Automated Short Answer Scoring

Thesis Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Science
In
Computer Science

Under the Supervision of

Mr. Moin Mostakim
And
Co-Supervision
of
Mr. Matin Saad Abdullah

By
M. Rayhan Ahmed Mithu (12101115)

Syed Mohammad Moinul Islam Robin (13201082)

Moumita Kamal (12141001)

School of Engineering and Computer Science
Department of Computer Science and Engineering
BRAC University

Declaration

This is to certify that the research work titled “Automated Short Answer Scoring” is submitted by M. Rayhan Ahmed Mithu, Syed Mohammad Moinul Islam Robin and Moumita Kamal to the Department of Computer Science & Engineering, BRAC University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science. We hereby declare that this thesis is based on results obtained from our own work. The materials of work found by other researchers and sources are properly acknowledged and mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted to any other University or Institute for the award of any degree or diploma. We carried our research under the supervision of Mr. Moin Mostakim and Mr. Matin Saad Abdullah

Signature of Supervisor:

Mr. Moin Mostakim
Supervisor
Department of CSE
BRAC University

Signature of Co-Supervisor:

Mr. Matin Saad Abdullah
Co-Supervisor
Department of CSE
BRAC University

Signature of Authors:

M. Rayhan Ahmed Mithu,
12101115

Moumita Kamal,
12141001

Syed Mohammad Moinul Islam Robin
13201082

FINAL READING APPROVAL

Thesis Title: Automated Short Answer Scoring

Date of Submission: 14-12-2016

This final report on our research is read and approved by the supervisor Mr. Moin Mostakim. Its format, citation and bibliographic style are consistent and acceptable. Its illustrative materials including figures, tables and charts are in place. The final manuscript is satisfactory and is ready for submission to the department of Computer Science and Engineering, School of Engineering and Computer Science, BRAC University.

Signature of Supervisor:

_____

Mr. Moin Mostakim
Supervisor
Department of CSE,
BRAC University

Signature of Co-Supervisor:

_____

Mr. Matin Saad Abdullah
Co-Supervisor
Department of CSE,
BRAC University

Acknowledgements

We would like to thank our supervisor Mr. Moin Mostakim for allowing us to work with this thesis topic under his supervision and for all the ideas, suggestions and motivations he provided us. We would also like to thank our co-supervisor Mr. Matin Saad Abdullah for all his valuable suggestions, help and support. Finally, we would like to acknowledge the help that we got from Kaggle.com for giving away such large collections of supervised data and help all the researchers like us all over the world.

Abstract

In our thesis we have worked to analyse text short answers then predict the score accordingly by using different extracted features. In our research we have used around 1700 data for each data_set and are scored by two different humans provided by the Hewlett foundation available in Kaggle. We have used different NLP techniques to process the data in order to use it to the classifiers. Sckit was used to implement the algorithms of the different classifiers. The data were divided into two different data sets, one of them was the training_set and the test_set. The training_set data was used to train the different classifiers afterwards the test_set data was given to the classifiers to predict the score. The predicted score was compared with the score given by the humans to find the efficiency and accuracy of the different classifiers.

TABLE OF CONTENTS

CHAPTER 6 OUR APPROACHES

CHAPTER 7 ANALYSIS AND COMPARISON

CHAPTER 8 CONCLUSION

LIST OF FIGURES

LIST OF CHARTS

LIST OF ABBREVIATION

AES:	Automated Essay Scoring
NLP:	Natural Language Processing
NLTK:	Natural Language Toolkit
SVM:	Support Vector Machine
SGD:	Stochastic gradient descent
POS:	Parts Of Speech
ASAS:	Automated Short Answer Scoring
ML:	Machine Learning
NB:	Naive Bayes
ES:	Expert System
RL:	Reinforcement Learning
KR:	Knowledge Recognition
OCR:	Optical Character Reader

CHAPTER 1 INTRODUCTION

1.1 Introduction

The concept of Automated Essay Scoring or AES is not a new one in the field of Computer Science. People have been trying to use machines to grade papers for as long as 50 years now [1]. Automated Essay Grading or Scoring is referred to the computer technology of evaluating and scoring of written works [4]. The automated scoring of short answers is considered to be one of the most complicated domains [2] because it hugely depends on the similarity in semantics which means that it relies on the extent to which two sentences are similar in meaning to each other. In case of short answered questions, the answers are written in short sentences and the student has to justify his arguments or responses in that short span of text. As a result, two sentences might use completely different words to state the same thing while similar words might be used to express two very different kinds of sentences. For example, a student may use a synonymous word for an answer if he forgets the target keyword and the answer would still be correct.

It is very easy for humans to judge whether two answers carry similar concepts. Machines, on the other hand, can not differentiate between the sentimental or logical meanings of the two texts as easily. Another fact about long essays is that they follow certain rules of prose and grammar and have to be graded on those bases as well as that of the length and meaningfulness. Short answers, however, are graded generally based on length and meaningfulness but they are not necessarily required to be correct grammatically [3].

In our thesis, we have used natural language processing and machine learning to find out how close our model can get to those of human in regard to grading short answers. Through machine

learning, an artificial agent learns and predicts the next possible result of a given problem. Using the training dataset, a machine can identify certain criterion and train itself to predict the next possible output. We have used several approaches in our thesis to establish a system that would help with short answer scoring.

1.2 Motivation

Script checking has always been a tiresome and tough job for people. It takes a lot of time and it might become pretty monotonous and boring. In most cases there are a lot of scripts to be checked within a short period of time and the faculty has to check the same answers over and over again. As an alternative to this, many institutions now a days conduct MCQ type tests instead of written responses. This approach, however, does not properly assess students' critical reasoning and writing skills.

Many standardized tests now a days require essays or short passages as writing tasks. Thousands of people worldwide take these tests every year. The essay part of these exams, however, can not be graded by the machines and thus they are saved and sent to thousands of graders. This is time consuming, expensive and effortful all at the same time. On the other hand, due to the same reason, where these tests could take short answers to have better view of a student's knowledge about the topic, they rather opt for multiple choice type questions. The problem with MCQs is that the student has the chance to guess an answer even without knowing the correct solution which results in poor evaluation.

Another problem with human grading is that there is almost always no way to assess one's writing skills while practising at home or giving mock tests. No softwares and a very few online practise tests include essay grading and even that involves human graders and the writer has to wait for

days for the results to come. So, the student most of the time does not have any way to judge whether his writing is up to the mark or not.

An automatised way of grading short answers may help avoiding all these problems while saving a lot of time, breath and money. Automated Essay Scoring (AES) will also give students an way to assess themselves and improve their skills by receiving quick and useful feedbacks.

1.3 Limitations

The real challenge for any research is finding the right sets of data and our thesis is no exception to this. To train our artificial agent, we need a large selection of training dataset containing short answers that have been scored at least by two different graders in order to maintain uniformity and ensure cogency of the raters. These sorts of selection is not easily accessible for free. Another problem with finding such datasets is that most of the educational institutions in Bangladesh prefer their students to write their answers on paper rather than typing in computers and thus these selection of data is really hard to find digitally. Thankfully, we found a public domain in which different companies and researchers post their data and it is available for everyone for free and thus, we collected our data for this thesis from Kaggle.com [5] which is used by thousands of statisticians and data miners for machine learning purposes all over the world.

The next hurdle we faced while doing our research was the fact that we were dealing with short answer questions. Any standard long essay, as we know it, has a length of around 500 words while the answers we had to grade were no more than 60 words. As a result of this, judging the answers with such little information became a daunting task for us. The length of the answers also affected the precision and accuracy of our system.

Another problem with grading short answers is the challenge of extracting the actual meaning of a sentence. The same sentence can be written in so many different ways and there are so many ways of expressing the same word that it becomes almost impossible to determine the perfect meaning of a sentence. Our training data consists of 10 datasets each of which has around 1700 different entries for the same short answer. Therefore, there are thousands of different possible answers for the same question with different synonymous words. Similarly, the correct keyword might be used in a wrong answer which makes it even more difficult to grade accurately.

1.4 Research Goal

As complex and challenging this research may be, it is gratifying at the same time. There are millions of students all over the world. As of 2015, the total literacy rate of Bangladesh was 75.4% and total enrollment in primary, secondary and higher secondary schools were 23,907,151 [14]. The secondary school enrollment [14] alone in the year 2015 was 7,400,000. Around 4 million students sit for secondary and junior secondary school exams every year and million others sit for various tests everyday. Unquestionably, one can guess the burden that falls over the graders of these exams which results in wrong evaluation, time consuming outcome, exhaustion, monotony and many other problems.

This research gives us an opportunity to resolve these problems. Automatizing the grading system might result in much more accurate results within a much shorter period of time. A large number of people have done researches on OCRs to convert handwritten or scanned texts into machine encoded texts and our research could use that and go one step further.

This research has also given us an opportunity to learn about Natural Language Processing and Machine Learning and how to use the different approaches of machine learning to attain a desired solution and help the teachers score their students or help students with better assessment of themselves.

Our main goal is to build an intelligent system that will help a student improve their critical reasoning and writing skills along with a better judgement of their caliber and at the same time, minimize the time and effort spent by the teachers in grading the scripts. With this thought, we started working with various supervised classification algorithms to see which method predicts the scores closest to those of the human graders. In order to fulfill this goal, we will have to accumulate the right sort of data that contains a large number of short answers to certain questions with at least two human graders marking each of the answers. After getting the desired dataset, we will have to extract features to feed our algorithms for training. This will give us a chance to learn and use NLTK to extract features like bag of words, length, bag of keywords, misspelling words etc. Then, after training the algorithms we will have to do the testing and automatically grade the short answers in the test dataset.

Our work does not finish at only grading the answers. We also plan to comparisons and calculations to ensure accuracy, precision and efficiency in our system.

CHAPTER 2 RELATED WORKS

2.1 Literature Review

Initially, when we were processing and planning to successfully carry out our thesis, we consulted a number of papers regarding natural language processing (NLP). Jana and John developed C-rater [13], where it is used to automatically score a student's answer. It is supported by NLP and Knowledge Representation (KR) techniques. In this system, initially, the model answers are generated with the help of given concepts and later, the student's answer is processed by NLP technique. But according to Shweta and Sonal there are a "few disadvantages of the system where there is No distinct concepts specified, unexpected similar lexicons, incorrect spelling mistakes and many more" [15]. Arshad and Mohammed made a system Automated essay grading [16], this system would "analyze text essays then predict the score". The system used Natural language toolkit (NLTK) to extract features like word count, tokenized and many more to process the data and run it in the system where the system is being trained with similar data to score the essay. Laurie and Maiga developed Auto-assessor in the year of 2011[17], their target was to automatically score the short answers of students by the semantic meaning of those answers. Auto-assessor is underpinned by NLP technique with component based architecture. The sentences are re-processed using the components to their canonical form so that later it could be used in processing of both supplied correct answers and student responses. Afterwards, evaluation of the student response with the correct answer takes place where each word from the correct answers in canonical form is compared with the word from student response to finally score the answer. Ade-Ibijola, Wakama and Amadi developed Automated Essay Scoring (AES) an Expert System (ES) for scoring free text answers [18]. AES is based on Information

Extraction (IE). This ES is composed of three primary modules namely: Knowledge Base, Inference Engine and Working Memory. Inference engine uses shallow NLP technique to promote the pattern matching from the inference rules to the data contained in knowledge base. The NLP module contains: a Lexical Analyzer, a Filter and a Synonyms Handler module. The correctness evaluation is performed by fuzzy model which generate the scores for student answer with the help of two parameters: the percentage match and the mark assigned. Steven Burrows, Iryna Gurevych and Benno Stein have shown the progress on development of Automated Short Answer Scoring (ASAS) system over the course of the years in their publication.

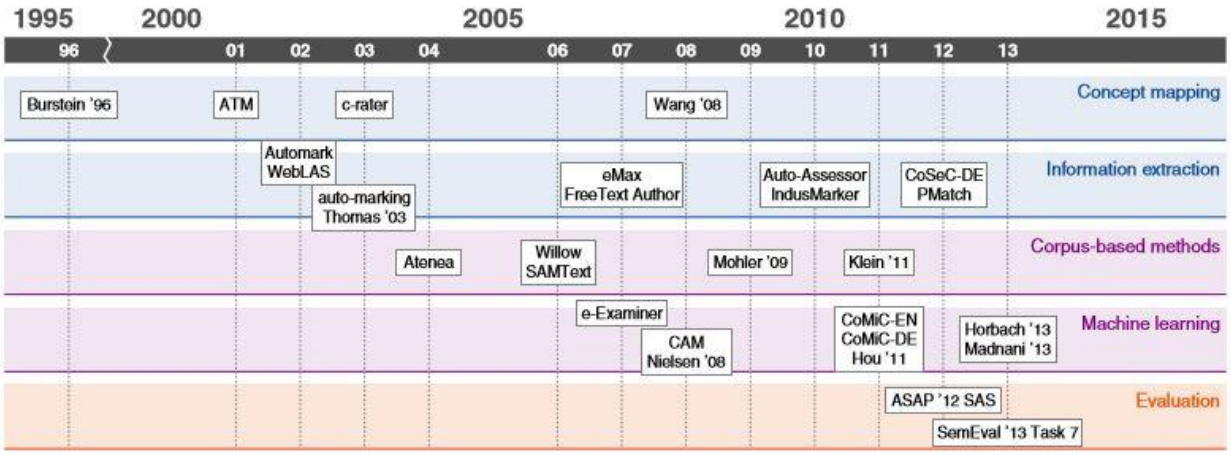

Figure 2.1: Historical analysis presenting the progress of ASAS developed over time.

2.2 Methodological Biases

Every research is bound to have a few biases and ours is no different from that. This research is about scoring short answers based on previously scored datasets. As a result, dissimilarity in the training dataset answers and the test dataset answers might result in erroneous scoring of the short answers. Again, a huge bias in the system that has been mentioned beforehand is the conceptual

barrier between a computer and a human grader. It is easier for humans to determine if two different sentences carry the similar concept but the same task is next to impossible for a computer. Another bias in the system is the irrelevant usage of keywords. In scoring the short answers, we have used bag of keywords as a feature but there might be a few questions that have the correct keywords in a wrong or unrelated sentence.

2.3 Problem Statement

Nowadays we can find a number of assessment tools on the market, but most of these tools are to score objective type questions like multiple choice questions. Using such assessment tools one can assess a student only at a very lower level of Blooms taxonomy [10] of educational objectives. Also, these type of tools fail to assess a student's spelling mistakes, critical reasoning skills and the correct order of words. In order to overcome the above stated problems, we are going to develop a system to evaluate the student's short answers using NLP and Machine Learning (ML). More details of the system is discussed in Chapter 4, Section 4.1.

The proposed system will try to score the student's short answer as evenly matched to human graders as possible.

CHAPTER 3 THESIS WORKFLOW

3.1 Thesis Workflow

In this chapter we will discuss the procedure that we followed throughout our thesis work. We can summarize our thesis workflow using a simple flow chart:

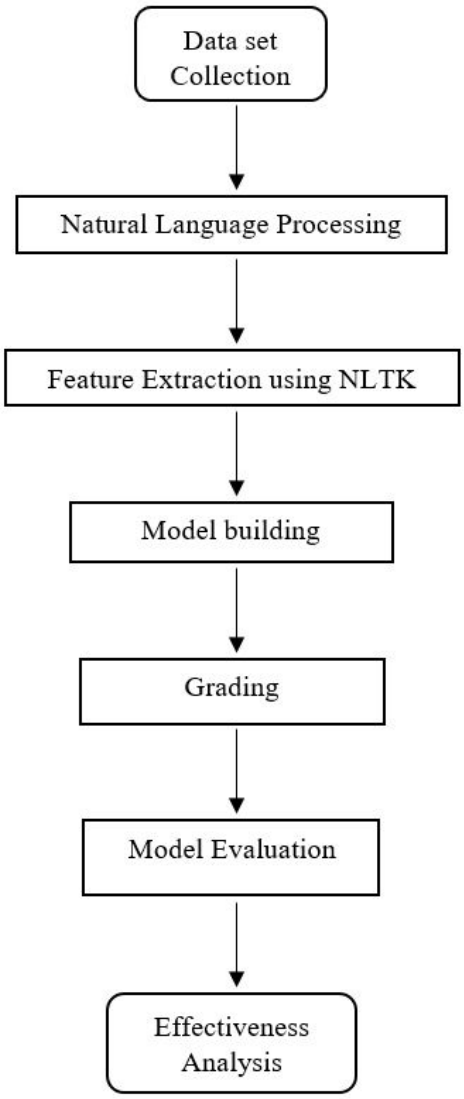

Figure: 3.1: Thesis Workflow

After selecting the problem that we wanted to work on, the very next step was to get the data. For topic like ours we needed a lot of data containing human graded answers along with some information. More about the data are discussed in Chapter 5, Section 5.1. Section 5.2 discusses the process of our data collection. At Kaggle we found a bulk amount of data which turned out to be very useful. We had to process those data to make it workable for our system by removing noise and corruption. More about data processing is discussed in Section 5.3. After data processing in order for our system to work we had to remove the stop words. We trained our machine with those data to develop a system that grades short answers accordingly by extracting features from the trained data. In Chapter 5, Section 5.6 we talked more about how we extracted features from the data.

After getting all the data sorted, we use those data to train our algorithms of classifiers and the test data are given into the algorithm to get the test data scored by the algorithms of classifiers. Algorithms uses different extracted features mentioned in Chapter 5, Section 5.6, and compare the test data with the trained data so that it can compare and categorize the score of the test data. In chapter 6, we talked more about the different classifiers and algorithms used and their results.

After we have the test results we start to compare the generated results by the different algorithms and classifiers to the scores given by the humans and analyse the results using Precision, F-measure, Recall and Accuracy. More about the analysis and their results are discussed at Chapter 7.

CHAPTER 4 EXPERIMENTAL ENVIRONMENT

4.1 Machine Learning

We are entering the era of big data. For example, there are about 1 trillion web pages, one hour of video is uploaded to YouTube every second, amounting to 10 years of content every day2; the genomes of 1000s of people, each of which has a length of 3.8 × 109 base pairs, have been sequenced by various labs; Walmart handles more than 1M transactions per hour and has databases containing more than 2.5 petabytes (2.5 × 1015) of information (Cukier 2010); and so on.

This deluge of data calls for automated methods of data analysis, which is what machine learning provides. In particular, we define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty. The best way to solve such problem is to use the tool of probability theory. Probability theory can be applied to any problem involving uncertainty.

Machine learns when they take a series of input data and based on mathematical calculations they correctly choose an algorithm to apply to that input so that the output is given in such a way which is useful to the user. Being accepted or not accepted is important because that feedback information accumulates and feeds into the selection criteria used to choose the algorithm. It’s closed feedback loop. The current machine learning algorithms that we can find are mostly designed in such a way so that it can solve a single well defined problem, often better than humans. However, significant work normally needs to be done to get the data into a form that is algorithmically operable.

Furthermore, these algorithms do not match human intelligence because they tend to learn one thing very precisely at a time but not more than one thing.

4.2 Types of Machine Learning

Machine learning is usually divided into two main types. In the predictive or supervised learning approach, the goal is to learn a mapping from inputs x to outputs y, given a labeled set of input-output pairs $D = \{(x_i, y_i)\}^N_{i=1}$. Here D is called the training set, and N is the number of training examples. In the simplest setting, each training input x_i is a D-dimensional vector of numbers, representing, say, the height and weight of a person. These are called features, attributes or covariates. In general, however, x_i could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc. Similarly the form of the output or response variable can in principle be anything, but most methods assume that y_i is a categorical or nominal variable from some finite set, $y_i \in \{1,...,C\}$ (such as male or female), or that y_i is a real-valued scalar (such as income level). When y_i is categorical, the problem is known as classification or pattern recognition, and when y_i is real-valued, the problem is known as regression. Another variant, known as ordinal regression, occurs where label space Y has some natural ordering, such as grades A–F. The second main type of machine learning is the descriptive or unsupervised learning approach. Here we are only given inputs, $D = \{x_i\}^N_{i=1}$, and the goal is to find "interesting patterns" in the data. This is sometimes called knowledge discovery. This is a much less well-defined problem, since we are not told what kinds of patterns to look for, and there is no obvious error metric to use (unlike supervised learning, where we can compare our prediction of y for a given x to the observed value). There is a third type of machine learning, known as reinforcement learning (RL), which is somewhat less

commonly used. This is useful for learning how to act or behave when given occasional reward or punishment signals. (For example, consider how a baby learns to walk.). Machine Learning [20] for more information on RL.

Natural Language Processing

Computer science, computational linguistics and artificial intelligence, combination of these three makes the Natural Language Processing. Natural Language Processing is the study that enables computers to understand the meaning of the human language or natural language. This line of study is related to the human-computer interactions. Moderns NLP algorithms are mainly based on Machine learning and statistics. Some of the major tasks in natural language processing are automatic summarization, morphological segmentation, named entity recognition, semantic analysis, optical character recognition, parsing, question answering, speech analysis, information extraction, information retrievals, translation and many more.

4.3 Natural Language ToolKit

The NLTK module is a massive tool kit, aimed at helping us with the entire Natural Language Processing (NLP) methodology. NLTK helped us with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, and then even with helping the machine to understand what the text is all about. Python's package NLTK is one of the most important package for our thesis. Using the NLTK we bring out features that we later use in our thesis. NLTK is a very handy tool to work with while working with natural language and machine.

4.4 Word Net

Word Net is a lexical database for the English language. It groups English words into sets of synonyms called synsets, provides short definitions and usage examples, and records a number of relations among these synonym sets or their members. Word Net can thus be seen as a combination of dictionary and thesaurus. While it is accessible to human users via a web browser, its primary use is in automatic text analysis and artificial intelligence applications. We have used it so that the program could know if the correct word was used in the answer.

CHAPTER 5 DATA COLLECTION AND PROCESSING

5.1 The Data

Consider the following GCSE biology question:

Statement of the question:

The blood vessels help to maintain normal body temperature. Explain how the blood vessels reduce heat loss if the body temperature falls below normal.

Marking Scheme (full mark 3) Any Three

Vasoconstriction; explanation (of vasoconstriction); less blood flows to / through the skin / close to the surface; less heat loss to air/surrounding/from the blood / less radiation / conduction / convection.

Here is a sample of real answers:

1. all the blood move faster and dose not go near the top of your skin they stay close to the moses 2. The blood vessels stops a large ammount of blood going to the blood capillary and sweat gland. This prents the presonne from sweating and loosing heat

3. When the body falls below normal the blood vessels 'vasoconstrict' where the blood supply to the skin is cut off, increasing the metabolism of the body. This prevents heat loss through the skin, and causes the body to shake to increase metabolism.

It will be obvious that many answers are ungrammatical with many spelling mistakes, even if they contain more or less the right content. Thus using standard syntactic and semantic analysis

methods will be difficult. Furthermore, even if we had fully accurate syntactic and semantic processing, many cases require a degree of inference.

Our data that we collect are similar to the example that I just mentioned earlier. We collected ten different sets of data, the one we are calling **data_set.** Each data_set represents a different question and different type of answer and marking schemes. Combining all the data we have almost twenty seven thousand five hundred and eighty eight data. In each data_set precious information on how to mark the answer are given also along with the information about that particular question of the data_set.

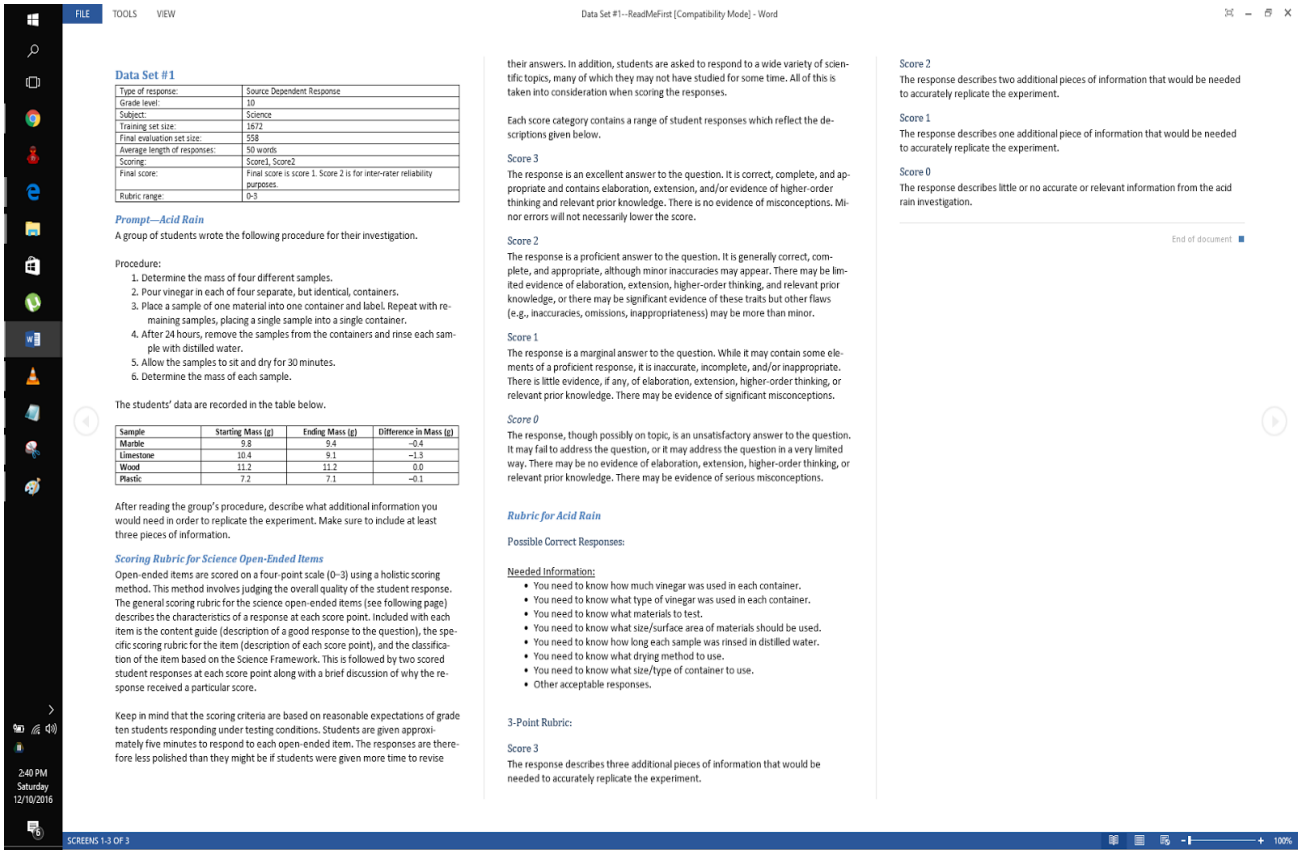

Figure 5.1: Screenshot of the Question and marking instructions

Again for every data_set there are train_data and test_data. More of which are discussed in the next sections on how to those data were used.

5.2 Data Collection

After we tool the data from Kaggle, we sorted all the data according to their data_set. In this thesis we are working with ten data_set. As described earlier each data_set is different we sorted them accordingly and have given them parameters. The parameters of the train data_set are as follow:

Figure 5.2: Screenshot of data

Id: A unique Id was introduced to each short answers.

DataSet: Data_set number represents the following set of data it represents.

Score1: Score1 are the scores that was given by first human examiners to the student for that answer of that particular data_set.

Score2: Score2 are the scores that was given by second human examiners to the student for that answer of that particular data_set.

DataText: Data_Text represents the answer that was given by the student on the following data_set.

5.3 Data Processing

In total combining all the data of all ten training data_set we are working with almost twenty seven thousand five hundred and eighty eight data. If we consider the first training data_set, it contains one thousand six hundred seventy two data. From those we filtered out one thousand three hundred and ninety one and used them to train our algorithm. Dropping two hundred and eighty one.

We only took those data where both the score, score1 and score2 are the same. Giving us train_data_set into four categories of each score from 0 to 3. Now for each score we roughly have three hundred to four hundred train_data_set with which we can train our algorithm to run the test_data to find out the score that our program gives and match it with the actual human score to find out the accuracy. Which we will be discussing later.

5.4 Word tokenization and Sentence tokenization

We used python NLTK's word tokenize module to our short answers. Tokenization is a process of converting a large text into single pieces of tokens.

We also used sentence tokenization to separate the sentences from the answers so that we can process it easily onwards.

The codes and the outputs are shown below.

```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
import random

from nltk.corpus import dataset1
print("Sentence Tokenized: ")
print(sent_tokenize(dataset1.raw('sc3/8.txt')))
print('Word Tokenized: ')
print(word_tokenize(dataset1.raw('sc3/8.txt')))
```

Figure 5.3: Code for word tokenize and sentence tokenization using python NLTK

```
Sentence Tokenized:
['You would need many more pieces of information to replicate the experiment .',
'You would need the type of samples to begin with in the procedure .', 'You would
also need to know the amount of vinegar used in each container .', 'You would als
o need to know exactly how to mass the samples and what types of container to use
( plastic for example , might alter the results ) . ']
Word Tokenized:
['You', 'would', 'need', 'many', 'more', 'pieces', 'of', 'information', 'to', 're
plicate', 'the', 'experiment', '.', 'You', 'would', 'need', 'the', 'type', 'of',
'samples', 'to', 'begin', 'with', 'in', 'the', 'procedure', '.', 'You', 'would',
'also', 'need', 'to', 'know', 'the', 'amount', 'of', 'vinegar', 'used', 'in', 'ea
ch', 'container', '.', 'You', 'would', 'also', 'need', 'to', 'know', 'exactly', '
how', 'to', 'mass', 'the', 'samples', 'and', 'what', 'types', 'of', 'container',
'to', 'use', '(', 'plastic', 'for', 'example', ',', 'might', 'alter', 'the', 'res
ults', ')', '.']
```

Figure 5.4: Output of the word tokenization and sentence tokenization of python NLTK

5.5 Removing Stop Words

The idea of Natural Language Processing is to do some form of analysis, or processing, where the machine can understand, at least to some level, what the text means, says, or implies. This is an obviously massive challenge, but there are steps to doing it that anyone can follow. The main idea, however, is that computers simply do not, and will not, ever understand words directly. Humans don't either. In humans, memory is broken down into electrical signals in the brain, in the form of neural groups that fire in patterns. There is a lot about the brain that remains unknown, but, the more we break down the human brain to the basic elements, we find out basic the elements really are. Well, it turns out computers store information in a very similar way! We need a way to get as close to that as possible if we're going to mimic how humans read and understand text. Generally, computers use numbers for everything, but we often see directly in programming where we use binary signals (True or False, which directly translate to 1 or 0, which originates directly from either the presence of an electrical signal (True, 1), or not (False, 0)). To do this, we need a way to convert words to values, in numbers, or signal patterns. The process of converting data to something a computer can understand is referred to as "pre-processing." One of the major forms of pre-processing is going to be filtering out useless data. In natural language processing, useless words (data), are referred to as stop words.

```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
import random
from nltk.corpus import dataset1
from nltk.corpus import stopwords
stop_words=set(stopwords.words('english'))
print("Stop Words in English: ")
print(stop_words)
sent_tokenized=sent_tokenize(dataset1.raw('sc3/8.txt'))

print('Text with stop words: ')
print(sent_tokenized)
print('Text after filtering stop words')
for i in sent_tokenized:
    words=word_tokenize(i)
    filtered=[w for w in words if not w in stop_words]
    print(filtered)
```

Figure 5.5: Code for removing stop words

```
Stop Words in English:
{'his', 'if', 'am', 'above', 'so', 'before', 'down', 'over', 's', 'was', 'by', 'a
bout', 'some', 'needn', 'she', 'i', 'being', 'didn', 'or', 'their', 'very', 'furt
her', 'does', 'but', 'won', 'who', 'do', 'hasn', 'until', 're', 'on', 'themselves
', 'here', 'be', 'her', 'him', 'ours', 'yourself', 'up', 'off', 've', 'and', 'has
', 'yours', 'there', 'them', 'me', 'from', 'shan', 'most', 'any', 'hers', 'd', 't
hose', 'will', 'y', 'had', 'more', 'of', 'mightn', 'in', 'such', 'don', 'm', 'you
r', 'doesn', 'mustn', 'into', 'when', 'ourselves', 'too', 'weren', 'than', 't', '
how', 'aren', 'after', 'ma', 'wouldn', 'this', 'where', 'nor', 'to', 'during', 'w
hom', 'between', 'under', 'are', 'have', 'couldn', 'at', 'haven', 'with', 'he', '
both', 'then', 'theirs', 'through', 'our', 'as', 'few', 'while', 'other', 'why',
'just', 'isn', 'they', 'again', 'now', 'll', 'below', 'doing', 'that', 'should',
'myself', 'it', 'because', 'for', 'you', 'is', 'o', 'the', 'against', 'its', 'own
', 'can', 'these', 'once', 'ain', 'been', 'yourselves', 'all', 'having', 'himself
', 'only', 'wasn', 'same', 'were', 'itself', 'which', 'no', 'we', 'what', 'did',
'a', 'shouldn', 'hadn', 'not', 'out', 'herself', 'an', 'my', 'each'}
Text with stop words:
['You would need many more pieces of information to replicate the experiment .',
'You would need the type of samples to begin with in the procedure .', 'You would
also need to know the amount of vinegar used in each container .', 'You would als
o need to know exactly how to mass the samples and what types of container to use
( plastic for example , might alter the results ) . ']
Text after filtering stop words
['You', 'would', 'need', 'many', 'pieces', 'information', 'replicate', 'experimen
t', '.']
['You', 'would', 'need', 'type', 'samples', 'begin', 'procedure', '.']
['You', 'would', 'also', 'need', 'know', 'amount', 'vinegar', 'used', 'container'
, '.']
['You', 'would', 'also', 'need', 'know', 'exactly', 'mass', 'samples', 'types', '
container', 'use', '(', 'plastic', 'example', ',', 'might', 'alter', 'results', '
)', '.']
```

Figure 5.6: Output for removing stop words

5.6 PyEnchant to extract the Misspelling words

PyEnchant is a spell checking library for python, based on the Enchant library. By using the Enchant library we can check if there is any spelling mistake in the given test data which will be added as another feature in order to score.

```
import nltk
from nltk.tokenize import sent_tokenize as st , word_tokenize as wt
from nltk.corpus import dataset1
import enchant

c=enchant.Dict("en_US")
words=set(wt(dataset1.raw('sc0/28.txt')))
print(words)
count=0
for w in words:
    check=c.check(w)
    if(check == False):
        count=count+1
print('Spelling Mistakes: ',count)
```

Figure 5.7: Code for PyEnchant to extract Misspelling words

```
{'Record', 'of', '24', 'hrs', 'container', 'different', '7', '4', ')', '6', 'eac
h', 'your', 'w/', 'plastic', 'min', 'containers', '1', 'marble', 'into', 'to', '
rinse', 'Get', '2', 'and', 'Pour', 'remove', 'separate', 'samples', 'vinegar', '
identical', 'dry', '3', ':', 'but', 'Allow', 'water', 'sample', 'sit', '.', 'dis
tilled', 'Determine', '5', 'After', 'from', ',', 'limestone', 'the', '30', 'data
', 'for', 'wood', 'Put', '^p', 'in'}
Spelling Mistakes:  5
```

Figure 5.8: Output of the PyEnchant to extract Misspelling words

5.7 Feature Extraction

After collecting and processing all our data by using different techniques of NLP we extracted features from the train_data_set to predict the score and are further used in our program. We will be discussing on the next chapter. The features that we used are as follow:

Bag of words: Tokenizer is used to remove stop words and trained to the program and where the word frequency is recorded as bag of words.

Length: The char lengths of the short answers are recorded

Bag of keywords: A bag of keywords are stored which are considered as possible key answers.

Misspelling words: Number of misspelled words are recorded from the short answers.

CHAPTER 6 OUR APPROACHES

6.1 Machine Learning Approach

Machine Learning is defined as the science of getting computers to perform without being explicitly programmed [7]. It explores the analysis and construction of algorithms that can learn from data and make predictions [6]. In general, Machine Learning is categorized into three types namely - (1) Supervised Learning, (2) Unsupervised Learning and (3) Reinforcement Learning. In our thesis, we have used supervised learning in order to grade the short answers of the students.

Supervised Learning includes the section of machine learning where the system predicts the next possible outcome using training data. To solve problems using supervised learning, one has to first determine the type of data to be used as the training data set and accumulate the data. In case of our research, the type of data that we needed to train our algorithms were bulk amount of short answers to certain questions that were scored by at least two human graders to ensure validation. Afterwards, we took the assistance of Kaggle for data accumulation. After determining the appropriate learning algorithm, one will have to complete the design and finally, evaluate the efficiency of the system which in case of our thesis, are SVM, SGD, Decision Tree Learning and Naive Bayes algorithm and have been discussed later in this section.

There are several approaches and algorithms to predict upcoming solutions using supervised learning. A few of them include Logistic Regression, Naive Bayes Classifier, Artificial Neural Networks, Decision Tree Learning, Support Vector Machines, Gaussian Process Regression, Nearest Neighbor Algorithm, Stochastic Gradient Descent etc. In our thesis, we used three different approaches of Naive Bayes Algorithm including Gaussian and Multinomial Naive Bayes along with

the regular Naive Bayes classifier and Support Vector Machine (SVM), Stochastic gradient Descent (SGD) and Decision Tree Learning algorithm in order to grade short answers.

We used scikit for implementing Multinomial Naive bayes, SVM, SGD and Decision Tree Learning algorithm. The general working procedure for all the classifiers are more or less the same where we have used feature union in order to train our data. Next, we created a pipeline containing all the features and the different classifiers. More about these classifiers and screenshots of the codes are added later in this chapter.

6.2 Naive Bayes Classifier

The Naive Bayes classifiers, in machine learning, are a set of supervised learning algorithms that use simple probability to determine the maximum likelihood of occurrence of a possible solution. This algorithm is based on applying the Bayes' Theorem with the naive assumption of independence between the features [11]. The Naive Bayes methods are also referred to as simple Bayes or independent Bayes. Although naive Bayes classifier is based on the Bayes' theorem, it is not necessarily a Bayesian method [8].

For a given class variable y and a dependent feature vector x_1 through x_n the Bayes' theorem is given as:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive independence assumption and after simplification, for all i, the relationship becomes:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \quad \text{.....[21]}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y), \quad \text{..... [21]}$$

This classifier is very popular because classification using Naive Bayes algorithm is easy, quick and efficient.

In our research, we trained our algorithm for naive bayes with the extracted features from the answers that contained the highest score (three marks). We, then, tested the test data set to predict the probability of all the answers of carrying a score of three. Next, we divided the result ranging from 0 to 1 into four divisions. Any answers having the probability of getting a score of three more than 0.75 upto 1 got a score of 3. Similarly, answers having the probability of getting three marks ranging from 0.51 to 0.75 got 2 marks, 0.26 to 0.50 probabilities got 1 marks and range of 0 to 0.25 got a 0.

6.3 Gaussian Naive Bayes

The Gaussian Naive Bayes classifier mainly deals with continuous data. It is assumed that the continuous values associated with each class are disseminated according to a Gaussian distribution (also known as Normal distribution).

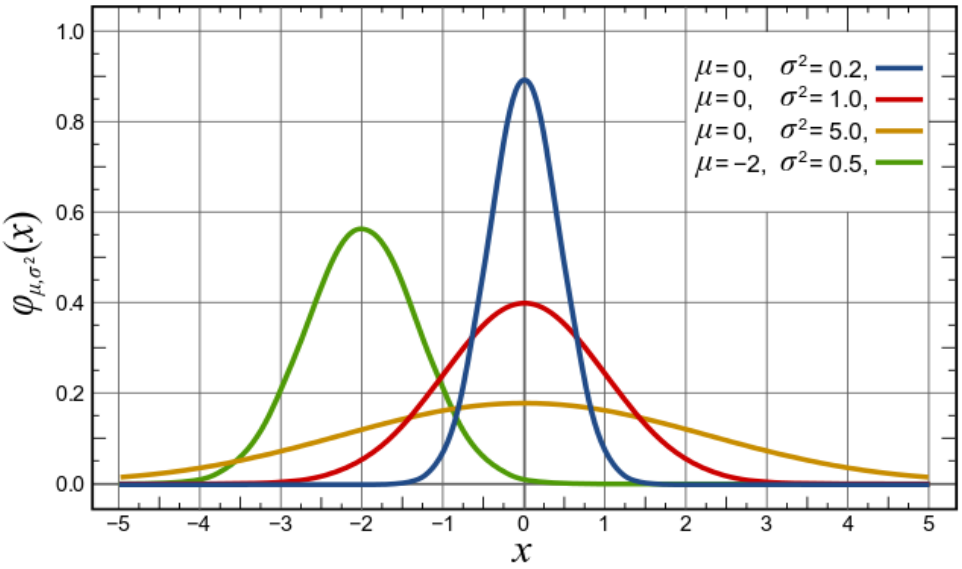

Figure 6.1: Gaussian or Normal distribution

The Gaussian Naive Bayes classifier relationship is expressed as:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \text{.....[22]}$$

Where, μ_y is the mean of the values in x_1 associated with class y and σ_y is the variance of the values in x_1 associated with class y. Both the parameters σ_y and μ_y are estimated using maximum likelihood.

6.4 Multinomial Naive Bayes

The Multinomial Naive Bayes is one of the two classic naive Bayes variants used in text classification. It implements the naive Bayes algorithm for multinomially distributed data [11]. The distribution is specified by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where, n is the number of features and θ_{yi} is the probability of feature i appearing in a sample located in class y. The parameter θ_y is estimated by maximum likelihood given by:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \text{.....[23]}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{|T|} N_{yi}$ is the total count of all features for class y [11].

```
import sklearn
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
from sklearn.feature_extraction import DictVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline, FeatureUnion
import numpy as np
from sklearn import metrics
from sklearn.datasets import load_files
from nltk.corpus import keywords as kw,stopwords as sw
import string
from sklearn.base import BaseEstimator, TransformerMixin

files = load_files('dataset1', shuffle=True)

class FeatureExtractor(BaseEstimator, TransformerMixin):
    def __init__(self,dict_vec=DictVectorizer(sparse=False),
                 key_words=None,stopwords=None,punc=None):
        self.stopwords=stopwords or set(sw.words('english'))
        self.punct=punc or set(string.punctuation)
        self.dict_vec=dict_vec
        self.key=key_words or list(kw.words('key1'))
    def transform(self, X, y=None):
        length=[]
        for i in X:
            words=set(i)
            feat={}
            feat['length']=len(i)
            for w in self.key:
                feat[w]=(w in words)
            length.append(feat)
        return self.dict_vec.fit_transform(length)
    def fit(self, X, y=None):
        return self

text_clf = Pipeline([
  ('features', FeatureUnion([
      ('fext', FeatureExtractor()),
    ('ngram_tf_idf', Pipeline([
      ('counts', CountVectorizer()),
      ('tf_idf', TfidfTransformer())
    ])),
  ])),
  ('clf', MultinomialNB())
```

Figure 6.2: Screenshot of Multinomial NB code (part 1)

```
text_clf = Pipeline([
  ('features', FeatureUnion([
      ('fext', FeatureExtractor()),
    ('ngram_tf_idf', Pipeline([
      ('counts', CountVectorizer()),
      ('tf_idf', TfidfTransformer())
    ])),
  ])),
  ('clf', MultinomialNB())
])

text_clf=text_clf.fit(files.data,files.target)
testdata=sklearn.datasets.load_files('testset',shuffle=True)
doc_test=testdata.data
predicted=text_clf.predict(doc_test)
accuracy=np.mean(predicted == testdata.target)
print('Accuracy of MultinomialNB: ',accuracy)
print("Classification report: \n",
      metrics.classification_report(testdata.target,
                                    predicted,
                                    target_names=testdata.target_names))
```

Figure 6.2: Screenshot of Multinomial NB code (part 2)

6.5 Support Vector Machine

Another common method that is used to perform supervised learning using different classifiers in order to predict upcoming possible solutions is Support Vector Machine (SVM). It is a discriminative classifier formally defined by a separating hyperplane. In other words, the algorithm outputs an optimal hyperplane for given training data which categorizes new examples. In spite of being a complex process, SVM is widely regarded as one of the best text classification algorithms because of its effectiveness, accuracy, efficiency and versatility.

```
import sklearn
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
from sklearn.feature_extraction import DictVectorizer
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline, FeatureUnion
import numpy as np
from sklearn import metrics
from sklearn.datasets import load_files
from nltk.corpus import keywords as kw,stopwords as sw
import string
from sklearn.base import BaseEstimator, TransformerMixin
from nltk.tokenize import word_tokenize as wt
import enchant
files = load_files('dataset1', shuffle=True)

class FeatureExtractor(BaseEstimator, TransformerMixin):
    def __init__(self,dict_vec=DictVectorizer(sparse=False),
                 key_words=None,stopwords=None,tokenizer=None,punc=None):
        self.stopwords=stopwords or set(sw.words('english'))
        self.punct=punc or set(string.punctuation)
        self.dict_vec=dict_vec
        self.wtokenize = tokenizer or wt
        self.key=key_words or list(kw.words('key1'))
    def transform(self, X, y=None):
        length=[]
        for i in X:
            words=list(self.wtokenize(str(i,'utf-8')))
            feat={}
            mistakes=0
            checker=enchant.Dict("en_US")
            for w in words:
                check=checker.check(w)
                if(check == False):
                    miskates= mistakes+1
            feat['mistakes']=mistakes
            feat['length']=len(i)
            for w in self.key:
                feat[w]=(w in words)
            length.append(feat)
        return self.dict_vec.fit_transform(length)
    def fit(self, X, y=None):
        return self
```

Figure 6.3: Screenshot of SVM code (part 1)

```
text_clf = Pipeline([
  ('features', FeatureUnion([
      ('fext', FeatureExtractor()),
    ('ngram_tf_idf', Pipeline([
      ('counts', CountVectorizer()),
      ('tf_idf', TfidfTransformer())
    ])),
  ])),
  ('clf',SVC(kernel='linear'))
])

text_clf=text_clf.fit(files.data,files.target)
testdata=sklearn.datasets.load_files('testset',shuffle=True)
doc_test=testdata.data
predicted=text_clf.predict(doc_test)
accuracy=np.mean(predicted == testdata.target)
print('Accuracy of MultinomialNB: ',accuracy)
print("Classification report: \n",
      metrics.classification_report(testdata.target,
                                    predicted,
                                    target_names=testdata.target_names))
```

Figure 6.4: Screenshot of SVM code (part 2)

6.6 Stochastic Gradient Descent

Stochastic gradient descent, also known as incremental gradient descent, is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions. It is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions [9]. SGD has been around in the machine learning community for a long time but it has received a considerable amount of attention just recently in the context of large-scale learning.

Stochastic Gradient Descent is considered advantageous for it's efficiency and ease of implementation.

```
import sklearn
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
from sklearn.feature_extraction import DictVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline, FeatureUnion
import numpy as np
from sklearn import metrics
from sklearn.datasets import load_files
from nltk.corpus import keywords as kw,stopwords as sw
import string
from sklearn.base import BaseEstimator, TransformerMixin
from nltk.tokenize import word_tokenize as wt
import enchant
files = load_files('dataset1', shuffle=True)

class FeatureExtractor(BaseEstimator, TransformerMixin):
    def __init__(self,dict_vec=DictVectorizer(sparse=False),
                 key_words=None,stopwords=None,tokenizer=None,punc=None):
        self.stopwords=stopwords or set(sw.words('english'))
        self.punct=punc or set(string.punctuation)
        self.dict_vec=dict_vec
        self.wtokenize = tokenizer or wt
        self.key=key_words or list(kw.words('key1'))
    def transform(self, X, y=None):
        length=[]
        for i in X:
            words=list(self.wtokenize(str(i,'utf-8')))
            feat={}
            mistakes=0
            checker=enchant.Dict("en_US")
            for w in words:
                check=checker.check(w)
                if(check == False):
                    miskates= mistakes+1
            feat['mistakes']=mistakes
            feat['length']=len(i)
            for w in self.key:
                feat[w]=(w in words)
            length.append(feat)
        return self.dict_vec.fit_transform(length)
    def fit(self, X, y=None):
        return self


    text_clf = Pipeline([
      ('features', FeatureUnion([
          ('fext', FeatureExtractor()),
        ('ngram_tf_idf', Pipeline([
          ('counts', CountVectorizer()),
          ('tf_idf', TfidfTransformer())
        ])),
      ])),
      ('clf',SGDClassifier(loss='hinge',
                           penalty='l2',alpha=1e-3, n_iter=5, random_state=42))
    ])

    text_clf=text_clf.fit(files.data,files.target)
    testdata=sklearn.datasets.load_files('testset',shuffle=True)
    doc_test=testdata.data
    predicted=text_clf.predict(doc_test)
    accuracy=np.mean(predicted == testdata.target)
    print('Accuracy of MultinomialNB: ',accuracy)
    print("Classification report: \n",
          metrics.classification_report(testdata.target,
                                        predicted,
                                        target_names=testdata.target_names))
```

Figure 6.5: Screenshot of SGD code

6.7 Decision Tree Learning

Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves) [12]. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values are called regression trees.

```
import sklearn
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
from sklearn.feature_extraction import DictVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline, FeatureUnion
import numpy as np
from sklearn import metrics
from sklearn.datasets import load_files
from nltk.corpus import keywords as kw,stopwords as sw
import string
from sklearn.base import BaseEstimator, TransformerMixin
from nltk.tokenize import word_tokenize as wt
import enchant
files = load_files('dataset1', shuffle=True)

class FeatureExtractor(BaseEstimator, TransformerMixin):
    def __init__(self,dict_vec=DictVectorizer(sparse=False),
                 key_words=None,stopwords=None,tokenizer=None,punc=None):
        self.stopwords=stopwords or set(sw.words('english'))
        self.punct=punc or set(string.punctuation)
        self.dict_vec=dict_vec
        self.wtokenize = tokenizer or wt
        self.key=key_words or list(kw.words('key1'))
    def transform(self, X, y=None):
        length=[]
        for i in X:
            words=list(self.wtokenize(str(i,'utf-8')))
            feat={}
            mistakes=0
            checker=enchant.Dict("en_US")
            for w in words:
                check=checker.check(w)
                if(check == False):
                    miskates= mistakes+1
            feat['mistakes']=mistakes
            feat['length']=len(i)
            for w in self.key:
                feat[w]=(w in words)
            length.append(feat)
        return self.dict_vec.fit_transform(length)
    def fit(self, X, y=None):
        return self
```

Figure 6.6: Screenshot of Decision Tree code (part 1)

```
text_clf = Pipeline([
  ('features', FeatureUnion([
      ('fext', FeatureExtractor()),
    ('ngram_tf_idf', Pipeline([
      ('counts', CountVectorizer()),
      ('tf_idf', TfidfTransformer())
    ])),
  ])),
  ('clf', DecisionTreeClassifier(max_depth=5))
])

text_clf=text_clf.fit(files.data,files.target)
testdata=sklearn.datasets.load_files('testset',shuffle=True)
doc_test=testdata.data
predicted=text_clf.predict(doc_test)
accuracy=np.mean(predicted == testdata.target)
print('Accuracy of DecisionTree: ',accuracy)
print("Classification report: \n",
      metrics.classification_report(testdata.target,
                                    predicted,
                                    target_names=testdata.target_names))
```

Figure 6.7: Screenshot of Decision Tree code (part 2)

CHAPTER 7 ANALYSIS AND COMPARISON

7.1 Analysis

For the analysis of our research, we have computed the precision, recall, F-measure and support for each class. These measures depend upon sensitivity and specificity. To understand these terms better, one has to first have a clear idea about True Positives, True negatives, False Positives and False negatives.

True Positives (TP): True positive occurs when positives are correctly identified as such. In machine learning, when the predicted outcome matches the actual value as positive, then it is said to be a true positive.

True Negatives (TN): Just like the true positives, True negativity occurs when negatives are correctly identified as such. In machine learning, when the predicted outcome matches the actual value as negative, it is said to be a true negative.

False Positives (FP): False positives or false alarms are errors that occur when the prediction indicates that the result is positive but the actual value is negative.

False Negatives (FN): False Negatives are errors where a test result indicates that a condition failed, while it actually was successful. In other words, when the predicted solution is negative but the actual value is positive, it is said to be a false negative.

This relations can be easily understood from the diagram below:

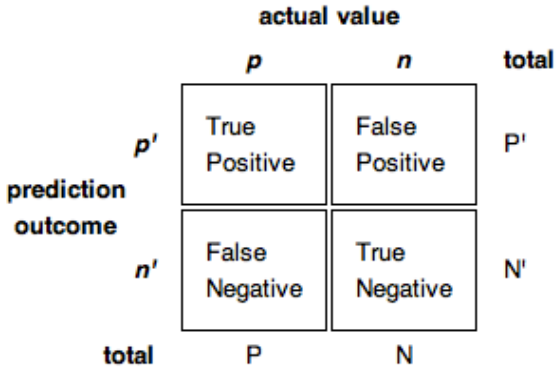

Figure 7.1: Sensitivity and specificity

Using these measures, we can calculate the precision, recall, F-measure and support.

Precision: The precision or positive predictive value is the ability of the classifier not to label a negative sample as positive. It is the ratio given by: *TP/(TP + FP)*, where, *TP* is the number of true positives and *FP* is the number of false positives.

Recall: The recall or true positive rate is the ability of the classifier to find all the positive samples. Recall is the ratio given by: *TP/(TP + FN)*, where, *TP* is the number of true positives and *FN* is the number of false negatives.

F-measure: F-measure or F-score is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score. The F-measure score can be interpreted as a weighted harmonic mean of the precision and recall, where it reaches its best value at 1 and worst score at 0. The F-measure or balanced F-score is the harmonic mean of precision and recall obtained by multiplying the constant of 2:

$$F\text{-measure} = 2((P * R) / (P + R))$$

Where, *P* is precision and *R* is recall.

Support: The support is the number of occurrences of each class in the system. In our thesis, support indicates how many rows of each score has occurred in the processing of the data.

For our analysis, we calculated the accuracy, precision, recall, F1-score and support for each algorithm and compared them against each other.

7.2 Analysis for Support Vector Machine

Calculating the precision, recall, and F-measure for Support Vector Machine (SVM), we got the following output:

```
0.653846153846
             precision    recall  f1-score   support

        sc0       0.82      0.82      0.82        22
        sc1       0.56      0.43      0.49        23
        sc2       0.50      0.68      0.58        28
        sc3       0.81      0.68      0.74        31

avg / total       0.67      0.65      0.66       104
```

Figure 7.2: Screenshot of SVM analysis output

We, then plotted these calculations against each score in a bar chart and compared them to find the following result for Support Vector Machine:

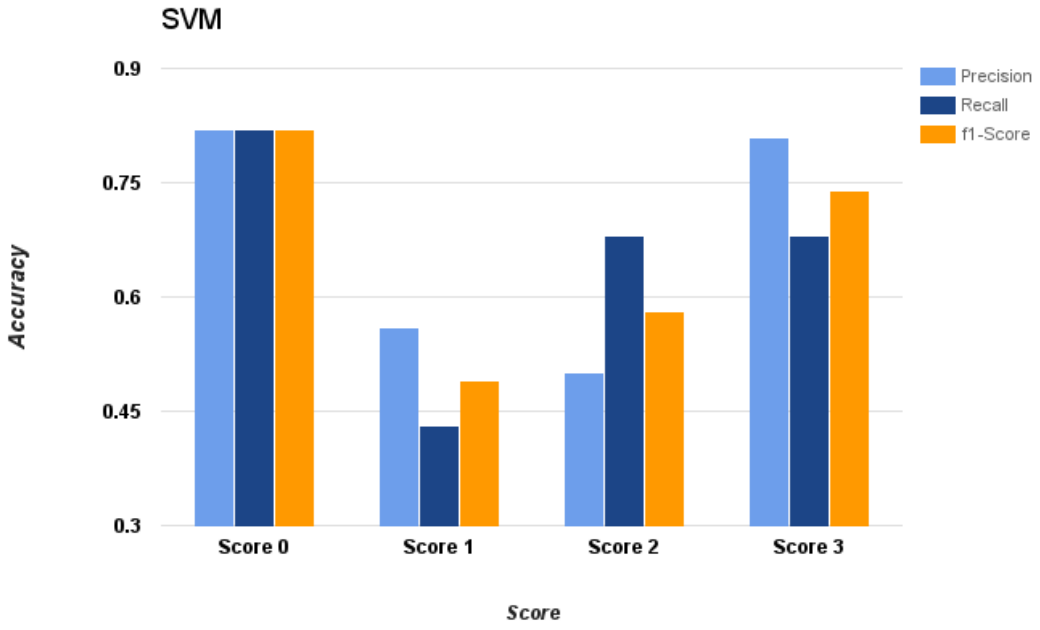

Chart 7.1: Precision, recall and f1-score comparison for SVM

7.3 Analysis for Stochastic Gradient Descent

We computed the precision, recall, F-measure and support for Stochastic Gradient Descent (SGD) and obtained the following output:

```
0.596153846154
             precision    recall  f1-score   support

        sc0       0.86      0.82      0.84        22
        sc1       0.62      0.35      0.44        23
        sc2       0.41      0.68      0.51        28
        sc3       0.71      0.55      0.62        31

avg / total       0.64      0.60      0.60       104
```

Figure 7.3: Screenshot of SGD analysis output

We, then plotted these calculations against each score in a bar chart and compared them to find the following outcome for Stochastic Gradient Descent:

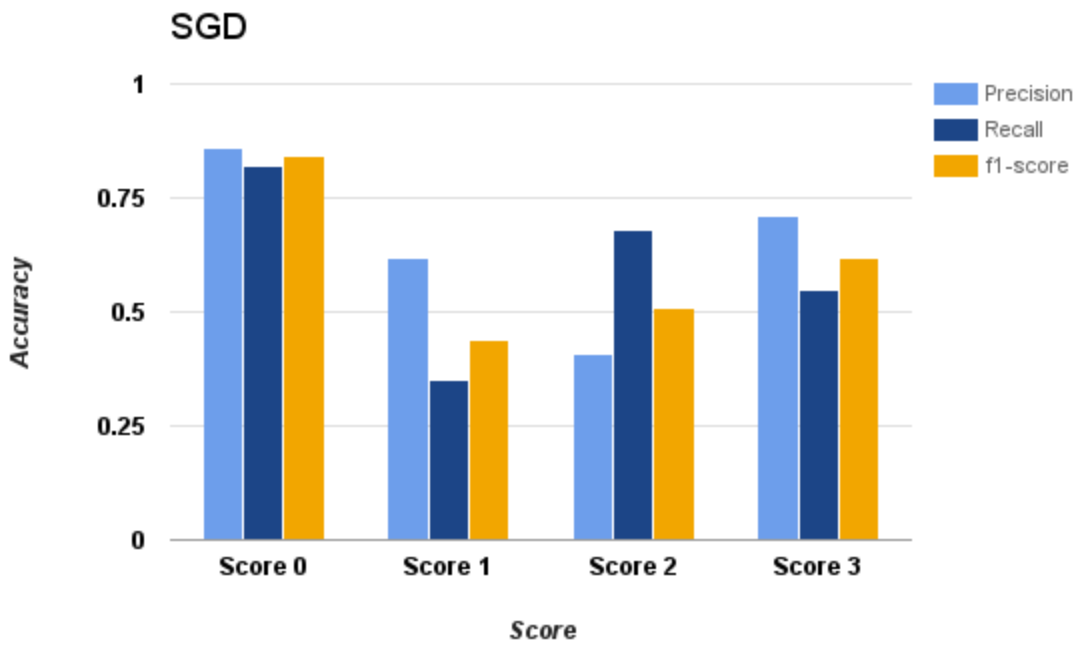

Chart 7.2: Precision, recall and f1-score comparison for SGD

7.4 Analysis for Multinomial Naive Bayes

We calculated the precision, recall, F-measure and support for Multinomial Naive Bayes and got the following output:

```
Accuracy of MultinomialNB:  0.394230769231
Classification report:
             precision    recall  f1-score   support

        sc0       0.89      0.36      0.52        22
        sc1       0.50      0.17      0.26        23
        sc2       0.33      1.00      0.49        28
        sc3       1.00      0.03      0.06        31

avg / total       0.68      0.39      0.32       104
```

Figure 7.4: Screenshot of Multinomial NB analysis output

We, then plotted these calculations against each score in a bar chart and compared them to find the following result for Multinomial Naive Bayes :

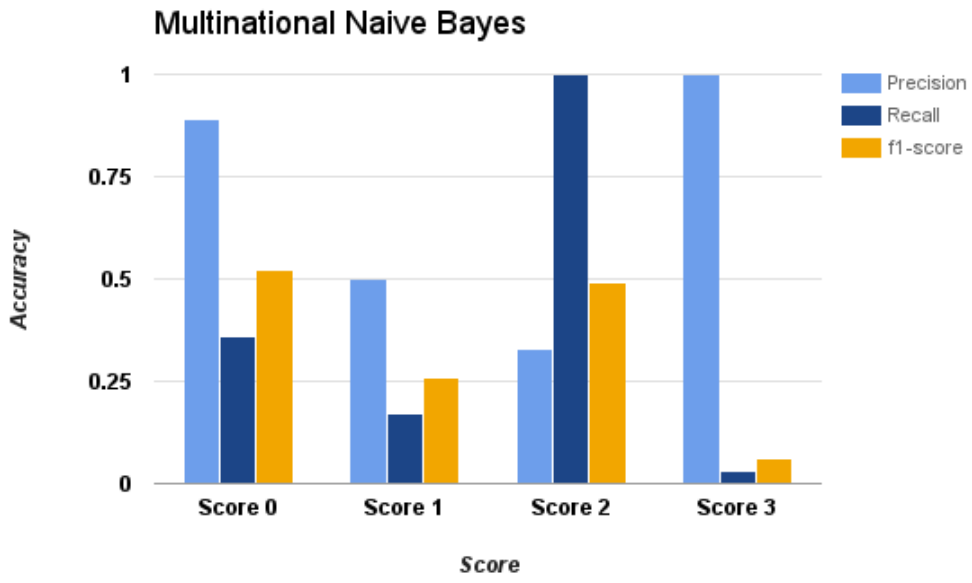

Chart 7.3: Precision, recall and f1-score comparison for Multinomial NB

7.5 Analysis for Decision Tree Learning

We calculated the precision, recall, F-measure and support for Decision tree and got the following output:

```
Accuracy of DecisionTree:  0.519230769231
Classification report:
             precision    recall  f1-score   support

        sc0       0.75      0.68      0.71        22
        sc1       0.50      0.48      0.49        23
        sc2       0.38      0.64      0.47        28
        sc3       0.71      0.32      0.44        31

avg / total       0.58      0.52      0.52       104
```

Figure 7.5: Screenshot of Decision tree analysis output

We, then plotted these calculations against each score in a bar chart and compared them to find the following result for Decision tree :

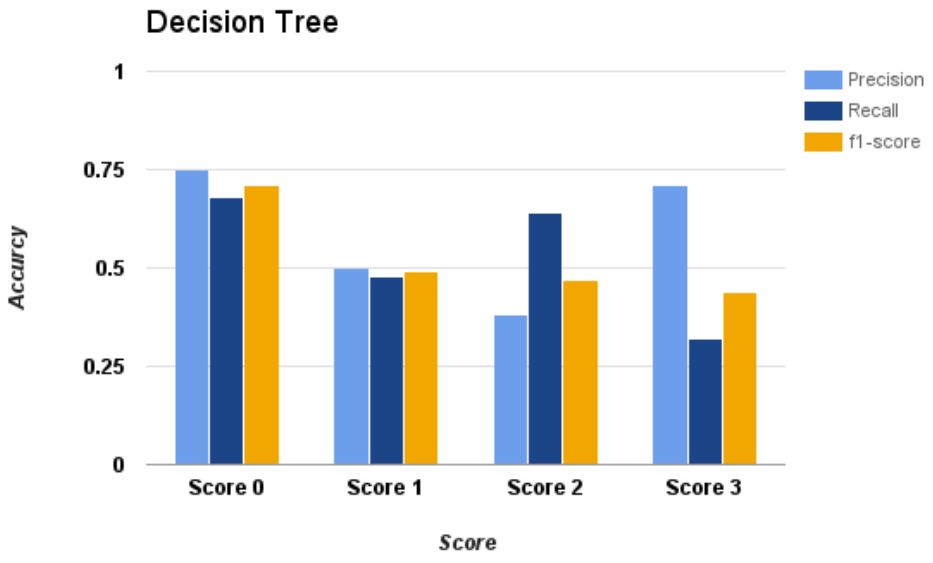

Chart 7.4: Precision, recall and f1-score comparison for Decision tree

7.6 Accuracy Comparison

After running the algorithm with the test_data the following predicted score was produced which is as follow:

id	text_set	text_score	Predicted_score
1673	1	1	1
1674	1	1	0
1675	1	3	3
1676	1	0	0
1677	1	0	0
1678	1	2	1
1679	1	0	0
1680	1	1	0
1681	1	3	2
1682	1	2	0
1683	1	1	1
1684	1	1	0
1685	1	1	0
1686	1	3	2
1687	1	1	1
1688	1	2	1
1689	1	3	1
1690	1	0	0
1691	1	2	2
1692	1	2	2
1693	1	0	0
1694	1	0	0
1695	1	0	1
1696	1	1	1
1697	1	1	1
1698	1	1	1
1699	1	1	0
1700	1	2	0
1701	1	3	0
1702	1	3	2
1703	1	2	2
1704	1	3	2
1705	1	2	2
1706	1	2	2
1707	1	1	1
1708	1	2	2

Figure 7.6 Actual score Vs predicted score using SVM

Finally, we compared the accuracy of each of these algorithms and came up with the following chart:

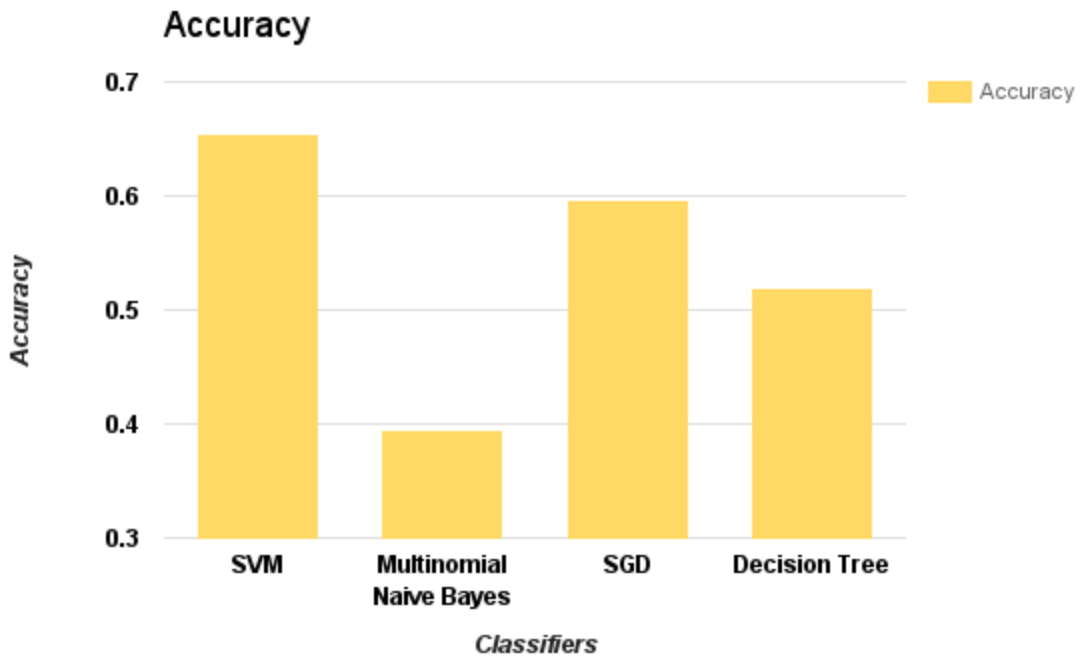

Chart 7.5: Accuracy comparison between SVM, SGD and Multinomial NB

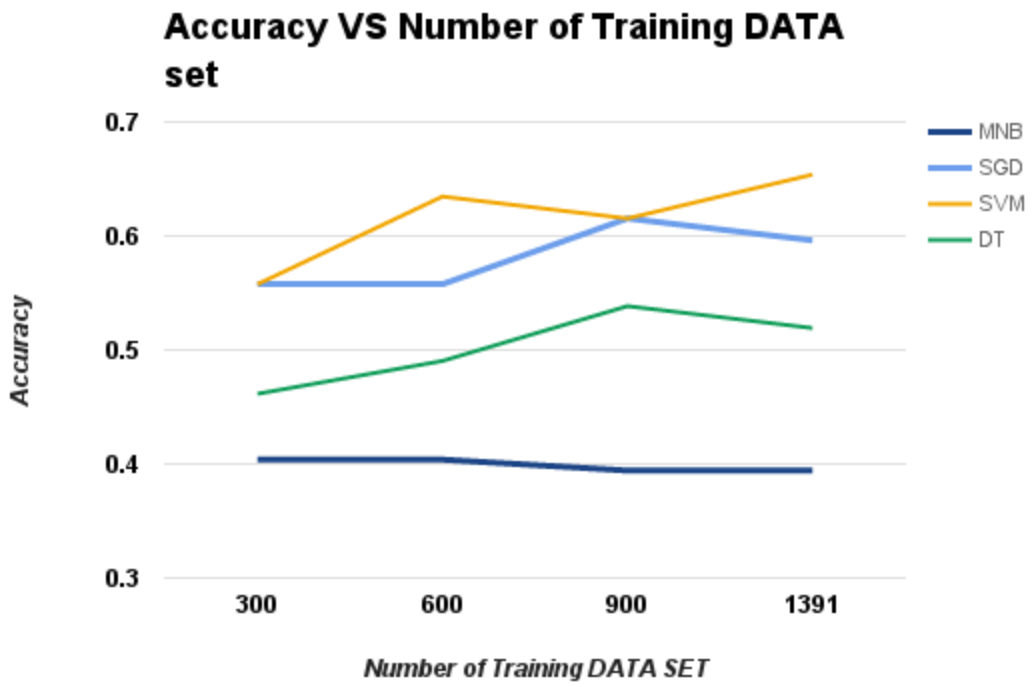

Figure 7.7 Accuracy VS Number of training DATA set

The predicted result are generated using all the classifiers. Among the classifiers we could find that the SVM classifier could predict closer to the human score. The output of the predicted score are shown in Figure 7.6.

Analysing all the algorithms and calculating their accuracies, we came to the decision that Support Vector Machine is the most accurate method among all the other methods that we tested. Stochastic Gradient Descent is also able to produce a somewhat accurate result, though not as precise as SVM. Multinomial Naive Bayes, on the other hand, as simple as it may be, is not very accurate at grading the responses of students.

Also as we have recorded the accuracy of the different classifiers using different number of training data and was plotted in the graph which is shown in Figure 7.7. In here we changed the number of data set as 300, 600, 900 and 1391. As we increase the number of training data set we can see most of the classifiers could identify and predict more accurate result. Throughout the process we can see that the SVM classifier gives us much more accurate result than the other classifiers. But while we were running the classifiers with all the data to predict the score the runtime of SVM classifier was much more higher than the other classifiers too.

CHAPTER 8 CONCLUSION

8.1 Conclusion

We started our thesis on Automated Short Answer Scoring thinking of establishing a system that will be able to grade short answer scripts with accuracy and preciseness as good as that of human graders. We tried and tested a number of algorithms in order to achieve our goal. Some of them worked better than the others in sense of efficiency, effectiveness, complexity and correctness. Multinomial Naive Bayes turned out to be the least accurate of all even though it was the simplest process among them. Decision tree and Stochastic Gradient Descent or SGD also gave good results but comparing all other features one algorithm stood out better than all the others. At the end of our thesis, after comparing all the algorithms that we used on the aforementioned scale, we came to the decision that Support Vector Machines or SVM would be the best choice to score short answers. Although our algorithm has some limitations and shortcomings of its own, it is able to predict scores that are the closest to that of human raters.

8.2 Future Works

Our primary goal was to establish a system that would make people's lives easier and assist in the improvement of a student's writing abilities. We have somewhat succeeded in achieving our goal but our work does not end here. We have a few plans for the future of our research. Firstly, we plan to have an online database management system where people can upload their data for our algorithm to train itself and have better, more accurate predictions of the scores. Moreover, we want to create

automated feedback system depending on various scores obtained so that students can assess themselves better at home. At the end of the day, our work is just a miniscule amount for what is about to come in the future in terms of natural language processing.

GLOSSARY

Continuous data: Continuous data is quantitative data that can be measured. It has an infinite number of possible values within a selected range e.g. temperature range. In Gaussian Naive Bayes classifier, this range spans from 0 to 1.

Normal (Gaussian) distribution: The Gaussian distribution is a continuous function which approximates the exact binomial distribution of events. If Gaussian distribution is normalized then the sum over all values of x gives a probability of 1.

Bloom's Taxonomy: Bloom's Taxonomy was created in 1956 under the leadership of educational psychologist Dr Benjamin Bloom in order to promote higher forms of thinking in education, such as analyzing and evaluating concepts, processes, procedures, and principles, rather than just remembering facts.

Multinomial distribution: The multinomial distribution is a generalization of the binomial distribution. If you perform an experiment that can have only two outcomes (either success or failure), then the number of times you obtain one of the two outcomes is a binomial random variable.

Supervised learning: Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consists of a set of training examples. In supervised learning, each example is a pair consisting of an input object and a desired output value.

OCR: Optical character recognition (a.k.a. optical character reader, OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image.

Pipeline: Pipeline is used to chain multiple estimators into one. This is useful as there is often a fixed sequence of steps in processing the data, for example feature selection, normalization and classification.

Feature Union: Feature union combines several transformer objects into a new transformer that combines their output.

REFERENCES

[1] Page, E.B. (1966), “The imminence of grading essays by computers”. Phi Delta Kappan, 47, 238-243

[2] Omran, A. and Aziz, M. (2013), “Automatic Essay Grading System For Short Answers In English Language”, *Computer Science,* published 10 September 2013, <http://www.thescipub.com/jcs/toc>

[3] O’Shea, J. and Bandar, Z. (2010), “Benchmarking Short Text Semantic Similarity”. Int. J. Intell. Inform. Database Syst., 4 : 103-120. DOI:10.1504/IJIIDS.2010.032437

[4] Tamrakar, A. and Dubey, D. (2012), “Query Optimisation Using Natural Language Processing”. Int. J. Tech., 3 : 307-310

[5] “The Hewlett Foundation: Short Answer Scoring | Kaggle” (2012) <https://www.kaggle.com/c/asap-sas>

[6] Ron Kohavi; Foster Provost (1998). "Glossary of terms". *Machine Learning*. **30**: 271–274

[7] Phil Simon (March 18, 2013). *Too Big to Ignore: The Business Case for Big Data*. Wiley. p. 89. ISBN 978-1-118-63817-0

[8] Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". *International Statistical Review*. **69** (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.

[9] Ferguson, Thomas S. (1982). "An inconsistent maximum likelihood estimate". *Journal of the American Statistical Association*. **77** (380): 831–834. doi:10.1080/01621459.1982.10477894. JSTOR 2287314.

[10] Bloom, B.S. (Ed.). Engelhart, M.D., Furst, E.J., Hill, W.H., Krathwohl, D.R. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain.* New York: David McKay Co Inc.

[11] H. Zhang (2004). "The optimality of Naive Bayes." Retrieved from <http://scikit-learn.org/stable/modules/naive_bayes.html>

[12] Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers

[13] J. Z. Sukkarieh and J. Blackmore, "c-rater: Automatic Content Scoring for Constructed Responses," *Twenty-Second International FLAIRS Conference*, 2009.

[14] "Bangladesh Education Stats". NationMaster. Retrieved 2016-09-12.

[15] M. S. M. Patil and P. M. S. Patil, "Evaluating Student Descriptive Answer Using Natural Language Processing," *International Journal of Engineering Research & Technology,* vol. 3, no. 3, Mar. 2014. [Online]. Available: www.ijert.org.

[16] A. Arafat and M. Raihanuzzaman, "Automated Essay Grading with Recommendation," *BRAC University,* pp. 73-74, April. 2016.

[17] Laurie Cutrone, Maiga Chang and Kinshuk, "Auto-Assessor: Computerized Assessment System for Marking Student's Short-Answers Automatically*", IEEE International Conference on Technology for Education*, 2011.

[18] Ade-Ibijola, Abejide Olu, Wakama, Ibiba, Amadi, J. Chioma, " An Expert System for Automated Essay Scoring (AES) in Computing using Shallow NLP Techniques for Inferencing", *International Journal of Computer Applications* (0975 – 8887) Volume 51– No.10, August 2012

[19] S. Burrows, I. Gurevych, and B. Stein, “The Eras and Trends of Automatic Short Answer Grading,” *International Journal of Artificial Intelligence in Education,* 25 (2015) 60 – 117, doi: 10.1007/s40593-014-0026-8

[20] K. P. Murphy “Machine Learning A Probabilistic Perspective,” *The MIT Press,* September 2012

[21] Narasimha Murty, M.; Susheela Devi, V. (2011). *Pattern Recognition: An Algorithmic Approach*. ISBN 0857294946

[22] Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". *International Statistical Review*. **69** (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.

[23] Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). *Tackling the poor assumptions of Naive Bayes classifiers* (PDF). ICML.