“BUG REPORT SUMMARIZATION”

A

PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

Computer Science and Engineering

Under the supervision

of

Dr. Hemraj Saini

by

Varnita Sachdeva (151258)

Ayushi Mittal(151279)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN– 173234

HIMACHAL PRADESH, INDIA, MAY- 2019

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"BUG REPORT SUMMARIZATION"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat, is an authentic record of my own work carried out over a period from August 2018 to May 2019 under the supervision of Dr. Hemraj Saini, Associate Professor Computer Science.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Varnita Sachdeva, 151258

Ayushi Mittal, 151279

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Hemraj Saini

Associate Professor

Computer Science & Engineering

Dated:

ACKNOWLEDGEMENT

I take this opportunity to acknowledge all who have been a great sense of support and inspiration throughout the project work. There are lot of people who inspired me and helped me in every possible way to provide the detail about various related topics, thus, making report work a success. I am very grateful to my supervisor **Dr. Hemraj Saini** for his guidance, encouragement, and support.

I am also thankful for all his diligence, guidance, and encouragement and helped throughout the period of this report, which has enabled me to complete project report work on time. I also thank him for the time that he spared for me, from his extremely busy schedule. His insight and creative ideas are always the inspiration for me during the dissertation work.

Thanking you,

Varnita Sachdeva (151258)

Ayushi Mittal (151279)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S No.	Abbreviations	Description
1	MMR	Maximum Marginal Relevance
2	BRC	Bug Report Corpus
3	EC	Email Corpus
4	EMC	Email Meeting Corpus
5	Tf-idf	Term frequency, Inverse-document frequency
6	NLP	Natural Language Processing
7	POS	Part of speech
8	ML	Machine Learning
9	ANN	Artificial Neural Network
10	NB	Naïve Bayes Model
11	RNN	Recurrent Neural Network
12	LSTM	Long short-term Memory

ABSTRACT

Bug Report Summarization is one of the most necessary need for developers these days. As developers come across many bugs while working on some project. It is very difficult to undergo long documents to find the solution of their bug. That is why Bug Report Summarization is done to generate summaries that are shorter and precise and can ease the work of coders, testers and other people as well. This report depicts our fourth year project "Bug Report Summarization", describing the method using which a summary can be generated using an algorithm.

CHAPTER -1
INTRODUCTION

1.1 Introduction

Natural Language Processing:

"Natural Language Processing is a field that does understanding and manipulation of human language into a language understandable by computer, and it's blooming with possibilities for news-gathering". "You usually hear about it in the context of analyzing large pools of legislation or other document sets, attempting to discover patterns or root out corruption." With the use of NLP computer analyses, understands and derive the meaning from human language into a summarized and useful, helpful way. By using NLP, developers have organized and structured knowledge to implement tasks of 'automatic summarization.'

"Apart from common word processor operations that have been treating text like a sequence of symbols, natural language processing takes into consideration the hierarchical structure of this language, that means, many different words combine to form a phrase and several phrases combine to form a sentences and, finally, these are the sentences that convey ideas," Natural Language Processing has been used since years to analyse text, and to make models through which machines can understand human language and their emotions.

Such human-computer interaction encourages processing of real-world applications like: sentiment analysis and automatic text summarization. NLP is most commonly used for text mining, text summarization, semantic analysis, machine translation, question answering and for many other applications. The main drawback while processing human language is they are not just plain text. They contains different meaning for different arrangements of words, also they carries emotions of speaker. Although, humans naturally learn their respective languages without any difficulty but for processing because of ambiguity this is a difficult task.

NLP algorithms are normally derived from machine learning algorithms itself. Instead of coding manually large sets of rules, rules are learnt automatically using Machine learning.

In general, more the training is done, more accuracy can be achieved by the model. This depends on following different features extraction:

Tokenization- Extracting main key words, reducing words to their root form, applying stemming and lemmatization.

Pos tagging- Finding the noun using pos tags.

Score Generation- Generating score for each token to know the value of each word in a document. And there are many more processing features.

Machine Learning:

Artificial Intelligence has many sub fields one of them is machine learning. In general, ML's goal is to understand the structure of data and then applying necessary model on it that can be used to process to gain necessary output by different people.

In conventional computing, "algorithms are set of explicitly programmed instructions used by computers for calculation and problem solving." Whereas, ML is completely different from conventional approach, it allows the computer to train on data-inputs and use statistical-analysis to order data output values that fall within an anticipated specific range. It's because of this that ML helps computer to process taking help from data-inputs training set.

Anyone using technology today has got the benefit of using ML. For example, technology of Image Processing has helped to read data from images, technology of summarization saves time of the user from reading large documents. Recommendation engines are based on ML, suggesting latest, trending videos and shows.

Machine Learning Methods

Processing in machine learning is classified under two broad category based on how learning is received by the system and how it is interpreted and developed into an output.

One method is supervised learning, which is based on a training algorithm having input-output data sets using which further processing is done. Another method is unsupervised algorithm which has algorithm where no pre data set is given. These models will be explained in further pages.

- Supervised Learning- For applying supervised learning, the computers are provided with the set of inputs which have corresponding set of outputs. The main reason for this method is that the algorithm should be “able to learn by comparing its actual output with the taught outputs to find errors and then modify the model in the required way.” This learning follows pattern to predict label values on the testing data set. For example, when supervised learning is used, it is fed with input data of shark image as fish and river image as water. So when user will input with shark image it will give output as fish and similarly for river. A very often use of this method is using earlier data to predict future events. It may be used for weather forecast, like when tides, and previous temperature is processed it will be helpful to predict weather for next few days of a place. It is also helpful for image reading, when it will be fed with previous input-output set of images.

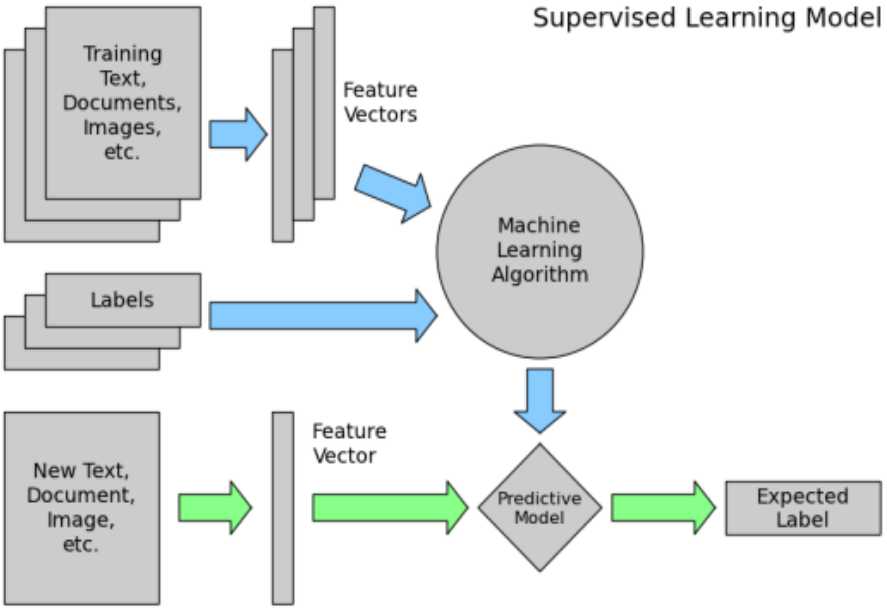


Figure 1: Supervised Learning Model [1]

- Unsupervised Learning - In case of unsupervised learning, data is not given pre hand. So the model itself find common grounds between different information to predict future data. As we all know learning by itself is more difficult than labelled data, this machine learning method is quite difficult but also an important learning method. The final goal of this learning is straight- forward to discover hidden

patterns within a data-set, it also have aim of feature learning as discussed above, which allows the machine to automatically find the representations that are required to classify raw-data. Unsupervised learning is generally used for data to be used as transactional one. The most appropriate example for this will be a recommendation engine. Suppose you being a service provider of latest movie, can't make sense from the data available of user that who prefer what. So in this case, unsupervised learning is used to recommend you with which age group likes to watch what type of genre of movies. Therefore, providing the users with their preferred information earn the service provider more marketing and user's interest.

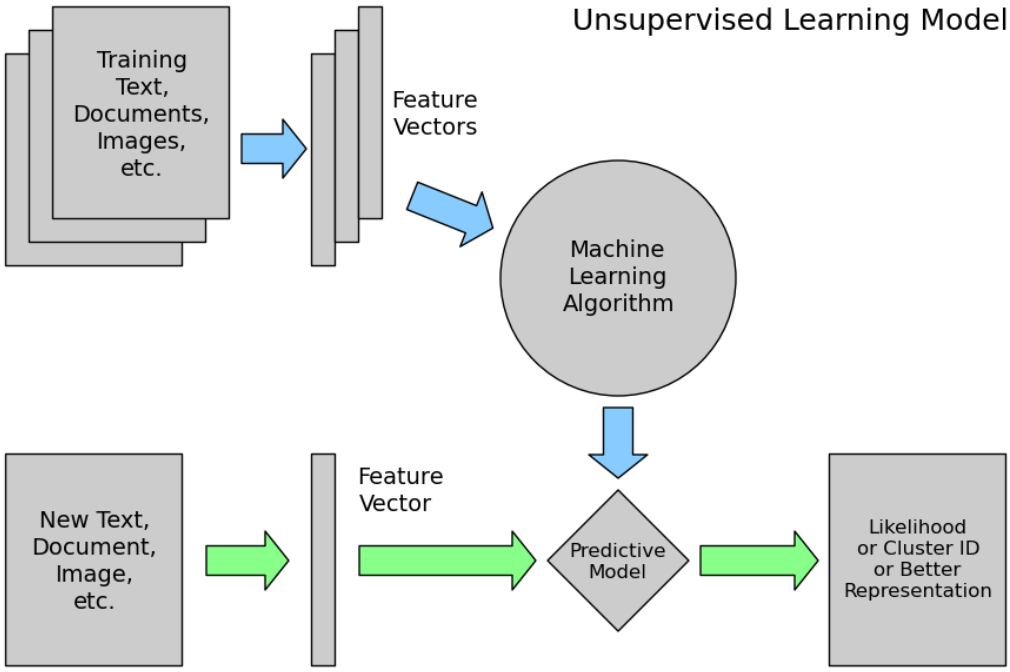


Figure 2: Unsupervised Learning Model [2]

Deep Learning

Deep learning is a field which reciprocate how human brain works and how it can process any information in any form (like sound, light etc). It's architecture by biological neural network of human body and similarly consist of multiple layers in an artificial neural network form.

Artificial Neural Network

A neural network is a three layer classifier operating in parallel. The first layer receives the input information similar to optic nerves in human visual processing. Second layer is hidden layer that works on some given algorithm to process the input information. The last layer produces the output of the system. Advantage of using neural network is that they are adaptive. They can modify themselves as they learn from initial training.

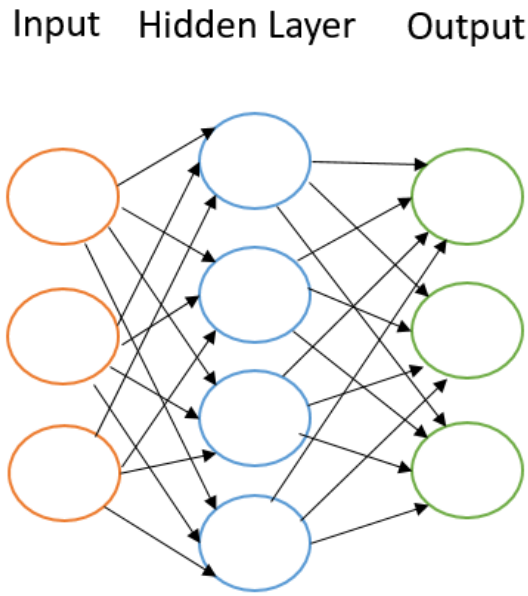


Figure 3: Artificial Neural Network Model

Text Summarization

Text Summarization is a process of making short, meaningful paragraphs from huge documents to ease the working of researcher, developer, or user. Based on the method of summary generation, text summarization systems are of two kinds: extractive and abstractive.

- In extractive text summarization, at first each word is separated as tokens by applying certain tokenization techniques. Noise words and stop are removed. After that there is score calculation for each token by applying any given algorithm like tf-idf etc. After that each score is compared and tokens with higher score are selected to be inserted in summary.

- In abstractive text summarization, instead of extracting sentences from the original document by applying a predefined algorithm and then using them as they are in summary in this method sentence is generated by the system itself which uses Natural Language Processing to do so. This type of summary generation is more complex than extractive.

Programming Language

While choosing a language to code and create a model, we should choose a language in which we have specialization and the one which has maximum libraries and modules to ease the work.

Python has been the most used programming language for this work.

Other languages like Java, R and C++ can be kept as optional (this information is taken from indeed.com December 2016).

Python is an object oriented programming language which is easy to interpret and learn because of its simplicity of syntax. Python being a portable language can be used on various operating systems such as Mac Os, UNIX, and various Windows version. Its popularity has increased in the field of deep learning because of the availability of large number of libraries including nltk kit, Tensor Flow, NumPy, Keras etc. The modules available are easy to use and direct in implementation.

Java is an HLL. Its syntax is similar to C++ but is strictly an object oriented programming language. Usually, it is not one of the favoured language by the developers who are new to coding but it is surely preferred by those who already have a good hand in working in Java. Also, Java is used more than Python for detection of false cases, cyber-attacks and security related problems.

1.2 Problem Definition

The volume of electronic information available on Internet has increased rapidly. As a result, dealing with such huge volume of data has lead to a problem and time consuming task. Mostly research shows that summary extraction is a shortening of information which means selecting only those words or sentences that feels to be logically correct. Humans also extract summary whose efficiency is very low as they don't perform any particular

algorithm to generate a summary. But there is a drawback of this approach. There is no focus on particular text weight and its position in the document. So our focus is on generating a summary that is generated by calculating tokens score and its uniqueness and positioning in a document.

1.3 Objective

A software project's bug repository provides an upscale source of data for a developer operating on the project. For instance, a developer may need to consult few bug reports reported in the past. Developer does so in order to find the bug he is dealing with and to find how the bug was resolved. While scrolling the bug reports repository, developer have to read several reports before he finds the correct solution. As information is growing day by day, length of each bug report has also increased. Sometimes a developer finds the required bug report by only reading the title or many times he has to read the complete lengthy reports. Thus, developer has to invest a lot of time while reading tens or hundreds of sentences of each bug report to find the solution. These searches are very tedious and has become an issue.

One solution to reduce the time the developer spends getting to the right bug report to perform their task is to provide them with summary of each report. This can be done by using a software to generate a summary to help the developer to get the solution in optimal time. This summary generation is known as Bug Report Summarization.

As till now, for performing extractive text summarization various models like Naïve Bayes Classifier have been proposed. Naïve Bayes treats the summarization problem as classification problem and calculates whether a sentence can be added to the summary or not. They compare each sentence with other one and scores are given to each sentence is given depending on the algorithm used. There is a threshold score which is dependent on the length of the summary required, and every sentence which is higher than the score is included in the summary.

The above method seems to be easy but only for some genre-specific summarization (hospital reports, news articles etc). This means that article which are trained using data set of same genre, gives great result but these techniques fails in case of general text summarization.

To solve this issue, and for in general text summarization Neural Network Models are most preferred model.

1.4 Proposed System Methodology

In our research, we have used two different algorithm to generate summaries of the bug reports that are Naïve Bayes Classifier and Artificial Neural Network.

Naïve Bayes Model performs in a way where Bug Reports are taken as input files and after performing the algorithm for Naïve Bayes, bug report summary is generated. The overall computation can be divided into two phase- Training Phase, Testing Phase. The corpus for input consisted of 10 bug reports (taken from "Automatic Summarization of Bug Reports", Sarah Rastkar, Gail C Murphy [6]). We checked the effectiveness of the summary generated by the classifier by comparing it with the human written summary. Effectiveness measure taken into account were precision, recall and F-score.

Neural Network Model performs in a way where all the bug reports are taken in a single .csv file. And after performing ANN algorithm, bug report's summary is generated. The overall computation can be divided into two phase- Training Phase, Testing Phase. The corpus for input consisted of 36 bug reports (taken from "Automatic Summarization of Bug Reports", Sarah Rastkar, Gail C Murphy [6]).

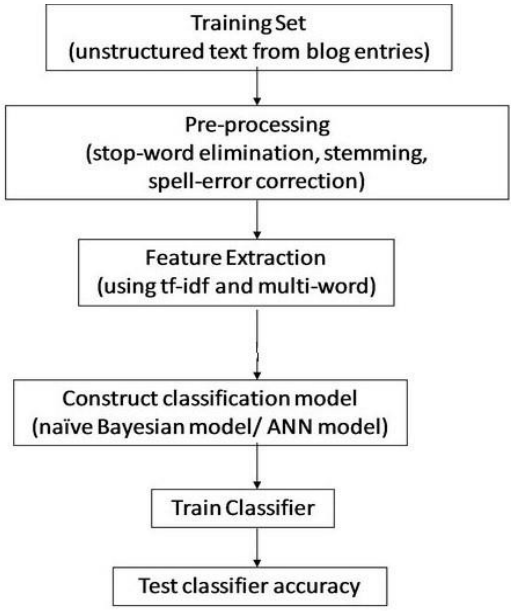


Figure 4: General working of Text Summarizer [3]

CHAPTER -2
LITERATURE SURVEY

2.1 Authors: Senthil Mani, Rose Catherine, Vibha Singhal Sinha, Avinava Dubey

Title: Approach for Unsupervised Bug Report Summarization [2012] [4]

Work done on this paper was it used four unsupervised techniques (Cen-troid, MMR, Divrank, Grasshopper) and compared their efficacy with supervised approach(BRC,EC,EMC). The efficacy of the unsupervised techniques is enhanced by noise removal technique and deducting the useless code. There were few significant limitation observed that were- Observed results are particular results can't be used for other setups and they need reconfiguration depending on subject to which noise reduction is applied .

2.2 Authors: Rafael Lotufo, Zeeshan Malik, Krzysztof Czarnecki

Title: Modelling the hurried bug reports reading process to summarize bug reports [2012] [5]

This paper focused on unsupervised techniques for bug summarization and apply noise reduction, two of the unsupervised technique became scalable for larger size bug reports. Its algorithm resulted into following limitations- it don't provide an evaluation showing how user would choose two input parameter, which are summary percentage length and minimum relevance threshold.

2.3 Authors: Sarah Rastkar, Gail C Murphy

Title: Automatic Summarization of Bug Reports [2014] [6]

They made a summarizer that produces summaries that are statistically better than produced by existing conversation- based generators. In this paper, they have researched the automatic generation of one sort of software skill, bug answers, to give designers the advantages others encounter every day in different areas. They found that current conversation based extractive summary generators can create summaries for bug reports that are better than any random classifier. They likewise discovered that an extractive summary generator prepared on bug reports delivers the best outcomes. It resulted into

following limitations that is Naïve/non-experts can't create summaries and more than one annotator is required.

2.4 Authors: Elder Cirilo, Fernando Mourao

Title: Bug Report Summarization: An Evaluation of Ranking Technique [2016] [7]

They provided a solution using extractive summaries where summaries are based on comments instead of one based on isolated sentences. In this paper, they propose a novel methodology where summaries depend on remarks, rather than the ones dependent on confined sentences, as proposed by past works. Exact outcomes prove with our arguments that positioning the most relevant comments would enable developers to discover more appropriate information. They could observe that summaries generated by conventional ranking algorithms are precise concerning developers expected data, when contrasted with reference summaries made manually, offers applicable summaries in general. Conclusion lead to limitation like Size and amount of reports may be a threat to the conclusion of their study and few bug reports contain different types of structured information which can't be treated in the algorithm.

2.5 Authors: Ha Nguyen Thi Thu

Title: An Optimization Text Summarization Method Based on Naïve Bayes and topic word for single syllable language [2014] [8].

A text summarization method based on Naïve Bayes algorithm and topic word sets is proposed. Processing time for word processing is long special when worked with more text.

2.6 Authors: HeJiang, Najam Narar, Tao Zhangi, Zhilai Ren

Title: A page rank based summarization technique for summarizing bug reports with duplicates [9]

It's a used page ranking which effectively utilized the textual information of duplicate bug reports for generating extractive summaries of master bug reports. While comparing the results they concluded that their algorithm outperformed the BRC algorithm in terms of precision, recall, F-score measures. Limitation founded in this algorithm was their

algorithm cannot slice paragraph into sentences/words independently they took help of external ling pipe toolkit.

2.7 Authors: Sarah Rastkar, Gail C Murphy, Gabriel Murphy

Title: Summarizing software Artifacts: A case study of bug reports [10]

They created an automatic generation technique for bug summarization and found out that existing conversation based extractive summary generator can produce better summaries for reports than a random classifier. Limitation concluded were they have used sufficient data set but training set (experts) size was limited.

2.8 Authors: Khosrow Kaikah

Title: Text Summarization using Neural Network [2004] [11]

A new technique for summarizing new articles using a neural network is presented.

2.9 Authors: Aakash Sinha, Abhishek Yadav, Akshay Gahlot

Title: Extractive Text Summarizing using Neural Networks [2018] [12]

Work done here was it took a fully data driven approach using feedforward neural networks for single document summarization. It is Limitation was that it assumed that their generated summary length < page_len.

2.10 Authors: Aditya Jain, Divij Bhatia, Manish K Thakur

Title: Extractive Text Summarization using Word Vector Embedding [2017] [13]

They proposed an approach to extract a good set of features followed by neural network for supervised extractive summarization. There was still a scope for performance improvement, by also using abstractive summaries

2.11 Authors: Meetkumar Patel, Adwaita Chokshi1 , Satyadev Vyas, Khushbu Maurya

Title: Machine Learning Approach for Automatic Text Summarization Using Neural Networks[2018] [14]

In this research paper, summaries were generated by using RNN model. There was an encoder that takes an input sequence and generates a vector output and there is a decoder which produces the final output. It also discusses the implementation of encoder-decoder model with respect to Keras using Tensor Flow. Their model still doesn't work for multi lingual and multi –documents

2.12 Authors: Ramesh Nallapati, Bowen Zhou, Cicero dos, Santos Çaglar, Bing Xiang
Title: Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond [2016] [15]

In this research paper, abstractive text summarization is done using Attentional Encoder-Decoder Recurrent Neural Networks. They made a model for multi-sentence summary generation. But still they need to prepare more robust model to increase their efficiency.

2.13 Authors: Sumit Chopra, Michael Auli, Alexander M. Rush
Title: Abstractive Sentence Summarization with Attentive Recurrent Neural Networks[2016] [16]

They made a simplified model of encoder-decoder model. Their model is trained on Gigaword corpus to generate headlines based on a particular approach. Their approach out-performed other models that were built for Gigaword corpus

2.14 Authors: Mikael Kageb, Olof Mogren, Nina Tahmasebi, Devdatt Dubhashi
Title: Extractive Summarization using Continuous Vector Space Models [2014] [17]

They evaluated different composition for sentence representation using ROGUE. They deduced the results which showed the advantages of using vector representation for summary generation. They used phrase embedding and showed this method significantly improved the performance. Their future aim will be to work on multiple kernels and to increase the performance from the above mentioned methods only.

CHAPTER -3

SYSTEM DEVELOPMENT AND DESIGN

3.1 Software Requirements:

- Python 3.6 /Anaconda3-5.2.0
- Jupyter Notebook

3.2 Hardware Requirements:

- CPU: 2.5 GHz Processor, above
- RAM – 4 GB, and above
- OS: Windows 10

3.3 Proposed Model

3.3.1 Technology Used

- Python is an object oriented programming language which is easy to interpret and learn because of its simplicity of syntax. Python being a portable language can be used on various operating systems such as Mac Os, UNIX, and various Windows version. Its popularity has increased in the field of deep learning because of the availability of large number of libraries including nltk kit, Tensor Flow, NumPy, Keras etc. The modules available are easy to use and direct in implementation.
- Anaconda is an open-source, free software used for the distribution of Python and R programming languages for deep learning, machine learning applications etc. This is a software which can be usable on many operating systems like Windows, UNIX, and MacOs having many important packages like NumPy, Keras, Tensorflow etc. Python development involves conda, which is a platform-independent package manager.
- Jupyter Notebook is an open-source, free web application that allows us to build and share documents that contain equations, live code, narrative text and visualization.

3.3.2 Libraries Used

- Natural Language Toolkit (NLTK) provides a platform to use Python to work it for human language data-sets for applying NLP. It consist of libraries for text processing like for finding stopwords, for tokenization, stemming etc.
- NumPy is a module of Python acronym for “Numeric Python”. It is based on data strcutres, having large number of mathematical functions to operate arious operations like matrices, arrays, multi-dimensional arrays. They help the developer to run the model even with a small unit of code available.
- TensorFlow is an open source library used for faster numeric computations. It was created by Google. Unlike other libraries, it can run on single CPU as well as like GPU on other mobile devices and is designed to be used in both research and development systems.
- Keras is a Python Library made to implement deep learning models as fast and as easily as possible.

3.3.3 System Design

Naïve Bayes Model

The algorithm is based on Naïve Bayes Classifier in which Bug Reports are taken as input files and after performing the algorithm bug report summary is generated. The overall computation can be divided into two phase- Training Phase, Testing Phase.

Figure 5 depicts Training Phase. In training phase there are various levels that training set document undergo to create a summary. Firstly, training set is selected. Training set is basically set of documents which contain bug reports of various errors. On that training set preprocessing is done. Preprocessing is a method of eliminating noise and stop words from the sentences and separating the key words and storing them as tokens. Then POS tool is used for extracting nouns. Then with the help of Human, a summary is generated. On evaluating human generated summary sentences are divided into two classes. Sentences selected in the summary are categorized as “class1” and sentences removed or not selected are categorized as “class0”.After this, feature extraction is performed where score is given

to each token by calculating word frequency for both “class1” and “class0”. This score shows the weight and uniqueness of each token in a sentence and in a document as whole.

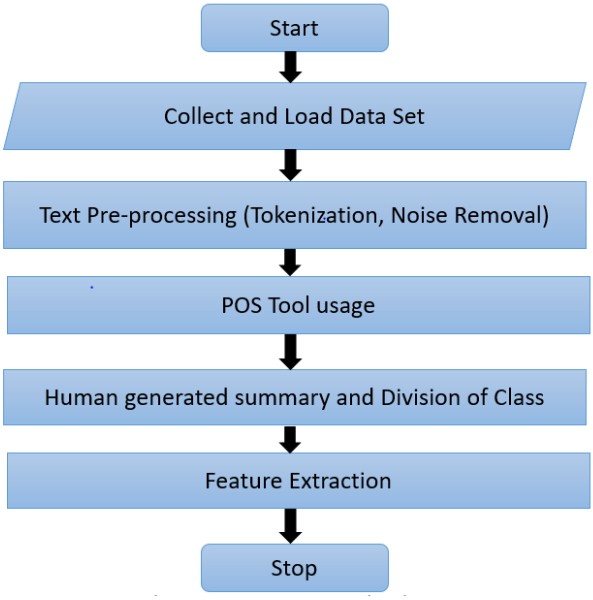


Figure 5: Flowchart of Training Phase of NB

Figure 6, depicts testing phase. In this, various levels include selecting a document. Sentence tokenization is done on the selected document. Then Naïve Bayes Classifier is applied. In this model, Bayes Theorem is applied to calculate, probability of each sentence for both “class1” and “class0” is calculated by taking total sum of probability of each word in a sentence. Then, both probabilities of a sentence are compared and if probability for “class1” is greater than that for “class0”, that sentence is added to the final summary.

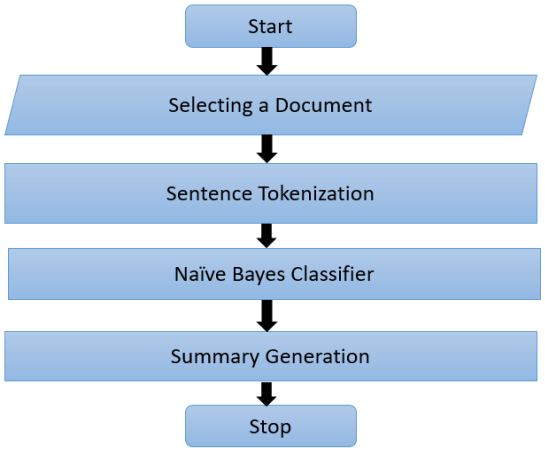
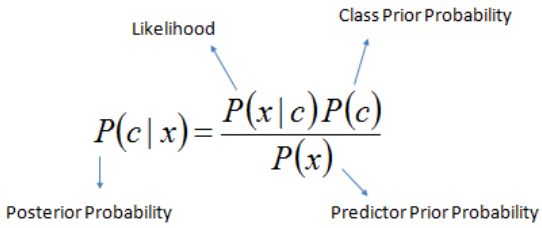


Figure 6: Flowchart of Testing Phase of NB

Figure 7, is the representation of basic Bayes Theorem on which Naïve Bayes is based. In this probability of "c" is calculated with respect to x which has already occur.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 7: Bayes Theorem [18]

Artificial Neural Network Model

Neural Network Model performs in a way where all the bug reports are taken in a single .csv file. And after performing ANN algorithm, bug report's summary is generated. The overall computation can be divided into two phase- Training Phase, Testing Phase. The corpus for input consisted of 32 bug reports.

In Figure 8, training phase is depicted, which follows steps like collecting and loading of data set. Then test pre-processing on the document is applied. After that feature extraction on sentences is performed by using GLOVE 100d. Then model is built and training for model is done on it.

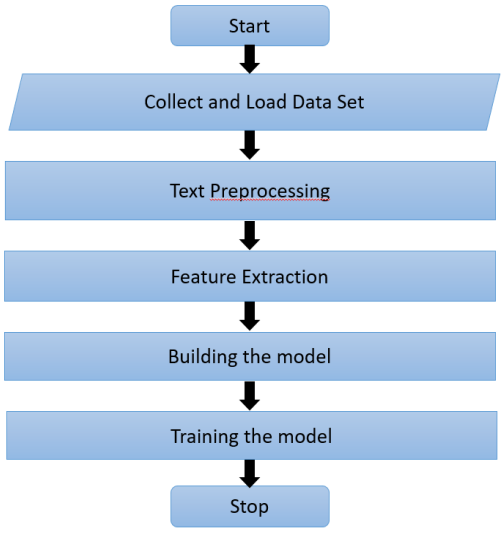


Figure 8, Flowchart of Training Phase of ANN

In figure 9, testing phase is depicted where text is prepared which needs to be applied on the model. Then model is applied and summaries are generated

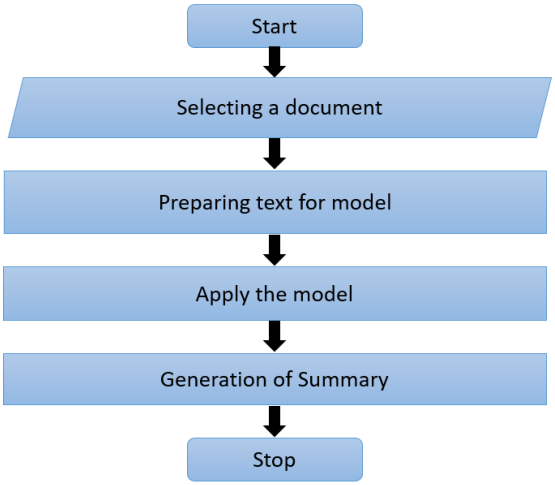


Figure 9: Flowchart of Testing Phase

Figure 10, is the general depiction of RNN model.

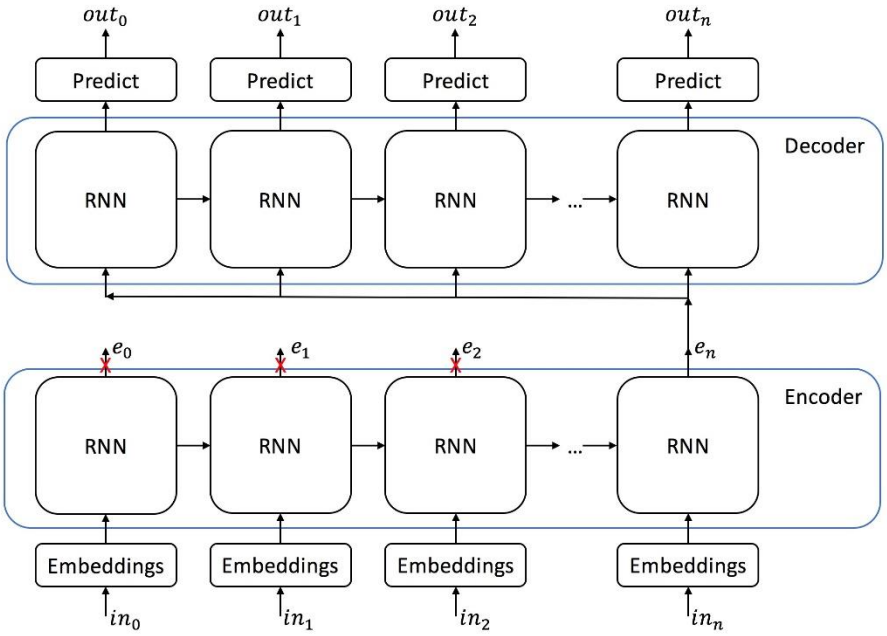


Figure 10: Neural Attention mechanism [19]

CHAPTER – 4

ALGORITHM

Text Summarization is the way toward making a short and sound adaptation of a more drawn out archive.

4.1 Naïve Bayes Model

4.1.1 Training Phase

In Training phase, algorithm is trained by using human generated summaries. Here humans according to their knowledge and logic creates a summary on which various training phase techniques are performed.

1. Collect and Load Data Set – A training data set is built of documents, D which is used by the algorithm to perform calculations. D= {d1, d2,…, d10}

2. Text Preprocessing – We clean the text by converting to lower, removing unwanted characters, replacing contractions by their lower forms and tokenization.

3. Using POS – POS tool is used to create the sets of nouns.

```
is_noun = lambda pos: pos[:2] == 'NN'
nouns = [word for (word, pos) in pos_tag(words) if is_noun(pos)]
```

Figure 11:Pos words

4. Human Generated Summary and Division of Classes – Bug Report summaries are generated by the help of humans. They will select most useful sentences to generate a summary according to their intelligence, knowledge and information given to them and save them in “class1”.Sentences that are not selected for summary are saved in “class0”.

5. Feature Extraction – Feature extraction is performed where score is given to each token by calculating word frequency for both “class1” and “class0”.

Given below, is the algorithm to be applied on training data set. This algorithm performs noun extraction and then calculating the score.

Input –

Class_1: tokens present in Human Generated Summary;

Class_0: tokens present in Human Generated not summary;

V: dictionary of nouns;

Output –

word_frequencies_insummary: list of words with their frequency in summary sentences;

word_frequencies_notinsummary: list of words with their frequency in not_summary sentences

Initialization

```
for k=1 to length(Class_1) do
    if (Class_1 [k] is noun) then
        if(Class_1[k] not in word_frequencies_insummary.keys()) then
            word_frequencies_insummary [word] = 1
        else
            word_frequencies_insummary [word] += 1
    else
        word_frequencies_insummary[word] =0
for k=1 to length(Class_0) do
    if (Class_0 [k] is noun) then
        if(Class_0[k] not in word_frequencies_notinsummary.keys()) then
            word_frequencies_notinsummary [word] = 1
        else
            word_frequencies_notinsummary [word] += 1
```

else

word_frequencies_notinsummary[word] =0

```
{'bugreport': 0, 'id': 10, '2': 0, 'title': 8, '449596': 0, 'firefox': 18, 'remove': 0, 'the'
: 0, 'browser': 15, 'sessionstore': 7, 'enabledpref': 0, 'that': 0, 'pref': 1, 'was': 0, 'tho
ught': 0, 'to': 0, 'be': 0, 'for': 0, 'extensions': 4, 'which': 0, 'wanted': 0, 'completely':
0, 'replace': 0, 'our': 0, 'own': 0, 'session': 3, 'restore': 2, 'functionality': 2, 'i': 30,
'would': 0, 'much': 0, 'rather': 0, 'encourage': 0, 'authors': 1, 'override': 0, 'both': 0, '
nssessionstartup': 0, 'and': 0, 'nssessionstore': 0, 'provide': 0, 'same': 0, 'api': 2, 'with
': 0, 'their': 0, 'or': 0, 'implementing': 0, 'a': 0, 'dummy': 0, 'making': 0, 'sure': 7, 'th
emselves': 0, 'they': 0, 'have': 0, 'correctly': 0, 'replaced': 0, 'all': 0, 'known': 0, 'con
sumers': 1, 'is': 0, 'what': 0, 'max': 2, 'tabs': 8, 'undo': 2, 'setting': 0, 'it': 0, '0': 0
, 'effectively': 0, 'disables': 0, 'feature': 1, 'gt': 22, 'others': 1, 'do': 10, 'not': 0, '
want': 0, 'any': 0, 'tracks': 1, 'stored': 0, 'in': 0, 'memory': 2, 'at': 0, 'then': 0, 'agai
n': 0, 'we': 0, 'save': 0, 'data': 3, 'anyway': 0, 'so': 0, 'resume': 3, 'from': 0, 'crash':
2, 'false': 2, 'fine': 0, 'now': 0, 'until': 0, 'proper': 0, 'private': 0, 'browsing': 1, 'su
pported': 0, 'xxxzenikoshould': 0, 't': 0, 'just': 0, 'disable': 0, 'this': 0, 'item': 1, 'as
': 0, 'tab': 3, 'multiple': 3, 'items': 1, 'above': 0, 'consistency': 1, 'mundoclosetabmenuit
em': 1, 'hidden': 0, 'cc': 1, 'mozilla': 18, 'org': 4, '1': 0, 'getservice': 1, 'ci': 1, 'nsi
sessionstore': 0, 'getclosedtabcount': 1, 'window': 17, 'r': 2, 'me': 0, 'bug': 14, '350731':
0, 'care': 1, 'convince': 0, 'mconnor': 1, 'he': 0, 'wrong': 0, 'pushed': 0, '17120': 0, 'e71
2e96d7861': 1, '17121': 0, 'adb1ef78dd21': 0, 'currently': 0, 'only': 0, 'one': 0, 'problem':
6, 'how': 0, 'after': 0, 'restart': 1, 'you': 0, 've': 0, 'got': 0, 'several': 0, 'options':
1, 'set': 0, 'prefsbrowser': 0, 'once': 0, 'early': 0, 'possible': 0, 'make': 0, 'startup': 1
, 'page': 14, '3': 0, 'delete': 0, 'file': 3, 'js': 2, 'e': 6, 'g': 4, 'when': 0, 'profile':
1, 'change': 5, 'notification': 2, 'dispatched': 0, 'respond': 1, 'state': 1, 'read': 0, 'by'
: 0, 'subject': 0, 's': 5, 'member': 1, 'an': 0, 'empty': 0, 'string': 4, 'cf': 1, '448741':
0, 'comment': 2, '6': 0, 'there': 0, 'use': 7, 'case': 3, 'am': 0, 'missing': 0, '491925': 0,
'multitouch': 0, 'rotate': 5, 'gesture': 10, 'cycling': 0, 'noticed': 0, 'frequently': 0, 'tr
igger': 2, 'accidentally': 0, 'usually': 0, 'while': 0, 'scrolling': 0, 'gestures': 3, 'impro
ved': 0, 'since': 0, 'original': 0, 'landing': 1, 'triggering': 0, 'really': 0, 'easy': 0, 'b
ut': 0, 'articular': 0, 'still': 0, 'problematic': 0, 'given': 0, 'should': 0, '5': 0, 'also'
: 0, 'open': 0, 'tweaking': 0, 'harder': 0, 'dunno': 0, 'if': 0, 'see': 5, '461376': 0, 'find
': 0, 'over': 0, 'osx': 4, 'wonder': 0, 'of': 0, 'us': 0, 'vs': 0, 'odd': 0, 'timings': 1, 'c
ausing': 0, 'misreads': 1, 'trackpad': 1, 'probably': 0, 'because': 0, 'zoom': 5, 'subtle': 0
, 'effect': 2, 'step': 1, 'relative': 1, 'distraction': 1, 'accidental': 0, 'switch': 1, 'cau
ses': 1, 'playing': 0, 'different': 0, 'twist': 3, 'settings': 1, 'found': 0, 'twisting': 0,
'through': 0, 'often': 0, 'resulted': 0, 'very': 0, 'awkward': 0, 'especially': 0, 'more': 0,
'than': 0, 'quarter': 1, 'turn': 1, 'been': 0, 'using': 0, 'swipe': 5, 'left': 7, 'right': 7,
'cmd': 1, 'go': 0, 'back': 5, 'forward': 2, 'well': 0, 'triggered': 0, 'maybe': 0, 'your': 0,
'rotating': 0, 'technique': 1, 'nice': 0, 'leave': 0, 'prefs': 2, 'visible': 0, 'those': 0, '
who': 0, 'them': 0, 'implemented': 0, 'on': 0, 'win7': 3, 'screens': 2, 'please': 2, 'windows
': 20, 'though': 0, 'bit': 1, 'wary': 0, 'having': 0, 'users': 3, 'hit': 0, 'trackpads': 1, '
7': 0, 'better': 0, 'discriminating': 0, 'between': 0, 'otoh': 0, 'big': 0, 'deal': 1, 'globa
lly': 0, 'safari': 0, 'does': 0, 'rob': 1, 'confirmed': 0, 'ie': 1, 'pinching': 2, 'natural':
0, 'widely': 0, 'used': 0, 'low': 0, 'penalty': 1, 'think': 0, 'most': 0, 'people': 1, 'consi
der': 0, 'claimed': 0, 'support': 2, 'multi': 2, 'touch': 0, 'pinch': 2, 'did': 0, 'anything'
```

Figure 12: Word Frequency of tokens in Class 1

{'bugreport': 0, 'id': 12, '2': 0, 'title': 6, '449596': 0, 'firefox': 11, 'remove': 0, 'the'
: 0, 'browser': 7, 'sessionstore': 6, 'enabledpref': 0, 'while': 0, 'this': 0, 'has': 0, 'wor
ked': 0, 'somehow': 0, 'for': 0, 'tab': 10, 'mix': 3, 'plus': 0, 'we': 0, 'have': 0, 'had': 0
, 'several': 0, 'issues': 4, 'with': 0, 'people': 4, 'ending': 0, 'up': 0, 'both': 0, 'sessio
n': 7, 'restore': 4, 'and': 0, 'disabled': 0, 'see': 3, 'bug': 9, '435055': 0, 'its': 0, 'dup
licates': 1, 'furthermore': 0, 'there': 0, 'are': 0, 'code': 3, 'points': 1, 'which': 0, 'wil
l': 0, 'also': 0, 'break': 0, 'when': 0, 'been': 0, 'such': 0, 'as': 0, 'list': 1, 'of': 0, '
recently': 0, 'closed': 0, 'tabs': 7, 'instead': 0, 'adding': 0, 'try': 3, 'catch': 1, 'block
s': 1, 'wherever': 0, 'use': 7, 'would': 0, 'make': 0, 'lives': 2, 'those': 0, 'other': 0, 'e
xtension': 5, 'authors': 2, 'simpler': 0, 'who': 0, 'so': 0, 'far': 0, 'can': 0, 'not': 0, 'b
e': 0, 'too': 0, 'sure': 5, 'that': 0, 'store': 1, 'component': 5, 'actually': 0, 'works': 0,
'through': 0, 'whatever': 0, 'implementation': 3, 'note': 3, 'privacy': 3, 'concerned': 0, 'u
sers': 6, 'still': 0, 'able': 0, 'to': 0, 'disable': 0, 'writing': 0, 'js': 1, 'resume': 1, '
from': 0, 'crash': 1, 'pref': 12, 'created': 0, 'an': 0, 'attachment': 0, '332726': 0, 'detai
ls': 10, '448725': 0, 'should': 0, 'wont': 0, 'fixed': 0, 'if': 0, 'is': 0, '333820': 0, 'bug
gy': 0, 'api': 6, 'comments': 1, 'update': 5, 'a': 0, 'problem': 5, 'patch': 7, 'data': 2, 's
tored': 0, 'in': 0, 'memory': 3, 'app': 1, 'running': 0, 'by': 0, 'removing': 0, 'no': 0, 'wa
y': 5, 'some': 0, 'do': 16, 'want': 0, 'menu': 3, 'others': 1, 'any': 0, 'tracks': 1, 'at': 0
, 'all': 0, 'gt': 11, 'i': 50, 'rather': 0, 'introduce': 0, 'different': 0, 'or': 0, 'means':
2, 'cater': 0, 'sensitive': 0, 'than': 0, 'half': 2, 'baked': 0, 'cut': 0, 'it': 0, 'afaict':
0, 'produced': 0, 'more': 0, 'solved': 0, 'then': 0, 'again': 0, 'save': 0, 'anyway': 0, 'fai
r': 0, 'point': 3, 'am': 0, 'yes': 0, 'agreed': 0, 'ideal': 0, 'purpose': 2, 'one': 0, 'men':
1, 'significantly': 0, 'affects': 0, 'disabling': 0, 'you': 0, 'now': 0, 'replace': 0, 'shipp
ing': 0, 'implements': 2, 'same': 0, 'keep': 0, 'minimal': 0, 'though': 0, 'just': 0, 'call':
1, 'your': 0, 'own': 0, 'whenever': 0, 'used': 0, 'using': 0, 'manager': 2, 'somewhat': 0, 't
hey': 0, 'without': 0, 'having': 0, 'worry': 0, 'much': 0, 'about': 0, 'behind': 0, 'offer':
0, 'option': 4, 'switching': 0, 'between': 0, 'our': 0, 'e': 5, 'g': 1, 'ignore': 1, 'history
': 16, 'torbutton': 1, 'how': 0, 'overwrite': 0, 'original': 0, 'being': 0, 'available': 0, '
internally': 0, 'either': 0, 'pass': 1, 'calls': 0, 'forward': 2, 'handle': 0, 'them': 0, 'yo
urself': 0, 'sounds': 0, 'like': 0, 'impact': 1, 'documented': 0, 'on': 0, 'mdc': 1, 'current
tabmix': 0, 'dev': 1, 'build': 1, 'already': 0, 'currently': 0, 'only': 0, 'after': 0, 'resta
rt': 1, 'add': 0, '3': 0, '491925': 0, 'multitouch': 0, 'rotate': 10, 'gesture': 13, 'cycling
': 0, 'basic': 0, 'reasoning': 1, 'highly': 0, 'disruptive': 0, 'switched': 0, 'another': 0,
'expecting': 0, 'happens': 0, 'know': 0, 'what': 0, 'happened': 0, 'until': 0, 'notice': 0, '
entirely': 0, 'page': 5, 'randomly': 0, 'left': 4, 'right': 9, '1': 0, 'thought': 0, 'were':
0, 'think': 0, 'nearly': 0, 'useful': 0, 'discoverable': 0, 'good': 0, 'fit': 1, 'gestures':
1, 'play': 0, 'around': 0, 'twist': 6, 'config': 1, 'temporary': 0, 'workaround': 1, 'justin'
: 0, 'odd': 0, 'thing': 1, 'here': 0, 'find': 0, 'trigger': 4, 'zoom': 3, 'frequently': 0, 'n
ever': 0, 'accidentally': 0, 'because': 0, 'probably': 0, 'decrease': 0, 'default': 3, 'thres
hold': 3, 'triggered': 0, 'repeatedly': 0, 'usually': 0, 'fix': 2, 'away': 0, 'but': 0, 'quic
k': 0, 'browse': 1, 'content': 1, 'prefs': 2, 'sqlite': 1, 'shows': 0, '20': 0, 'sites': 1, '
settings': 1, 'did': 0, 'set': 0, 'my': 0, 'fingers': 3, 'spread': 0, 'join': 0, 'ends': 0, '

Figure 13: Word frequency of tokens in Class 0

4.1.2 Testing Phase:

1. Selecting a document – We select a document from our dataset for which we have to generate summary.

2. Sentence tokenization – It is done on the selected document.

test_sentences = sent_tokenize(wholedoc)

3. Naïve Bayes Classifier – In this, Bayes Theorem is applied to calculate, probability of each sentence for both “class1” and “class0” is calculated by taking total sum of probability of each word in a sentence.

Input –

C: original text;

V: topic words (nouns);

word_frequencies_insummary: list of words with their frequency in summary sentences;

word_frequencies_notinsummary: list of words with their frequency in not_summary sentences;

n1: Number of total words in summary

n2: Number of total words in not summary

Output –

w_class1: sentences score for class1

w_class0: sentences score for class0

Initialization

```
for each sentence si in C do
    for j=1 to length(si) do
        if w(j) in V then
            if w(j) in word_frequencies_insummary then
                counts_class1[word]←word_frequencies_insummary[word]
            if w(j) in word_frequencies_notinsummary then
                counts_class0[word]←word_frequencies_notinsummary[word]
            m=m+1
            w_class1=w_class1+log((counts_class1[word]+1)/(n1+len(V)))
w_class0 = w_class0+log((counts_class0[word]+1)/(n2+len(V)))
    w_class1=w_class1+ log((len_class1/(len_class1+len_class0))) + log(m/len_class1)
+ log((1/i)/len_class1
     w_class0=w_class0 + log((len_class0/(len_class1+len_class0))) + log(m/len_class0)
+ log((1/i)/len_class0)
```

4. Summary Generation – Then, both probabilities of a sentence are compared and if probability for “class1” is greater than that for “class0”, that sentence is added to the final summary.

```
if(w_class1>w_class0):
    summary_sentences.append(sentence)
print(len(summary_sentences))
summary = ' '.join(summary_sentences)
```

print(summary)

4.2 Artificial Neural Network Model

4.2.1 Training Phase

In Training phase, algorithm is trained by using human generated summaries. Here humans according to their knowledge and logic creates a summary on which various training phase techniques are performed.

1. Collect and Load Data Set – A training data set is built of documents, D stored in a .csv file which is used by the algorithm to perform calculations. D= {d1, d2,…, d32}

2. Text Pre-processing –We clean the text and the human generated abstracted summaries by converting to lower, removing unwanted characters and replacing contractions by their lower forms.

3. Feature Extraction- In this vectorization is done. Vectorization is a process of converting string representation of data into vector representation. For this, we have used Global Vectors for Word representation (GLOVE) of 100d vector representation.

4. Building the Model-

```
def model_inputs():
    input_data = tf.placeholder(tf.int32, [None, None], name='input')
    targets = tf.placeholder(tf.int32, [None, None], name='targets')
    lr = tf.placeholder(tf.float32, name='learning_rate')
    keep_prob = tf.placeholder(tf.float32, name='keep_prob')
    summary_length = tf.placeholder(tf.int32, (None,), name='summary_length')
    max_summary_length = tf.reduce_max(summary_length, name='max_dec_len')
    text_length = tf.placeholder(tf.int32, (None,), name='text_length')
    return input_data, targets, lr, keep_prob, summary_length, max_summary_length, text_length
```

Figure 14: Model [20]

Now, we used Bidirectional Recurrent Neural Network(RNN) with Long Short-term Memory(LSTM) for building our encoding layer.

```
def encoding_layer(rnn_size, sequence_length, num_layers, rnn_inputs, keep_prob):
    '''Create the encoding layer'''
    layer_input = rnn_inputs
    for layer in range(num_layers):
        with tf.variable_scope('encoder_{}'.format(layer)): #for defining ops that creates variables
            cell_fw = tf.contrib.rnn.LSTMCell(rnn_size,
                                              initializer=tf.random_uniform_initializer(-0.1, 0.1, seed=2))
            cell_fw = tf.contrib.rnn.DropoutWrapper(cell_fw,
                                                    input_keep_prob = keep_prob)

            cell_bw = tf.contrib.rnn.LSTMCell(rnn_size,
                                              initializer=tf.random_uniform_initializer(-0.1, 0.1, seed=2))
            cell_bw = tf.contrib.rnn.DropoutWrapper(cell_bw,
                                                    input_keep_prob = keep_prob)

            enc_output, enc_state = tf.nn.bidirectional_dynamic_rnn(cell_fw,
                                                                    cell_bw,
                                                                    layer_input,
                                                                    sequence_length,
                                                                    dtype=tf.float32)
            layer_input = tf.concat(enc_output, 2)
    # Join outputs since we are using a bidirectional RNN
    enc_output = tf.concat(enc_output,2)
    return enc_output, enc_state
```

Figure 15: Encoding Layer [20]

Now, we built our training and inference decoding layers.

```
def training_decoding_layer(dec_embed_input, summary_length, dec_cell, output_layer,
                            vocab_size, max_summary_length,batch_size):
    training_helper = tf.contrib.seq2seq.TrainingHelper(inputs=dec_embed_input,
                                                        sequence_length=summary_length,
                                                        time_major=False)

    training_decoder = tf.contrib.seq2seq.BasicDecoder(cell=dec_cell,
                                                       helper=training_helper,
                                                       initial_state=dec_cell.zero_state(dtype=tf.float32, batch_size=batch_size),
                                                       output_layer = output_layer)

    training_logits = tf.contrib.seq2seq.dynamic_decode(training_decoder,
                                                        output_time_major=False,
                                                        impute_finished=True,
                                                        maximum_iterations=max_summary_length)
    return training_logits
```

Figure 16: Training Decoding Layer [20]

```
def inference_decoding_layer(embeddings, start_token, end_token, dec_cell, output_layer,
                             max_summary_length, batch_size):
    '''Create the inference logits'''

    start_tokens = tf.tile(tf.constant([start_token], dtype=tf.int32), [batch_size], name='start_tokens')

    inference_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(embeddings,
                                                                start_tokens,
                                                                end_token)

    inference_decoder = tf.contrib.seq2seq.BasicDecoder(dec_cell,
                                                        inference_helper,
                                                        dec_cell.zero_state(dtype=tf.float32, batch_size=batch_size),
                                                        output_layer)

    inference_logits = tf.contrib.seq2seq.dynamic_decode(inference_decoder,
                                                            output_time_major=False,
                                                            impute_finished=True,
                                                            maximum_iterations=max_summary_length)

    return inference_logits
```

Figure 17: Inference Decoding Layer [20]

Although it may seem that decoding layer is a bit complex, so it is preferable to break it down into 3 parts:

- Decoding cell – 2 layer LSTM with dropout
- Attention – It helps to train the model faster and to produce better results.
- Getting logits

```
def decoding_layer(dec_embed_input, embeddings, enc_output, enc_state, vocab_size, text_length, summary_length,
                   max_summary_length, rnn_size, vocab_to_int, keep_prob, batch_size, num_layers):
    '''Create the decoding cell and attention for the training and inference decoding layers'''
    dec_cell = tf.contrib.rnn.MultiRNNCell([lstm_cell(rnn_size, keep_prob) for _ in range(num_layers)])
    output_layer = Dense(vocab_size,kernel_initializer=tf.truncated_normal_initializer(mean=0.0, stddev=0.1))
    attn_mech = tf.contrib.seq2seq.BahdanauAttention(rnn_size,
                                                     enc_output,
                                                     text_length,
                                                     normalize=False,
                                                     name='BahdanauAttention')
    dec_cell = tf.contrib.seq2seq.AttentionWrapper(dec_cell,attn_mech,rnn_size)
    with tf.variable_scope("decode"):
        training_logits = training_decoding_layer(dec_embed_input,summary_length,dec_cell,
                                                  output_layer,
                                                  vocab_size,
                                                  max_summary_length,
                                                  batch_size)
    with tf.variable_scope("decode", reuse=True):
        inference_logits = inference_decoding_layer(embeddings,
                                                    vocab_to_int['<GO>'],
                                                    vocab_to_int['<EOS>'],
                                                    dec_cell,
                                                    output_layer,
                                                    max_summary_length,
                                                    batch_size)
    return training_logits, inference_logits
```

Figure 18: Decoding Layer [20]

Now, the previous functions are ready to be used to build the model.

```
def seq2seq_model(input_data, target_data, keep_prob, text_length, summary_length, max_summary_length,
                  vocab_size, rnn_size, num_layers, vocab_to_int, batch_size):
    '''Use the previous functions to create the training and inference logits'''

    # Use Glove's embeddings and the newly created ones as our embeddings
    embeddings = word_embedding_matrix
    enc_embed_input = tf.nn.embedding_lookup(embeddings, input_data)
    print(enc_embed_input)
    enc_output, enc_state = encoding_layer(rnn_size, text_length, num_layers, enc_embed_input, keep_prob)
    dec_input = process_encoding_input(target_data, vocab_to_int, batch_size) #shape=(batch_size, senquence length) each seq start with index of<GO>
    dec_embed_input = tf.nn.embedding_lookup(embeddings, dec_input)
    training_logits, inference_logits  = decoding_layer(dec_embed_input,
                                                        embeddings,
                                                        enc_output,
                                                        enc_state,
                                                        vocab_size,
                                                        text_length,
                                                        summary_length,
                                                        max_summary_length,
                                                        rnn_size,
                                                        vocab_to_int,
                                                        keep_prob,
                                                        batch_size,
                                                        num_layers)
    return training_logits, inference_logits
```

Figure 19: Seq2seq Model [20]

5. Training the model

The hyperparameters that we used to train our model.

```
# Set the Hyperparameters
epochs = 50
batch_size = 5
rnn_size = 256
num_layers = 2
learning_rate = 0.005
keep_probability = 0.95
```

Figure 20: Hyperparameters [20]

```
# Train the Model
learning_rate_decay = 0.95
min_learning_rate = 0.0005
display_step = 20 # Check training loss after every 20 batches
stop_early = 0
stop = 10 # If the update loss does not decrease in 20 consecutive update checks, stop training
per_epoch = 1 # Make 1 update checks per epoch
update_check = (len(sorted_texts_short)//batch_size//per_epoch)-1

update_loss = 0
batch_loss = 0
summary_update_loss = [] # Record the update losses for saving improvements in the model

checkpoint = "./output/model.ckpt"
with tf.Session(graph=train_graph) as sess:
    sess.run(tf.global_variables_initializer())

    # If we want to continue training a previous session
    #loader = tf.train.import_meta_graph("./" + checkpoint + '.meta')
    #loader.restore(sess, checkpoint)

    for epoch_i in range(1, epochs+1):
        update_loss = 0
        batch_loss = 0
        for batch_i, (summaries_batch, texts_batch, summaries_lengths, texts_lengths) in enumerate(
                get_batches(sorted_summaries_short, sorted_texts_short, batch_size)):
            start_time = time.time()
            _, loss = sess.run(
                [train_op, cost],
                {input_data: texts_batch,
                 targets: summaries_batch,
                 lr: learning_rate,
                 summary_length: summaries_lengths,
                 text_length: texts_lengths,
                 keep_prob: keep_probability})
```

```
        batch_loss += loss
        update_loss += loss
        end_time = time.time()
        batch_time = end_time - start_time
        if batch_i % display_step == 0 and batch_i > 0:
            print('Epoch {:>3}/{} Batch {:>4}/{} - Loss: {:>6.3f}, Seconds: {:>4.2f}'
                  .format(epoch_i,
                          epochs,
                          batch_i,
                          len(sorted_texts_short) // batch_size,
                          batch_loss / display_step,
                          batch_time*display_step))
            batch_loss = 0

        if batch_i % update_check == 0 and batch_i > 0:
            print("Average loss for this update:", round(update_loss/update_check,3))
            summary_update_loss.append(update_loss)

            # If the update loss is at a new minimum, save the model
            if update_loss <= min(summary_update_loss):
                print('New Record!')
                stop_early = 0
                saver = tf.train.Saver()
                saver.save(sess, checkpoint)


            else:
                print("No Improvement.")
                stop_early += 1
                if stop_early == stop:
                    break
            update_loss = 0


    # Reduce learning rate, but not below its minimum value
    learning_rate *= learning_rate_decay
    if learning_rate < min_learning_rate:
        learning_rate = min_learning_rate

    if stop_early == stop:
        print("Stopping Training.")
        break
```

Figure 21: Algorithm of Training Model [20]

4.2.2 Testing Phase

1. Selecting a document – We select a document from our dataset for which we have to generate summary.

2. Prepare the text for the model- We converted “string representation” to “integer representation”.

3. Apply the model

```
# Load saved model
    loader = tf.train.import_meta_graph(checkpoint + '.meta')
    loader.restore(sess, checkpoint)

    input_data = loaded_graph.get_tensor_by_name('input:0')
    logits = loaded_graph.get_tensor_by_name('predictions:0')
    text_length = loaded_graph.get_tensor_by_name('text_length:0')
    summary_length = loaded_graph.get_tensor_by_name('summary_length:0')
    keep_prob = loaded_graph.get_tensor_by_name('keep_prob:0')

    #Multiply by batch_size to match the model's input parameters
    answer_logits = sess.run(logits, {input_data: [text]*batch_size,
                                      summary_length: [np.random.randint(10,150)],
                                      text_length: [len(text)]*batch_size,
                                      keep_prob: 1.0})[0]
```

Figure 22: Algorithm of applying the model [20]

4. Generation of summary

CHAPTER -5
TEST PLAN

Testing is a process to check and evaluate whether the results obtained from any model are accurate enough in comparison with the expected results. It also detects errors on running the model with different number of test cases to check whether how stable a model is. It also involves execution of system components to evaluate needed properties.

Software testing can be done in either of the two ways- White Box Testing or Black Box Testing.

Different Levels of testing that can be applied are:

1. Unit Testing- During this phase, the program is assessed on specific units or components involved in the system. In this white-box testing is usually applied to do unit testing.
2. Integration Testing- In this phase, it allows the tester to group all the units together and then test them as one. It is performed to find the interface/links defect between the different functions. It is beneficial to calculate the effectiveness of all the units when run together.
3. System Testing- In this phase, whole code is executed. Its aim is to check whether the complete program is able to run all the requirements of the user without any error or faults. It is very important as it checks all the functional, non-functional, technical requirements of the user.
4. Acceptance Testing- This is this last phase of testing. Its main aim is to verify that whether the software is ready to be released for the use of user or not. In this phase, the user gets to check the model. At last validation is done, where all the test and levels are performed.

Limitations-

- The main limitation of Naïve Bayes classifier is that it assumes any two features are independent of each other. But in real life, it is not possible.
- The main limitation of ANN is that it is hardware dependent which means it requires processors with parallel processing power.

Test Data Set-

$$T = \{ W_{1,1}\ W_{1,2} \ldots\ldots W_{1,n}\}$$
$$\{ W_{2,1}\ W_{2,2} \ldots\ldots W_{2,n}\}$$
$$\{ W_{m,1}\ W_{m,2} \ldots\ldots W_{m,n}\}$$

This is the matrix of text representation. In this form sentences or words appear after sentence tokenization is applied.

Bug Report ID="1"

Title: "(495584) Firefox - search suggestions passes wrong previous result to form history"

MattN noticed a problem with the WIP patch from bug 469443 applied.
When typing in the search box, sometimes search-suggestion entries would be displayed above the divider (where entries for previous matching searches are).The problem here is that nsSearchSuggestions.js is passing the wrong previousResult to form history. Instead of it being the previous form history search result, it's the SuggestAutoCompleteResult result (which contains the union of the form-history and search-suggest entries).
So, when form history refines its results as you time, it can actually add *more* entries as data leaks from the suggestions result into form history result, and it thus looks like the divider is being drawn in the wrong place. This bug wasn't visible before 469443, because nsFormFillController::StartSearch tries to QI the provided result to a nsIAutoCompleteSimpleResult. The search-suggestion result is only implements nsIAutoCompletResult (no \"Simple\"), so the QI fails, historyResult nee previousResult becomes null, and thus Satchel was doing a new search every time. form history finds 1 entry (\"blah\"), search-suggestions finds \"baaa\", \"bloop\", \"bzzz\", the autocompete menu shows these in order with a divider between \"blah\" and \"baaa\". startHistorySearch()'s previous result contains [\"blah\", \"baaa\", \"bloop\", \"bzzz\"], Satchel filters this down to [\"blah\", \"bloop\"] to match the new \"bl\" search string nsSearchSuggestions's onReadyState() change is called with updated search suggestions, builds up a new list of results, but sees that the form history result now has *two* entries. Created an attachment (id=380567) [details] Patch v.1 (WIP) this fixes the problem, but isn't quite correct... If you type \"a<backspace>b\", satchel trying to use the results from the \"a\" search for the \"b\" search, and so nothing is found. I suspect nsSearchSuggestions needs to throw away the old form history result when the search string changes like this, but I'm not entirely sure it's responsible for doing so, maybe satchel should be smarter about throwing away a previous result when the previous result's search string doesn't have a common prefix. That seems to be handled somewhere else for normal form field entries, oddly enough. Created an attachment (id=383211) [details] Patch v.2 Ah. So, there's a ._formHistoryResult in the SuggestAutoCompleteResult wrapper (used to combine form history with search suggestions), and also a ._formHistoryResult in SuggestAutoComplete (the service itself, used to hold onto a form history result until a search suggestion is available).The simple fix it to just discard the service's form history result copy when startSearch() is called with a null previous result. Otherwise it's trying to use a old form history result that no longer applies for the search string. (From update of attachment 383211 [details]). Perhaps we should rename one of them to _fhResult just to reduce confusion? (From update of attachment 383211 [details] [details]) Perhaps we should rename one of them to _fhResult just to reduce confusion? Good point. I renamed the one in the wrapper to _formHistResult. fhResult seemed maybe a bit too short.Pushed http://hg.mozilla.org/mozilla-central/rev/097598383614

Figure 23: Doc 1 for NB [6]

Bug Report ID="13"
Title: "(364852) GIMP - screenshot plug-in incomplete for Windows platform"

Please describe the problem: When going to File-> Acquire-> Screenshot in the Windows port of the development build GIMP 2.3.11, firstly there are two different entries \"Screen Shot\" and \"Screenshot\" and secondly, functionality to grab only a portion of the screen is broken. There is no option to \"Select a Region to Grab\" when the user chooses \"Screen Shot.\" However, this option exists in the dialogue box that appears after the user selects: Screenshot.\" However, the functionality does not work, even after selecting a long delay period. Instead, the whole screen is captured each time. I was expecting the mouse cursor onscreen to change to \"crosshairs\" or something similar to allow me to select the region I required. Steps to reproduce:
1. Launch GIMP 2.3.11 Development for Windows on a Windows 2000 platform.
2. Go to File-> Acquire-> Screenshot. (Note the two different entries for \"Screen Shot\" and \"Screenshot\".)
3. Tick \"Select a Region to Grab.\"
4. Set the delay.
5. Click \"Grab.\"
Actual results: The whole screen is captured each time. Expected results: The ability to select a portion of the screen would be available, probably via mouse click-and-drag functionality. Does this happen every time? Yes. Other information: (I presume the two entries for screenshots under the submenu are there in order to test old and new code, by the way but I just noted it in case there was an issue.); D. Le Sage; APF; Hobart, Tasmania. We know that the port of the screenshot plug-in for Windows is incomplete. What's this bug-report about? Do you want to contribute the missing bits or do you want us to disable the plug-in for the Windows platform? We are waiting for a volunteer to implement the missing functionality for more than a year now. Perhaps it's about time to drop support for the Win32 platform completely. The bug report was to alert you in case you did not know it was broken. You are obviously aware it is incomplete. Perhaps you should disable it to prevent more questions from users. I am not a programmer so I cannot implement the functionality. I am trying to do my part by reporting issues for you. You have complained before about how short of volunteers you are. If this is the case, perhaps it is a very good idea to drop support for Win32 and consolidate on making sure the *Nix implementation is as good as possible. Supporting Win32 is probably just draining more resources away. IIRC Tor is working on this. Apt, you are using a development version. Of course there are incomplete and even broken features in there. If you are just a user, you should be using the stable release. Tor said that it is impossible to implement the look and feel of the X11 screenshot plug-in on the Windows platform. He suggested that we stick with Win snap for the Win32. 2006-10-27 Sven Neumann <sven@gimp.org> * configure.in; * plug-ins/common/plugin-defs.pl; * plug-ins/common/Makefile.am: reverted changes from 2006-04-06 and build screenshot plug-in optionally again. Fixes bug #364852. * plug-ins/common/mkgen.pl: reverted addition of 'extralibs' parameter that was introduced for the screenshot plug-in only.

Figure 24: Doc 2 for NB [6]

Bug Report ID="17"
Title: "(514396) GTK - Add gtk_show_uri"

Currently apps use gnome help functions to show help files, but gnome_help is part of the deprecated gnome help. See also GNOME goal proposal http://live.gnome.org /action/show/Gnome Goals /RemoveGnomeOpenGnomeHelp Created an attachment (id=105668) [details] Patch adding gtk_show_uri and gtk_show_help to GTK+, The gtk_show_help function is missing |return ret;|. You forgot to add the new funcs to gtk/gtk.symbols.

+ * modify it under the terms of the GNU Lesser General Public

+ * License as published by the Free Software Foundation; either

+ * version 2 of the License, or (at your option) any later version.

There is no \"Lesser GPL version 2\". It's the Lesser GPL version 2.1 (or the Library GPL version 2). Somehow I feel that gtk_show_help () is too GNOME oriented to be included in this form. I have been told that gtk_show_uri () will work nicely on the major supported platforms. This is however not the case for gtk_show_help (). Since the functionality of gtk_show_help () can be easily implemented on top of gtk_show_uri(), it should be sufficient to provide gtk_show_uri() for now. Created an attachment (id=105671) [details]. Updated patch. Adding ret and gtk symbols. Christian, I copied the license from other files in GTK+. I opened about 5 to 6 files and they are the same. I guess it's a better idea to change all licenses in one go. What's the consensus on this? Can I commit gtkshow.c to gtk+ when I remove gtk_show_help? Update your patch to remove gtk_show_help() to start with. Created an attachment (id=106302) [details] Patch removing gtk_show_help. Can one of the maintainers decided what to do with this patch? Thanks, gtk_show_uri allows a NULL parent, but that is not multi-head safe. Why not take a mandatory GdkScreen argument since the function couldn't care less about the parent widget beyond its screen? I can make it such that the parent has to be set mandatory. Having a GtkWidget *parent saves calling a gtk_widget_get_screen everytime you do a gtk_show_uri What's the opinion on this of others? I don't think the parent should be mandatory, because there are situations, where they may not be a parent at all. A mandatory parent would be bad, as Ross explained in comment #11. IMO the best solution is to change the function to take a GdkScreen

Figure 25: Doc 1 for ANN [6]

CHAPTER -6

RESULT AND PERFORMANCE ANALYSIS

6.1 Result and Performance Analysis for Naïve Bayes Model

For analysis of results and performance we have taken 2 documents here for which we will generate the summary.

Figure 26, Figure 27 are sentence tokenization are for Doc 1, Doc 2 respectively.

```
['MattN noticed a problem with the WIP patch from bug 469443 applied.When typing in the searc
h box, sometimes search-suggestion entries would be displayed above the divider (where entrie
s for previous matching searches are).The problem here is that nsSearchSuggestions.js is pass
ing the wrong previousResult to form history.', "Instead of it being the previous form histor
y search result, it's the SuggestAutoCompleteResult result (which contains the union of the f
orm-history and search-suggest entries).", 'So, when form history refines its results as you
time, it can actually add *more* entries as data leaks from the suggestions result into form
history result, and it thus looks like the divider is being drawn in the wrong place.', "This
bug wasn't visible before 469443, because nsFormFillController::StartSearch tries to QI the p
rovided result to a nsIAutoCompleteSimpleResult.", 'The search-suggestion result is only impl
ements nsIAutoCompletResult (no \\"Simple\\"), so the QI fails, historyResult nee previousRes
ult becomes null, and thus Satchel was doing a new search every time.', 'form history finds 1
entry (\\"blah\\"), search-suggestions finds \\"baaa\\", \\"bloop\\", \\"bzzz\\", the autocom
pete menu shows these in order with a divider between \\"blah\\" and \\"baaa\\".', 'startHist
orySearch()\'s previous result contains [\\"blah\\", \\"baaa\\", \\"bloop\\", \\"bzzz\\"], Sa
tchel filters this down to [\\"blah\\", \\"bloop\\"] to match the new \\"bl\\" search string
nsSearchSuggestions\'s onReadyState() change is called with updated search suggestions, build
s up a new list of results, but sees that the form history result now has *two* entries.', "C
reated an attachment (id=380567) [details] Patch v.1 (WIP) This fixes the problem, but isn't
quite correct...", 'If you type \\"a&lt;backspace&gt;b\\", satchel trying to use the results
from the \\"a\\" search for the \\"b\\" search, and so nothing is found.', "I suspect nsSearc
hSuggestions needs to throw away the old form history result when the search string changes l
ike this, but I'm not entirely sure it's responsible for doing so, maybe satchel should be sm
arter about throwing away a previous result when the previous result's search string doesn't
have a common prefix.", 'That seems to be handled somewhere else for normal form field entrie
s, oddly enough.', 'Created an attachment (id=383211) [details] Patch v.2 Ah.', "So, there's
a ._formHistoryResult in the SuggestAutoCompleteResult wrapper (used to combine form history
with search suggestions), and also a ._formHistoryResult in  SuggestAutoComplete (the service
itself, used to hold onto a form history result until a search suggestion is available).The s
imple fix it to just discard the service's form history result copy when startSearch() is cal
led with a null previous result."]
```

Figure 26: Sentence Tokenization of Doc 1

['BugReport\xa0ID="13"Title : "(364852) GIMP - screenshot plug-in incomplete for
 Windows platform"Please describe the problem: When going to File->Acquire-&g
t;Screenshot in the Windows port of the development build GIMP 2.3.11, firstly t
here are two different entries \\"Screen Shot\\" and \\"Screenshot\\" and second
ly, functionality to grab only a portion of the screen is broken.', 'There is no
 option to \\"Select a Region to Grab\\" when the user chooses \\"Screen Shot.\\
" However, this option exists in the dialogue box that appears after the user se
lects :Screenshot.\\" However, the functionality does not work, even after selec
ting a long delay period.', 'Instead, the whole screen is captured each time.',
'I was expecting the mouse cursor onscreen to change to \\"crosshairs\\" or some
thing similar to allow me to select the region I required.', 'Steps to reproduce
: 1.', 'Launch GIMP 2.3.11 Development for Windows on a Windows 2000 platform.2.
', 'Got o File->Acquire->Screenshot.', '(Note the two different entries fo
r \\"Screen Shot\\" and \\"Screenshot\\".)3.', 'Tick \\"Select a Region to Grab.
\\"4.', 'Set the delay.5.', 'Click \\"Grab.\\"Actual results: The whole screen i
s captured each time.', 'Expected results: The ability to select a portion of th
e screen would be available, probably via mouse click-and-drag functionality.',
'Does this happen every time?', 'Yes.', 'Other information: (I presume the two
entries for screenshots under the submenu are there in order to test old and new
 code, by the way but I just noted it in case there was an issue.', '); D. Le Sa
ge; APF; Hobart, Tasmania.', 'We know that the port of the screenshot plug-in fo
r Windows is incomplete.', "What's this bug-report about?", 'Do you want to cont
ribute the missing bits or do you want us to disable the plug-in for the Windows
 platform?', 'We are waiting for a volunteer to implement the missing functional
ity for more than a year now.', "Perhaps it's about time to drop support for the
 Win32 platform completely.", 'The bug report was to alert you in case you did n
ot know it was broken.', 'You are obviously aware it is incomplete.', 'Perhaps y
ou should disable it to prevent more questions from users.', 'I am not a program
mer so I cannot implement the functionality.', 'I am trying to do my part by rep
orting issues for you.', 'You have complained before about how short of voluntee
rs you are.', 'If this is the case, perhaps it is a very good idea to drop suppo
rt for Win32 and consolidate on making sure the *Nix implementation is as good a
s possible.', 'Supporting Win32 is probably just draining more resources away.',
 'IIRC Tor is working on this.', 'apf, you are using a development version.', 'O
f course there are incomplete and even broken features in there.', 'If you are j
ust a user, you should be using the stable release.', 'Tor said that it is impos
sible to implement the look and feel of the X11 screenshot plug-in on the Window
s platform.', 'He suggested that we stick with Winsnap for the Win32.', '2006-10
-27 Sven Neumann <sven@gimp.org> * configure.in; * plug-ins/common/plugin
-defs.pl; * plug-ins/common/Makefile.am: reverted changes from 2006-04-06 and bu
ild screenshot plug-in optionally again.', 'Fixes bug #364852.', "* plug-ins/com
mon/mkgen.pl: reverted addition of 'extralibs' parameter that was introduced for
 the screenshot plug-in only."]

Figure 27: Sentence Tokenization of Doc 2

Figure 28, shows the sentence score for sentences of Doc 1, Doc 2.

```
Sentence 1 Score for Class1: -65.71859111334032
Sentence 1 Score for Class0: -69.08405345168107


Sentence 2 Score for Class1: -40.61796646558207
Sentence 2 Score for Class0: -44.70268399990956


Sentence 3 Score for Class1: -69.41357765279335
Sentence 3 Score for Class0: -75.63904768604564


Sentence 4 Score for Class1: -17.754646852787705
Sentence 4 Score for Class0: -19.113887508944018


Sentence 5 Score for Class1: -25.64658176059062
Sentence 5 Score for Class0: -25.797051429786002


Sentence 6 Score for Class1: -46.12458087532733
Sentence 6 Score for Class0: -44.17861314939356


Sentence 7 Score for Class1: -76.01012225540286
Sentence 7 Score for Class0: -84.52786401832607


Sentence 8 Score for Class1: -8.070906088787817
Sentence 8 Score for Class0: -8.070906088787817


Sentence 9 Score for Class1: -45.87991618795924
Sentence 9 Score for Class0: -44.83230149321295


Sentence 10 Score for Class1: -74.56765762281593
Sentence 10 Score for Class0: -81.28121739651249
```

```
Sentence 11 Score for Class1: -20.123094924259775
Sentence 11 Score for Class0: -20.67140536419976


Sentence 12 Score for Class1: -8.13989896027477
Sentence 12 Score for Class0: -8.13989896027477


Sentence 13 Score for Class1: -96.1329072927109
Sentence 13 Score for Class0: -103.48204062906382
```

Figure 28: Sentence Score of Doc 1

```
Sentence 1 Score for Class1: -74.87346228146015
Sentence 1 Score for Class0: -81.23684335807908

Sentence 2 Score for Class1: -63.379627642177
Sentence 2 Score for Class0: -62.598146504551394

Sentence 3 Score for Class1: -15.49786273788889
Sentence 3 Score for Class0: -15.722124765532966

Sentence 4 Score for Class1: -59.977342417056995
Sentence 4 Score for Class0: -57.37573366100509

Sentence 5 Score for Class1: -9.356436724596655
Sentence 5 Score for Class0: -9.435574045155379

Sentence 6 Score for Class1: -9.53875828139061
Sentence 6 Score for Class0: -9.617895601949332

Sentence 7 Score for Class1: -9.692908961217867
Sentence 7 Score for Class0: -9.772046281776593

Sentence 8 Score for Class1: -15.839612031610944
Sentence 8 Score for Class0: -16.862381755472796

Sentence 9 Score for Class1: -9.906483061515926
Sentence 9 Score for Class0: -9.98562038207465

Sentence 10 Score for Class1: -10.011843577173753
Sentence 10 Score for Class0: -10.090980897732477

Sentence 11 Score for Class1: -16.409380191010385
Sentence 11 Score for Class0: -16.633642218654465

Sentence 12 Score for Class1: -44.86451865986942
Sentence 12 Score for Class0: -46.038605577347205

Sentence 13 Score for Class1: -10.073537146179092
Sentence 13 Score for Class0: -10.152674466737817

Sentence 14 Score for Class1: -10.147645118332814
Sentence 14 Score for Class0: -10.22678243889154

Sentence 15 Score for Class1: -60.27233190884057
Sentence 15 Score for Class0: -58.096807548099555

Sentence 16 Score for Class1: -10.088804618309881
Sentence 16 Score for Class0: -10.167941938868605

Sentence 17 Score for Class1: -23.06550930481503
Sentence 17 Score for Class0: -24.756651879526785

Sentence 18 Score for Class1: -10.157797489796831
Sentence 18 Score for Class0: -10.236934810355557

Sentence 19 Score for Class1: -23.861102432833157
Sentence 19 Score for Class0: -22.405443720701065

Sentence 20 Score for Class1: -32.932302751887576
Sentence 20 Score for Class0: -31.736698713514826
```

```
Sentence 21 Score for Class1: -30.146628790099143
Sentence 21 Score for Class0: -29.526388896629953


Sentence 22 Score for Class1: -29.04463585861247
Sentence 22 Score for Class0: -27.71022941735925


Sentence 23 Score for Class1: -10.170297652561064
Sentence 23 Score for Class0: -10.249434973119788


Sentence 24 Score for Class1: -18.324518681398946
Sentence 24 Score for Class0: -17.673311971689124


Sentence 25 Score for Class1: -18.346991537251007
Sentence 25 Score for Class0: -17.695784827541182


Sentence 26 Score for Class1: -30.237100693263223
Sentence 26 Score for Class0: -28.338780023314367


Sentence 27 Score for Class1: -18.353748319713883
Sentence 27 Score for Class0: -17.70254161000406


Sentence 28 Score for Class1: -44.76175892591165
Sentence 28 Score for Class0: -45.060377106035546


Sentence 29 Score for Class1: -17.229745169038004
Sentence 29 Score for Class0: -18.370297928556234


Sentence 30 Score for Class1: -10.231905461950555
Sentence 30 Score for Class0: -10.31104278250928
```

```
Sentence 31 Score for Class1: -16.993486998954104
Sentence 31 Score for Class0: -18.42172183092412


Sentence 32 Score for Class1: -32.83602592145216
Sentence 32 Score for Class0: -31.82274343987336


Sentence 33 Score for Class1: -10.267496407053256
Sentence 33 Score for Class0: -10.34663372761198


Sentence 34 Score for Class1: -30.656866351513578
Sentence 34 Score for Class0: -31.90534696840857


Sentence 35 Score for Class1: -10.283777292657394
Sentence 35 Score for Class0: -10.362914613216118


Sentence 36 Score for Class1: -29.46922996650734
Sentence 36 Score for Class0: -31.63400131527648


Sentence 37 Score for Class1: -15.58242012591695
Sentence 37 Score for Class0: -15.238698115955092


Sentence 38 Score for Class1: -32.46706972240089
Sentence 38 Score for Class0: -31.636108797616046
```

Figure 29: Sentence Score of Doc 2

With the proposed method we built the Naïve Bayes Classifier Model which when performed on the different bug reports generated following results.

After summary is generate by our system, we have calculated precision, recall and f-score for comparing system generated summary and human generated summary. This will tell how accurate and precise our model is.

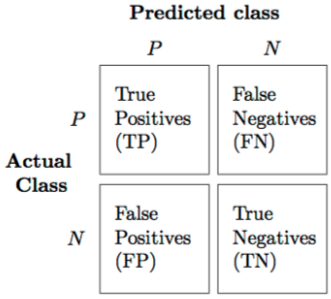


Figure 30: Confusion Matrix for Precision, Recall, F-Score [21]

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Figure 31: Formulas for Precision, Recall, F-score [22]

Figure 32, Figure 33 shows the summary generation and performance analysis

MattN noticed a problem with the WIP patch from bug 469443 applied.When typing i
n the search box, sometimes search-suggestion entries would be displayed above t
he divider (where entries for previous matching searches are).The problem here i
s that nsSearchSuggestions.js is passing the wrong previousResult to form histor
y. Instead of it being the previous form history search result, it's the Suggest
AutoCompleteResult result (which contains the union of the form-history and sear
ch-suggest entries). So, when form history refines its results as you time, it c
an actually add *more* entries as data leaks from the suggestions result into fo
rm history result, and it thus looks like the divider is being drawn in the wron
g place. This bug wasn't visible before 469443, because nsFormFillController::St
artSearch tries to QI the provided result to a nsIAutoCompleteSimpleResult. The
search-suggestion result is only implements nsIAutoCompletResult (no \"Simple\")
, so the QI fails, historyResult nee previousResult becomes null, and thus Satch
el was doing a new search every time. startHistorySearch()'s previous result con
tains [\"blah\", \"baaa\", \"bloop\", \"bzzz\"], Satchel filters this down to [\
"blah\", \"bloop\"] to match the new \"bl\" search string nsSearchSuggestions's
onReadyState() change is called with updated search suggestions, builds up a new
 list of results, but sees that the form history result now has *two* entries. I
f you type \"a<backspace>b\", satchel trying to use the results from the \
"a\" search for the \"b\" search, and so nothing is found. I suspect nsSearchSug
gestions needs to throw away the old form history result when the search string
changes like this, but I'm not entirely sure it's responsible for doing so, mayb
e satchel should be smarter about throwing away a previous result when the previ
ous result's search string doesn't have a common prefix. That seems to be handle
d somewhere else for normal form field entries, oddly enough. Created an attachm
ent (id=383211) [details] Patch v.2 Ah. So, there's a ._formHistoryResult in the
 SuggestAutoCompleteResult wrapper (used to combine form history with search sug
gestions), and also a ._formHistoryResult in SuggestAutoComplete (the service i
tself, used to hold onto a form history result until a search suggestion is avai
lable).The simple fix it to just discard the service's form history result copy
when startSearch() is called with a null previous result.
Precision: 63.63636363636363
Recall: 70.0
F-score: 66.66666666666667

Figure 32: Summary Generation of Doc 1

Please describe the problem: When going to File->Acquire->Screenshot in th
e Windows port of the development build GIMP 2.3.11, firstly there are two diffe
rent entries \"Screen Shot\" and \"Screenshot\" and secondly, functionality to g
rab only a portion of the screen is broken. Instead, the whole screen is capture
d each time. Steps to reproduce: 1. Launch GIMP 2.3.11 Development for Windows o
n a Windows 2000 platform.2. Got o File->Acquire->Screenshot. (Note the tw
o different entries for \"Screen Shot\" and \"Screenshot\".)3. Tick \"Select a R
egion to Grab.\"4. Set the delay.5. Click \"Grab.\"Actual results: The whole scr
een is captured each time. Expected results: The ability to select a portion of
the screen would be available, probably via mouse click-and-drag functionality.
Does this happen every time? Yes.); D. Le Sage; APF; Hobart, Tasmania. We know
that the port of the screenshot plug-in for Windows is incomplete. What's this b
ug-report about? You are obviously aware it is incomplete. If this is the case,
perhaps it is a very good idea to drop support for Win32 and consolidate on maki
ng sure the *Nix implementation is as good as possible. Supporting Win32 is prob
ably just draining more resources away. IIRC Tor is working on this. apf, you ar
e using a development version. If you are just a user, you should be using the s
table release. Tor said that it is impossible to implement the look and feel of
the X11 screenshot plug-in on the Windows platform. He suggested that we stick w
ith Winsnap for the Win32. 2006-10-27 Sven Neumann <sven@gimp.org> * confi
gure.in; * plug-ins/common/plugin-defs.pl; * plug-ins/common/Makefile.am: rever
ted changes from 2006-04-06 and build screenshot plug-in optionally again.
Precision: 66.66666666666666
Recall: 94.11764705882352
F-score: 78.04878048780488

Figure 33: Summary Generation of Doc 2

6.2 Result and Performance Analysis for Neural Network Model

All the given below results and performance analysis are for a single document.

```
Word embeddings: 400000
Number of words missing from Glove: 10
Percent of words that are missing from vocabulary: 0.27%
Total number of unique words: 3680
Number of words we will use: 3117
Percent of words we will use: 84.7%
Length of word embedding matrix 3117
Total number of words in summaries: 34640
Total number of UNKs in summaries: 1009
Percent of words that are UNK: 2.91%
Summaries:
           counts
count   36.000000
mean    95.694444
std     30.411294
min     38.000000
25%     75.750000
50%     93.500000
75%    121.000000
max    166.000000

Texts:
            counts
count    36.000000
mean    867.527778
std     428.052032
min     303.000000
25%     548.250000
50%     782.500000
75%    1060.750000
max    2125.000000
```

Figure 34: Mathematical Analysis of Data Set

```
Average loss for this update: 10.282
New Record!
Average loss for this update: 5.68
New Record!
Average loss for this update: 5.093
New Record!
Average loss for this update: 4.999
New Record!
Average loss for this update: 4.905
New Record!
Average loss for this update: 4.847
New Record!
Average loss for this update: 4.829
New Record!
Average loss for this update: 4.717
```

```
Average loss for this update: 4.253
No Improvement.
Average loss for this update: 4.123
New Record!
Average loss for this update: 4.036
New Record!
Average loss for this update: 3.962
New Record!
Average loss for this update: 3.838
New Record!
Average loss for this update: 3.77
New Record!
Average loss for this update: 3.671
New Record!
Average loss for this update: 3.554
New Record!
```

```
Average loss for this update: 2.627
New Record!
Average loss for this update: 2.493
New Record!
Average loss for this update: 2.407
New Record!
Average loss for this update: 2.216
New Record!
Average loss for this update: 2.029
New Record!
Average loss for this update: 1.931
New Record!
Average loss for this update: 1.877
New Record!
Average loss for this update: 1.639
New Record!
```

```
Average loss for this update: 1.428
New Record!
Average loss for this update: 1.173
New Record!
Average loss for this update: 1.003
New Record!
Average loss for this update: 0.884
New Record!
Average loss for this update: 0.841
New Record!
Average loss for this update: 0.725
New Record!
Average loss for this update: 0.609
New Record!
Average loss for this update: 0.474
New Record!
Average loss for this update: 0.039
No Improvement.
Average loss for this update: 0.038
New Record!
Average loss for this update: 0.037
New Record!
Average loss for this update: 0.034
New Record!
Average loss for this update: 0.034
New Record!
Average loss for this update: 0.032
New Record!
Average loss for this update: 0.033
No Improvement.
Average loss for this update: 0.031
New Record!
Average loss for this update: 0.026
No Improvement.
Average loss for this update: 0.024
New Record!
Average loss for this update: 0.026
No Improvement.
Average loss for this update: 0.026
No Improvement.
Average loss for this update: 0.03
No Improvement.
Average loss for this update: 0.026
No Improvement.
Average loss for this update: 0.024
No Improvement.
Average loss for this update: 0.026
No Improvement.
```

Figure 35: epochs

```
 Response Words: the gnome help functions which are currently being used are de
precated so a patch is offered to switch to gtk show uri and gtk show help this
is missing updates to a symbol table has some licensing details wrong and gtk sh
ow help does not work on all platforms updates patches are given which fix these
 it is pointed out that gtk show uri is not multi head safe due to null parent i
t is suggested that it be forced to take a parent but this is rejected instead a
 version which takes <UNK> is suggested and then provided it is suggested that <
UNK> should be able to be null and that the documentation should be improved and
 this is accepted auto mounting is also requested someone questions the utility
of catering to so
```

Figure 36: Summary of Doc 1 through ANN

Figure 37, represents the performance comparison i.e. F-score between both the models. And it is clearly reflected that Neural Network gives better value and hence is a preferred model for summary generation.

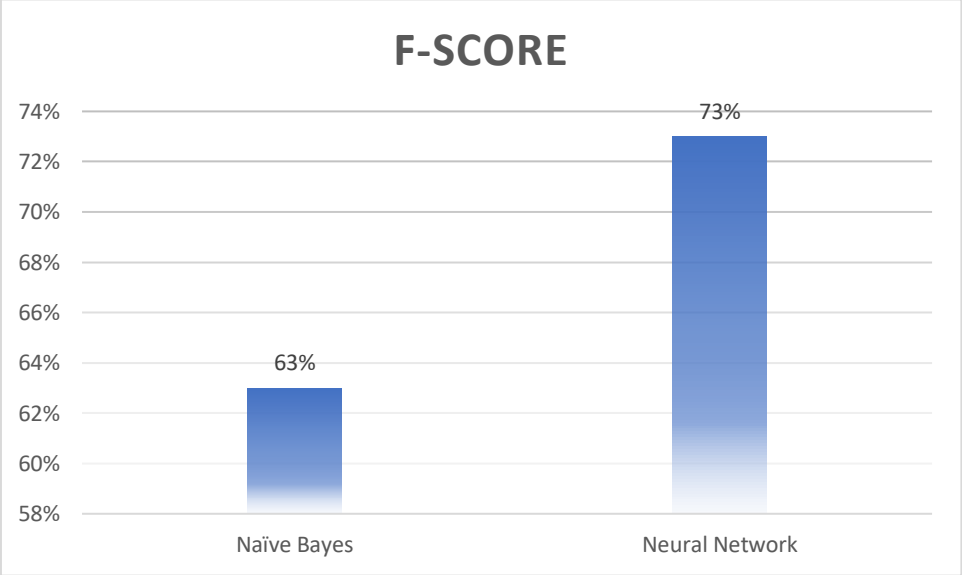


Figure 37: Bar chart of F-score of both model

CHAPTER -7
CONCLUSION

In this report, we presented Naïve Bayes Model and Artificial Neural Network Model for generating summary for a document.

For Naïve Bayes Model, we have compared the model's summary with human generated summary by using Precision, Recall and F-Score. For our model, F-Score ranges from 60% to 75%.

The varying difference in the f-score was because of the reason that human generating the summary was not an expertise on the topic chosen. If an expertise will be chosen f-score will increase.

REFERENCES

[1] Amit Kumar (2018). Introduction to Machine Learning. Retrieved from: https://www.allprogrammingtutorials.com/tutorials/introduction-to-machine-learning.php Retrieved on: 7 May, 2019.

[2] Morgan Polotan (2015). What is Machine Learning? Retrieved from: https://morganpolotan.files.wordpress.com/2015/04/unsupervised_learning.png Retrieved on: 7May, 2019.

[3] Das, S., 2017. Sentiment Analysis for Web-based Big Data: A Survey. *International Journal of Advanced Research in Computer Science*, *8*(5).

[4] Mani, S., Catherine, R., Sinha, V.S. and Dubey, A., 2012, November. Ausum: approach for unsupervised bug report summarization. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering* (p. 11). ACM.

[5] Lotufo, R., Malik, Z. and Czarnecki, K., 2015. Modelling the 'hurried'bug report reading process to summarize bug reports. *Empirical Software Engineering*, *20*(2), pp.516-548.

[6] Rastkar, S., Murphy, G.C. and Murray, G., 2014. Automatic summarization of bug reports. *IEEE Transactions on Software Engineering*, *40*(4), pp.366-380.

[7] Ferreira, I., Cirilo, E., Vieira, V. and Mourao, F., 2016, September. Bug report summarization: an evaluation of ranking techniques. In *2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*(pp. 101-110).

[8] Thu, H.N.T., 2014. An optimization text summarization method based on naive bayes and topic word for single syllable language. *Applied Mathematical Sciences*, *8*(3), pp.99-115.

[9] Jiang, H., Nazar, N., Zhang, J., Zhang, T. and Ren, Z., 2017. PRST: a page rank-based summarization technique for summarizing bug reports with duplicates. *International Journal of Software Engineering and Knowledge Engineering*, *27*(06), pp.869-896.

[10] Sinha, A., Yadav, A. and Gahlot, A., 2018. Extractive text summarization using neural networks. *arXiv preprint arXiv:1802.10137*. [2018]

[11] Cheng, J. and Lapata, M., 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.Khosrow, Kaikkah. "Text summarization using neural networks." (2004).

[12] Sarda, A.T. and Kulkarni, A.R., 2015. Text summarization using neural networks and rhetorical structure theory. *International Journal of Advanced Research in Computer and Communication Engineering*, *4*(6), pp.49-52.

[13] Jain, A., Bhatia, D. and Thakur, M.K., 2017, December. Extractive text summarization using word vector embedding. In *2017 International Conference on Machine Learning and Data Science (MLDS)* (pp. 51-55). IEEE.

[14] Patel, M., Chokshi, A., Vyas, S. and Maurya, K., 2018. Machine Learning Approach for Automatic Text Summarization Using Neural Networks‖. *International Journal of Advanced Research in Computer and Communication Engineering*, *7*(1).

[16] Nallapati, R., Zhou, B., Gulcehre, C. and Xiang, B., 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

[15] Chopra, S., Auli, M. and Rush, A.M., 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 93-98).

[17] *Kågebäck*, M., Mogren, O., Tahmasebi, N. and Dubhashi, D., 2014. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)* (pp. 31-39).

[18] Naïve Bayesian. Retrieved from: https://www.saedsayad.com/naive_bayesian.htm Retrieved on: 7 May, 2019

[19] Tal Baumel (2015). Sequence To Sequence Attention Models In DyNet. Retrieved from: https://talbaumel.github.io/blog/attention/ Retrieved on: 7 May, 2019.

[20] David Currie (2017). Text Summarization with Amazon Reviews.

Retrieved from : https://towardsdatascience.com/text-summarization-with-amazon-reviews-41801c2210b . Retrieved on: 3 April, 2019.

[21] Sebastian Raschka. Machine Learning FAQ. Retrieved from: https://sebastianraschka.com/faq/docs/multiclass-metric.html Retrieved on: 7 May, 2019.

[22] Tensorflow Precision/Recall/F1 Score and Confusion Matrix. Retrieved from: https://stackoverflow.com/questions/35365007/tensorflow-precision-recall-f1-score-and-confusion-matrix Retrieved on: 7 May, 2019