WORD DIFFICULTY LEVEL PREDICTION SYSTEM USING DEEP LEARNING APPROACH

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

BY

Abhinav Panse (EN16CS301008)
Shivam Parihar (EN16CS301245)

Under the Guidance of

Prof. (Dr.) Ruchi Patel

Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331

May, 2020

WORD DIFFICULTY LEVEL PREDICTION SYSTEM USING DEEP LEARNING APPROACH

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

BY

Abhinav Panse (EN16CS301008)
Shivam Parihar (EN16CS301245)

Under the Guidance of

Prof. (Dr.) Ruchi Patel

Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331

May, 2020

Report Approval

The project work **"Word Difficulty Level Prediction System (Using Deep Learning Approach)"** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the "Project Report" only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation

Affiliation

External Examiner

Name:

Designation

Affiliation

Declaration

We hereby declare that the project entitled **"Word Difficulty Level Prediction System (Using Deep Learning Approach)"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Prof. (Dr.) Ruchi Patel, Assistant Professor, Department of Computer Science and Technology,** Faculty of Engineering, Medi-Caps University, Indore is an authentic work.

Further, we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Date: _____

ABHINAV PANSE (EN16CS301008)

SHIVAM PARIHAR (EN16CS301245)

Certificate

I, **Prof. (Dr.) Ruchi Patel** certify that the project entitled **"Word Difficulty Level Prediction System (Using Deep Learning Approach)"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Abhinav Panse (EN16CS301008)** and **Shivam Parihar (EN16CS301245)** is the record carried out by them under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

Prof. (Dr.) Ruchi Patel	Dr. Suresh Jain
Assistant Professor	Head of the Department
Computer Science and Engineering	Computer Science and Engineering
Medi-Caps University, Indore	Medi-Caps University, Indore

Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Sunil K. Somani,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department (**Dr.) Suresh Jain** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **Internal Guide, Prof. (Dr.) Ruchi Patel,** Assistant Professor, Department of Computer Science and Engineering, Medi-Caps University without whose continuous help and support, this project would ever have reached to the completion.

I would also like to thank to my team who extended their kind support and help towards the completion of this project.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

Abhinav Panse (EN16CS301008)
Shivam Parihar (EN16CS301245)

B.Tech. IV Year
Department of Computer Science & Engineering
Faculty of Engineering
Medi-Caps University, Indore

Abstract

This study is aimed to examine the difficulty of word is the measure of how hard the word is for the particular user. Research on this topic has a long history. Our study is based on the LSTM based approach for measuring the difficulty level of word. Long Short-Term Memory (LSTM) networks are recurrent neural network mainly used to solve sequence prediction problem. The model basically uses LSTM Cell for training the dataset, it is a type of supervised learning approach, apart from this, model also uses phonetics which is the study of production, transmitting and classification of speech sound.

A readability test is a way to measure difficulty level of word, it gives a prediction as to how difficult readers will find a particular word. The Common Core State Standards define word complexity as two closely related component out of which first one is qualitative dimensions of word complexity which refer to those aspects of word complexity only measurable by an attentive human reader, such as depth of meaning or purpose; structure; language conventionality clarity and knowledge demands. And the second is quantitative dimensions of word complexity: The terms quantitative dimensions and quantitative factors refer to those aspects of word complexity, such as length of the word or frequency of occurrence of that word, sentence length, that are difficult for a human reader to evaluate efficiently and therefore today computer software are used for the evaluation.

The pronunciation of a word is governed by the phonetical translation so using phonetics in determining the difficulty of word gives better prediction then using the actual words or character set which was used in previous models. This model is divided into two modules: phonetic translation of 1800 word set and training the model using LSTM which further uses five layers for training and prediction: Embedding layer, Bidirectional Layer, dense (ReLU activation function) Layer, Dropout layer, dense layer (SoftMax activation function). Using the dataset and the trained model the accuracy obtained is 97.5%.

Keywords: Long Short-term Memory Network (LSTM), Embedding layer, Bidirectional Layer, dense (ReLU activation function) Layer, Dropout layer, dense layer (sigmoid function), Document difficulty, Word difficulty.

Table of Contents

		Page No.
	Report Approval	ii
	Declaration	iii
	Certificate	iv
	Acknowledgement	v
	Abstract	vi
	Table of Contents	vii
	List of figures	ix
	List of tables	x
Chapter 1	Introduction to Word Difficulty Predictor	1
	1.1 Introduction	1
	1.2 Literature Review	1
	1.3 Objective	3
	1.3.1 Word Difficulty, Phonetics and Pronunciations	4
	1.3.2 Word Difficulty with Neural Networks	5
	1.4 Significance	5
Chapter 2	Project Analysis	
	2.1 Information Gathering	6
	2.1.1 Neural Network	6
	2.1.2 Recurrent Neural Network	8
	2.1.3 Proposed Word Difficulty Architecture	12
	2.2 System Feasibility	13
	2.2.1 Economic Feasibility	14
	2.2.2 Technical Feasibility	14
	2.2.3 Behavioral Feasibility	14
	2.3 Experimental Setup	14
	2.3.1 Hardware	14
	2.3.2 Software	15
Chapter 3	System Analysis	
	3.1 Information Flow Diagram	16
	3.1.1 Activity Flow Diagram	17
Chapter 4	Design	
	4.1 Architectural Design	18
	4.2 Data Design	20
	4.2.1 Data Structures in the Model	22
	4.3 Phonetics	22
	4.4 Dataset	23
	4.5 Proposed Word Level Difficulty Prediction System	24
	4.5.1 Embedding Layer	24
	4.5.2 Bidirectional Layer	24
	4.5.3 Dense (ReLU) Layer	25
	4.5.4 Dropout Layer	25

	4.5.5 Output Layer (Softmax)	25
	4.6 Training	25
	4.7 Result of Training and Further Discussion	26
	4.8 Further Discussions	27
Chapter 5	Testing	
	5.1 Testing Objectives	30
	5.2 Testing Principle	30
	5.3 Testing Methods Used	30
	5.4 Test Cases	32
	5.5 Sample Test Data and Result	33
Chapter 6	Limitations	34
Chapter 7	Future Scope	35
Chapter 8	Conclusion	36
Chapter 9	Bibliography and References	37

List of Figures

Serial No.	Title	Page No.
Fig. 2.1	Neural Network Architecture	7
Fig. 2.2	Embedding Layer	10
Fig. 2.3	ReLU Activation Function	11
Fig. 2.4	Word Difficulty Prediction Architecture	13
Fig. 3.1	Activity Diagram of Word Difficulty Prediction	17
Fig. 4.1	Word Difficulty Prediction Model	18
Fig. 4.2	Word Difficulty Prediction Architecture	19
Fig. 4.3	Preprocessing Code	21
Fig. 4.4	Phonetic Symbols	22
Fig. 4.5	Phonetic Translation of few words used in Training Model	23
Fig. 4.6	Graph Representing Accuracy with respect to Dataset Split	27

List of Tables

Serial No.	Title	Page No.
Table 2.1	GPU Characteristics	15
Table 4.1	Summary of Training and Testing Dataset	23
Table 4.2	Comparison of performance	26
Table 4.3	Comparison of performance using confusion matrix	28
Table 4.4	Final Confusion Matrix	28
Table 4.5	Precision and Recall Table	29
Table 5.1	Test Data and Result	33

Chapter-1

Introduction

1.1 Introduction

Readability gives a prediction as to how hard readers will discover a unique word. The Common Core State Standards outline phrase complexity as two intently related thing out of which first one is qualitative dimensions of word complexity which refer to these aspects of word complexity only measurable with the aid of an attentive human reader, such as depth of meaning or purpose; structure; language conventionality readability and knowledge demands. And the second is quantitative dimensions of phrase complexity: The terms quantitative dimensions and quantitative elements refer to these factors of phrase complexity, such as size of the word or frequency of occurrence of that word, sentence length, that are hard for a human reader to evaluate correctly and therefore nowadays laptop software program are used for the evaluation. This model makes use of phonetics and LSTM based deep mastering method to discover the complexity or problem degree of word.

This study is aimed to examine the difficulty of word is the measure of how hard the word is for the particular user. Research on this topic has a long history. Our study is based on the LSTM based approach for measuring the difficulty level of word. Long Short-Term Memory (LSTM) networks are recurrent neural network mainly used to solve sequence prediction problem. The model basically uses LSTM Cell for training the dataset, it is a type of supervised learning approach, apart from this, model also uses phonetics which is the study of production, transmitting and classification of speech sound.

1.2 Literature Review

This field has a long history of experiments and tests which are conducted with new and emerging technologies and techniques. Initial researches date back to 1993 when Ala P. Rudell[1] proposed an experiment using a set of 870 five-letter word. The words were selected from three word-frequency categories that are 1, 5-10, and 50-3, 562/million and it was estimated that Reaction time for valid word identification if we keep the frequency constant is much larger [2].

Also, [3] Hunter M. Breland in March, 1996 published a paper in the experiment of estimating correlation between word difficulty and word frequency concluded that the relation between the word difficulty ranks and word frequency indices shows that word frequencies for all four text sample that are used for the experiment are related with word difficulty and the results also shows that the difficulty of few words is not estimated accurately by word frequency.

[4] Pedro Curto, Nuno Mamede and Jorge Baptista in 2015 proposed an experiment in which the system uses existing Natural Language Processing (NLP) to extract linguistic features from texts, the system extracts near about 52 features, grouped in 7 groups chunks and phrases, parts-of-speech, syllables, words, averages and frequencies, and other features. This paper presents an automatic classifier based on a variety of linguistic features.

In the year (1977) [5] Kibby, M. W. note on relationship of word frequency and word difficulty. Reports that correlation of word difficulty as determined by the order of words on 5 standardized vocabulary tests with word frequency. The 5 tests or subtests used were as follows: Peabody Picture Vocabulary Test, Forms A and B; WISC; WISC-R; and Stanford-Binet Intelligence Scale. Keith Rayner and Susan A.

Duffy conducted two experiments in order to investigate the effect of lexical complexity on word's processing time. Subjects read sentences, each containing a target word, meanwhile the eye movement of the reader was noted. The results of this experiment suggest that verb complexity has no effect on lexical access time, but the word frequency and the presence of two highly similar meanings may affect lexical access and post access integration. [6] Jose M. Tamayo in the year 1987 conducted an experiment to verify the effect of word frequency in determining the difficulty level in vocabulary test construction and translation, three 24-word lists, two Spanish and one English, were used. The first list of Spanish words matched the meaning and frequency of use of the corresponding English words, whereas the second list of words matched the meaning but not the frequency of use. The result concluded that the frequency of use can be used as a valid measure to determine the word level difficulty in vocabulary test construction and translation.

In 1976, [7][8][9] Ronald P. Carver use the newly developed Rauding Scale to estimate the passage difficulty Ronald P. Carver conducted two experiment in his first experiment he studied the relationship between Passage difficulty and word length to be linear from Grade 1

to Grade 17 difficulty levels Average word length was measured two ways one is character spaces per word and the other one is letters per word.

Whereas in second experiment the reading rate of college students was found to decrease from about 315 to 200 words per min. as difficulty increased from about Grade 2 to about Grade 17. These datasets were completely contradictory of the current theory relating redundancy to reading rate and eye movements and it was also concluded that the Rauding Scale most reliable and valid technique used at that time.

In the year 2017 [10] Xiabin Chen and Detmar Meurers conducted an experiment in which use of lexical level attribute is done in order to predict the text level redability. The conclusion of this experiment advances the understanding of the relation between the word frequency and text readability.

Another experiment in the year 1957 was conducted by [11] L.L. Haseley in which cue-value of the word is determined and its relationship with the frequency of occurrence of the word is taken as a parameter to determine the difficulty level classification. [12] W. N. Francis, H. Kučera, and A. W. Mackie conducted an experiment in which word frequency is used as a major parameter in predicting the word level difficulty.

These are some of the studies which provided us with existing techniques and experience in this domain. These studies greatly helped us in achieving our objective of improving upon existing techniques and provided us with a proper guiding way to improve upon the performance of word difficulty prediction methods.

1.3 Objective

The objective of this project is to predict how difficult the word is to pronounce in a particular accent. For this we will train the machine learning model and make it such that it adapts to a particular accent. The training will be done by the native speaker of that accent. After training the model, whenever a new word will be introduced for the labeling i.e. to check if the word is easy, medium or hard then, the prediction will be done.

Through this, various users can use it in different domains, like in teaching, reading, training chat bots and vocal bots. It can also be used in Natural Language Processing and Natural Language Generation.

The models will be different for each accent and can be trained from people of different countries or states. This way we can have customized solution for each accent. And to further increase its accuracy we can crowd source the training process.

1.3.1 Word Difficulty, Phonetics And Pronunciations

The word difficulty is simply the struggle or trouble faced by the individual to pronounce the word. And this may be different for every individual but it is largely same for the people with same demographics.

Phonetics is a branch of linguistics that studies the sounds of human speech, or in the case of sign languages, the equivalent aspects of sign. Phoneticians—linguists who specialize in phonetics—study the physical properties of speech. The field of phonetics is traditionally divided into three sub disciplines based on the research questions involved such as how humans plan and execute movements to produce speech (articulatory phonetics), how different movements affect the properties of the resulting sound (acoustic phonetics), or how humans convert sound waves to linguistic information (auditory phonetics). Traditionally, the minimal linguistic unit of phonetics is the phone—a speech sound in a language—which differs from the phonological unit of phoneme; the phoneme is an abstract categorization of phones.

Pronunciation is the way in which a word or a language is spoken. This may refer to generally agreed-upon sequences of sounds used in speaking a given word or language in a specific dialect ("correct pronunciation"), or simply the way a particular individual speaks a word or language. A word can be spoken in different ways by various individuals or groups, depending on many factors, such as: the duration of the cultural exposure of their childhood, the location of their current residence, speech or voice disorders, their ethnic group, their social class, or their education.

Syllables are counted as units of sound (phones) that they use in their language. The branch of linguistics which studies these units of sound is phonetics. Phones which play the same role are grouped together into classes called phonemes; the study of these is phonemics or phonematic or phonology. Phones as components of articulation are usually described using the International Phonetic Alphabet.

1.3.2 Word Difficulty With Neural Networks

In this project we have applied neural networks for text. You can think of text as a sequence of characters, words or anything else. But we have used the phonetic transcription of this text. We will continue to think of text as a sequence of words or tokens with their phones.

We have every word and forever distinct word that we have in our dataset, and then we have a feature column. And we actually effectively vectorize each word with one-hot-encoded vector

that is a huge vector of zeros that has only one non-zero value which is in the column corresponding to that particular word. So, you split your text into phonetic transcripts. For each token you take an embedding vector and you just sum them up. So, that is a baseline model and it can actually work pretty well. Another approach which is a little bit better is to use 1D convolutions. And this way you train neural network end to end. So, we have an input sequence and you have a result that you want to predict, and you use back propagation and train all those convolutions, to train the specific features that this neural network needs to classify your word.

1.4 Significance

This project has its significance in variety of fields. The following are some of the foreseeable fields in which this project can be used and provide good value.

- Can be used to label thousands of words for its difficulty level. Many apps and software can benefit from this project. They would not have to use expensive human resource.
- Can be used to train vocal robots, so that they talk in regional accent.
- Can be helpful in teaching and language field.
- Can be used to optimize text according to ease of readers.

Chapter-2

Project Analysis

2.1 Information Gathering

Requirements analysis is the tasks that an analyst performs to structure and organize requirements, specify and model requirements and designs, validate and verify information, identify solution options that meet business needs, and estimate the potential value that could be realized for a solution option.

The most important phase is to analyze what all are the requirements of the project. During analyzing I found research on existing the existing topic has a long history. Different tools, techniques and methods are discussed in the process of determining the complexity or difficulty level of word. This system is proposed for the same and is based on the concept of LSTM and phonetics. Phonetics is the branch of linguistics which deals with the study of production, transmitting and classification of speech sound and its representation in written symbols.

2.1.1 Neural Network

An artificial neutral network (ANN) is a system that is based on the biological neural network, such as the brain. The brain has approximately 100 billion neurons, which communicate through electro-chemical signals. The neurons are connected through junctions called synapses. Each neuron receives thousands of connections with other neurons, constantly receiving incoming signals to reach the cell body. If the resulting sum of the signals surpasses a certain threshold, a response is sent through the axon. The ANN attempts to recreate the computational mirror of the biological neural network, although it is not comparable since the number and complexity of neurons and the used in a biological neural network is many times more than those in an artificial neutral network.

An ANN is comprised of a network of artificial neurons (also known as "nodes"). These nodes are connected to each other, and the strength of their connections to one another is assigned a value based on their strength: inhibition (maximum being -1.0) or excitation (maximum being +1.0). If the value of the connection is high, then it indicates that there is a strong connection. Within each node's design, a transfer function is built in. There are three types of neurons in an ANN, input nodes, hidden nodes, and output nodes.

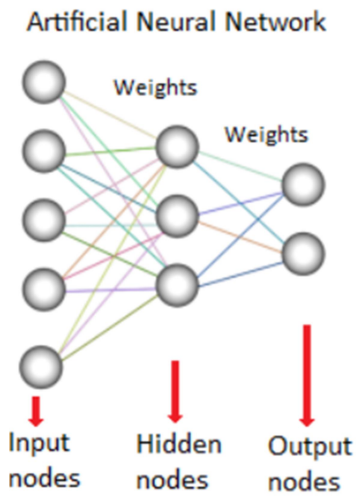


Fig. 2.1 Neural Network Architecture

There are different types of neural networks, but they are generally classified into feed-forward and feed-back networks.

2.1.1.1 Feed-Forward Neural Network

A Feed-Forward network is a non-recurrent network which contains inputs, outputs, and hidden layers; the signals can only travel in one direction. Input data is passed onto a layer of processing elements where it performs calculations. Each processing element makes its computation based upon a weighted sum of its inputs. The new calculated values then become the new input values that feed the next layer. This process continues until it has gone through all the layers and determines the output. A threshold transfer function is sometimes used to quantify the output of a neuron in the output layer. Feed-forward networks include Perceptron (linear and non-linear) and Radial Basis Function networks. Feed-forward networks are often used in data mining.

2.1.1.2 Feed-Back Neural Network

A Feed-Back network has feed-back paths meaning they can have signals traveling in both directions using loops. All possible connections between neurons are allowed. Since loops are present in this type of network, it becomes a non-linear dynamic system which changes continuously until it reaches a state of equilibrium. Feed-back networks are often used in associative memories and optimization problems where the network looks for the best arrangement of interconnected factors.

2.1.1.3 Backpropagation Neural Network

Backpropagation is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning. Essentially, backpropagation is an algorithm used to calculate derivatives quickly.

Artificial neural networks use backpropagation as a learning algorithm to compute a gradient descent with respect to weights. Desired outputs are compared to achieved system outputs, and then the systems are tuned by adjusting connection weights to narrow the difference between the two as much as possible. The algorithm gets its name because the weights are updated backwards, from output towards input.

The difficulty of understanding exactly how changing weights and biases affects the overall behavior of an artificial neural network was one factor that held back wider application of neural network applications, arguably until the early 2000s when computers provided the necessary insight. Today, backpropagation algorithms have practical applications in many areas of artificial intelligence (AI), including optical character recognition (OCR), natural language processing (NLP) and image processing.

Because backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient, it is usually classified as a type of supervised machine learning. Along with classifiers such as Naïve Bayesian filters and decision trees, the backpropagation algorithm has emerged as an important part of machine learning applications that involve predictive analytics.

2.1.2 Recurrent Neural Network

Recurrent neural networks, of which LSTMs (long short-term memory) units are the most powerful and well known subset, are a type of artificial neural network designed to recognize patterns in sequences of data, such as numerical times series data emanating from sensors, stock markets and government agencies (but also including text, genomes, handwriting and the spoken word). What differentiates RNNs and LSTMs from other neural networks is that they take time and sequence into account, they have a temporal dimension.

Research shows them to be one of the most powerful and useful types of neural network, although recently they have been surpassed in language tasks by the attention mechanism, transformers and memory networks. RNNs are applicable even to images, which can be decomposed into a series of patches and treated as a sequence. Since recurrent networks possess a certain type of memory, and memory is also part of the human condition, we'll make repeated analogies to memory in the brain.

Recurrent networks are distinguished from Feed-Forward networks by that feedback loop connected to their past decisions, ingesting their own outputs moment after moment as input. It is often said that recurrent networks have memory. Adding memory to neural networks has a purpose, firstly there is information in the sequence itself, and secondly recurrent nets use it to perform tasks that Feed-Forward networks can't.

That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called "long-term dependencies", because an event downstream in time depends upon, and is a function of, one or more events that came before. One way to think about RNNs is this: they are a way to share weights over time.

Just as human memory circulates invisibly within a body, affecting our behavior without revealing its full shape, information circulates in the hidden states of recurrent nets. The English language is full of words that describe the feedback loops of memory. When we say a person is haunted by their deeds, for example, we are simply talking about the consequences that past outputs wreak on present time. The French call this "Le passé qui ne passe pas," or "The past that does not pass away."

We'll describe the process of carrying memory forward mathematically:

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

RNN Layers

- **Embedding Layer**

Embedding layer used to compress the input feature space into a smaller one. Imagine that we have 80,000 unique words in a text classification problem and we select to preprocess the text and create a term document matrix. This matrix will be sparse and a sequence of the sequence ['i', 'love', 'you'] is a 80,000-dimensional vector that is all zeros except from 3 elements that correspond to those words. In the case, we pass this matrix as input to the model it will need to calculate the weights of each individual feature (80,000 in total). This approach is memory intensive.

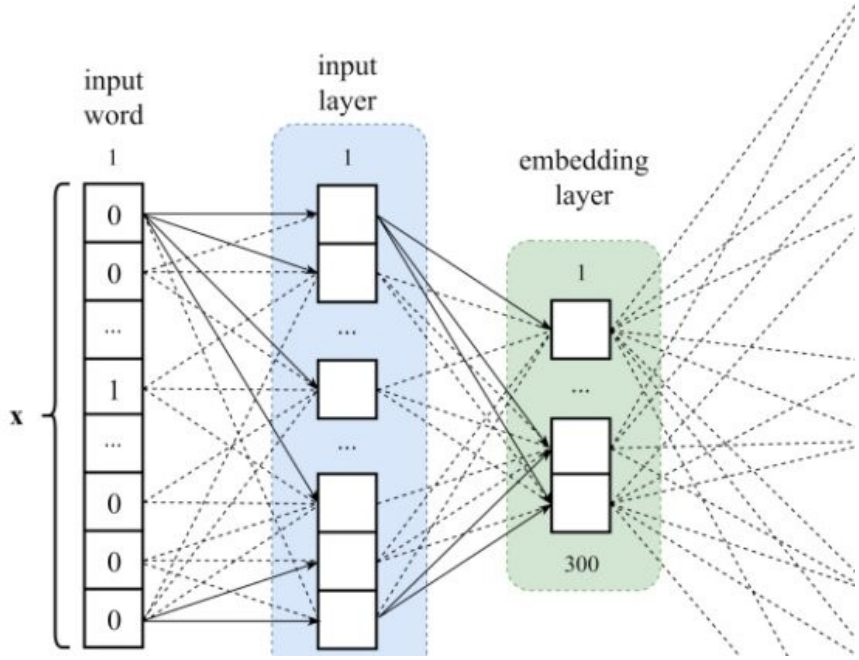


Fig. 2.2 Embedding Layer

- **Bidirectional Layer**

Bidirectional recurrent neural networks (RNN) are really just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step, though there are other options, e.g. summation.

This structure allows the networks to have both backward and forward information about the sequence at every time step. The concept seems easy enough. But when it comes to actually implementing a neural network which utilizes bidirectional structure, confusion arises.

- **Dense (ReLU activation function) Layer**

A dense layer is a Layer in which Each Input Neuron is connected to the output Neuron, like a Simple neural net, the parameters units just tells you the dimensionality of your Output.
The name suggests that layers are fully connected (dense) by the neurons in a network layer.

Each neuron in a layer receives an input from all the neurons present in the previous layer—thus, they're densely connected. In other words, the dense layer is a fully connected layer, meaning all the neurons in a layer are connected to those in the next layer.

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as y = max(0, x). Visually, it looks like the following:

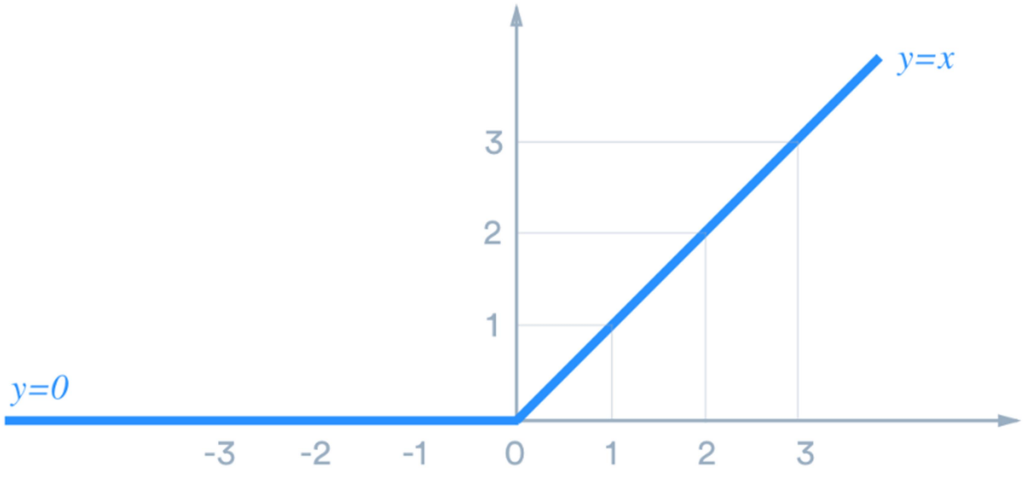


Fig. 2.3 ReLU Activation Function

- **Dropout Layer**

Dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. By "ignoring", I mean these units are not considered during a particular forward or backward pass.

- **Output Layer (Softmax activation function)**

The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model. This is why softmax is sometimes referred to as a multinomial logistic regression.

2.1.3 Proposed Word Difficulty Architecture

The proposed model works in two steps: Phonetic translation of the words and then training the data set using the Long Short-Term Memory, but before, there are five layers through which the dataset is passed which are termed as Embedding layer, Bidirectional Layer, dense (Relu activation function) Layer, Dropout layer, dense layer(sigmoid function). Figure 3.2 shows the architecture of deep learning based proposed Word difficulty prediction system.

The description of each component is mentioned below.

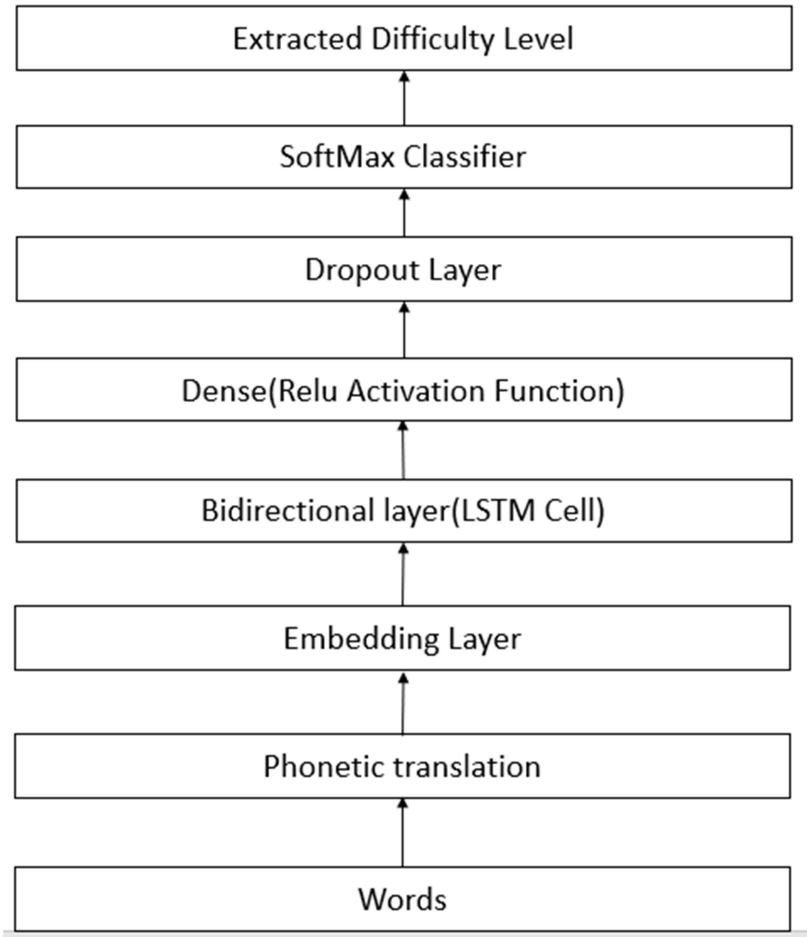


Fig.2.4 Word Difficulty Prediction Architecture

2.2 System Feasibility

A feasibility study is an assessment of the practicality of a proposed project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.

In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

2.2.1 Economic Feasibility

- Since the system doesn't consist of any multimedia data transfer, bandwidth required for the operation of this application is very low.
- The system will follow the freeware software standards.
- No cost will be charged from the potential customers.
- Bug fixes and maintaining tasks will have an associated cost.

2.2.2 Technical Feasibility

Word Difficulty Prediction is a complete deep learning project using RNN Architecture. The main technologies and tools that are associated with this project are:

- Python
- Python Libraries
- Dataset
- TensorFlow
- GPU Processor

2.2.3 Behavioral Feasibility

Being a project, which is used in the vocabulary field which needs Phonetic Translation API, this project will contain significant amount of code lines. The file sizes and the complete project size will not exceed 500MB. For predicting the result, it takes 5ms in the in the CPU computing, if the GPU is used then time would be reduced which suggest that its giving the solution in real time. From these it's clear that this project is behavioral feasible.

2.3 Experimental Setup

2.3.1 Hardware

- A system that supports latest version of IDE (Anaconda) and other technologies for smooth accessing of the project.
- Integrated software system to build the project.
- 2 X Tesla GK210B GPU for training machine learning models.

2.3.2 Software

- The project will require application of Artificial Intelligence with various sub topics like Natural Language Processing. Integrated software system to build the project.
- Anaconda's integrated development environment (IDE) - Spyder.
- Phonetic Conversion API
- Python programming language & libraries like Regular expression operations.

GPU Processor	2 X Tesla GK210B
CUDA Cores	2496
Core clock	560MHz
Memory	24 GB

Table 2.1 GPU Characteristics

Chapter-3

System Analysis

3.1 Information Flow Diagram

An information flow diagram or data flow diagram is a Unified Modeling Language (UML) representation that illustrates how information/data is being exchanged between system entities within a process. An information flow diagram assists user in understanding the entire flow of data from beginning to end.

3.1.1 Activity Flow Diagram

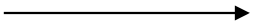
Activity diagram is defined as a UML diagram that focuses on the execution and flow of the behavior of a system instead of implementation. It is also called object-oriented flowchart. Activity diagrams consist of activities that are made up of actions which apply to behavioral modeling technology.

Activity Diagram Notations –

1. **Initial State –** The starting state before an activity takes place is depicted using the initial state.

2. **Action or Activity State –** An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically, any action or event that takes place is represented using an activity.

3. **Action Flow or Control flows –** Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.



4. **Decision node and Branching –** When we need to make a decision before deciding the flow of control, we use the decision node.

5. **Final State or End State** – The state which the system reaches when a particular process or activity ends is known as a Final State or End State.

The purpose of an activity diagram can be described as −

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Below is the representation of Activity diagram for Word Prediction-

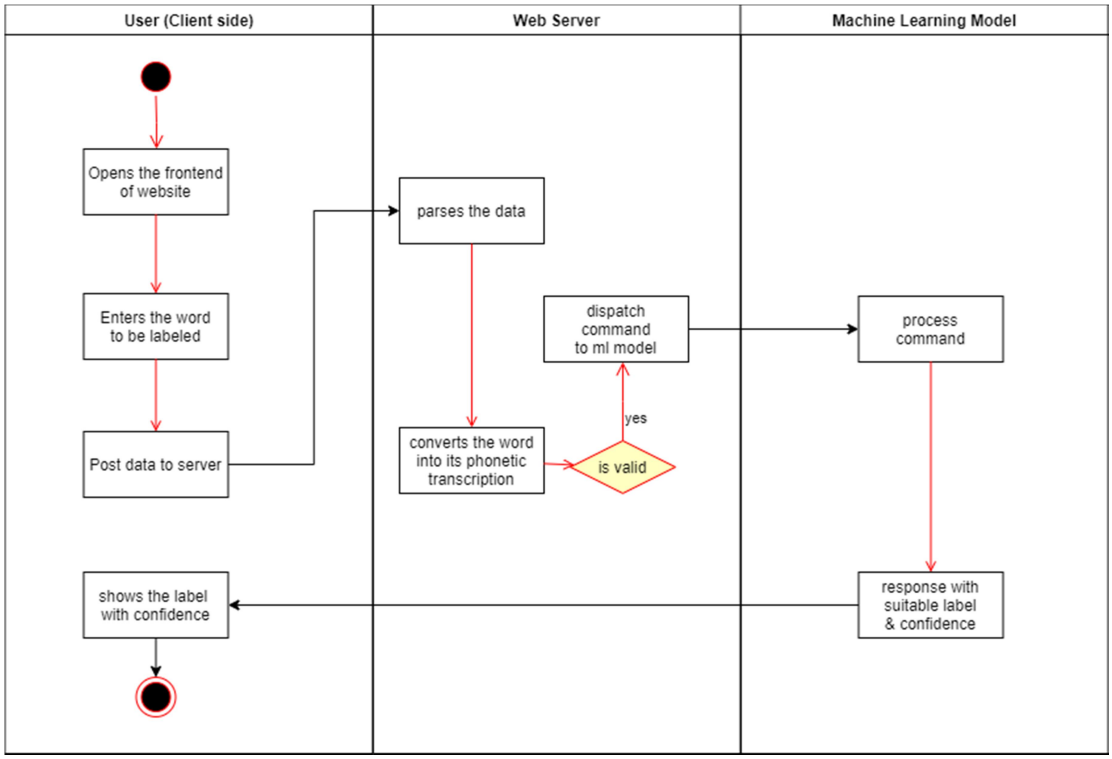


Fig. 3.1 Activity Diagram Of Word Difficulty Prediction

Chapter-4

Design

4.1 Architectural Design

The software needs the architectural design to represents the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectural styles.

Each style will describe a system category that consists of :-

- A set of components (e.g.: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 14, 128)           5760
_________________________________________________________________
bidirectional (Bidirectional (None, 256)               263168
_________________________________________________________________
dense (Dense)                (None, 32)                8224
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 99
=================================================================
Total params: 277,251
Trainable params: 271,491
Non-trainable params: 5,760
```

Fig.4.1 Word Difficulty Prediction Model

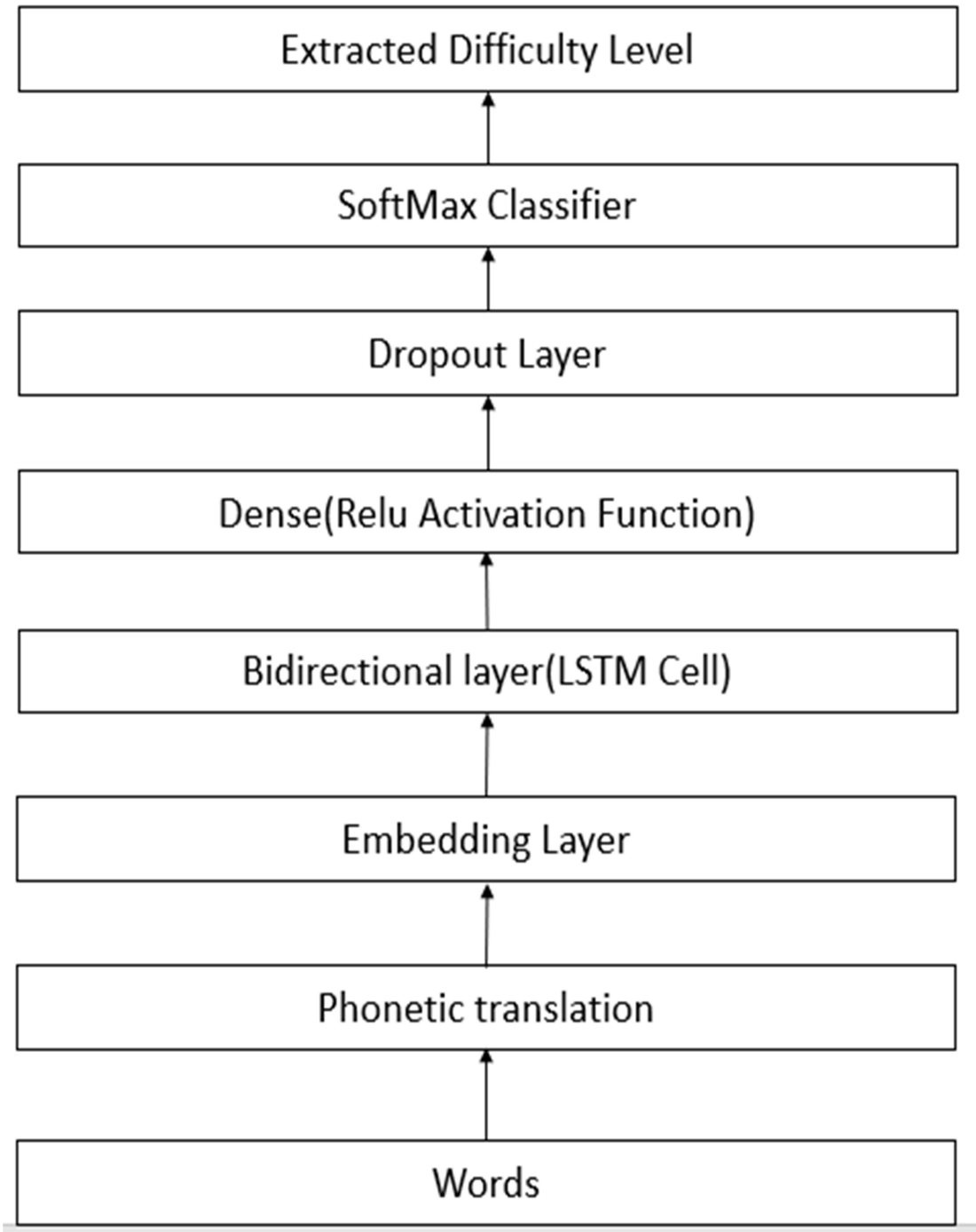


Fig. 4.2 Word Difficulty Prediction Architecture

4.2 Data Design

Data design is the first design activity, which results in less complex, modular and efficient program structure. The information domain model developed during analysis phase is transformed into data structures needed for implementing the software. The data objects, attributes, and relationships depicted in entity relationship diagrams and the information stored in data dictionary provide a base for data design activity. During the data design process, data types are specified along with the integrity rules required for the data. For specifying and 15 designing efficient data structures, some principles should be followed.

These principles are listed below.

- The data structures needed for implementing the software as well-as the operations that can be applied on them should be identified.
- A data dictionary should be developed to depict how different data objects interact with each other and what constraints are to be imposed on the elements of data structure.
- A data dictionary should be developed to depict how different data objects interact with each other and what constraints are to be imposed on the elements of data structure.
- Only those modules that need to access data stored in a data structure directly should be aware of the representation of the data structure.

```python
def preprocessing(sentences):
    words = []
    for x in list(sentences):
        characters_list = list(x)
        print(characters_list)
        temp = []
        for index in range(0, len(characters_list)):
            y = characters_list[index]
            if y == 'ɒ' or y == 'ʌ' or y == 'æ' or y == 'ɛ' or y == 'p' or y == 'b' or y == 't' or y == 'd'
               or y == 'ʧ' or y == 'ʤ' or y == 'k' or y == 'g' or y == 'f' or y == 'v' or y == 'θ' or y == 'ð'
               or y == 's' or y == 'z' or y == 'ʃ' or y == 'ʒ' or y == 'm' or y == 'n' or y == 'ŋ' or y == 'h'
               or y == 'l' or y == 'r' or y == 'w' or y == 'j':
                temp.append(y)
            elif y == 'i' or y == 'u' or y == 'ɜ' or y == 'ɔ' or y == 'ɑ':
                if (index + 1)  < len(characters_list) and characters_list[index + 1] == 'ː':
                    temp.append(y+'ː')
                elif (index + 1)  < len(characters_list) and characters_list[index + 1] == 'ɪ':
                    temp.append('ɔɪ')
                else:
                    temp.append('i')
                index = index + 1
            elif y == 'ɪ':
                if (index + 1)  < len(characters_list) and characters_list[index + 1] == 'ə':
                    index = index + 1
                    temp.append('ɪə')
                else:
                    temp.append('ɪ')
            elif y == 'ʊ':
                if (index + 1)  < len(characters_list) and characters_list[index + 1] == 'ə':
                    index = index + 1
                    temp.append('ʊə')
                else:
                    temp.append('ʊ')
            elif y == 'e':
                if characters_list[index + 1] == 'ɪ':
                    temp.append('eɪ')
                else:
                    temp.append('eə')
                index = index + 1
            elif y == 'ə':
                if (index + 1)  < len(characters_list) and characters_list[index + 1] == 'ʊ':
                    index = index + 1
                    temp.append('əʊ')
                else:
                    temp.append('ə')
            elif y == 'a':
                if characters_list[index + 1] == 'ɪ':
                    temp.append('aɪ')
                else:
                    temp.append('aʊ')
                index = index + 1
        words.append(temp)
    return words
```

Fig. 4.3 Preprocessing Code

4.2.1 Data Structures In The Model

Data structures which is used in our project:

- **Data Dictionary**: - A set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database.
- **List**: - A sequential set of elements to which you can add new elements and remove or change existing ones.
- **NumPy Arrays: -**
  - It is extension package to Python for multi-dimensional arrays.
  - It is closer to hardware (efficiency).
  - It is designed for scientific computation (convenience).
  - It is also known as array-oriented computing.

4.3 Phonetics

Phonetics is the branch of linguistics which deals with the study of production, transmitting and classification of speech sound and its representation in written symbols. Phonetic Symbols are given in below figure.

```
phonetic_tags = {'i': 0, 'iː': 1, 'ɪ': 2, 'ʊ': 3, 'uː': 4, 'ɪə': 5, 'eɪ': 6, 'ɛ': 7,
                 'ə': 8, 'ɜː': 9, 'ɔː': 10, 'ʊə': 11, 'ɔɪ': 12, 'əʊ': 13, 'æ': 14,
                 'ʌ': 15, 'ɑː': 16, 'ɒ': 17, 'eə': 18, 'aɪ': 19, 'aʊ': 20, 'p': 21,
                 'b': 22, 't': 23, 'd': 24, 'ʧ': 25, 'ʤ': 26, 'k': 27, 'g': 28,
                 'f': 29, 'v': 30, 'θ': 31, 'ð': 32, 's': 33, 'z': 34, 'ʃ': 35,
                 'ʒ': 36, 'm': 37, 'n': 38, 'ŋ': 39, 'h': 40, 'l': 41, 'r': 42,
                 'w': 43, 'j': 44}
```

Fig. 4.4 Phonetic Symbols

Phonetic translation of some of the words used in the model to train the dataset is shown in the below table.

Word	Phonetic Translation	Difficulty Level
boss	bɒs	Easy
significant	sɪgˈnɪfɪkənt	Easy
goggle	gɒgl	Medium
timorous	tɪmərəs	Hard
adversity	ədˈvɜːsɪti	Medium
cartilaginous	kəˈtælɪsɪs	Hard
language	læŋgwɪʤˈ	Easy
success	səkˈsɛs	Easy
oversized	ˈəʊvəsaɪzd,	Medium
tench	kəˈlaɪdəskəʊp	Hard
testis	tɪld	Hard
fresh	frɛʃ	Easy
smile	smaɪl	Easy
diadem	drʌʤ	Hard
hubris	ˈhjuːbrɪs	Medium
normalize	lɒnˈʤɛvɪti	Medium
longevity	ˈnɔːməlaɪz	Medium
gift	gɪft	Easy
timorous	ˈtɔːri	Hard
green	griːn	Easy
disruptive	dɪsˈrʌptɪv	Medium
clamorous	ˈkɒləkwi	Hard
discredit	dɪsˈkrɛdɪt	Medium

Fig. 4.5 Phonetic Translation of Few Words Used in Training Model

4.4 Dataset

Difficulty Level	Training Data (figure)	Testing Data (figure)
Easy	451	141
Medium	473	132
Hard	476	127
Total	**1400**	400

Table. 4.1 Summary of Training and Testing Dataset

Balanced dataset is prepared for training and testing of the model with total entities of about 1800 out of which 1400 is used to train the model and 400 is for testing. The summary of each dataset with their difficulty level type is shown in table.

4.5 Proposed Deep Learning Based Word Difficulty Level Prediction System

The proposed model works in two steps: Phonetic translation of the words and then training the data set using the Long Short-Term Memory, but before, there are five layers through which the dataset is passed which are termed as Embedding layer, Bidirectional Layer, dense (Relu activation function) Layer, Dropout layer, dense layer(sigmoid function). Figure 3.2 shows the architecture of deep learning based proposed Word difficulty prediction system. The description of each component is mentioned below.

4.5.1 Embedding Layer

Before passing the training dataset to the Embedding layer every word is translated in phonetics and all the fourth five characters of the phonetics is assigned a numeric value and hence the phonetic translated words are also converted into its numeric form. Now the numeric data set is passed through embedding layer which after considering all the dimensions convert the word into a vector in the range of (-1, 1)

4.5.2 Bidirectional Layer

Bidirectional layer is the main layer of the LSTM model It duplicates the first recurrent layer in the network and so that two-layer work side by side now providing the input to the first layer and provides a copy of input to the second layer.

Working of LSTM cell: Using LSTM cell for training the dataset gives better accuracy because LSTM evaluates the overall effect of each character of the word. And the output obtained by one character is influenced by its preceding character as well so the overall difficulty of the word is obtained.

4.5.3 Dense (Relu) Layer

Dense Layer is a type of connecting layer, it is basically connecting the Bidirectional layer from Dropout Layer. It is an activation function stands for Rectified Linear Unit. Its mathematical formula is y=max (0, x).

It is linear for all positive value and Zero for all negative value.

4.5.4 Dropout Layer

This layer prevents the model from over-fitting, all the data set that is coming as an input in this layer is filtered and half of the values are only transmitted to other layer in other words dropout can help a model generalize by setting the output of few neurons to 0. In setting the output to 0, the cost function becomes more sensitive to neighboring neurons changing the way the weights will be updated during the process of backpropagation.

4.5.5 Output Layer (Softmax)

It is the final layer of neural network. It gives the confidence of each relation type. In the work, Softmax classifier is used as multi class identification of relations. The output layer is applied on the joint vector to convert the output values into probabilities for relation classification. It returns the probabilities of each relation and target relation is having highest probability.

4.6 Training

Training is done on 1400 words and LSTM is used to train the model all the layer have functions associated with it and dataset is passed through all the layers of the neural network in forward direction in case error the data is send back to the layers in reverse order and in this way the bidirectional neural network trains the model.

The standard formula for RELU Activation function is defined as:

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

The standard formula for SoftMax activation function is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \; for\; i = 1, \ldots, k\; and\; z = (z_1, \ldots\ldots, z_k) \in R^k$$

4.7 Results Of Training And Further Discussions

After the training and testing the result is obtained and the dataset is divided such that the maximum accuracy can be obtained after the division and it is noted that on increasing the dataset for training the accuracy is increased.

Table - 4.2 shows the accuracy graph after the division of dataset is done. It can be concluded that as the dataset for training the model is increased the accuracy is increased.

Train Data	Test Data	% of test data	Accuracy
900	**900**	**0.5**	**62.21%**
1260	**540**	**0.3**	**76.25%**
1296	**504**	**0.28**	**88.6%**
1400	**400**	**0.22**	**97.5%**

Table 4.2 Summary Of Result With Respect To Dataset Split

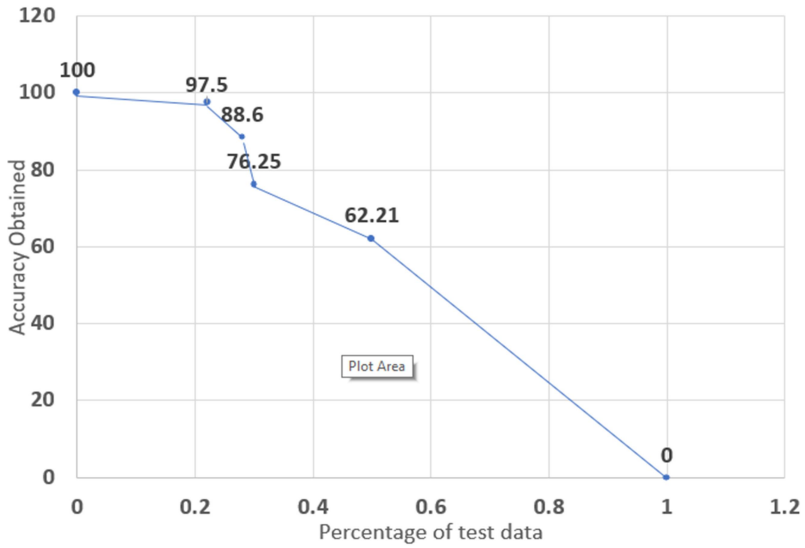


Fig. 4.6 Graph Representing Accuracy With Respect To Dataset Split

We can understand with these observations clearly that the increase in training dataset results in more accuracy and this can be seen in the Table 4.2 and Fig 4.6.

4.8 Further Discussions

In Table 4.2 it is shown that the dataset is divided for training and testing purpose and is divided such that highest precision can be obtained so the dataset is divided in different percentage and relative accuracy value is obtained. In the above table it can be seen that on using 50% of the dataset for both training and testing the model accuracy obtained is 62.21%, on using 30% of total dataset for testing the accuracy obtained is 76.25%, similarly on increasing the data for training the accuracy increased to 88.6% when 16.7% is used for the same highest accuracy of 97.5% is obtained and Fig 4.6 represents the graph obtained after the division of the dataset.

Table 4.3 shows the comparison of the performance obtained when the dataset is trained and tested by changing the various parameter and the output accuracy is also compared, number of nodes in Embedding layer as well as Dense layer is altered and the relative accuracy is compared, in bidirectional layer GRU cell as well as LSTM cell both are used and accuracy obtained by using them is also compared in the above table shows the relative accuracy by

altering the different parameters in the layers which clearly shows that using LSTM cell gives better precision then using GRU cell and the using 128 nodes in embedding layer and 32 nodes in dense layer results in the highest accuracy.

Embedding Layer	Bidirectional Layer	Dense Layer	Dropout Layer	Output Accuracy
64	GRU	16	0.4	88.5%
128	GRU	32	0.5	95.2%
256	GRU	64	0.6	91.1%
562	GRU	128	0.5	87.5%
64	LSTM	16	0.4	91.5%
128	LSTM	32	0.5	97.5%
256	LSTM	64	0.6	93.6%
562	LSTM	128	0.5	89.4%

Table 4.3 Comparison of Performance

Difficulty Level	Easy	Medium	Hard
Easy	138	1	0
Medium	2	128	3
Hard	1	3	124

Table 4.4 Final Confusion Matrix

Table 4.4 shows the confusion matrix obtained from the testing data set there are two easy word which are predicted as Medium and one is predicted as hard and similarly in case of medium difficulty level words there are four words which are giving wrong prediction and in case of Hard words three wrong prediction has been taken place so the overall accuracy obtained is 97.5%.

Difficulty Level	Precision	Recall
Easy	**0.9787**	**0.9928**
Medium	**0.9696**	**0.9624**
Hard	**0.9763**	**0.9687**
Macro average	**0.9748**	**0.9746**

Table 4.5 Precision And Recall Table

Whereas recall is the ratio of the correctly labeled positive by our program to all who are diabetic in reality.

Macro average represents the overall performance of the system across the dataset

$$PRECISION = \frac{TRUE\ POSITIVE}{TRUE\ POSITIVE + FALSE\ POSITIVE}$$

$$RECALL = \frac{TRUE\ POSITIVE}{TRUE\ POSITIVE + FALSE\ NEGATIVE}$$

Chapter-5

Testing

A crucial part of any software development lifecycle is testing. This involves carrying out certain procedures and operations to understand the limitations of the software. It is evident that with testing the constraints of the application that particular bugs and errors are picked up and documented through test cases. This will improve the overall standard and quality of the chatbot and enhance the user experience.

Testing is the major quality control that can be used during software development. Its basic function is to detect the errors in the software. During requirement analysis and design, the output is a document that is usually textual and non-executable. After the coding phase, computer program is available that can be executed for testing purposes. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during previous phases. Thus, the goal of the testing is to uncover requirement, design and coding errors in the program.

5.1 Testing Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers an as-yet undiscovered error.

5.2 Testing Principle

- All tests should be traceable to customer requirements.
- Tests should be planned long before testing begins.
- Testing should begin “in the small” and progress toward testing “in the large”.
- Exhaustive testing is not completely possible.
- To be most effective, testing should be conducted by an independent third party.

5.3 Testing Methods Used

Software Testing Strategies

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. As important, a

software testing strategy provides a road map. Testing is a set of activities that can be planned in advance and conducted systematically.

Various strategies are given below:

- Unit Testing
- Integration Testing
- Validation Testing
- User Acceptance Testing
- System Testing

Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design of module. This is also known as "Module Testing". Acceptance of package is used for computerization of module. Machine Utilization was prepared and approved by the project leader. In this testing step, each module is found to be working satisfactory as regards to the expected output from the module. The suggested changes were incorporated into the system. Here each module in the Machine Utilization has been tested.

Integration Testing

After the package is integrated, the user test version of the software was released. This testing consists of testing with live data and various stress tests and result were noted down. Then the corrections were made based on the user's feedback. Integration testing is systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. Thus, the integration testing step, all the errors uncovered are corrected for the next steps.

Validation Testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been uncovered and corrected, and a final series of software tests - Validation testing - may begin.

User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system users at time of development and making changes wherever required. After performing all the tests, the system was found to be running successfully according to the user requirements i.e., (constraints).

System Testing

Software is only one element of a larger computer-based system. Ultimately, software is incorporated with other system elements and a series of system integration and validation tests are conducted. The various types of system testing are:

- Recovery Testing: Many computer-based systems must recover from faults and resume processing within a pre specified time
- Security Testing: Security testing attempts to verify that protection mechanisms built into a system will in fact protect it from improper penetration.
- Stress Testing: Stress tests are designed to confront programs with abnormal situations.
- Performance Testing: Performance testing is designed to test run-time performance of software within the context of an integrated system.

5.4 Test Cases

A total of 20 simulated queries were tested and the results are shown in table. Each query tested was uniquely phrased including variations of the same query to accumulate a range of results so and thoroughly classify the Word Prediction understanding can be thoroughly classified. This also outlines how it handles unexpected input, such as misspelt words. The confidence score is returned along with the intent the test user word was matched to.

5.5 Sample Test Data and Result

Simulated User Utterance	Confidence Score	Matched Intent(s)	Unrecognized Intent(s)	Response Time*(ms)*
Boss	1	✓	×	3480
Significant	0.9999	✓	×	3414
Competition	1	✓	×	3564
Downturn	1	✓	×	3446
Outdated	1	✓	×	2181
Egregious	0.9700	×	✓	3480
Coerce	0.9600	✓	×	3649
Upheaval	0.9400	✓	×	3308
Nightfall	0.9999	✓	×	2504
Barrister	1	✓	×	7339
Telegram	1	✓	×	N/A
Conceptual	0.9000	✓	×	3228
Timorous	0.9499	✓	×	3182
Blockhouse	1	✓	×	1613
Diadem	0.6499	×	✓	3381
Aftermost	1	✓	×	3128
Attributive	0.9600	✓	×	3380
Catalepsy	1	✓	×	3345

Table 5.1 Test Data and Result

Chapter-6

Limitations

The provided solutions, while possessing obvious advantages, have some important limitations in terms of functionalities and use cases. We will discuss them below.

Since user experience has not been strongly defined in the literature, we focused on the most mentioned aspects of user experience (usability, usefulness, and satisfaction). It is debatable if these three constructs are valid to measure the total user experience. This is a decision that was made based on time restrictions. To give a broader image of user experience, other measurements should have been used in addition (e.g. EEG). We tried to tackle this by using a combination of quantitative and qualitative user testing.

The dataset, used to train the machine learning model, is fixed and limited and cover goal-specific intents of the user, thus can fail while dealing with a new intent query.

The application is web-based and requires the internet for word to phonetics transcription, retrieving phonetics of the word. Thus, the application cannot be used without an active internet connection.

Chapter-7

Future Scope

Due to time and technical limitations, not all aspects of Word Difficulty Predictor were possible to research. In this section we discuss some of the aspects that are interesting to be researched further.

In our research we chose the three constructs of usability, usefulness and satisfaction to measure user experience. How-ever, there could be other constructs relevant to measure user experience. Also, it would be interesting to research Word Difficulty Predictor with a different kind of UX framework or evaluation method. Also, looking into measuring user engagement of Word Difficulty Predictor could be interesting.

The future scope of this project is in increasing its throughput by decreasing the response time of the Word Difficulty Predictor. We can achieve this goal by making a better machine learning model and using better computational devices like faster GPUs.

We can also make an API for this, so that other developers and companies can also easily use our Word Difficulty Predictor for their own requirements.

The Machine learning model can classify 3 different intents from the user's query. But we have to add more intents in the Machine learning model to make the project superior. The dataset and the algorithm used for training the model also affects the accuracy of the model. We have used the Recurrent Neural Network with LSTM cell to train our model after comparing it with other algorithms. Our model has an accuracy of 97.5%, which can be improved further either by adding more data in the training set or by using other algorithms.

Chapter-8

Conclusion

Difficulty level of word prediction is used to identify how easy the word is to read and the written word is used to communicate whole idea behind the text. It measures whether content is likely to be understood by your intended reader. The project aims to identify appropriate tools and techniques for predicting the difficulty level of words used in a given text. In this paper, various tools and technologies are used and tested by changing various parameter and the precision value is obtained for each of them. The performance of the system is compared by using both LSTM and GRU cell and relative accuracy obtained by them is compared. The precision obtained shows that it is better to use LSTM cell and hence the model is trained using LSTM based approach of recurrent neural network and the accuracy obtained by training the model is 97.5%.

Chapter-9

Bibliography And References

REFERENCES

[1] A. P. Rudell and J. Hua, "The recognition potential, word difficulty, and individual reading ability: On using event-related potentials to study perception," *Journal of Experimental Psychology: Human Perception and Performance,* vol. 23, p. 1170, 1997.

[2] H. M. Breland, "Word frequency and word difficulty: A comparison of counts in four corpora," *Psychological Science,* vol. 7, pp. 96-99, 1996.

[3] H. M. Breland, *Assessing Writing Skill. Research Monograph No. 11*: ERIC, 1987.

[4] P. Curto, N. Mamede, and J. Baptista, "Automatic text difficulty classifier," *Assisting the selection of adequate reading materials for European Portuguese teaching. Proceedings of CSEDU,* pp. 36-44, 2015.

[5] M. W. Kibby, "Note on relationship of word difficulty and word frequency," *Psychological Reports,* 1977.

[6] J. M. Tamayo, "Frequency of use as a measure of word difficulty in bilingual vocabulary test construction and translation," *Educational and Psychological Measurement,* vol. 47, pp. 893-902, 1987.

[7] R. P. Carver, "Word length, prose difficulty, and reading rate," *Journal of Reading Behavior,* vol. 8, pp. 193-203, 1976.

[8] R. P. Carver, "Measuring prose difficulty using the Rauding Scale," *Reading research quarterly,* pp. 660-685, 1975.

[9] R. P. Carver, "Manual for the Rauding Scale Qualification Test," *Kansas City, MO: Revrac Publi-cations,* 1974.

[10] X. Chen and D. Meurers, "Word frequency and readability: Predicting the text-level readability with a lexical-level attribute," *Journal of Research in Reading,* vol. 41, pp. 486-510, 2018.

[11] L. L. Haseley, "The relationship between cue-value of words and their frequency of prior occurrence," Ohio University, 1957.

[12] W. N. Francis, H. Kučera, and A. W. Mackie, *Frequency analysis of English usage: Lexicon and grammar*: Houghton Mifflin Harcourt (HMH), 1982.