddy (221710302065)

Bollempally Laxman Reddy (221710302011)

Bompelli Sai Krishna (221710302012)

Under the esteemed guidance of

Mr. Jethya

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

HYDERABAD

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "**NATURAL LANGUAGE WORD PREDICTION MODEL USING DEEP LEARNING**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s)	Name	Signature(s)
221710302061	**Sripath Cherukuri**	
221710302065	**Yelgonda Aniketh Reddy**	
221710302011	**Bollempally Laxman Reddy**	
221710302012	**Bompelli Sai Krishna**	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**NATURAL LANGUAGE WORD PREDICTION MODEL USING DEEP LEARNING**" is a bonafide record of work carried out by **Sripath Cherukuri (221710302061), Yelgonda Aniketh Reddy (221710302065), Bollempally Laxman Reddy (221710302011), Bompelli Sai Krishna (221710302012)** submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Head of the Department**
Mr. Jethya	**Dr.S.Phani Kumar**
(Assistant Professor)	**Professor**

ACKNOWLEDGEMENT

Our Mini Project would not have been successful without the help of several people. We want to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the workshop.

We are incredibly thankful to our honorable Pro-Vice-Chancellor, Prof. N.Siva Prasad, for providing the necessary infrastructure and resources for the accomplishment of our seminar. We are highly indebted to Prof. N. Seetharamaiah, Principal, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved Prof. S. Phani Kumar, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this seminar and encouragement in completion of this seminar.

We now wish to express our deep sense of gratitude to Dr. S Aparna, Assistant Professor, Department of Computer Science and Engineering, School of Technology, and to Mr. Jethya, Assistant Professor, Department of Computer Science and Engineering, School of Technology, for the esteemed guidance, moral support and invaluable advice provided by them for the success of the Summer Internship.

We are also thankful to all the Computer Science and Engineering department staff members who have cooperated in making our seminar a success. We want to thank all our parents and friends who extended their help, encouragement and moral support directly or indirectly in our seminar work.

Sincerely,

Sripath Cherukuri (221710302061)

Yelgonda Aniketh Reddy (221710302065)

Bollempally Laxman Reddy (221710302011)

Bompelli Sai Krishna (221710302012)

TABLE OF CONTENTS

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
GRU	Gated Recurrent Unit

1. ABSTRACT

Generally, word predictions or word suggestions are given based on occurrence statistics, which may have massive data storage requirements and don't consider the text's original meaning. The word prediction is used widely in many applications these days. With the release of texting apps like WhatsApp, Telegram, and similar apps like these, there is more emphasis on text data in recent years.

Having Word prediction at hand, we can search for information efficiently and convey what we want to others quickly, saving a lot of time. The best example is Google search, as it predicts the words based on the last word typed. The predictive keyboards in smartphones these days are also one of the best examples of word prediction. Most people rely upon these words prediction tools and keyboards.

There are various methods to implement this model, and we can use the LSTM neural network which is recurrent neural network having memory gates or Naïve Bayes and other methods like Decision tree, Random forest algorithms to predict the next word. This paper will implement the LSTM model, which is best known for Language modeling tasks such as summarizing the text, anticipating the next term, and identifying the context.

2. INTRODUCTION

Users are enormously using web search engines more and more these days, and with messengers like WhatsApp, Telegram, and more apps like these, people are more used to chatting these days. So, there is more emphasis on text data in recent years.

Generally, the websites store the text data in a database. While the user is filling up a form or typing something, or searching in their website, the algorithm shows the relevant words while typing without typing the whole term or text. According to this, what we can understand is that the website can predict which word users might need shortly.

Word prediction is a technology that predicts the words based on the sequence entered by the user or a person simply by reducing the number of keystrokes necessary for typing words. There are several methods in which this technology can be implemented. The first is to fetch the relevant terms from the database by using some fetching algorithm. The second is to have a machine learning model trained on a massive amount of text data so that the model can predict the words based on sequences entered by users.

Word prediction is one of the simple applications of natural language processing (NLP), which uses a language model to understand the relation between the word sequences. Some words or characters are used to predict the chances or the probability of the upcoming characters or words. The traditional language models based on probability, such as the Hidden Markov models, depending on Random Field models, and Decision Trees, are used widely to find solutions to the word prediction problem. With the advancement of deep learning technology in recent years, the language models based on deep neural networks have been extensively used in NLP. A recurrent neural network (RNN) can learn the sequence relation between the words within a sentence and can apply that semantic information to the present situation, enhancing the semantic relevance within the sentence.

Moreover, the information which is useful will be lost in processing the long-time relation due to problems such as gradient dispersion. To solve such problems, Hochreiter and Schmidhuber adopted a long short-term memory (LSTM) network by adding order dependence between word sequences to make fully use past information or memory sequences. The gated recurrent unit (GRU) network in LSTM solved the dispersion problems with RNN's.

As we look further into word prediction, there are different ways of prediction, such as word-to-word prediction and letter-to-word prediction. Here we are only focusing on word-to-word prediction. Before we see what the problem definition is, the advantages and limitations of the project, and its applications, we see what machine learning is, what deep learning is, what artificial intelligence is, and other terms.

Artificial Intelligence: Artificial Intelligence means the replication of human cognitive skills in machines, that are developed to think like humans and behave like the humans or replicate their activities. Moreover, the word artificial intelligence is connected to any computer or a machine that shows characteristics similar to a human mind, examples such as learning and problem-solving. A machine is said to be AI if and only if it can thinks rationally like humans and chooses the best possible option in the given situation.

Machine Learning: Machine Learning (ML) is the branch of computer calculations that get better automatically upon encounter. It is a subset of artificial intelligence. Machine learning calculations construct a depiction based on test sample information, called as training data, to create expectations or probabilities without being unequivocally coded to perform such operations. Machine learning calculations are used in a extensively in broad range of applications, such as computer vision, where it is difficult or unfeasible to develop normal calculations to perform the required tasks. A subset of machine learning is closely associated with computational information, focusing on making predictions or assumptions using computers.

Deep Learning: Deep Learning is the subset of machine learning strategies based on artificial neural systems with representation learning. Learning can be directed, semi-supervised or unsupervised. Deep-learning models such as neural networks, conviction systems, redundant neural systems, and convolutional network systems have been connected to areas counting OpenCV, machine vision, machine interpretation, bioinformatics, medicate plan, where they have delivered comes about comparable to and in a few cases outperforming human master execution. There are particular learning algorithms that we can implement based on the problem we are dealing with, so let us see those learning algorithms.

Supervised Learning: Supervised Learning, also known as Administered Learning, is the most famous worldview for AI. It is the least demanding to comprehend and the most straightforward to carry out. It is the same as showing a kid with the utilization of glimmer cards. Given the data as models with identities, we can supervise a learning operation. These model combines separate elements, giving access to the calculation to anticipate the identity for every model, and providing it input regarding if it expected the right answer. Over the long run, the calculation will figure out how to vague the specific idea of the relationship among models and their names. When wholly prepared, the regulated learning calculation will want to notice another, at no other time seen the model and anticipate a decent mark for it. Supervised learning is regularly portrayed as an assignment situated along these lines. It is profoundly centered around a solitary undertaking, taking care of an ever-increasing number of guides to the calculation until it can precisely perform on that task. It is the

learning type that we will most likely experience, as displayed in many primary applications, such as Ad Popularity and Spam Classification.

Unsupervised Learning: Unsupervised Learning Solo Learning is especially something contrary to Directed Learning or Supervised Learning. It includes no marks. All things being equal, our calculation would be taken care of a ton of information and given the devices to comprehend the properties of the information. From that point, it can figure out how to gather, bunch, and put together the information in order with the result that a human (or other clever calculation) can come in and figure out the recently coordinated information. For instance, imagine a scenario where we had an enormous data set of each exploration paper ever distributed. We had a solo learning calculation that realized how to bunch these in such a manner, so we were consistently mindful of the current movement inside a specific examination area. Presently, we start to begin an examination project ourselves, guiding our data into the neural network that the calculation can see. We review our work and make list of points, the calculation does, ideas about related works, we wish to refer, and operations that may assist us in pushing that area of examination forward. With such a device, our profitability can be incredibly helped. Since unaided learning depends on the information and its properties, we can say that unaided learning is information-driven. The results from an unaided learning task are constrained by the information and how it is organized. We might see solo learning crop up a few regions are Recommender frameworks, purchasing propensities, and gathering client log.

Reinforcement Learning: Reinforcement Learning is the process of preparation of AI models to make a grouping of choices. The model finds out how to perform a task which is uncertain, an set up in a extreme climate. In support of learning, computerized reasoning appearances it appears a game. The computer uses different ways to think of solution to the problem. To get the learning model to do the required job, the fake insight gives information which is correct or punishments due to the actions it does. Its objective is to extend all-out remuneration.

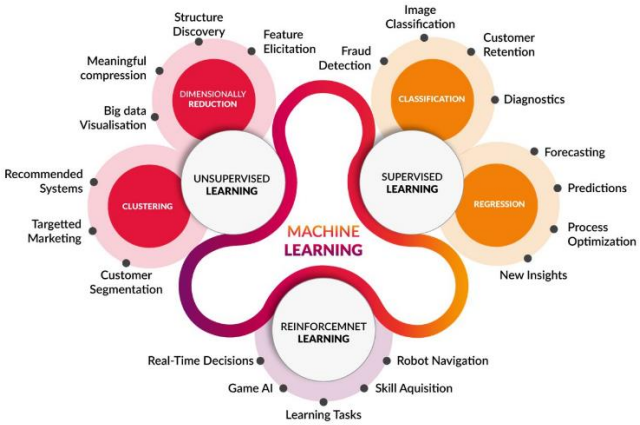

Figure 2.1 Types of Machine Learning

2.1 MOTIVATION FOR THE PROJECT

The main motto behind this project is to get valuable insights and knowledge regarding the word prediction technology or tool. These days with the evolution of computer technology, we hear terms like Machine Learning, Deep Learning, and Artificial Intelligence. So, in this project, we will be predicting words based on the user's sequence using the Deep Learning model. It helps us to learn the basics of Deep Learning and Machine Learning.

By taking up this project we can learn so many useful insights about python, Deep learning and Machine learning. We get to work with the language models and gain valuable knowledge from this project.

Although there are many Deep Learning models out there used for word prediction, we will be trying to achieve good accuracy by developing a very simple sequence prediction model that will be trained on text files and then used to make predictions.

2.2 APPLICATIONS OF NATURAL LANGUAGE WORD PREDICTION

There are several applications of word prediction. Some of them are listed below:

• A good next word prediction model or application can be useful to normal users, as it reduces the reducing the number of keys pressed and minimizing grammatical errors and spellings, which are these days.

• People having issues with the movement of muscles are not required to enter all the keys, they might enter just few words without entering all words.

• Children having issues with writing may require word prediction system as additional help, so they can express their ideas properly.

• Users who are suffering from typing mistakes or spell checks can broaden the range of vocabulary and learn proper words for the context while writing.

• It can be used on websites for form filling and other applications.

• It is also used in smartphones for quick texting in messengers like WhatsApp or Telegram and similar apps.

2.3 LIMITATIONS

• Even though the Deep Learning model is trained on text data, it might not give good word suggestions every time.

• To get proper word suggestions, the Deep Learning model might have to train on massive amounts of text data which takes much time.

3. LITERATURE REVIEW

3.1 INTRODUCTION

A literature review is very crucial phase in the project or application building process. Before building any software, it is always a good choice to discuss about factors such as, time, economy, and company strength. After this data about these things are acquired, the coming steps will let us know what operating system and, what language can be used to develop the project. Once the developers start building, they need additional assistance. We can obtain this external support from experienced developers, books, or websites or forums. Before proceeding with the project, we will take the above points into consideration for building the required system.

3.2 RECENT WORK/RESEARCH

There has been a lot of research in past few years on topics related to natural language processing and Deep Learning. These have been the buzz words past decade and there were massive developments in the field of NLP and Deep learning. Word predictions is one of the most used application of NLP. There has been lot of research going on how to predict the next word basing on what the user enters. Details of those research articles is mentioned below.

In 2000 Yair Even-Zohar and Dan Roth worked on a classification approach to word prediction. It was an approach based on learning representations, for words. This approach is completely context dependent. They used the SNOW architecture. This model was a sparse network having linear units. One important characteristics of the sparse network is that the complexity of processing an example depends on the number of features which are active in it. It was a multi-class predictor. The input sentence, along with expected word of interest in it, was mapped into a set of features which were active. This representation is presented to the input layer of SNOW network and passes forward to the target nodes.

In 2007 Al-Mubaid made research on a learning-classification approach for predicting the next word. This method suggested that each word is treated as a feature vector. Then to develop a machine 8 learning model to train on word classifiers and identify the context of the word. Suppose if there is a set of words for example consider weak and week both sound similar. But there each and every word has a context to be used in. Now the machine learning model was used to predict what word to be used when the letter ‘w’ occurs. A SVM with training inductive learning model was also used, which has literature support and achieved good results.

In 2011 Ilya Sutskever, James Martens and Geoffrey Hinton worked on generating text using recurrent neural networks. In this paper they demonstrate the power of RNNs trained with the new Hessian-Free optimizer (HF), by applying them to character-level language modelling tasks. The

standard RNN structure, while effective, is not suited for all tasks, so they introduced a new RNN variant that uses multiplicative (or "gated") connections which allowed the current inputted character to determine the transition matrix from one hidden state vector to the next one. They trained the multiplicative RNN with the HF optimizer for almost five days on eight high-end Graphics Processors, they were able to surpass the performance of the best previous single method for character-level language modelling – a hierarchical non-parametric sequence model.

In 2012 Qiang Liu, Ainur, Jonh.C.platt studies the different computational approaches for sentence completion. They approached the problem with two methods. First one is to use local lexical information, such as n-grams of a language model. The second is to evaluate global coherence, like latent semantic analysis. They used the N-gram language model which was a trained model. They also studied about the RNN its recent development. They used the language modelling for sentence reconstruction.

In 2014 Rui Lin, Shujje Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li proposed a paper with a novel hierarchical recurrent neural network language model (HRNNLM) for document modelling. They designed the two-step training approach, in which sentence-level and word-level language models are approximated for the combining in a pipeline style. Examined by the standard sentence reordering scenario, HRNNLM is proved that it achieved better results in modelling the sentences Experimental results also suggested a significant lower model perplexity, followed by a practical better translation result when applied to a Chinese-English document translation reranking task.

In 2015 Md. Masudul Haque, Md. Tarek Habib and Md. Mokhlesur Rahman produced a paper named automated word prediction in bangla language using stochastic language models. They proposed the N-grams method. N-gram model is a probabilistic language model where the approximate matching of next element is more. Generally, probability is based on counting the most number of occurrences of word in most cases. The chance of a word depends on the last word. Unigram means single item from a given sequence. Bigram is known as first-order Markov model which looks one word into the back and trigram is known as second-order Markov model which looks two words into the back and quadrigram is another third-order Markov model which looks three words back and similarly an Ngram language model is N-1 Markov model which looks N-1 words back.

Carmelo Spiccia, Agnese Augello, Giovanni Pilato also produced a paper in the same year named as word prediction methodology for automatic sentence completion. They proposed another methodology, based on Latent Semantic Analysis. They employed asymmetric Word-Word frequency matrix is to achieve higher scalability with large training datasets compared to the normal Word Document approach. They also proposed a function for scoring candidate terms for the missing words in a sentence. In the pre-processing step they removed irrelevant information from the text.

Moreover they separated the text into sentences. In the frequency analysis step they extracted the statistical data. Then they built the frequency matrices. Then in the dimensionality reduction step they applied the TSVD to frequency matrices in order to associate a small vector of real values to each word present in the dictionary. Finally, in the word prediction step, they used those values to find the missing word in a sentence.

In 2016, Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba worked on sequence level training with recurrent neural networks. They proposed a sequence level training algorithm, that directly optimizes the metric used at time of testing, such as BLEU or ROUGE. They suggested a simple baseline which uses its model prediction during training and also has the ability to back propagate the elements through the entire sequence.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, Wang Ling in the same year made research on composing words into sentences using reinforcement learning. Their model included two steps the first, a sentence representation model and a reinforcement learning algorithm and the second, to learn the tree structure that is used by the sentence representation mode. Their sentence representation model followed the Stack-augmented Parser-Interpreter Neural Network, SPINN is a shift-reduce parser that implements Long Short-Term Memory as its function. They used reinforcement learning to discover the best tree structure for the task.

Melissa Roemmele also in the same year worked on paper name writing stories with the recurrent neural network. A simple RNN model that has an one input layer, one hidden layer, and output layer connected respectively by three weight matrices is used in this paper. In this paper a SoftMax 10 classifier was applied to the output layer to get the probability distribution of the next word over all dictionary words. The training is done by minimizing the function, which was defined as the negative log-likelihood of the probability of each actual word in the training sequence. Then the gradient of this cost was back-propagated in order to update the weight matrices.

In 2017 Dzmitry Bahdanau Philemon BrakelKelvin Xu Anirudh Goyal a papers was published named as an actor-critic algorithm for sequence prediction. This paper implemented an approach in which neural network models are trained on actor-critic methods that use reinforcement learning. They proposed and studied a procedure for training sequence prediction networks that aims to improve their test time metric. They also trained an extra network called the critic to output, where the value of each token, defined as the expected task-specific score. Moreover, they showed how predicted values can be used to train the main sequence prediction network, which they referred to as the actor. The theoretical foundation of their method is that, under the assumption that the critic computes exact values.

In 2018 Henrique X. Goulart, Mauro D. L. Tosi, Daniel Soares-Gonc¸alves, Rodrigo F. Maiaand Guilherme Wachs-Lopes produces a paper that tells us about a hybrid model that uses Naive Bayes and latent information for word prediction. Naive Bayes is a model based on probability that is used in NLP as a n-gram, which was developed by Bayesian networks that are applied in the machine learning area. Full n-gram models are not feasible to be implemented due to its complexity, as they require a combined probability table, which grow exponentially with the insertion of new variables. They stated that its variables can be divided in to cause and effect behaviours, thereby, it can be assumed that the effects are conditionally independent between themselves, which in turn reduces computational cost of the model.

In 2019 Pratheek, Joy Paulose published a paper, this paper proposes a novel approach for finding answer keywords from a given body of news text or headline, based on the questions that was asked, where the questions would be on the nature of current affairs or recent news, with the use of Gated Recurrent Unit (GRU) variant of RNNs. Thus, this ensures that the answers provided are relevant to the content of query that was based on previous words. A GRU unit is a recurrent unit that is used to memorize things like words. Adding this GRU unit gives additional abilities to RNN.

In the same year as above Omor Faruk Rakib, Shahinur Akter, Md Azim Khan, Amit Kumar Das, Khan Mohammad Habibulla published a paper on Bangla word predictions using the RNN based on N-grams with a GRU unit added to it. RNN's are used for word prediction in general, but they do not have the ability to remember the words multiple timestamps backward. So as this problem persists the RNN's are not completely good enough for word predictions, there are some flaws. To overcome this problem they added a GRU unit to it. This Gated recurrent unit helps the model to remember multiple timestamps backwards so it can give adequate predictions based on the entered sequence.

4. PROBLEM IDENTIFICATION & OBJECTIVES

Problem analysis is the essential step while working on projects like this. It gives us a complete idea of the problem and the requirements for proceeding with the project. Problem analysis deals with the problem definition, hardware requirements, and software requirements, existing methods, and what methods can be implemented to achieve the final result, scope, and limitations. This detailed analysis provides us with valuable insights, which helps us understand better. In this chapter, we deal with the following:

- Problem definition
- Objective
- Requirement analysis
- Existing method
- Methods to implement the project
- Feasibility study such as Financial, Operational, Resource, and Time feasibility.

4.1 PROBLEM DEFINITION

As we discussed earlier, the problem here is to predict the next word for a sequence entered by the user and develop and train a Deep Learning model on text data to predict the words adequately based on what the user enters. The emphasis here is only on word-to-word prediction. The Deep Learning model should give word suggestions after the user enters his/her sentence or a sequence of a certain length. The model will be trained on a text file and will be used to make word suggestions.

4.2 OBJECTIVE

The main aim of this project is to predict words for a sequence entered by the user. Not only predicting or giving suggestions, but we also will look into how we can improve the model's accuracy and make better predictions while training on a short amount of text data. We will also learn how these Deep Learning models were developed and implemented and learn from this project.

4.3 REQUIREMENTS ANALYSIS

Now coming on to the requirements for this project, it depends on which method you will choose while implementation as there are many ways in which this project can be implemented. But the generalized requirements of both hardware and software are mentioned below:

Hardware requirements:

• Processor: Intel-i5/Intel-i7 or any AMD equivalent

• RAM: 8GB minimum or higher

• GPU: A decent graphic card with CUDA rating 7.5 - 4GB memory (This is only needed if we choose to develop a model from scratch and train it on large datasets)

Software requirements:

• Operating system: Windows 10/Linux

• Python installation: Anaconda

• IDE: Visual studio code or PyCharm

4.4 EXISTING METHOD

Human beings do not think new thoughts every time, every second and every day. While reading through a book, humans acknowledge each letter, and word, and sentence based on our cognitive skills of the past words. People can remember thoughts that are occurred in the past, and our ideas are relatable. Old Deep Learning neural networks cannot work properly during this kind of situations or circumstances, which is a significant disadvantage.

For example, imagine we want to classify what kind of food is cooked by a person every night throughout the week. As a fellow human being people can find what the person cooks today basing on the previous night's food. But old neural networks fail to do this because they do not have a memory of the previous ones. To overcome this problem of not memorizing the previous data or sequence Recurrent neural networks are introduced.

A cycle forwards the data from one step of the network layer to the next layer. For example, let us consider a language model performing operations to guess the upcoming word based on the last words. If we are trying to make predictions for the final word in "Earth revolves around the sun," we do not require any other conditions as the sun is the pretty perfect or assertive next word. In this type of sentences, the space between the adequate data and its required data has a relatively small gap.

Moreover, there are also situations where other conditions are required. Now try to give suggestions for the terminal word in the sentence "he is bad at singing and recently he stood up for my country in singing." previous data or context suggests that the next word might be cultural activity the name of an activity. But here, if we would further like to be specific down to what was the name of the activity, we need the knowledge about what is singing from way back in time. In this example the relation between the given data and the point of time where, it is required becomes huge. As RNN's can only

remember only single timestamp backward in time they have problems dealing with examples mentioned above.

4.5 METHODS TO IMPLEMENT THE PROJECT

i. **Using the pre-trained model**: Using the pre-trained model gives us a head-start as we do not need to develop the model from scratch. We can load the model and use it in their programs we like. This method saves time and reduces the burden on us. But in this method, we might not be able to change the structure of the model.

ii. **Developing the model from scratch**: Developing the model from scratch gives us access to change the layers of the Deep Learning model according to their use. We can also make any changes to the model in between. Now we can pick a data set and train the model on it until good accuracy is achieved. Now, this model can be saved and used whenever needed.

4.6 FINANCIAL FEASIBILITY

The cost indulged in this project is zero as this is done for academic purposes. But if this project was to be released into the real-world market even then, it would not incur any costs. We can call it a low budget project. The cost incurred on the first two methods mentioned above is almost negligible.

4.7 OPERATIONAL FEASIBILITY

The operational feasibility deals with how good the mentioned model finds a solution to the problem. Any one of the above methods satisfies the problem partially but needs to be investigated further as the machine learning model's accuracy may improve over time or maybe in the future due to technological advancements. Method one and method two mentioned above use the systems resources as mentioned in the requirements section. This project also fits into real-world business as machine learning is one of the recent advancements in the technology sector.

Various organizations are working on this type of project and have demand for this type of project. This could be an excellent real-world application if it could be enhanced more. It is also affordable in terms of expenditure, and it should be continuously refined wherever there is scope for improvement if deployed as a real-world application. Process designing and building require the adequate and time to time fixes and maintenance systems to meet the stated parameters. A system may perform its stated purpose more appropriately when its hardware and functioning elements are appropriately designed.

4.8 RESOURCE FEASIBILITY

The number of resources used by the program depends on what method is being to train and develop the machine learning model. The methods mentioned above (Developing the model from scratch) method use most system resources while running. When the model trains on such massive datasets,

it will require much memory for processing that text data and store the results. The CPU and GPU are also continuously working to provide speedy access to the data for the machine learning model to train. The other method uses limited resources, probably less than the second method, as it does not have a training process in that method.

4.9 TIME FEASIBILITY

A time feasibility study is the amount of time taken for project completion. The time taken for the project completion is wholly based on the technical expertise, that is, the programmer's knowledge about the project and his or her capacity to do programs. But one particular thing that is known is that it will take a lot of time to train the model on test samples because sometimes the datasets may be massive. For example, a dataset may have one or two million words sometimes. So, while working on such colossal text files, it takes time.

4.10 BOUNDARIES

This project is only specific to word-to-word prediction. The predictions made are entirely random and generated dynamically based on the user's input and the Deep Learning model's ability to find adequate words based on its experience and training. However, this is not all, and there is much more to it.

Generally, the word prediction happens based on the user's search data, and word suggestions are given based on this data. Here the word suggestions are not specific to the user. It just suggests words based on the previous search data and newly entered sequence. In the future, this next word prediction may extend so that, with the help of Deep Learning models, we might be able to predict text for a particular user based on his words, and how he chats or texts with others tells us his character. In this way, we can extend this application, and there is more research to be done.

5. SYSTEM METHODOLOGY

This chapter deals with complete workflow in which this project was implemented. Every step is explained clearly explained below the diagram.

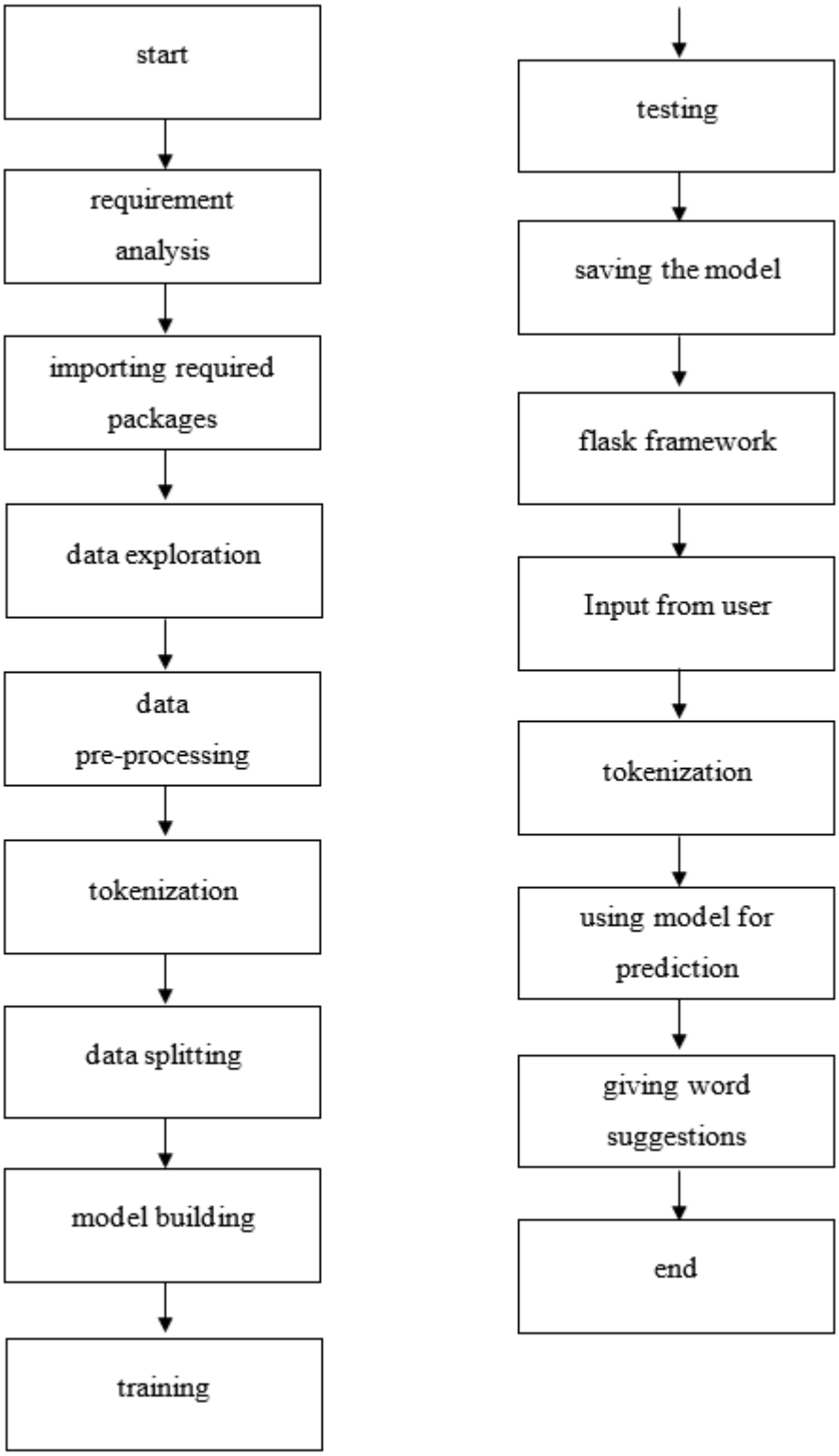

Figure 5.1 Overall project Workflow

5.1 REQUIREMENT ANALYSIS

Before starting any project, there is certain process to be done. Information regarding the project and its details must be gathered. These details include libraries required for implementation of the project, software and hardware requirements of the project. The generalized hardware and software requirements are mentioned below:

Hardware requirements:

• Processor: Intel-i5/Intel-i7 or any AMD equivalent

• RAM: 8GB minimum or higher

• GPU: A decent graphic card with CUDA rating 7.5 - 4GB memory (This is only needed if we choose to develop a model from scratch and train it on large datasets)

Software requirements:

• Operating system: Windows 10/Linux

• Python installation: Anaconda

• IDE: Visual studio code or PyCharm

5.2 IMPORTING REQUIRED PACKAGES

For every project there are certain libraries or modules which provide certain functionality for the program. These are very essential for the program in order to run the program properly. So, at the start of the program required packages are imported. Some of the library/module names are mentioned below.

• TensorFlow

• String

• Requests

• NumPy

• Keras

In-depth details about these packages or modules will be discussed in the next chapter clearly.

5.3 DATA EXPLORATION

Now for every research project almost all of them have a dataset. So, there is a need for the gathering a proper dataset. In our case as the title of the project suggests, it deals with the natural language i.e., text data for example consider English language. So any text data in proper format is acceptable for

this kind of project. So we here used that text file of a book named “A Tale of Two Cities” written by Charles Dickens which is available on Gutenberg (a free eBook website).

5.4 DATA PRE-PROCESSING

In accordance to provide the sufficient information to the model, the data should be cleaned or processed first. In this case our text data needs to be cleaned and tokenized in order to pass it to the Deep Learning model. There are various unnecessary symbols in a text file that needs to be removed, so this implies cleaning the text file is mandatory in our case. The text data even after cleaning it is raw data. The Deep Learning model can only understand only numbers or weights. So, tokenization is needed here.

5.5 TOKENIZATION

Tokenization is a class provided by keras for setting up text documents for computer projects related to Deep learning. Now this tokenizer class provides functions to convert a raw textual data either by dividing the text into sequences of numerical values or into a vector where every word has a binary coefficient. By using this class raw text is now vectorized or changes into integers.

5.6 DATA SPLITTING

Now in this project is completely based on text and text predictions. So the text will be framed into sequences of certain length and be divided into train inputs an train targets. Now every sequence we obtain will be of certain length in this case it is (5+1). The first five words will be given to model and 18 it predicts the next words. The total length of text sequence is six. So the model will also be trained like this.

5.7 MODEL BUILDING

To train a model first a model should be built. LSTM also known as Long short-term memory is a recurrent neural network was used in this project. There are five layers in this model. Those five layers are namely one Embedding layer, two LSTM layers with 100 units each, and two dense layers. This model is implement using keras module.

i. **Embedding layer**: This is layer provided by keras. This layer is used neural networks are used for training on text data. It requires the text data to be numerical data or integer data. This is an improvement over old method called bag of word model. In this layer the words are represented by integer vectors mostly having zero values.

ii. **LSTM layer**: This is also a layer that is provided by the keras module. It is a recurrent neural network layer, that has hidden units and memory gates that are used to learn the relation between the words in the text. In our model there are two LSTM layers used so that we can improve the accuracy.

iii. **Dense layer**: The dense layer is most used layer in the neural networks. It is deeply connected layer. This layer helps us to perform some output functions such as SoftMax. It helps us to perform matrix-vector multiplications and scaling and rotation. There are two dense layer in this project.

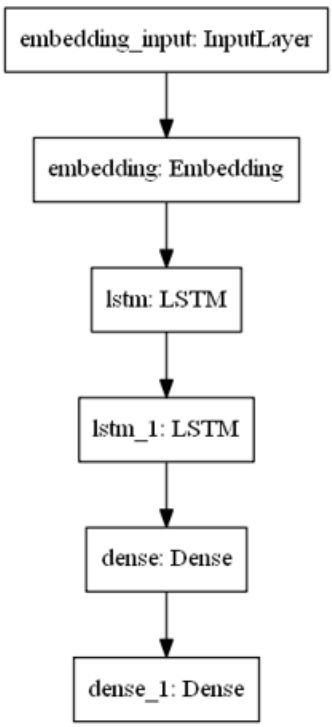

Figure 5.2 Model Architecture

5.8 TRAINING

After building the LSTM model using keras we need to train the model in-order to use the model for making predictions on what the next word will be. As it was discussed earlier the total sequence length maintained in this project is six. So, its like if five words were given as an input it gives the suggestions for the next word.

5.9 TESTING

In this phase the model is tested by giving various inputs to see how it is able to make suggestions on what the next word will be. If the model is able to make adequate predictions for the next word, then then the model is saved to the disk for further use or else it is checked and changes are made for improving the performance further more.

5.10 SAVING THE MODEL

After training the model it will be checked for its performance. Then the model is saved to the disk in h5 format which is a keras format for saving models. Keras also provides save model function to save its models.

5.11 FLASK FRAMEWORK

Flask is a small framework built for python. It can be seen as a smaller version of Django which is also a python framework with additional features. After training and testing, the saved model needs to be deployed so that the user can use it for making next word suggestions. So, we deploy this model using flask framework which has an html page and a stylesheet for it. There is also another python file where the predictions happens when user clicks the button oh html page it prints the results to the page. The file structure of the flask framework is shown below.

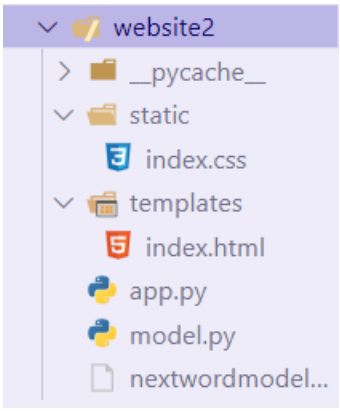

Figure 5.3 Flask File Structure

5.12 INPUT FROM USER

After running the python app file, the user will get the link for html site which is called localhost. So, by clicking on this link the user will be directed to the page where he or she needs to enter the sequence of five words or more to get word suggestions. After the user enters and clicks on predict button, the next word suggestions will be given on the screen.

5.13 USING MODEL FOR PREDICTIONS

The user after entering the sequence that he wants to make predictions, the user clicks on the predict button then the control will be passed to the app python file where the prediction function runs for the user given sequence and send the next word suggestions to the html to display to the user.

5.14 GIVING WORD SUGGESTIONS

The saved model will be loaded into the flask when the user clicks on predict. The user's sequence will be passed to the model for getting predictions. So, the model is bale to make predictions or next word suggestions on its training experience. The next word suggestions are completely random and predicted on spot.

6. OVERVIEW OF TECHNOLOGIES

This chapter deals with the technologies used in the making of the project. Every project has some technologies used for its implementation. So, in this project few libraries related to Deep learning, String handling, URL handling, Natural Language processing were used. Those are namely:

• TensorFlow

• String

• Requests

• NumPy

• Keras

• Tokenizer

• to_categorical

• pad_sequnces

• load_model

• Matplotlib

In-depth information regarding the libraries/modules mentioned above is discussed in the upcoming sections.

TensorFlow: This is the free machine learning library which is developed by senior developers and engineers working in the core team under the Google's organization for machine learning intelligence which carries experiments in field conducting machine learning and deep neural network research. This is the Machine Learning library for computation operations related to numerical works that require data flow graphs. TensorFlow has nodes which represents the numerical operations and terminals which represent the multi-dimensional data arrays called tensors, communicated between them. There is a diagram shown below. In this diagram add is a node and A and B are input tensors and resultant is the C.

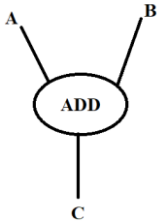

Figure 6.1 TensorFlow Operation

TensorFlow API's (Application Program Interfaces) are of two types. Those two types are namely Low-Level API and High-Level API.

i. **Low Level API**: It helps the developers to have control over programming. The best example for the TensorFlow low level API is TensorFlow core.

ii. **High Level API**: This is an API developed using the TensorFlow core. This makes repeated tasks easy and heavily stable for various users.

String: The String module contains a number of functions to process standard Python strings. This module/library is used when there is a need of handling strings or string variables in python. There are several functions in this library such as split, join, replace, lower, upper etc. Some of the functions provided by String library are shown below in the figure.

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable

Figure 6.2 String Methods

Requests: This the module used in python for sending http requests, this requests gives a response object containing page data like content of page, encoding format of page, status of a URL etc. This library is used when there is a need for handling URL's. By using this library data can be received from a certain URL. For installation we should use the command line code as shown below: "pip

install requests" There are various methods in this library. Some of them are shown in the figure below.

Method	Description
Delete(url,args)	Delete request to specified URL
Get (url,params,args)	Get request to specified URL
Head(url,args)	head request to specified URL
Patch(url,data,args)	patch request to specified URL
Post(url,data,json,args)	Post request to specified URL
Put(url, data, args)	put request to specified URL
Request(method, url, args)	Request to the specified method and URL

Figure 6.3 Request Methods

NumPy: NumPy was developed in 2005 by Travis Oliphant and it is a project which is freely available. This is the python library for working on large arrays and has a function for working in the domain of algebra, and matrices. In python we have lists that acts as arrays but they are slow when a large set of elements are present in the array so here comes the NumPy which tries to facilitate an array object that is fifty times quick than general lists in python as they are stored at one unbroken memory locations where as lists do not, so programs can access and manage them very effectively. And such behaviour is called locality reference. In NumPy the object is called as ndarray which has a lot of functions that help us in working with arrays in Deep Learning projects more effectively. In data science arrays are implemented when speed is required and resources are low.

Keras: Keras was developed by a researcher at google named Francois Chollet. This is the opensource deep learning framework in python. Features of keras are consistent, simple and have extensible API, which have minimal structure easy to bring the result without any frills, it supports multiple platforms and backends. Keras runs upon google's Deep Learning modules like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a module in python implemented for quick numerical computational operations. Both Theano and TensorFlow are one of the most powerful modules in python but some might find it quite hard to conceptualize and, use it for creating neural networks.

Tokenizer: As the word suggests tokenizing means dividing the sentence into a series of tokens or in layman words, we can say that whenever there is a space in a sentence, we add a comma between them so our sentence will get break down into tokens and each word gets a unique value of an integer.

to_categorical: This is library that is imported from "tensorflow.keras.utils" which is used to generate binary class matrix for a given class vector. A NumPy array or a vector that has integers representing

various categories, can be converted into a NumPy array or a binary matrix which has values and has columns equal to the number of categories in the given data.

pad_sequences: pad_sequences is required to assure the developers that all tuples or sequences in a list are of equal length. Pad_sequences have the normal padding value as zero. Generally, at the start of each sequence these zero's are placed to make them equal in length, but we can mention post or pre as we like. As shown in the example below to make all the tuples into same length it added zero's to the tuples with small length to make all of same length.

```
>>> sequence = [[1], [2, 3], [4, 5, 6]]
>>> tf.keras.preprocessing.sequence.pad_sequences(sequence)
array([[0, 0, 1],
       [0, 2, 3],
       [4, 5, 6]], dtype=int32)
```

Figure 6.4 pad_sequences example

load_model: This method is imported from "tensorflow.keras.models" library. This method is used to load the saved models after training. The advantage of using this is the model training is not needed every time, once trained the weights or the model can be saved using this load_model function. This can be used anywhere in the program when required.

Matplotlib: Matplotlib is a library which is used for plotting in python programming language. It is also used for plotting while using NumPy. It offers an object-oriented API for plotting graphs into applications for general use. It was initially released in 2003, that was almost 18 years ago. Since then, it was in continuous development.

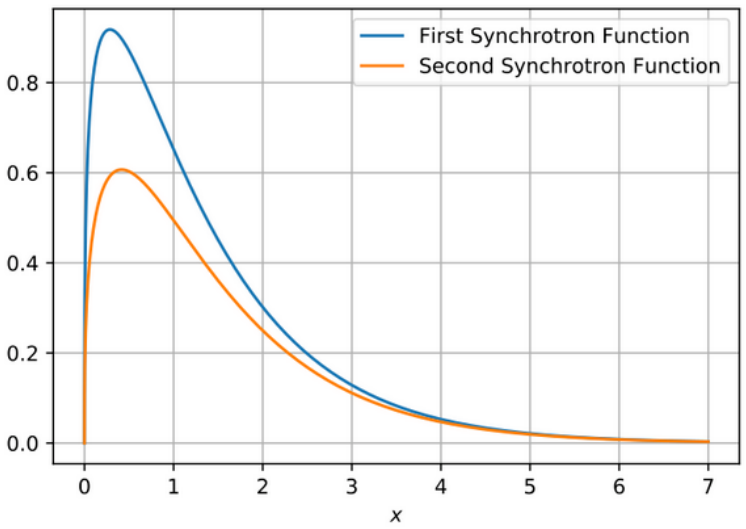

Figure 6.5 General Matplotlib Graph

7. IMPLEMENTATION

This chapter will deal with the code implementation and the testing of the model. There are two sets of code, first is the model development code and next the flask website implementation. The testing section shows how the model made suggestions for the next word based on entered sequence.

7.1 CODING

The code for files named buildingmodel.py, app.py, is shown below as these are the main files.

buildingmodel.py:

```
"""Importing required packages"""
import tensorflow as tf
import string
import requests
import numpy as np
import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
import matplotlib.pyplot as plt

"""Getting data from the site suing Http request"""
response = requests.get('https://www.gutenberg.org/files/98/98-0.txt')
data = response.text.split('\n')
data = data[108:]
data = " ".join(data)

"""Clean function to clean the text file"""
def clean_text(doc):
  tokens = doc.split()
  table = str.maketrans('', '', string.punctuation)
  tokens = [w.translate(table) for w in tokens]
  tokens = [word for word in tokens if word.isalpha()]
  tokens = [word.lower() for word in tokens]
  return tokens

"""Cleaning the data and getting tokens means words"""
tokens = clean_text(data)

"""Setting the train length and getting sequences"""
train_len = 5+1
text_sequences = []
for i in range(train_len,len(tokens)):
    seq = tokens[i-train_len:i]
    text_sequences.append(seq)

sequences = {}
count = 1
for i in range(len(tokens)):
    if tokens[i] not in sequences:
        sequences[tokens[i]] = count
        count += 1
```

```
45
46  """Now we actually use the tokenizer from nltk library to convert the
47  words or tokens in numeric values"""
48  tokenizer = Tokenizer()
49  tokenizer.fit_on_texts(text_sequences)
50  sequences = tokenizer.texts_to_sequences(text_sequences)
51
52  """Now we get the vocabulary size and set train inputs and targets"""
53  vocabulary_size = len(tokenizer.word_counts)+1
54
55  n_sequences = np.empty([len(sequences),train_len], dtype='int32')
56  for i in range(len(sequences)):
57      n_sequences[i] = sequences[i]
58
59  train_inputs = n_sequences[:,:-1]
60  train_targets = n_sequences[:,-1]
61  train_targets = to_categorical(train_targets, num_classes=vocabulary_size)
62  seq_len = train_inputs.shape[1]
63
64  """Functions to calculate recall, precision, f1-score"""
65  from keras import backend as K
66
67  def recall_m(train_inputs, train_targets):
68      true_positives = K.sum(K.round(K.clip(train_inputs * train_targets, 0, 1)))
69      possible_positives = K.sum(K.round(K.clip(train_inputs, 0, 1)))
70      recall = true_positives / (possible_positives + K.epsilon())
71      return recall
72
73  def precision_m(train_inputs, train_targets):
74      true_positives = K.sum(K.round(K.clip(train_inputs * train_targets, 0, 1)))
75      predicted_positives = K.sum(K.round(K.clip(train_targets, 0, 1)))
76      precision = true_positives / (predicted_positives + K.epsilon())
77      return precision
78
79  def f1_m(train_inputs, train_targets):
80      precision = precision_m(train_inputs, train_targets)
81      recall = recall_m(train_inputs, train_targets)
82      return 2*((precision*recall)/(precision+recall+K.epsilon()))
83
84  """Now we build our prediction model"""
85  model = Sequential()
86  model.add(Embedding(vocabulary_size, 50, input_length=seq_len))
87  model.add(LSTM(100, return_sequences=True))
88  model.add(LSTM(100))
89  model.add(Dense(100, activation='relu'))
90  model.add(Dense(vocabulary_size, activation='softmax'))
91  model.summary()
92  model.compile(loss='categorical_crossentropy', optimizer='adam',
93  metrics=['accuracy',f1_m,precision_m, recall_m])
94  history=model.fit(train_inputs,train_targets,batch_size=256,epochs=500,verbose=1).history
95  model.save("nextwordmodel1.h5")
96
97  """Code to plot the graphs for accuracy, f1-score, precision, recall"""
98  plt.plot(history['accuracy'])
99  plt.plot(history['f1_m'])
100 plt.plot(history['precision_m'])
101 plt.plot(history['recall_m'])
102 plt.title('Performance Measure')
103 plt.ylabel('model metrics')
104 plt.xlabel('epoch')
105 plt.legend(['accuracy', 'f1-score','precision','recall'],loc='lower right')
```

Figure 7.1 Code for Building the Model

app.py:

```python
from flask import Flask, render_template, request
import model


app = Flask(__name__)


@app.route('/', methods=['GET'])
def index():
    return render_template("index.html")

@app.route('/senddata', methods=['POST'])
def predictdata():
    predictions=[]
    input_text=request.form['textinp']
    input_text=input_text.strip().lower()
    encoded_text = model.tokenizer.texts_to_sequences([input_text])[0]
    pad_encoded = model.pad_sequences([encoded_text], maxlen=model.seq_len, truncating='pre')
    for i in (model.nextwmodel.predict(pad_encoded)[0]).argsort()[-3:][::-1]:
        pred_words = model.tokenizer.index_word[i]
        predictions.append(pred_words)
    return render_template("index.html", textent="The sequence entered is: {}".format(input_text),
    predictions=predictions)

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 7.2 Code for Flask App

7.2 TESTING

In this section the results which are obtained after the successful development and training of the model are mentioned here. Now we can enter some meaningful text to get the next word suggestions for the given sequence. Three test cases will be considered here. The first case predictions will be made on the text from the trained file. The other two cases will be the text sequences from general English statements or sentences. For all the three cases a maximum of three possible next word suggestions will be given by the model.

```
It was the age of
[8, 11, 1, 674, 3] [[  8  11   1 674   3]]
Next word suggestion: foolishness
Next word suggestion: wisdom
Next word suggestion: which
```

Figure 7.3 Next Word Suggestion for text from trained data (test-1)

```
what are you trying to
[58, 56, 15, 1049, 4] [[  58   56   15 1049    4]]
Next word suggestion: be
Next word suggestion: have
Next word suggestion: see
```

Figure 7.4 Next Word Suggestion for a random text (test-2)

```
My friend is dealing with
[30, 228, 23, 13] [[  0  30 228  23  13]]
Next word suggestion: your
Next word suggestion: this
Next word suggestion: my
```

Figure 7.5 Next Word Suggestion for a random text (test-3)

Test cases	Input text	Expected suggestions	Actual output	Status
Test-1	It was the age of	wisdom, foolishness, darkness, science, technology,	wisdom, foolishness, which	PASS
Test-2	What are you trying to	be, get, see, have, look, do	be, have, see	PASS
Test-3	My friend is dealing with	his, your, my, this, other, their	your, this, my	PASS

Figure 7.6 Testing Results

8. RESULTS & DISCUSSIONS

This chapter deals with the results obtained after training and testing the model. Now after the training is done. The model was tested on different cases as discussed in the above chapter. Now for a user to use the model, it has to be deployed into a website or and application. Now this deployment is done using the flask framework. Flask framework is small python framework that allows people to build websites or small web apps. The file construct for this flask implementation is already explained in chapter five. We created a small website like thing to deploy our deep learning model for giving next word suggestions. The image of the html page is shown below.

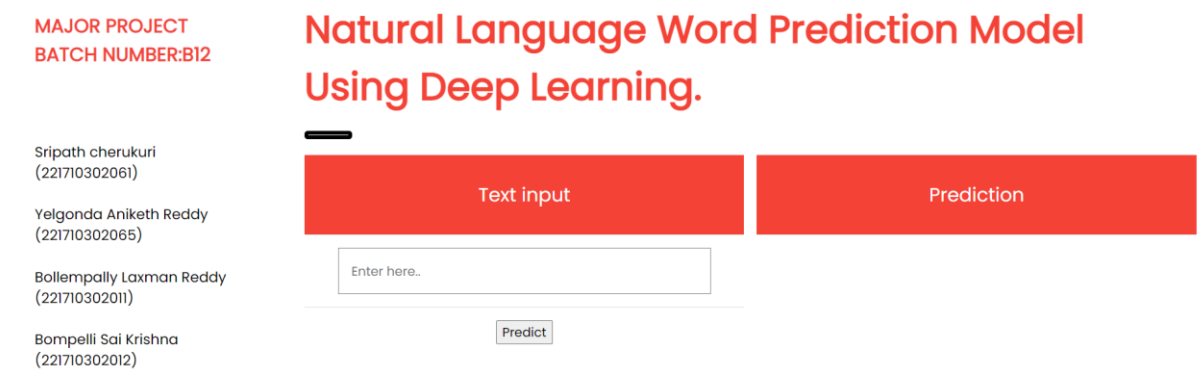

Figure 8.1 Localhost

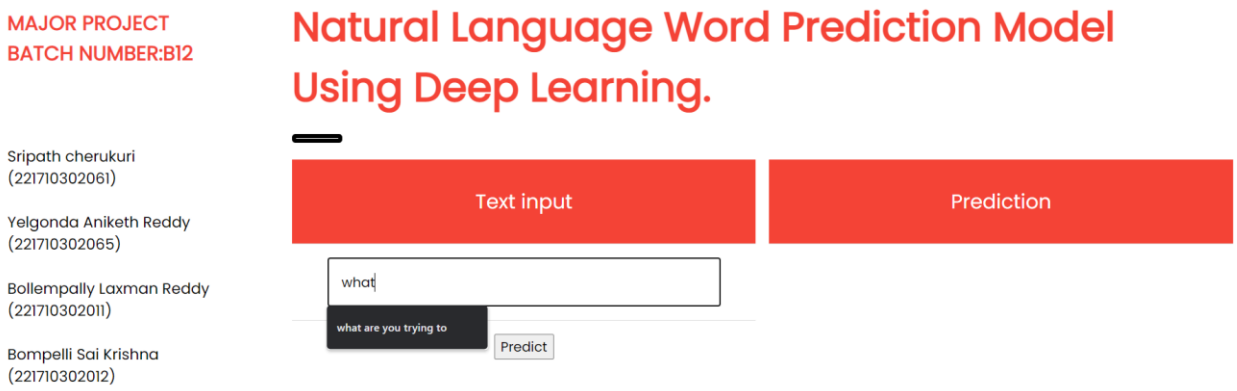

Figure 8.2 Text Entry in HTML.

Now coming to the performance of the model which we created, it produced an accuracy of 91% in the first but with a little change, and more training we were able to achieve and accuracy of 94.3%. Coming to the loss of the model when it was first built, we got a loss of 0.32% and for the next try we got 0.19% which is quite good improvement over the previous one. The graphs for the accuracy and loss are shown below.

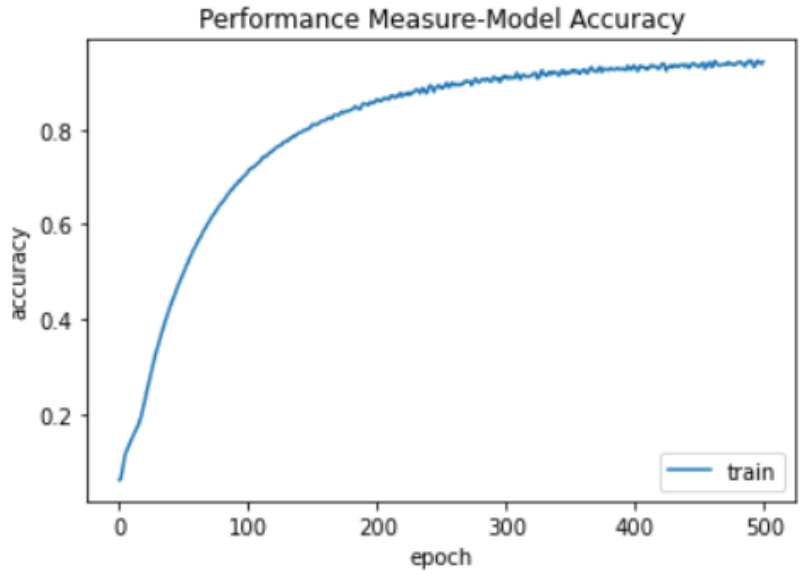

Figure 8.3 Model Accuracy

As we can see that the graph is consistent in the ninety to hundred range. The accuracy was almost constant after reaching ninety four percent and didn't change much.

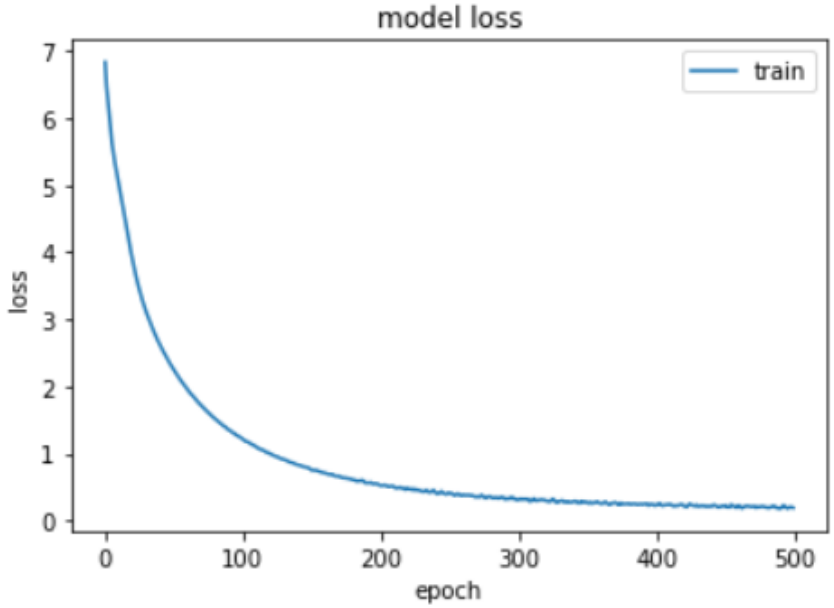

Figure 8.4 Model Loss

The model's performance was also measured on other metrics such as F1-score, Precision score and, Recall value. A Figure is shown below of the model's metrics mentioned above.

Accuracy Measure	Value Obtained
F1 Score	0.94
Precision Score	0.95
Recall Value	0.93

Figure 8.5 Metrics

Now the graph for the f1-score, precision and, recall is shown below which we obtained while training the model.

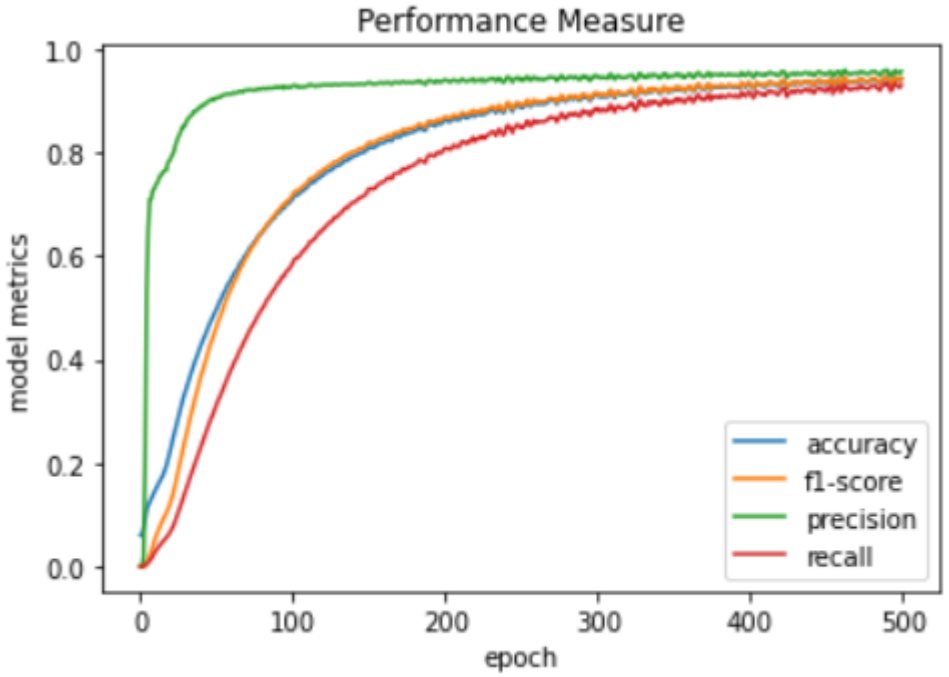

Figure 8.6 Metrics graph

F1 score, the recall value and the precision are almost same while coming to the end of the training. The values are almost similar to each other. The model which we developed shows good performance measures as discussed above. Now we compare our model with other models just for research purpose. Even though our model shows good performance values it might not predict or suggest adequate words every time 34 considering the number of words it's trained on. The comparison table is shown below basing on the accuracy measure.

Model	Accuracy
SNOW model	59.1%
3-grams	63%
RNN	88.2%
Our model	94.3%

Figure 8.7 Comparison with Other Models

Now let use see some of the results regarding the next word suggestions which our model suggested, which is implemented in flask framework.

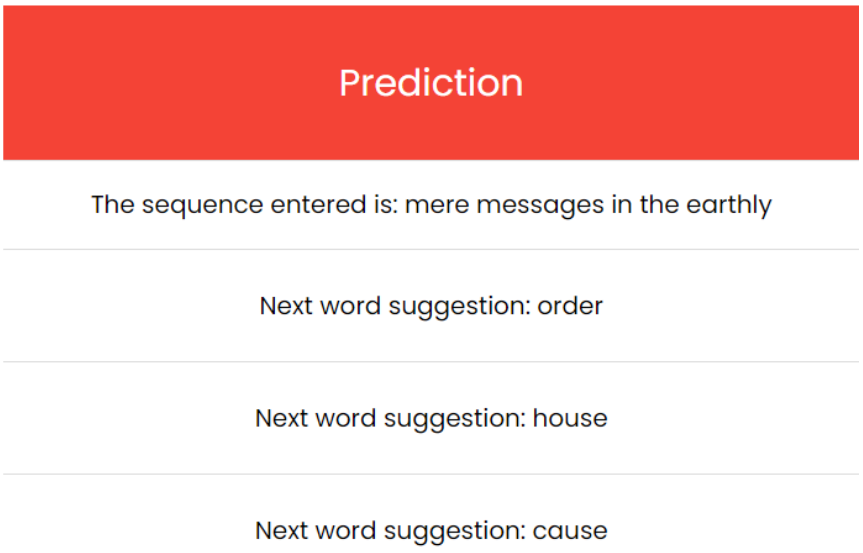

Figure 8.8 Output-1

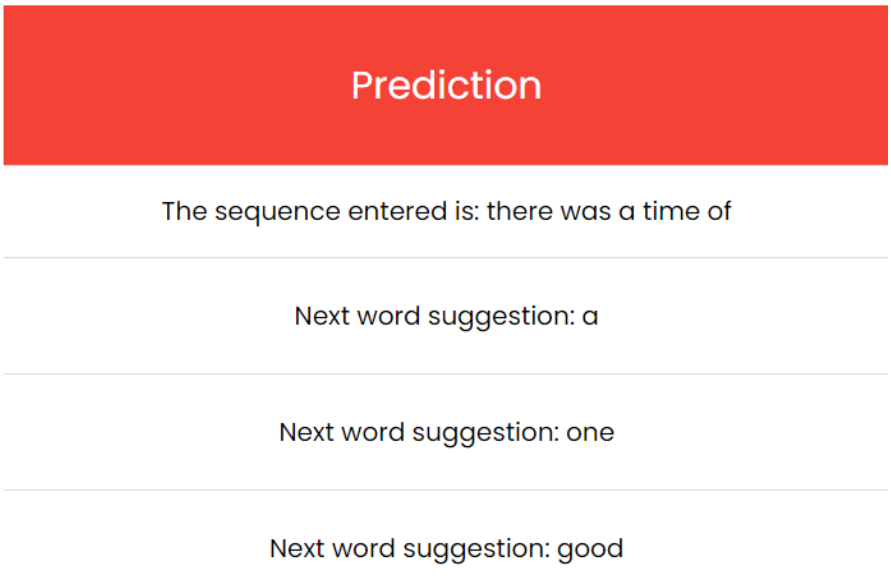

Figure 8.9 Ouput-2

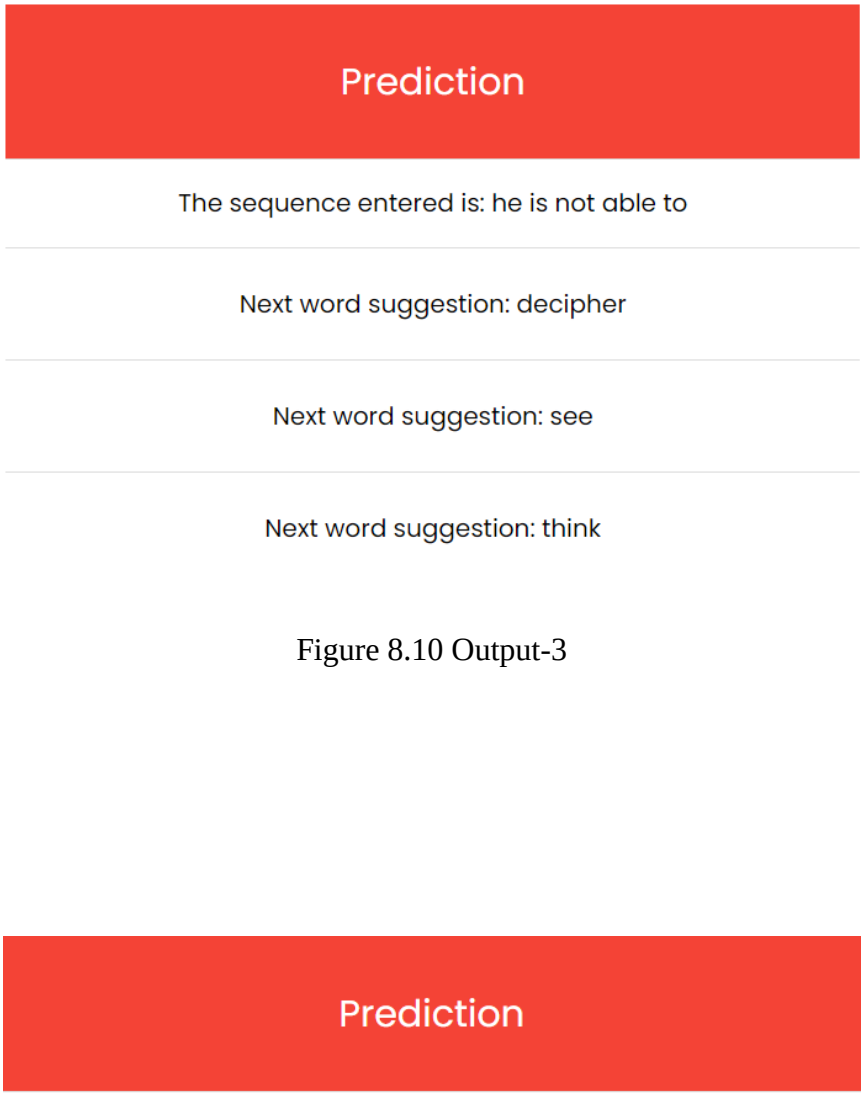

Figure 8.10 Output-3

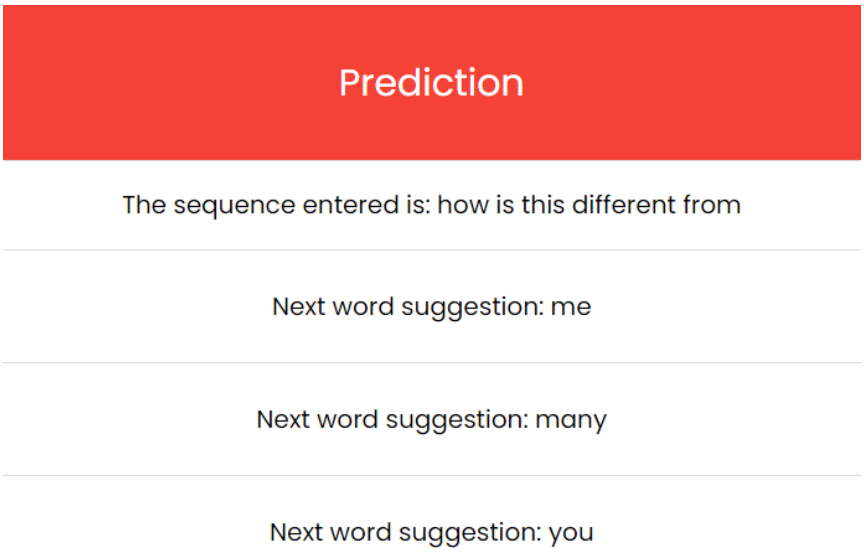

Figure 8.11 Output-4

9. CONCLUSION & FUTURE SCOPE

There is more emphasis on text data in recent years. In the past people made calls and wrote letters to people at the other end. The digitalization of data and availability of information to almost everyone in the world has been increasing widely, and also with the introduction of messenger apps like WhatsApp, Telegram people these days chat more than they speak. So, to communicate faster they need a quick way to express themselves. Software's like android use keyboards with next word prediction feature which allows them to communicate quickly.

For the task of predicting next word based on the given sentence or text we proposed a LSTM model with an embedding layer. As discussed earlier this model was able to perform better even though it is not trained oh massive text data. Considering the size of our data our model performed good and also gives the next word suggestions adequately for almost all given text sequences.

This research helped us to gain knowledge regarding the language models. But it is never complete, there is always a better way to do things and improve the performance of the models as technology evolves day by day. This project can be extended such that this next word suggestion can be extended to speech also, like predicting the next word while a person is speaking. Moreover, the model can be implemented for other application also, that is to make the model suggest next words based on a person's character. There is always more to learn and research on. But this project provided us many useful insights about Deep learning and language models.

10. REFERENCES

[1] Y. K. Xing and S. P. Ma, ‘‘A survey on statistical language models,’’ Comput. Sci., vol. 30, no. 9, pp. 22–26, 2003.

[2] O. Irsoy and C. Claire, ‘‘Opinion mining with deep recurrent neural networks,’’ in Proc. Empirical Methods Natural Lang. Process., 2014, pp. 720–728.

[3] Y. Bengio, P. Simard, and P. Frasconi, ‘‘Learning long-term dependencies with gradient descent is difficult,’’ IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157–166, Mar. 1994.

[4] S. Hochreiter and J. Schmidhuber, ‘‘Long short-term memory,’’ Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[5] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, ‘‘Empirical evaluation of gated recurrent neural networks on sequence modeling,’’ Dec. 2014, arXiv:1412.3555. [Online]. Available: http://arxiv.org/abs/1412.3555

[6] L. Wang, L. Liu, L. Niu, F. Y. Hu, and T. Peng, ‘‘Relation extraction method based on relation trigger words and single-layer GRU model,’’ J. Jilin Univ., Sci. Ed., vol. 58, no. 1, pp. 95–103, 2020.

[7] W. Wang, Y. Sun, Q. Qi, and X. Meng, ‘‘Text sentiment classification model based on BiGRU-attention neural network,’’ Appl. Res. Comput., pp. 3558–3564, 2019, doi: 10.19734/j.issn.1001-3695.2018.07.0413.

[8] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, ‘‘Minimal gated unit for recurrent neural networks,’’ Int. J. Autom. Comput., vol. 13, no. 3, pp. 226–234, Jun. 2016.

[9] A. Dong, Z. Du, and Z. Yan, ‘‘Round trip time prediction using recurrent neural networks with minimal gated unit,’’ IEEE Commun. Lett., vol. 23, no. 4, pp. 584–587, Apr. 2019.

[10] J. X. Liu and S. C. Chen, ‘‘Non-stationary multivariate time series prediction with MIX gated unit,’’ J. Comput. Res. Develop., vol. 56, no. 8, pp. 1642–1651, 2019.

[11] https://github.com/Bharath-K3/Next-Word-Prediction-with-NLP-and-Deep-Learning

[12] https://en.wikipedia.org/wiki/TensorFlow

[13] https://flask.palletsprojects.com/en/1.1.x/

[14] https://en.wikipedia.org/wiki/Deep_learning