Automated Essay Grading with Recommendation

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

Bachelor of Science
In
Computer Science
And
Engineering

Under the Supervision of

Professor Md. Haider Ali
And
Co Supervision
of
Annajiat Alim Rasel

By
Arshad Arafat (12101114),

Mohammed Raihanuzzaman (12101029)

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

Declaration

This is to certify that the research work titled “Automated Essay Scoring With Recommendation” is submitted by Arshad Arafat and Mohammed Raihanuzzaman 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 and Engineering. 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 Professor Md. Haider Ali and co supervision of Annajiat Alim Rasel.

Signature of Supervisor:

_____

Professor Md. Haider Ali
Supervisor
Department of CSE
BRAC University

Signature of Co Supervisor:

_____

Annajiat Alim Rasel
CoSupervisor
Department of CSE
BRAC University

Signature of authors:

_____

Arshad Arafat
12101114

_____

Mohammed Raihanuzzaman
12101029

FINAL READING APPROVAL

Thesis Title: Automated Essay Scoring with Recommendation

Date of submission: 21 – 04 - 16

This final report on our research is read and approved by the supervisor Professor Md. Haider Ali. 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 & Engineering, School of Engineering & Computer Science, BRAC University

Signature of Supervisor:

_____

Professor Md. Haider Ali
Supervisor
Department of CSE
BRAC University

Signature of Co Supervisor:

_____

Annajiat Alim Rasel
CoSupervisor
Department of CSE
BRAC University

Acknowledgements

At first we would like to thank our thesis supervisor Professor Md. Haider Ali Sir for allowing us to work on this thesis under his supervision and for his inspiration, ideas and suggestions to improve this work. In many stages of our work we have found the support and help from our supervisor. We would also like to thank our co supervisor Annajiat Alim Rasel for all the help, support, valuable time he spent discussing ideas with us.

Abstract

In our thesis we have worked to analyze text essays then predict the score accordingly and also recommend similar essays as well as other noticeable required changes to the readers who want to improve their essay writing skills.

In our research we have used a dataset of 13000 essays scored by two human graders provided by the Hewlett foundation available in Kaggle. We have used different natural language processing techniques and enormous natural language tools and tried to see different patterns present in the essays to score them. We have extracted noticeable features from these essays created dataset with necessary formation then again used supervised machine learning models to build an artificial system that could score further user given essays and also make suggestion.

We have implemented a machine learning agent which is trained by linear regression algorithm on the extracted features to predict the score and then calculates cosine distance to determine similar helpful essays and recommends those essays to the users. Also we have developed our system to suggest the writer necessary correction of their mistakes and writing patterns.

Index

List of Figures

List of Abbreviation

NLP :	Natural Language Processing
ML :	Machine Learning
NLTK :	Natural Language Tool Kit
POS :	Parts Of Speech
IN :	Preposition or subordinating conjunction
JJ :	Adjective
JJR :	Adjective, comparative
NN :	Noun, singular or mass
NNS :	Noun, plural
NNP :	Proper noun, singular
NNPS :	Proper Noun, plural
PRP :	Personal pronoun
VB :	Verb, base form
VBG :	Verb, gerund or present participle
VBN :	Verb, past participle
VBP :	Verb, non-3rd person singular present
VBZ :	Verb, 3rd person singular present

CHAPTER 1

1.1 Introduction

Automated essay scoring (AES) has been in the research area of computer science since the early 1966. Predicting the score of an essay so that the score might seem like it has come from a human reader is a bit daunting task because there are numerous quantified features that have to be extracted from the essay as well as many unquantifiable properties like the perceptions of the writer while writing the essay and his thoughts that he is trying to inscribe on the paper. Therefore, the behavior of the essay inherently noisy, non-stationary and deterministically chaotic. The quantifiable data that we have extracted from the essay is relatively easy for the computer to process rather than processing the ideas or thoughts of the writer in the essay, which may or may not affect the scoring of an essay by a computer.

Analyzing natural language, or free-form text used in everyday human-to-human communications, is a vast and complex problem for computers regardless of the medium chosen, be it verbal communications, writing, or reading. Ambiguities in language and the lack of one "correct" solution to any given communication task make grading, evaluating or scoring a challenging undertaking. In general, this is a perfect domain for the application of machine learning techniques with large feature spaces, and huge amounts of data containing interesting patterns.

Automated essay scoring (AES) is the use of specialized computer programs to assign grades to essays written in an educational setting. It is a method of educational assessment and an application of natural language processing. Its objective is to classify a large set of textual entities into a small number of discrete categories, corresponding to the possible grades—for example, and the numbers 1 to 6.

Our research focuses on the part where to make an artificial environment learn and predict scores from the essays that it will receive and then also to find the essays that we could recommend the reader so that they can get the proper idea to improve their writing skills reading the recommended essays. Again we have also focused to find anomalies or patterns present in the essays that needs to be changed to improve the writing. Using a training dataset, the artificial environment can identify the patterns and tries to predict the next possible output. Our paper explores different natural language processing and machine learning approaches and how they can help with essay scoring.

1.2 Motivation

Many large-scale testing programs around the world include at least one essay writing item. Examples include the GMAT® test administered by the Graduate Management Admission Council®, the GRE ® revised General Test administered by ETS, as well as the Pearson® Test of English (PTE). The written responses to such items are far more complex than responses to multiple-choice items, and are traditionally scored by human judges. Human raters typically gauge an essay's quality aided by a scoring rubric that identifies the characteristics an essay must have to merit a certain score level. Some of the strengths of scoring by human graders are that they can (a) cognitively process the information given in a text, (b) connect it with their prior knowledge, and (c) based on their understanding of the content, make a judgment on the quality of the text. Trained human raters are able to recognize and appreciate a writer's creativity and style (e.g., artistic, ironic, rhetorical), as well as evaluate the relevance of an essay's content to the prompt. A human rater can also judge an examinee's critical thinking skills, including the quality of the argumentation and the factual correctness of the claims made in the essay. However, even having many positivity on human scoring unfortunately it has some hindrance. Starting with, qualified human raters must be recruited after that they must be clearly informed how to use the scoring metric system properly and their rating competencies must be certified before engaging into operational grading. Lastly, they must be carefully watched to ensure the quality and consistency of their ratings. Besides only in 2012, more than

655,000 test takers worldwide took the GRE revised General Test (ETS, 2013), with each test taker responding to two essay prompts, producing a total of more than 1.3 million responses. Obviously, involving humans in grading such high volumes, especially in large-scale assessments like the GRE test, can be labor intensive, time consuming, and expensive. Humans can also make mistakes due to cognitive limitations that can be difficult or even impossible to quantify, which in turn can add systematic biases to the final scores. That's what we are talking from the rater's perspective. Now let's think from the examinees perspective. Would it be helpful to get an automatic system where the examinee can practice their essay writing skills and see how they are doing? Would it help if the system not only just scores the essays but also find some similar relevant essays so that the examinee can read the essays and get the idea to make changes in his or her writing to score better? Also what if the examinee gets a system that not only says how many errors they have but also recommends them how to overcome them? The answer is yes. As a student we think we will be very grateful to get a system like that. And that is what motivated us to work on this topic.

Automated scoring has the potential to provide solutions to some of the obvious shortcomings in human essay scoring. Our research tends to build a system for computer-based scoring involve construct-relevant aggregation of quantifiable text features in order to evaluate the quality of an essay. This system work exclusively with variables that can be extracted and combined mathematically. Humans, on the other hand, make holistic decisions under the influence of many interacting factors. The primary strength of automated scoring compared to human scoring lies in its efficiency, absolute consistency in applying the same evaluation criteria across essay submissions and over time, as well as its ability to provide fine-grained, instantaneous feedback. Computers are neither influenced by external factors (e.g., deadlines) nor emotionally attached to an essay. Computers are not biased by their stereotypes or preconceptions of a group of examinees. Automated scoring can therefore achieve greater objectivity than human scoring. Automated scoring systems are often able to evaluate essays across grade levels (e.g. the e-rater engine,

Intelligent Essay Assessor, Vantage Learning's IntelliMetric®). Human graders, in contrast, are usually trained to focus on a certain grade range associated with a specific rubric and a set of tasks. Shifting a human rater to a new grade range -may therefore require considerable retraining.

1.3 Limitation of data and ambiguities in it

Finding the right sets of data for any research is really a challenge and for our research it is basically the same. The problem is that in order to make our environment learn and extract features from the essays we need to have a large collection of essays which have been already been marked by at least two different raters to maintain the consistency among the raters and this collection is not readily available for free. Moreover, it is equally harder to get that collection digitally since most of the educational institutions in Bangladesh prefer their students to write on the paper instead of typing to a computer. And on top of that, those few institutions who do let their candidates to type essays on a computer are not so cooperative to hand over data to us due to their own policies. Fortunately, due to the advancement in technology particularly the internet there is one if not many public domain who likes to give data away for free in order to make the world a better for research students like us thus, we gathered our data from Kaggle.com which is being used by data scientists around the world for performing different machine learning and data analysis work.

The data set we gathered from Kaggle.com contain 8 different sets of collections of essays and each collection have around 1700 essays which have already been rated by two different human raters. However, out of those 8 different sets we first put our concentration in the first set and found ourselves in a pool of ambiguity. English language or any other language around the world have so much confusion init that it becomes a very indomitable task to pinpoint which way of expression is actually the right way because the same sentence could be written in some many ways along with way too much variations in it. And on top of that, to bring all these forms which represent the same idea in a sentence to a binary level which is the

only form a computer can understand takes the complexity level to a whole new dimension. Luckily, Natural Language Processing (NLP) is also a huge research area in computer science arena since the early 60's or even before that, that many researches have come a long way and developed a python package Natural Language Toolkit (NLTK). Even though NLTK does a very good job in parsing and processing many difficult things like Stop word parsing or finding the spelling mistakes but all these things are quantitative and there is no efficient way to make an automated system understand the actual meaning or the creative skills of any essay. Consider this example:

"A ship-shipping ship shipping shipping ships (A boat-delivering boat delivering delivery boats)"

This particular example above is hard for even humans to interpret and it is couple of times harder to make it understand to a computer which only knows binary. However, we believe as the analysis tools improve and understand the perspective of the user, text analysis will be more relevant and accurate.

1.4 Research Goal

Automated essay scoring has been a very fascinating research area among the computer scientists since early nineties. As much as it is challenging it is also rewarding with its significant knowledge gain. Simply by making a computer understand to one of the human languages along with the complexities and dimensions of that language and on top of that rating a person's essay while suggesting some improvements is absolutely very demanding brainstorming thing as well as a solution for many real world scenarios. Scenarios where thousands of students around the world are writing some sort of essays for various reasons need to be checked by hundreds of qualified people is really a time consuming and laborious task.

This research has been a platform for us to learn about NLP, how it works and learning about the methods of machine learning along with many existing algorithms and combine them with NLP to make a stable AI that can help the teachers for grading the essays at the same time providing relevant suggestions to a writer that would help him to improve his writing. We have also been trying to see how accurate the

current prediction algorithm works and how this would help us to attain our goal. Furthermore, this research has enabled us to know more about the automated suggestion system and experiment it with the machine learning algorithm to generate a stable model that would serve our purpose.

The main objective of this research is to build an intelligent cloud based system which could automatically grade an essay as well as giving valuable suggestions that would improve the writer's ability to write a better essay so that he could become a commendable writer. By putting this thought in our mind we have started working with the existing different machine learning algorithms to see which one predicts the better score of an essay with less prone for error. To begin this process we must carefully select a dataset where there are relatively large number of essays on a particular topic and those essays are already been marked by at least two different qualifies rates to maintain the consistency among the raters. Once we get our desired dataset we must start extracting the features from those collection of essays before feeding them into the algorithm. As a result, we have turned our attention to python's Natural Language Toolkit (NLTK) for Natural Language Processing (NLP) to extract features like word count, sentence count, and sentence to word ratio and many more, in fact we have extracted 19 different features from every single essay before running the machine learning algorithm. Moreover, after prediction of score we are suggesting few improvements to the test essay from those collection of essay by which an user could go through those suggested essays just to make that person's next essay little bit better from the previous one. We are putting our effort with the thought in the back of our minds that someday in the future computers will be able to check any kind of essay with highest degree of precision and giving relevant academic suggestions as much as possible to anyone who uses this system. We strongly feel that this is quite achievable due to advances in the Internet which will remove the space related issues where almost all the essays around will be saved into a cloud server in order to make a machine learn more and more about different topics of essays. Therefore, this research is just a tip of iceberg compare to what we think is waiting for us in the future.

Chapter 2

2.1 Literature Review

As mentioned above essay scoring using computers has been a piece of attraction to the researchers since early sixties though it became very much fruitful in late 90s.IBM the corporate giant focused to rate essays automatically in 1938. They manufactured IBM 805 that can score an essay selecting some responses. However the earliest noticeable piece of work found in this field is the Project Essay Grade (PEG) by Ellis Page in 1960. Page believed that the information of an essay can be hypothetically divided so as the style of the essay. That's why he developed the PEG to score the essays based on the style of the candidate essays. In 1968 he published that there are two kinds of variables that are present in essays. He defined them to be proxes and trains. Proxes are the variables or indicators that a computer can recognize. Trins are on the other hand only a human rater can understand. So basically he worked with the proxes that is the surface variables like the word count, length of the essay and 30 other indicators and then weighed some regression equation to predict the score.

In 1998 Burstein, Braden-Harder, Chodorow, Hua, Kaplan, Kukich, Lu, Nolan, Rock, & Wolff developed some model that is named the E-Rater which used multiple linear regression model to score essays. The second version of this model is still in use by ETS where the researchers have narrowed down the number of indicators to be used to score the essays.

As both E-Rater and PEG works based on surface features that's why it got criticized by researchers that's why in 2007 Ben-Simon and Bennett worked with semantic analysis of essays to predict the scores. It mainly focused on the meaning of words sentences or paragraphs. Prior to Bennet in 1999 Foltz,Landauer and Laham also worked with semantic analysis based automatic scoring which mainly focused on essay content rather than the style. In this case they developed the model that can score the candid essay based on the similarity of the essays based on the system was trained. Whenever some unique essays appeared as a candid essay this system used to mark it as anomalous essay and that essay was manually scored by a

human rater. Latent semantic analysis based tools were proved much more efficient in scoring the essays than the surface indicator based system.

Another approach towards automatically scoring the essays emerged later on that is based upon the text categorization technology. Williamson in 2001 used Bayesian classification techniques in text categorization and came up with an algorithm to predict the score of the essay. Larkey in 1998 developed a tool that can predict the score of an essay using text categorization with 12 surface indicators and regression model.

2.2 Methodological Biases

In our research as we have used previously scored data values to predict the future upcoming essays that's why it involves some sort of biases to the context. If the test essay doesn't match the training essays our system may score wrongly. Again there may be some key texts present in the essays that is not relevant to the essays that might change the weight of the essays. Again automatic essay scoring tool cannot understand the depth of the writing that a human reader can feel as writing in some of the cases are mostly measured by the art of writing rather than the indicators. However machine cannot understand these art of writing so in these kind of abstract writing relying on the tools cannot be efficient. Again tools to score essays automatically are pros to manipulation of the user. If the user knows how these tools work then they might manipulate the system and get higher scores without writing quality essays.

Chapter 3:

3.1 Thesis Workflow

In this chapter we will discuss the workflow that we have maintained throughout our thesis work. The following flowchart demonstrates the procedures or work research approach.

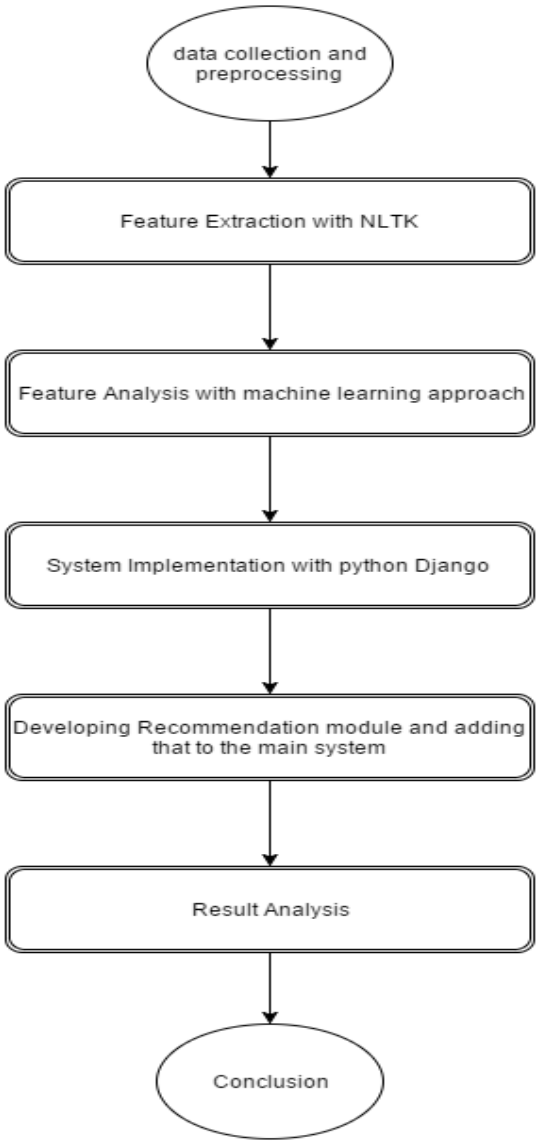


Figure 3.1: Thesis workflow

At the very beginning right after finding the problem we would like to work on, the first challenge was to collect relevant data to train the system. In most the cases we find data that is full of so much noises that we have to abandon most of it. So finding a usable dataset is very is very important for the research. After much searching we finally got some usable format of data in Kaggle. Though we had to clean the data in some places but still they helped us a lot. In chapter 5 we have discussed this in detail how we have collected all the data and cleaned them to extract noticeable indicators to predict the score. Later on in that same chapter 5 we have discussed how with the help of natural language toolkit we extracted all the features needed for our research.

After getting a desired dataset in proper format we then analyzed each of them individually to see whether they have some relation with the score updated or not. We have used Octave an open source tool to implement Simple Linear regression and Random Forest regressor for each of these case. Right after the analysis of the features we then used the noticeable features in our final Linear Regression algorithm and Random Forest Algorithm. In chapter 6 we have discussed it vastly.

After all the necessary research done to build the main regression model. We concentrated on building the server based system that can interact with the user. The system that we have developed can collect essay from users and predict the score that the essay may get by an experienced human grader. It can then make suggestion of proper changes to make in the essay to make it better. Also suggest the user some more essays reading which the user can have a very good understanding to make his or writing better. The system is developed with the python Django framework and in the software we have implemented all our previous research findings. The whole procedure is discussed in the Chapter 7.

Chapter 8 discusses how we came up with the idea to make the recommendation module. We studied different ways to find similarities among objects or documents. We implemented them to see the result. And finally we developed the module and added them in the Server Based system.

In chapter 9 we have demonstrated the results and errors that we have found and shown the result and compared them with each other. And finally in chapter 10 we have concluded our thesis and represented our thoughts on the future work that we are passionate to achieve.

Chapter 4

4.1 Machine learning

Instead of writing programs that explain to computers how to perform specific task, in machine learning, programmers write algorithms that explain computers how to learn by themselves to perform tasks. Machine Learning algorithms is a computer program that teaches computers how to program themselves so that every time a human programmer does not have to explicitly describe a computer to perform the task that we want to achieve. The need to make a program that explains to computers how to perform each task is the greatest limitation faced by traditional computer science programming. It has prevented computers from further extending our intelligence to solve more complex tasks. To truly extend our intelligence, we need computers to accomplish tasks that we don't even know how to do. However, machines do not learn in the way humans do, humans have specialized pattern discovering abilities that allow humans to extract patterns but machines have no idea of what so ever a pattern is. Fortunately, there is something called feedback mechanism that in some ways mildly similar. Machines "learn" when they take a series of input data items and, based on some mathematical criteria, they correctly chose an algorithm (a pattern of sorts) to apply to that input so that the output is acceptable to the user. Being accepted or not accepted is important because that feedback information accumulates and feeds into the selection criteria used to select the algorithm to use. It's a closed feedback loop. The current machine learning algorithms that exist are designed to solve a single well defined problem, often better that 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 people's expectation of human intelligence because they tend to learn on thing very precisely at a time but not more than one thing.

Machine Learning has been categorized into 3 categories precisely:

Supervised Learning, in supervised learning the data has been divided into two subgroups. First group is called the training set and the later called the test set, with the training set a model has been prepared where

a machine has to predicts outcome based on that model and corrected when there is a wrong prediction. The training process continues until a desired level of accuracy has been attained. After that, the model that has been prepared has to be tested with the test set to see how it actually works in the real life.

Unsupervised Learning, input data is not labelled or categorized and does not have a known result. Therefore, a model is prepared by examining the features that have extracted from the input data. The data may be extracted through a mathematical process to systematically reduce unwanted things, or by categorizing the data by some similarity.

Semi-Supervised Learning, Input data is a mixture of labelled and unlabeled examples. There is a desired prediction problem but the model must learn the structures to organize the data as well as make predictions.

4.2 Linear Regression

If we have a bunch of points on a chart in 2D (for example), often we want to fit a straight line through those points. We can draw it by hand, and that would be a form of linear regression. But if we want to do it mathematically, with some replicable rationale for the line we've drawn, we need to figure out how. Or in other words linear regression is a mathematical technique which is used to find a straight line (hence linear) which best fits some set of data. This is usually done by finding the line that minimizes the least squares error (which is the square of the distance from the data point to the line).

One very popular technique is to find the line that minimizes the sum of the square of the distances from the line to the points. This happens to work particularly well when the errors (distances of the points from the line) follow a normal distribution, which they often do. This gives you the line that best predicts where future points would lie, if more were added.

4.3 Random Forest Algorithm

In 1980 Leo Breiman and Adele Cutler came up with an idea of an algorithm then can do both classification and regression. Later on Ho Tin Kam, Dietterich, Amit and Geman joined them and initiated the early foundation of the Random Forest algorithm. In addition to classification this algorithm can also handle the missing values or other outliers in data set, or other data exploration work .Other top pros that invoked us to use this algorithm are the following: it can handle thousands of variables without deleting them, maintains accuracy when large proportion of the dataset is missing in fact it estimates the values for the missing data and many more.

The insight Behind Random forest is that instead of training one single classifier it trains multiple weak classifier that classifies some particular attributes and the target variables. Later on Random forest collects the result from each of the weak classifier and comes up with the actual classification. In other words we could say each weak classifier votes for its defined attributes and then creates the ultimate result. In case of regression the weak regressors comes with some continuous values and then their average is taken as the predicted output. It is mainly based on the popular decision tree or we could say it is based on the popular bagging approach. The algorithm is as follows (for both classification and regression)

1. If the number of cases in the training set is N, sample N cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

Now with Random forest algorithm there comes error that depends these following two things

The *correlation* between any two trees in the forest. Increasing the correlation increases the forest error rate.

And

The *strength* of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

4.4 Octave

GNU Octave is an open source, free mathematics program modeled after MATLAB (matrix laboratory). It is very similar to Matlab and will run most Matlab code with minor modifications. Matlab differs from most mathematical software in that the focus is on matrix manipulation. For example a time series is a 1xN matrix. Once data is captured in matrix variable, most other mathematical and signal processing functions can work on the variable like it was a standard algebra variable. We would prefer for ourselves to go with Octave or MatLab since it abstracts out a lot of implementation details from us. We need not really know the internals of Octave, we just need to know what is required for us to solve the problem. As Andrew NG says, with Octave / Matlab, we will not need to bother about implementing logic doing matrix multiplication or for fitting curves. We only have to solve the problem in hand applying Machine Learning techniques. Of course, we will have the libraries for doing the same in Python as well, but we are better off learning that later. It's better to take one thing at a time.

4.5 Natural Language Processing

Natural language processing is the combination of computer science, computational linguistics and artificial intelligence. This field of study is related to the human-computer interactions. Natural language processing is the study that enables computers to understand the meaning of natural languages. The language research with the help of machines started back in 1950’s. Modern NLP algorithms are mainly based on Machine learning and Statistics. Some of the major tasks in natural language processing are automatic summarization, translation, morphological segmentation, named entity recognition, semantic

analysis, optical character recognition, parsing, question answering, speech analysis, information extraction, information retrieval etc.

4.6 Natural Language Toolkit (NLTK) & NLP

One of the most important ingredients for our research is python's package NLTK. We have used NLTK for the little bit of NLP that we have in our research to find out many features as possible from the test essays before putting those features into our octave machine learning code. After getting the data set from Kaggle site our next step would be to come up with a NLP system. NLTK would provide us with most of the basic requirements. There is not much say about NLTK because this package is so well known that whenever anyone wants to do some twitching with natural language thing with a machine and that person is new in this sort of field then his best option to go for NLTK and start playing with it. However, even NLTK has go to a long way albeit it has come a long from where it started but in order to handle the complexities in language this package has lots of potential rooms for improvements.

4.7 Numpy

The python programming language was not originally designed for numerical computational jobs however, it soon began to capture the attention of many scientific/engineering communities early on. Therefore, people have started working onto something that would allow researchers to use python for heavy calculation. Finally, in the early 2005 Numpy was released as a standalone package that would be very easy to install and light weighted as well. So Numpy is an extension of python programming language, adding support for large, multidimensional arrays and matrices along with a very elaborate library of high-level mathematical functions to operate on these arrays. Nonetheless, Numpy arrays must be views on

contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation. Algorithms that are not expressible as a vectored operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include scipy. weave, numexpr and Numba. Cython is a static-compiling alternative to these.

4.8 pandas

One of the most versatile and powerful data analysis toolkit in python is pandas. It is a python package that provides fast, reliable and expressive data structures designed to make working with both relational and labelled data. . It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal. We have come across pandas when we have to feed our **CSV** file into our regression algorithm. Is has the Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format

4.9 Language-check

Language-Check is also a python package, we have used this to check the type of grammar errors are there in any particular essay and also count those errors. This package is still under beta version so we are not expecting it to detect some complex grammatical errors but it does manage to handle common errors very easily. On top that this package provides suggestion which we have used in our recommendation system in order to let the writer know the whereabouts of his/her mistakes.

4.10 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 to suggest better words so that the writer can improve his writing to earn better score next time when he submits an essay.

4.11 Django

Django is a high-level python web framework that is built to for rapid development of any kind of web related work. In our research we used Django 1.9 which is currently the latest one at the time of doing this research. Django comes with built in sqlite database although different database can be used but for this research we decided that sqlite would suffice. The main idea of using Django is that, most of our code has been written in python so it is lot easier to run to python code on the platform that understands it. Moreover, one of our main objectives is that we want to make a server where essays around the world be stored after that from those stored essays. From there we have a plan to extract important features and feed those featured into the machine learning algorithm. Basically, the more essays the server have the better a machine can learn therefore the better that machine can predict sores and suggests improvements.

4.12 Recommender system

While browsing internet we often come across some automatic system that pushes various information to our browsers. For example an online shopping cart. We can see some of these sites recommends us to buy particular things. The question is how they know what to recommend us. Not

necessarily what they recommend us are the same as what they recommend others. This part of the system is said to be recommender system or recommendation system. It is derived from the information filtering system and became very popular recently. Recommender system is now widely used all over the world. Corporate giants like Amazon, Netflix, Hulu and many others use them all the time. Basically recommender system are of the following few categories

Collaborative Filtering: This is what happens when the recommender system analyses our previous records to recommend newer products. For example if I buy a sunglass from some online shop and then revisit the site again the recommender system of that site will recommend us newer collection of sunglasses. In this case the system processes large amount of data regarding user's behavior. There are many algorithm that works very effectively in this regard, such as K-nearest Neighbors, Pearson Correlation and many others. .

Content Based Filtering: In this case the system analyzes the description of the items and also the profile created for the user. It is mainly originated from information filtering or information retrieval research. Bayesian classifiers, Artificial Neural Network. Cluster analysis etc works very well if we choose Content based filtering as a method of recommendation.

Hybrid Filtering: Hybrid is the combination of both the filtering method discussed above. Netflix uses this type of filtering system recommending films to the users. It analyses the previous records of the user as well as previous record of the other similar users. As well as the system analyses the movies the user has highly rated. Then finally the system makes the decision of the films to be recommended.

Chapter 5

5.1 Data Collection

Supervised machine learning is totally dependent upon the condition of the data. A good and credible dataset can create the difference between good and bad machine learning agent. In our case finding a large dataset of essay that could be used in our research wasn't easy. After much hassle we have found the right usable dataset in Kaggle.com. Kaggle is a platform of the data science and predictive modeling. Numerous organizations put their valuable data in this platform to draw the attention of the researchers who could find a pattern and in the dataset or predict valuable output from that dataset. The dataset that we have used was provided by the William and Flora Hewlett foundation mostly known as the Hewlett foundation. They mostly sponsor data scientists and machine learning experts to solve social problems. Recently they have sponsored Automatic Essay Scoring and provided some quality data.

The dataset that we have used is a spreadsheet that contains nearly 17000 essays written by students of grade 7 to grade 10. The essay is divided into 8 sets. And these essays are marked by two human rater.

The dataset contains the following parameters

Essay_id: Its an unique identifier used to differentiate each students essay

Essay_set : The essays varies in 8 different essay set number. Each set contains similar topic of the essay

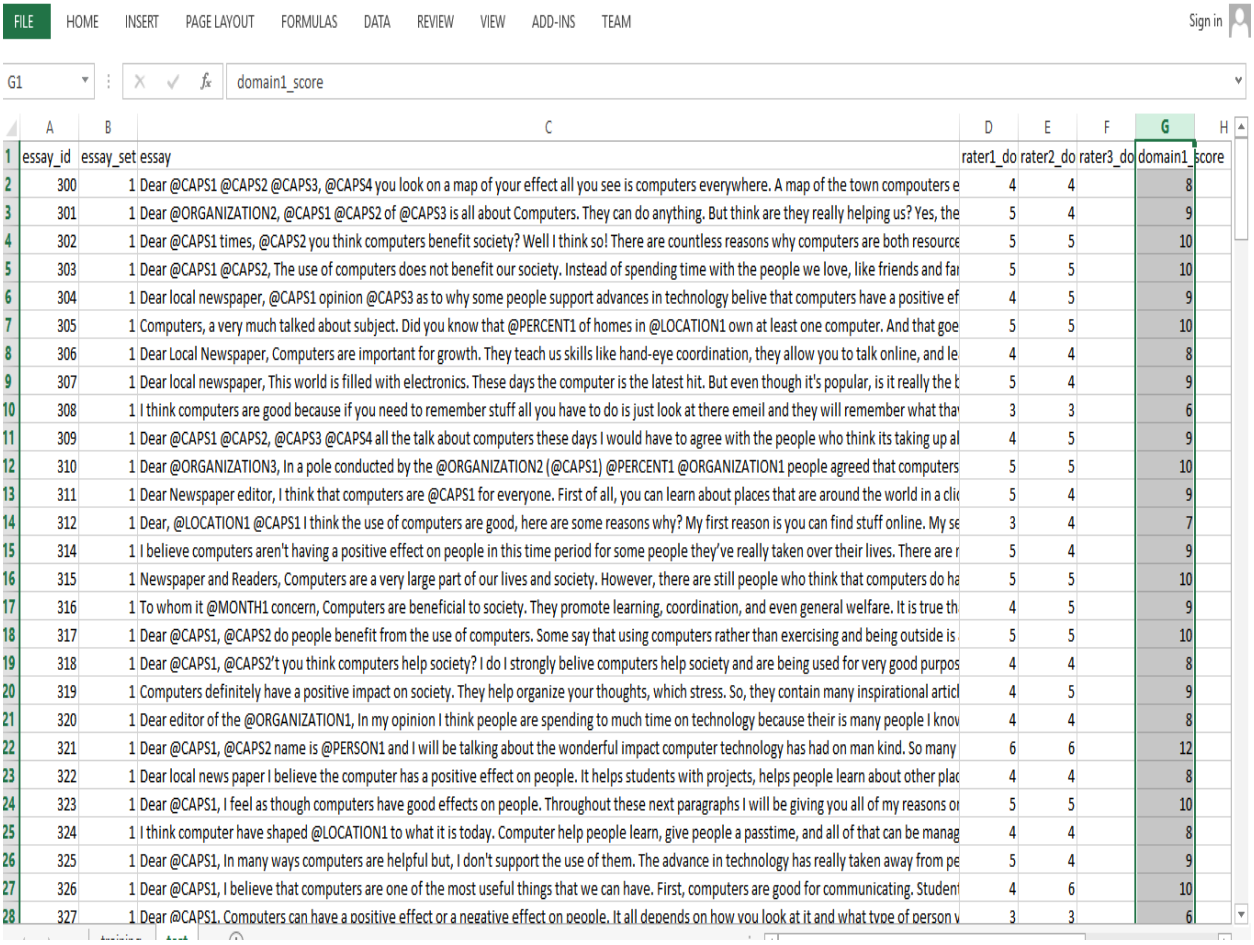
Essay: this column contains the ASCII value of the essays or the essay texts written by the students.

Rater1_domain1: the next column contains the essay score rated by a teacher

Rater2_domain1: It contains the essay score rated by second teacher

Domain1_score: It is the resolved score between the raters

A screenshot of the dataset is presented here



	A	B	C	D	E	F	G	H
1	essay_id	essay_set	essay	rater1_do	rater2_do	rater3_do	domain1_score	
2	300	1	Dear @CAPS1 @CAPS2 @CAPS3, @CAPS4 you look on a map of your effect all you see is computers everywhere. A map of the town compouters e	4	4		8	
3	301	1	Dear @ORGANIZATION2, @CAPS1 @CAPS2 of @CAPS3 is all about Computers. They can do anything. But think are they really helping us? Yes, the	5	4		9	
4	302	1	Dear @CAPS1 times, @CAPS2 you think computers benefit society? Well I think so! There are countless reasons why computers are both resource	5	5		10	
5	303	1	Dear @CAPS1 @CAPS2, The use of computers does not benefit our society. Instead of spending time with the people we love, like friends and fa	5	5		10	
6	304	1	Dear local newspaper, @CAPS1 opinion @CAPS3 as to why some people support advances in technology belive that computers have a positive ef	4	5		9	
7	305	1	Computers, a very much talked about subject. Did you know that @PERCENT1 of homes in @LOCATION1 own at least one computer. And that goe	5	5		10	
8	306	1	Dear Local Newspaper, Computers are important for growth. They teach us skills like hand-eye coordination, they allow you to talk online, and le	4	4		8	
9	307	1	Dear local newspaper, This world is filled with electronics. These days the computer is the latest hit. But even though it's popular, is it really the b	5	4		9	
10	308	1	I think computers are good because if you need to remember stuff all you have to do is just look at there emeil and they will remember what tha	3	3		6	
11	309	1	Dear @CAPS1 @CAPS2, @CAPS3 @CAPS4 all the talk about computers these days I would have to agree with the people who think its taking up al	4	5		9	
12	310	1	Dear @ORGANIZATION3, In a pole conducted by the @ORGANIZATION2 (@CAPS1) @PERCENT1 @ORGANIZATION1 people agreed that computers	5	5		10	
13	311	1	Dear Newspaper editor, I think that computers are @CAPS1 for everyone. First of all, you can learn about places that are around the world in a clic	5	4		9	
14	312	1	Dear, @LOCATION1 @CAPS1 I think the use of computers are good, here are some reasons why? My first reason is you can find stuff online. My se	3	4		7	
15	314	1	I believe computers aren't having a positive effect on people in this time period for some people they've really taken over their lives. There are r	5	4		9	
16	315	1	Newspaper and Readers, Computers are a very large part of our lives and society. However, there are still people who think that computers do ha	5	5		10	
17	316	1	To whom it @MONTH1 concern, Computers are beneficial to society. They promote learning, coordination, and even general welfare. It is true th	4	5		9	
18	317	1	Dear @CAPS1, @CAPS2 do people benefit from the use of computers. Some say that using computers rather than exercising and being outside is	5	5		10	
19	318	1	Dear @CAPS1, @CAPS2't you think computers help society? I do I strongly belive computers help society and are being used for very good purpos	4	4		8	
20	319	1	Computers definitely have a positive impact on society. They help organize your thoughts, which stress. So, they contain many inspirational articl	4	5		9	
21	320	1	Dear editor of the @ORGANIZATION1, In my opinion I think people are spending to much time on technology because their is many people I knov	4	4		8	
22	321	1	Dear @CAPS1, @CAPS2 name is @PERSON1 and I will be talking about the wonderful impact computer technology has had on man kind. So many	6	6		12	
23	322	1	Dear local news paper I believe the computer has a positive effect on people. It helps students with projects, helps people learn about other plac	4	4		8	
24	323	1	Dear @CAPS1, I feel as though computers have good effects on people. Throughout these next paragraphs I will be giving you all of my reasons or	5	5		10	
25	324	1	I think computer have shaped @LOCATION1 to what it is today. Computer help people learn, give people a passtime, and all of that can be manag	4	4		8	
26	325	1	Dear @CAPS1, In many ways computers are helpful but, I don't support the use of them. The advance in technology has really taken away from pe	5	4		9	
27	326	1	Dear @CAPS1, I believe that computers are one of the most useful things that we can have. First, computers are good for communicating. Student	4	6		10	
28	327	1	Dear @CAPS1, Computers can have a positive effect or a negative effect on people. It all depends on how you look at it and what type of person v	3	3		6	

Figure 5.1: Screenshot of the dataset when collecting

5.2 Data Preprocessing

Data preprocessing is the first step of any data mining approach. Data preprocessing is needed to convert the raw unordered unusable data to structured usable format. Also dataset contains a lot of outliers and other noises that could affect the research in a negative manner. However the dataset that we have collected was

very much in shape to be used in our research. Slightly modification was made to remove the outliers and other noises like some missing values removal and so on. Later on we have made the following modification to our dataset while using them so that we could use them in better way.

5.2.1 Word tokenization

Tokenization is a processing of converting a large text into single pieces of tokens. In our case we have tokenized each essay before using them for feature extraction. We have mainly used python NLTK's word Tokenize and Sentence Tokenize module to tokenize our essay here is a output of a tokenized essay using nltk word tokenizer.

Figure 5.2 List of all the tokenized words from one single essay

5.2.2 Removing Stop Words

In natural language processing Stop words is referred as a collection of words that is mostly present in each and every texts and that are mostly useless in natural language processing research. In fact removing those gives better result. These lists contains words like a, an, the etc. And also the punctuations like brackets and other symbols. In nltk there is a module named stop words that contains the list of words that could be removed from our essays. Though we need a slight modification to enrich the corpus of stop words a it does not include some of the most common words used today. Here is a code snippet that shows how to use

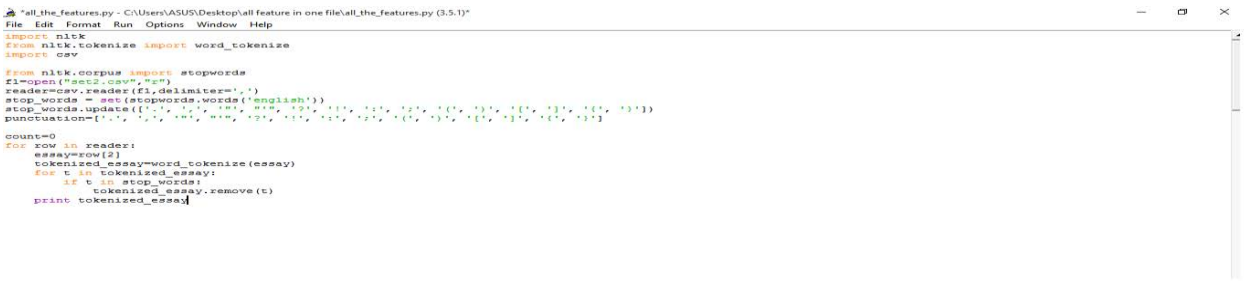


Figure 5.3: Code to remove the stop words

And after removing stop words the above printed essay looks like the following one

Figure 5.4: list of tokens after removing stop words

This time the essay contains less tokens than the previous one.

5.2.3 Stemming and lemmatization

Stemming is a process used in natural language processing that helps to transform the word to its base form or the stem form. For example “am, is, are” all of these can be transformed into base word “be”. Car, cars, car’s, etc. can be traced down to car. For grammatical reason words are used differently in texts. If we can trace them down to their base form it will help us greatly in processing our data. Among many algorithms used to stem the words the porter algorithm is well appreciated. python nltk has a build in support for porter algorithm . We have used this algorithm and reduced the dimensionality of each essays.

5.3 Feature Extraction

After collecting and preprocessing our data we have used different techniques from natural language processing to extract various noticeable features or indicators from the essays so that we can train our regression model to predict score. We have gathered 19 features from nearly 17000 essays. The features are the following.

Word count after removing stop words: after removing the stop words we have counted the number of words present in each essays

Sentence Count : using nltk sentence tokenizer we have counted the number of sentence each essay contains

Ratio of words and Sentence: we have calculated the ratio of the words and the sentences and kept them in the feature list

English words: as our experiment is totally based on the so we have extracted the number of English words present in the essay. Python nltk has two enriched corpus of English words those are the Word net and the Words. We have used these corpus to find the word's and made the count.

Non English word: The essays may contain non English words and other numeric values or in some cases they contains symbols or other languages so we made a count of these words and compiled them in the feature list.

Error: Errors are the most important part of essay writing. The number of grammatical errors, style errors, spelling errors and other errors affect the score of a quality essay. We have used a python nltk based framework to find the number of errors and counted them. Also we have manually used python word net and words to check the number of misspelled words in the text.

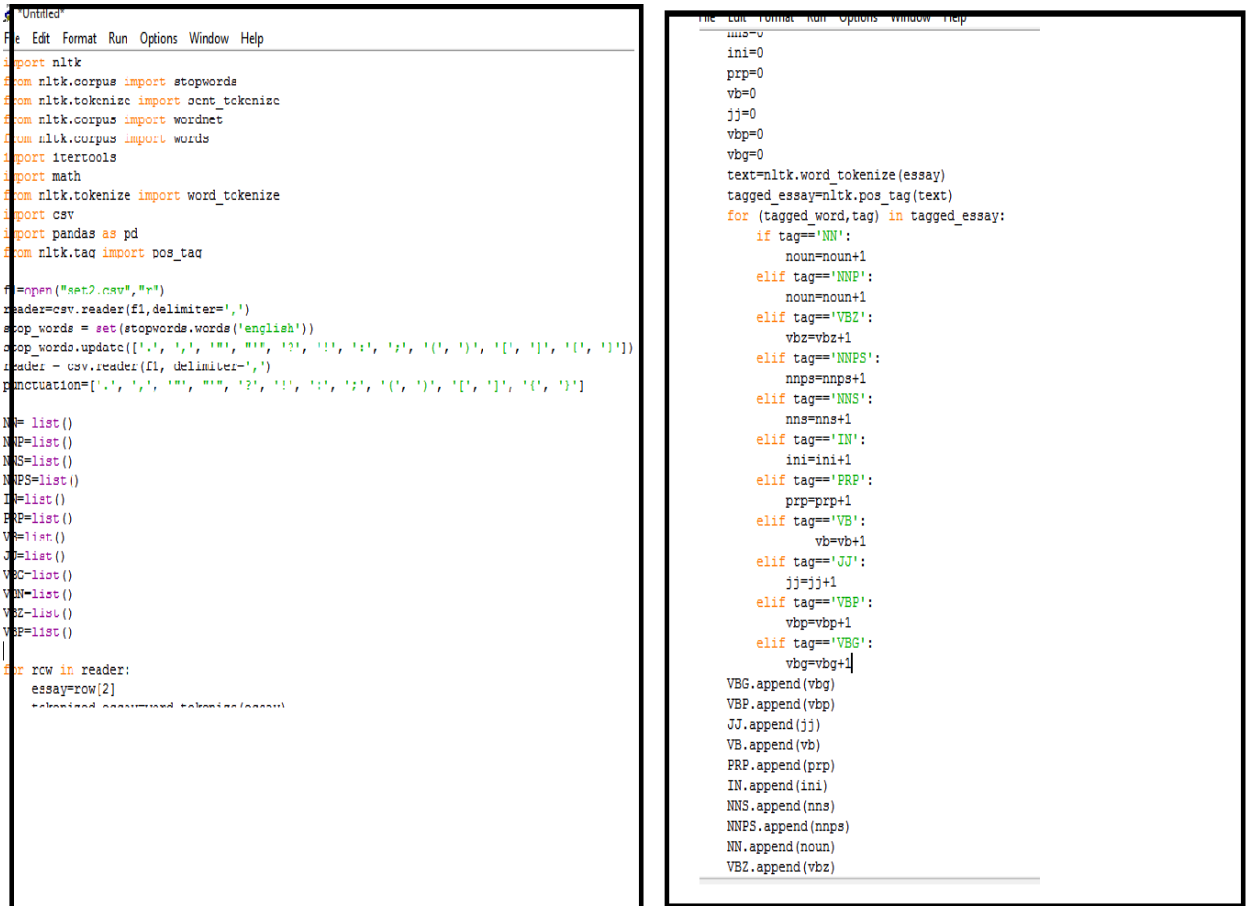
Total number of characters: Using python nltk we have calculated the number of characters present in the essays.

Parts of Speech (POS Tagging) : In natural language processing pos tagging plays a great role. POS tagging means tagging the role of the tokens in the sentence. We have used nltk pos_tag a module

from the python nltk to find different category of the parts of speech present in the text. We have used this module to find the following few categories of tokens.

IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PRP	Personal pronoun
VB	Verb, base form
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present

In the following figure we have shown two screenshot from the code we have used to identify and compile these parts of speech ..



```
*Untitled*
File  Edit  Format  Run  Options  Window  Help
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize
from nltk.corpus import wordnet
from nltk.corpus import words
import itertools
import math
from nltk.tokenize import word_tokenize
import csv
import pandas as pd
from nltk.tag import pos_tag

f1=open("set2.csv","r")
reader=csv.reader(f1,delimiter=',')
stop_words = set(stopwords.words('english'))
stop_words.update(['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}'])
reader = csv.reader(f1, delimiter=',')
punctuation=['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}']

NN= list()
NNP=list()
NNS=list()
NNPS=list()
IN=list()
PRP=list()
VB=list()
JJ=list()
VBG=list()
VBN=list()
VBZ=list()
VBP=list()

for row in reader:
    essay=row[2]
    tokenized_essay=word_tokenize(essay)
```

```
File  Edit  Format  Run  Options  Window  Help
    nns=0
    ini=0
    prp=0
    vb=0
    jj=0
    vbp=0
    vbg=0
    text=nltk.word_tokenize(essay)
    tagged_essay=nltk.pos_tag(text)
    for (tagged_word,tag) in tagged_essay:
        if tag=='NN':
            noun=noun+1
        elif tag=='NNP':
            noun=noun+1
        elif tag=='VBZ':
            vbz=vbz+1
        elif tag=='NNPS':
            nnps=nnps+1
        elif tag=='NNS':
            nns=nns+1
        elif tag=='IN':
            ini=ini+1
        elif tag=='PRP':
            prp=prp+1
        elif tag=='VB':
            vb=vb+1
        elif tag=='JJ':
            jj=jj+1
        elif tag=='VBP':
            vbp=vbp+1
        elif tag=='VBG':
            vbg=vbg+1
    VBG.append(vbg)
    VBP.append(vbp)
    JJ.append(jj)
    VB.append(vb)
    PRP.append(prp)
    IN.append(ini)
    NNS.append(nns)
    NNPS.append(nnps)
    NN.append(noun)
    VBZ.append(vbz)
```

Figure 5.5: code to find Part of speeches in essays

And in the following figure we have presented some screenshots of the python code written to collect the above stated features and compile them in the feature list.

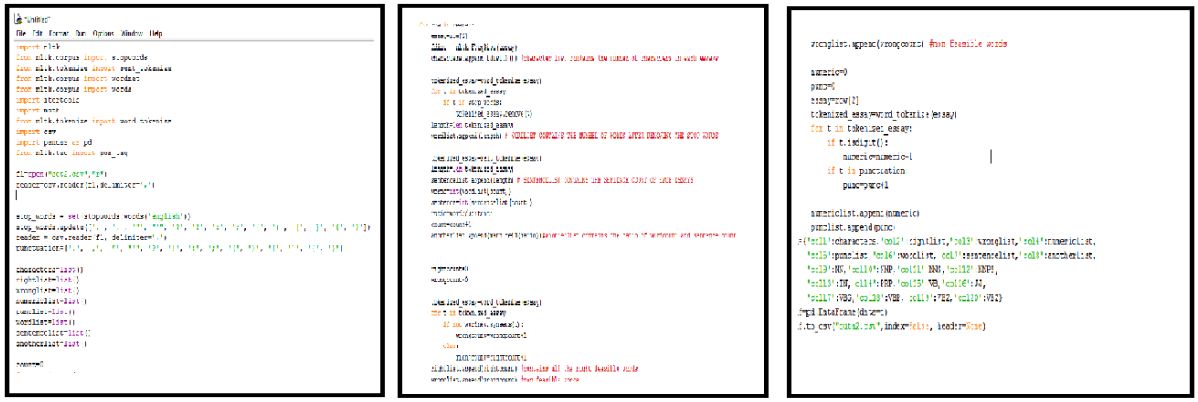
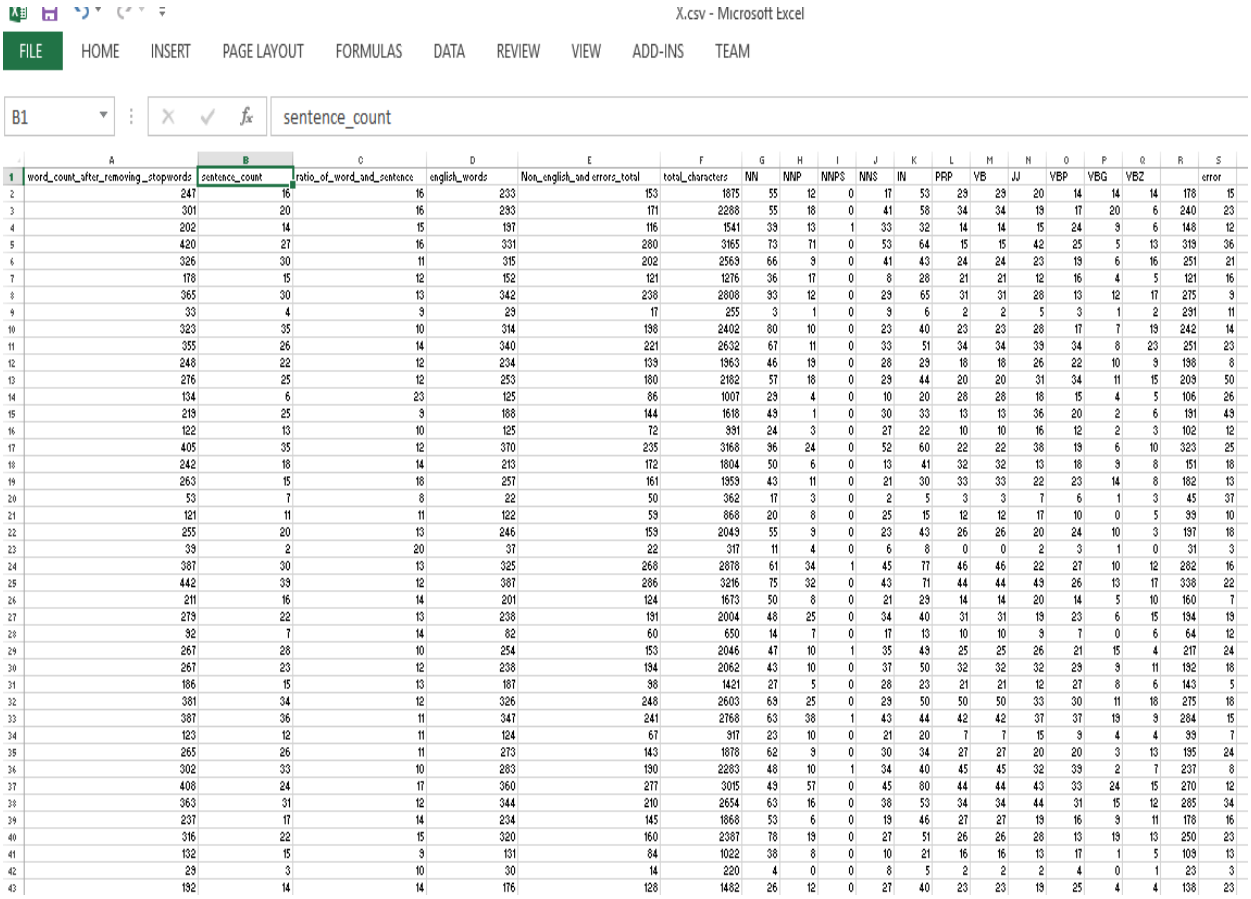


Figure 5.6: code to extract all the features and save them in the feature list

After running these code for a nearly 15 minutes in our environment composed of intel core i5 2.39 GHz processor and 4GB Memory. We have finally got the features created in a csv file where each column contains one feature for all the ~17000 essays .In the following page we have showed a screenshot of the csv feature dataset where we have kept all the features that we have generated from each of the essays.



word_count_after_removing_stopwords	sentence_count	ratio_of_word_and_sentence	english_words	Non_english_and errors_total	total_characters	NN	NNP	NNPS	NNS	IN	PRP	VB	JJ	VBP	VBG	VBZ		error
247	16	16	233	153	1875	55	12	0	17	53	29	29	20	14	14	14	178	15
301	20	16	293	171	2288	55	18	0	41	58	34	34	19	17	20	6	240	23
202	14	15	197	116	1541	39	13	1	33	32	14	14	15	24	9	6	148	12
420	27	16	331	280	3165	73	71	0	53	64	15	15	42	25	5	13	319	36
326	30	11	315	202	2569	66	9	0	41	43	24	24	23	19	6	16	251	21
178	15	12	152	121	1276	36	17	0	8	28	21	21	12	16	4	5	121	16
365	30	13	342	238	2808	93	12	0	29	65	31	31	28	13	12	17	275	9
33	4	9	29	17	255	3	1	0	9	6	2	2	5	3	1	2	291	11
323	35	10	314	198	2402	80	10	0	23	40	23	23	28	17	7	19	242	14
355	26	14	340	221	2632	67	11	0	33	51	34	34	39	34	8	23	251	23
248	22	12	234	139	1963	46	19	0	28	29	18	18	26	22	10	9	198	8
276	25	12	253	180	2182	57	18	0	29	44	20	20	31	34	11	15	209	50
134	6	23	125	86	1007	29	4	0	10	20	28	28	18	15	4	5	106	26
219	25	9	188	144	1618	49	1	0	30	33	13	13	36	20	2	6	191	49
122	13	10	125	72	991	24	3	0	27	22	10	10	16	12	2	3	102	12
405	35	12	370	235	3168	96	24	0	52	60	22	22	38	19	6	10	323	25
242	18	14	213	172	1804	50	6	0	13	41	32	32	13	18	9	8	151	18
263	15	18	257	161	1859	43	11	0	21	30	33	33	22	23	14	8	182	13
53	7	8	22	50	362	17	3	0	2	5	3	3	7	6	1	3	45	37
121	11	11	122	59	868	20	8	0	25	15	12	12	17	10	0	5	99	10
255	20	13	246	159	2049	55	9	0	23	43	26	26	20	24	10	3	197	18
39	2	20	37	22	317	11	4	0	6	8	0	0	2	3	1	0	31	3
387	30	13	325	268	2878	61	34	1	45	77	46	46	22	27	10	12	282	16
442	39	12	387	266	3216	75	32	0	43	71	44	44	43	26	13	17	338	22
211	16	14	201	124	1673	50	8	0	21	29	14	14	20	14	5	10	160	7
279	22	13	238	191	2004	48	25	0	34	40	31	31	19	23	6	15	194	19
92	7	14	82	60	650	14	7	0	17	13	10	10	9	7	0	6	64	12
267	28	10	254	153	2046	47	10	1	35	43	25	25	26	21	15	4	217	24
267	23	12	238	134	2062	43	10	0	37	50	32	32	32	29	9	11	192	18
166	15	13	167	98	1421	27	5	0	28	23	21	21	12	27	8	6	143	5
381	34	12	326	248	2603	69	25	0	29	50	50	50	33	30	11	18	275	18
387	36	11	347	241	2768	63	38	1	43	44	42	42	37	37	19	9	284	15
123	12	11	124	67	917	23	10	0	21	20	7	7	15	9	4	4	99	7
265	26	11	273	143	1878	62	9	0	30	34	27	27	20	20	3	13	195	24
302	33	10	283	130	2283	48	10	1	34	40	45	45	32	39	2	7	237	8
408	24	17	360	277	3015	43	57	0	45	80	44	44	43	33	24	15	270	12
363	31	12	344	210	2654	63	16	0	38	53	34	34	44	31	15	12	285	34
237	17	14	234	145	1868	53	6	0	19	46	27	27	19	16	9	11	178	16
316	22	15	320	160	2387	78	19	0	27	51	26	26	28	13	19	13	250	23
132	15	9	131	84	1022	38	8	0	10	21	16	16	13	17	1	5	109	13
29	3	10	30	14	220	4	0	0	8	5	2	2	2	4	0	1	23	3
192	14	14	176	128	1482	26	12	0	27	40	23	23	19	25	4	4	138	23

Figure 5.7: Final Dataset with all the features

Chapter 6

6.1 Feature analysis

In this chapter we will discuss how we have analyzed different features and selected them for ultimately training the regression model. We have analyzed each of the feature‘s relation with the score of the essay. In order to do so we have made 15 different dataset each containing only one type of feature. After that we have implemented linear regression model using that feature and documented the result. Similarly we have implemented random forest regressor model for each features too. In some of the cases we have also drawn some scatter plots and other graphs to see the outcome.As there are only one feature in this cases the formula for the linear regression will be

$$h(x) = theta0 + theta1 * x$$

Where theta0 and theta1 are the coefficients and the x is the feature. And the equation or we can say the cost function to find the value of the coefficients the theta0 and theta1 is the following

$$\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$$

Where function J is the following

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

And in order to find the minimized J we will use Gradient Descent and the formula for Gradient Descent is the following

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$

Another approach or another model we have applied in the same dataset is the random forest regressor model. Random forest is a regression and classification model based on decision tree in this model we apply multitude decision tree model to the training dataset and observed the result. Random forest applies multiple weak learner in the dataset and the n ultimately comes with a properly trained classifier or regressor. The application of these two algorithms on the features are discussed here.

6.2 Regression Model Analysis on Word count

We have applied linear regression to predict the score based on the word count only .We have first drawn a scatter plot with octave to see if there exist any relation or not . We have found a linear relation to be present between the word count and the score predicted. The relation is shown in the first left figure of the 6.1 figure. There the X axis denotes the number of words in an essay and Y axis represents the score. The second picture in that row shows the contour that we have generated to find the minimum value of the coefficients. In the third figure which is 6.1.3 We have shown the different values of the coefficients once again but this time we have shown the three dimensional view. While applying linear regression we have come across the following result.

Coefficients value:

theta 0=1 , theta1= 0.00521795

Intercept of linear regression: 2.91348485

Mean squared error: 0.390321713206

Residual sum of squares: 0.39

Then applying Random forest regressor we have found the following result

Mean squared error: 0.398938624457

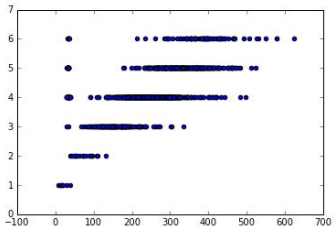


figure 6.1.1 : word count vs score

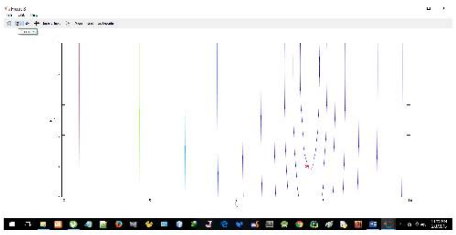


figure 6.1.2 : contour showing the value of coefficient

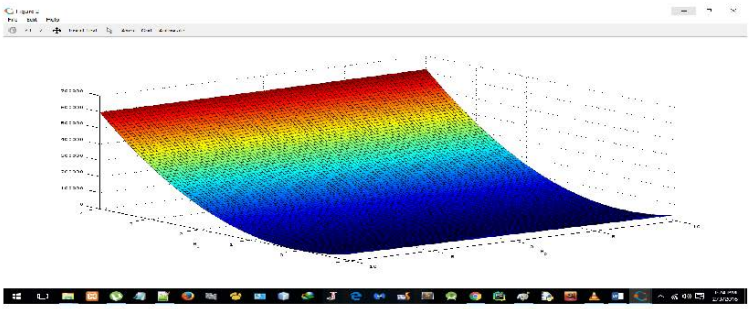


figure 6.1.3 : 3d plot of finding the value of coefficients (theta) for linear regression

Figure 6.1: relation of word count with score

6.3 Regression Model Analysis on Sentence Count

Just like 6.2 we have applied the similar process on the dataset of sentence count and the score. The first figure of the figure 6.2 which is figure 6.2.1 is the relation we have found between the sentence count (represented by the X axis) and the score (represented by the y axis). The other two figures denotes the contour to find the values of the coefficients just like the previous case.

Now the result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.04906209

Intercept of linear regression: 3.19317904

Mean squared error: 0.462485510962

Residual sum of squares: 0.46

Then applying Random forest regressor we have found the following result

Mean squared error : 0.458886352555

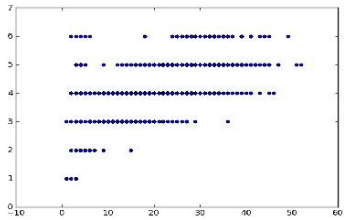


figure 6.2.1 : sentence count vs score

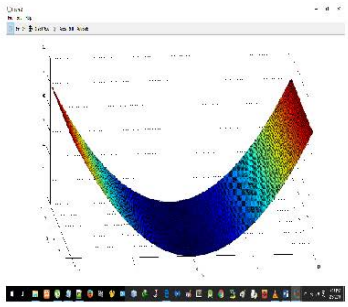


figure 6.2.2 : 3d view of finding coefficients(theta)

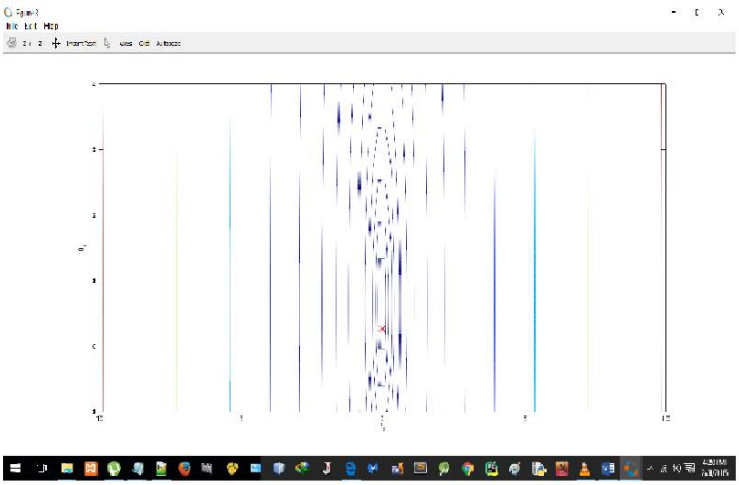


figure 6.2.3 : contour view of the coefficients

Figure 6.2: Relation of Sentence count with Score

6.4 Regression model analysis on Ratio of word and sentence count

In this case we have taken ratio of words count and sentence count as the input of the machine learning algorithm and tried to see how it responds the scatter diagram in figure 6.3 displays the relation of the ratio of the word and sentence with the score. Here the X axis represents the ratio of the words and sentences and the y axis represents the score. Other result noticed during the procedure are here:

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.04906209

Intercept of linear regression: 3.19317904

Mean squared error: 0.656234055647

Residual sum of squares: 0.66

Then applying Random forest regressor we have found the following result

Mean squared error: 0.65441103131

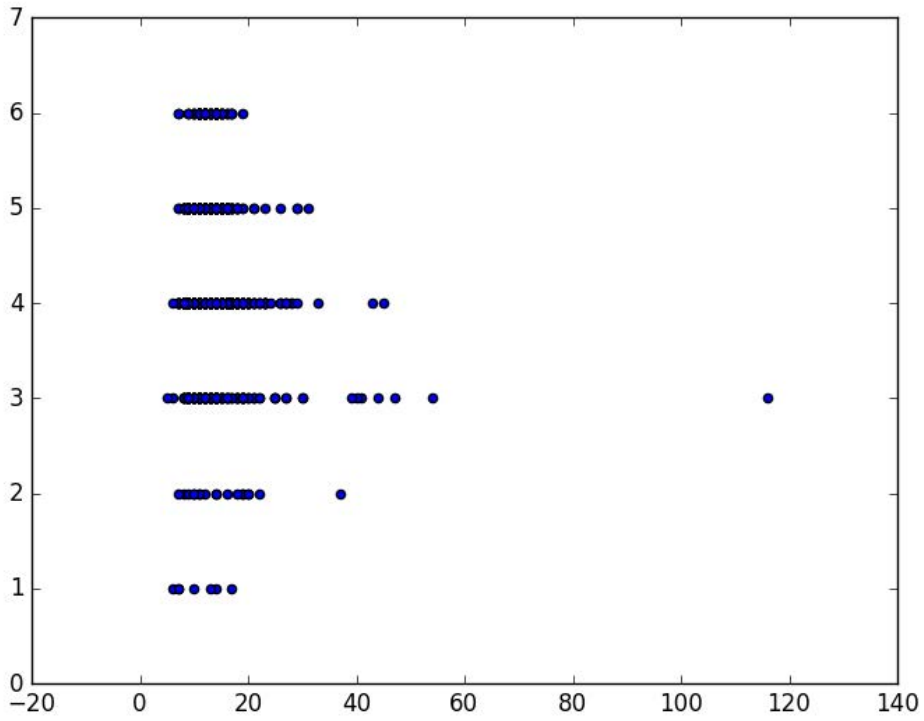


Figure 6.3: Relation of Sentence ratio of word and sentence count with Score

6.5 Regression model analysis on the number of English words

In this case we have taken the number of English words present in each essay as an input and figured out the relationship with the score. The relationship is shown in the figure 6.4 where the X axis denotes the number of English words and the Y axis denotes the score. Other noticeable outcomes are demonstrated here:

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.00552043

Intercept of linear regression: 2.93541532

Mean squared error: 0.402704353127

Residual sum of squares: 0.40

Then applying Random forest regressor we have found the following result

Mean squared error: 0.462912680188

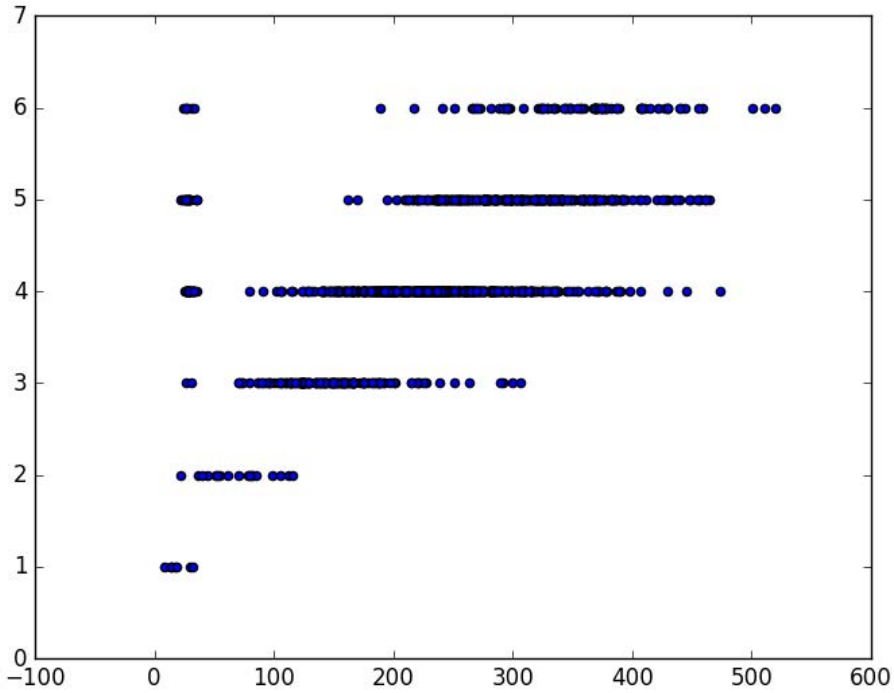


Figure 6.4: the relationship between English word count and the score

6.6 Regression model analysis on the number of non-English words

In this case we have taken the number of non-English words (punctuations numbers and symbols) present in each essay as an input and figured out the relationship with the score. The relationship is shown in the figure 6.5 where the X axis denotes the number of non-English words and the Y axis denotes the score. Other noticeable outcomes are demonstrated here:

result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.00779769

Intercept of linear regression: 3.01379251

Mean squared error: 0.412005858631

Residual sum of squares: 0.41

Then applying Random forest regressor we have found the following result

Mean squared error: 0.429546530851

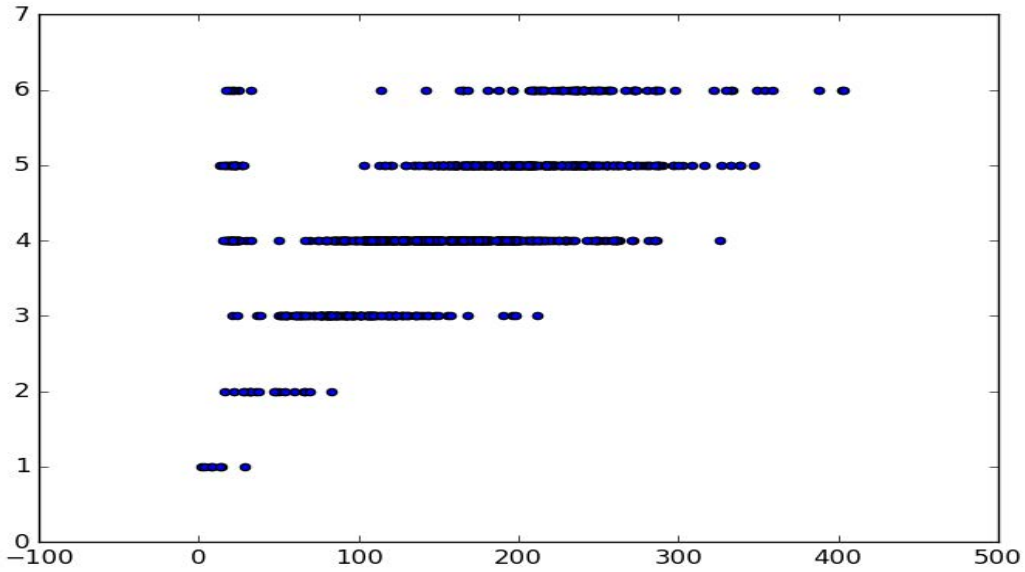


Figure 6.5: the relationship between the number of Non English words and the score

6.7 Regression model analysis on the number of Total Characters

In this case we have taken the number total characters present in each essay as an input and figured out the relationship with the score. The relationship is shown in the figure 6.6 where the X axis denotes the number of non-English words and the Y axis denotes the score. Other noticeable outcomes are demonstrated here:

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.00070541

Intercept of linear regression: 2.88916752

Mean squared error: 0.386679742165

Residual sum of squares: 0.39

Then applying Random forest regressor we have found the following result

Mean squared error: 0.432382673115

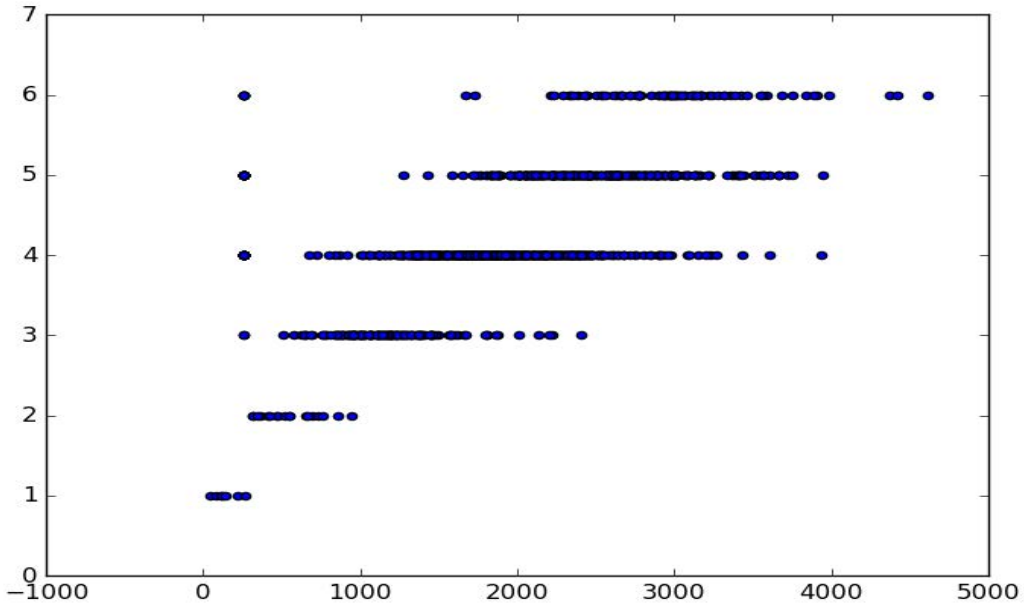


Figure 6.6: the relationship between the number of Characters and the score

6.8 Regression Model analysis on different parts of speech

In the following few discussions we will demonstrate how different parts of speeches present in the essays responds in predicting the score.

6.8.1 NN

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.00070541

Intercept of linear regression: 3.1044111

Mean squared error: 2561.7418978

Applying Random forest on the dataset

Mean squared error: 0.412428045164

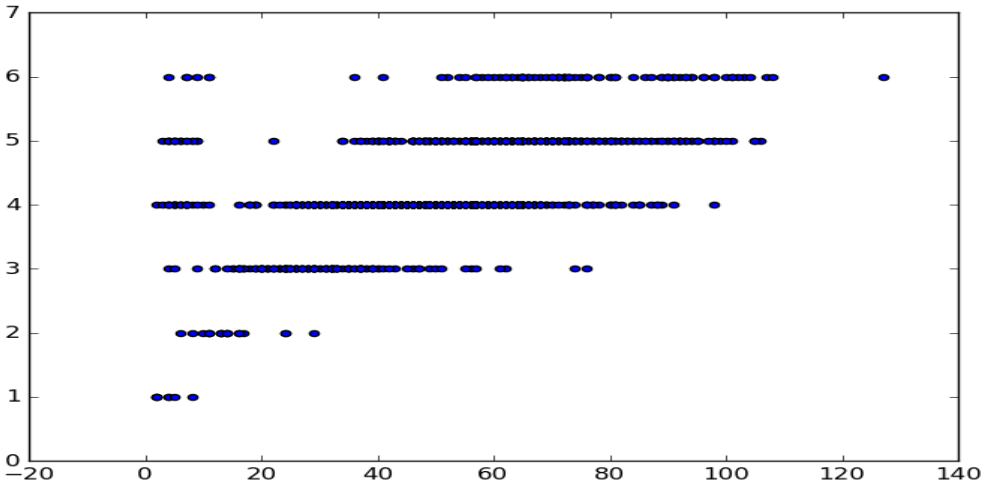


Figure 6.7: relationship between NN and the score.

6.8.2 NNP

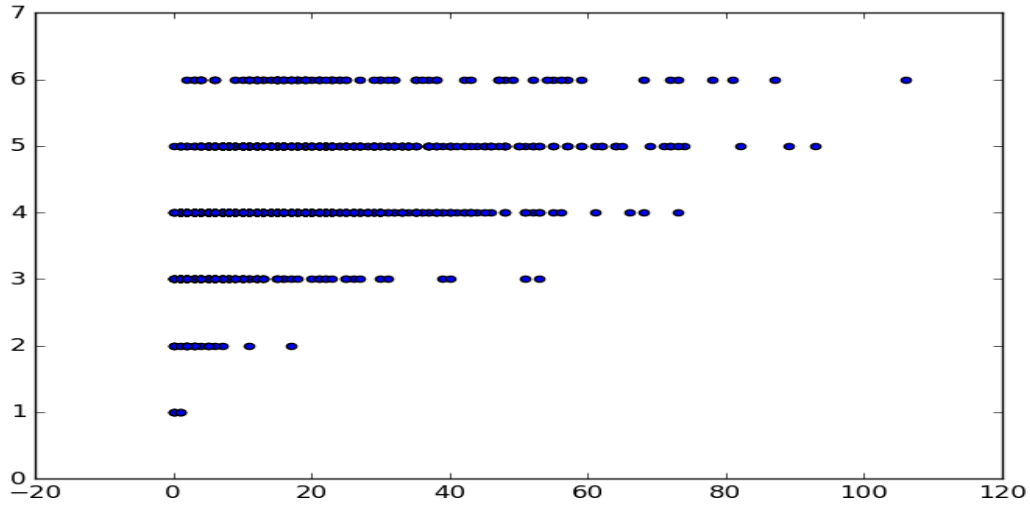


Figure 6.8: relationship between NNP and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1

theta1= 0.02174692

Intercept of linear regression:

3.90146375

Mean squared error

0.572256287847

Residual sum of squares: 0.57

Then applying Random forest regressor we have found the following result

mean squared error

0.560123684731

6.8.3 NNS

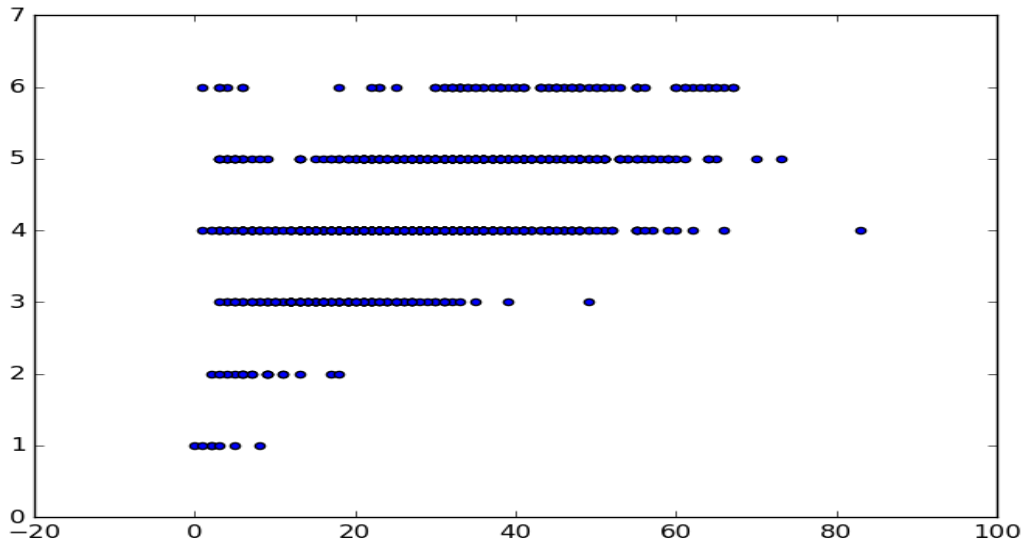


Figure 6.9: relationship between NNS and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1

theta1= 0.03225513

Intercept of linear regression:

3.35448958

Mean squared error

0.519548959044

Residual sum of squares: 0.52

Then applying Random forest regressor we have found the following result

Mean squared error

0.534230245256

6.8.4 IN

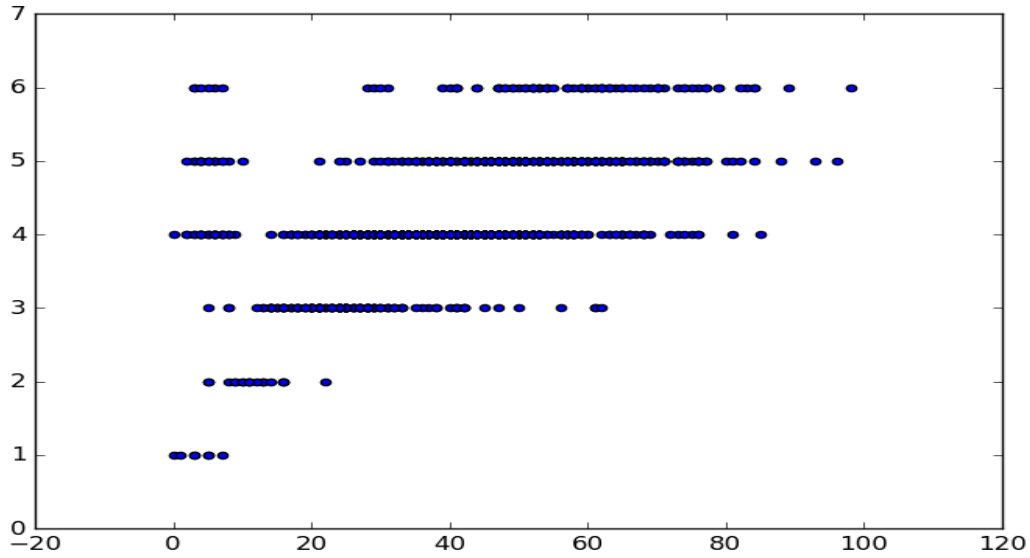


Figure 6.10: relationship between IN and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1 , theta1= 0.02792319

Intercept of linear regression:

3.13346339

Mean squared error

0.461192389186

Residual sum of squares: 0.46

Then applying Random forest regressor we have found the following result

Mean squared error

0.436169620154

6.8.5 PRP

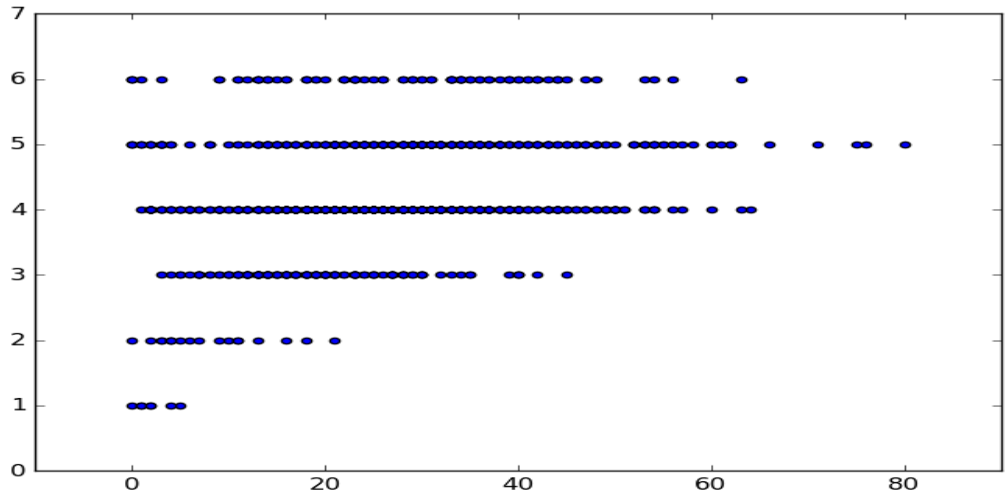


Figure 6.11: relationship between IN and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.02022658

Intercept of linear regression: 3.7290351

Mean squared error : 0.580044147173

Residual sum of squares: 0.58

Then applying Random forest regressor we have found the following result

Mean squared error

0.624930014876

6.8.6 VB

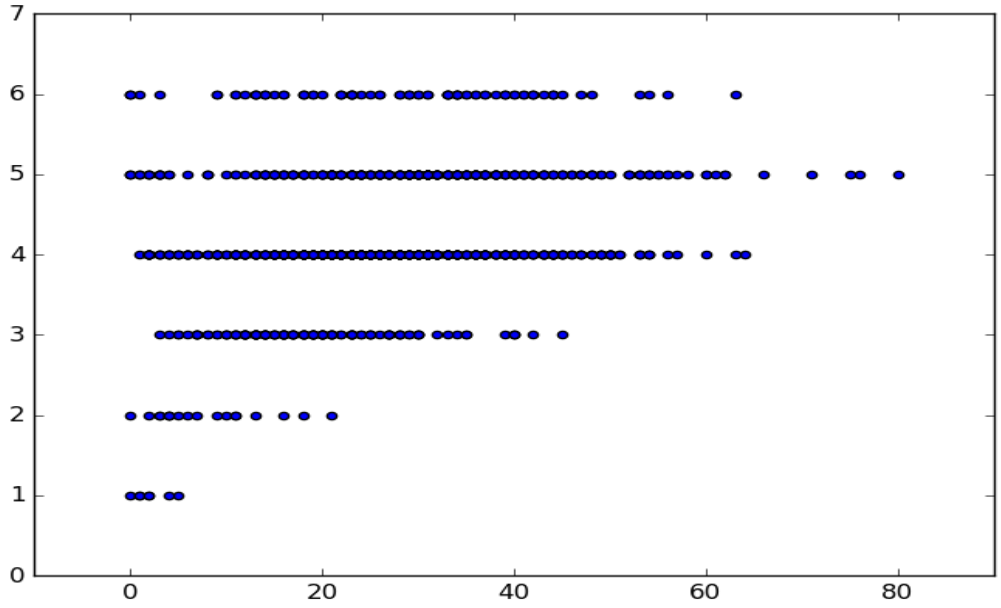


Figure 6.12: relationship between VB and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1, theta1= 0.02022658

Intercept of linear regression: 3.7290351

Mean squared error: 0.580044147173

Residual sum of squares: 0.58

Then applying Random forest regressor we have found the following result

Mean squared error

0.606420482632

6.8.7 JJ

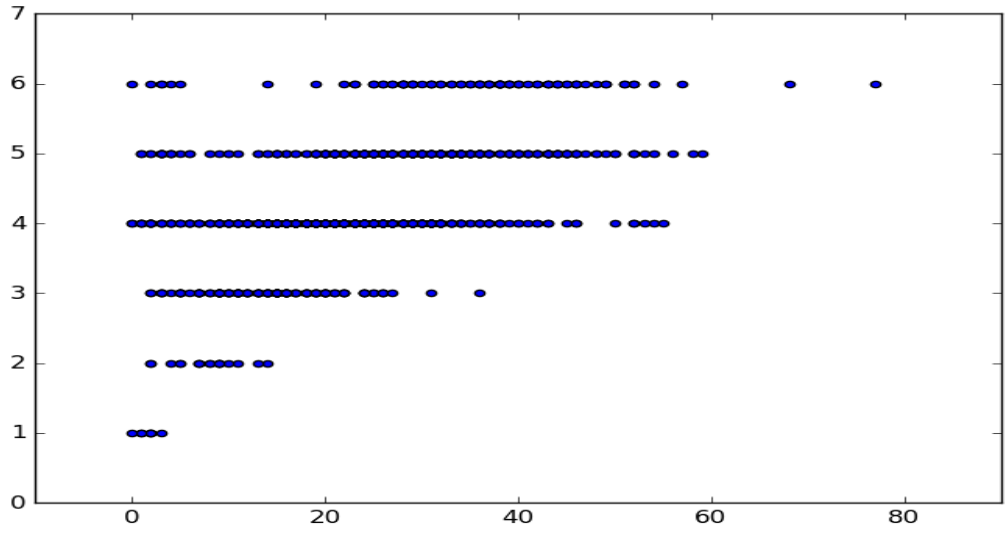


Figure 6.13: relationship between JJ and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1,

theta1= 0.0412841

Intercept of linear regression :

3.30252534

Mean squared error :

0.45983790596

Residual sum of squares:

0.46

Then applying Random forest regressor we have found the following result

Mean squared error: 0.496058956677

6.8.8 VBG

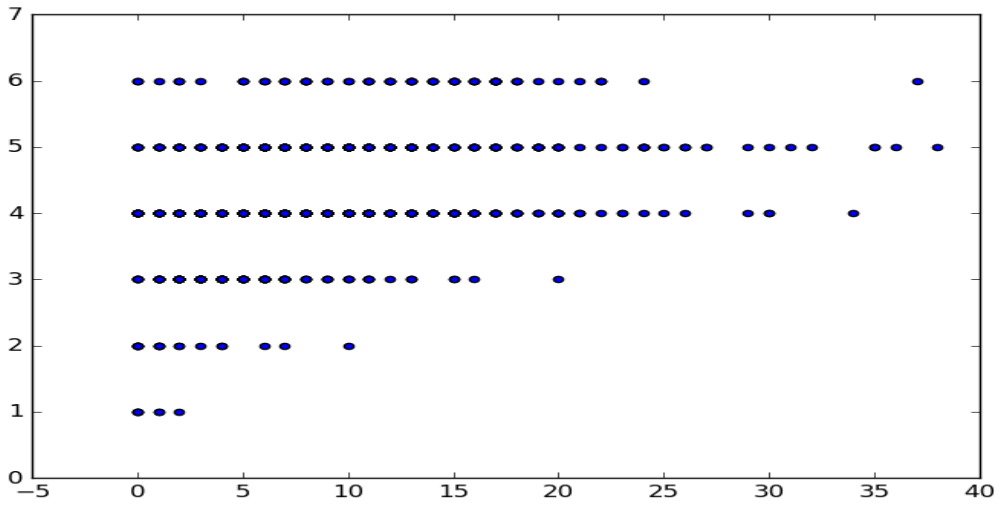


Figure 6.14: relationship between VBG and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1

theta1= 0.05184511

Intercept of linear regression:

3.81102707

Mean squared error

0.578750200793

Residual sum of squares: 0.58

Then applying Random forest regressor we have found the following result

Mean squared error

0.576086679739

6.8.9 VBZ

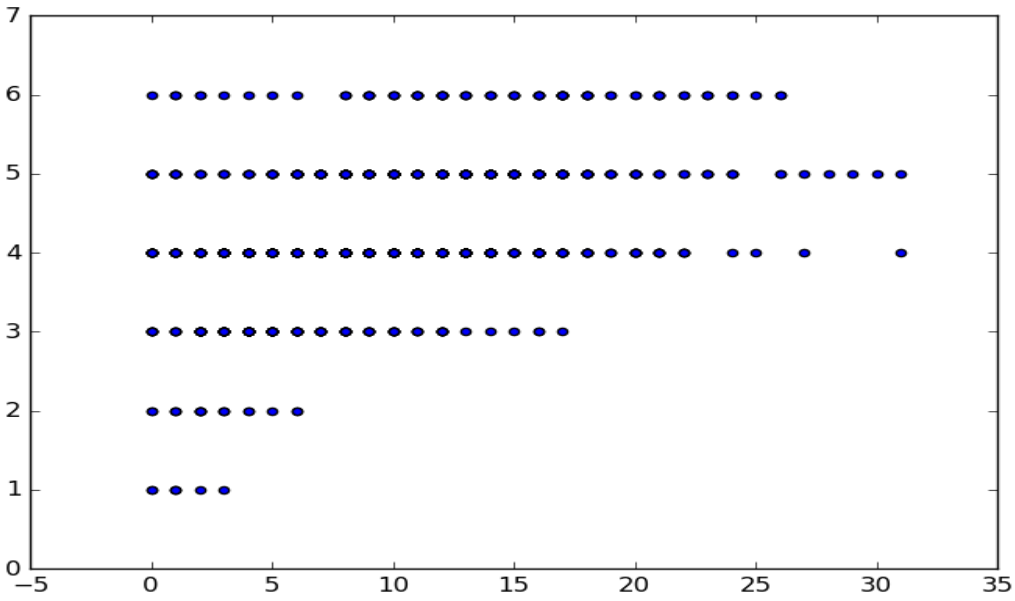


Figure 6.15: relationship between VBZ and the score.

Result found applying linear regression on the dataset:

Coefficients value:

theta 0=1

theta1= 0.06597526

Intercept of linear regression:

3.62587663

Mean squared error

0.558771217239

Residual sum of squares: 0.56

Then applying Random forest regressor we have found the following result

Mean squared error

0.5400289975

6.9 Regression model applied to all the features combined

After we saw all the features and their role predicting the score so based on the output we decided to run a linear regression algorithm where instead of one feature we used all the 19 features at a time and saw how it helped us predicting the score. That means in this case the linear equation would be

$h\theta(x)=\theta0 +\theta1X1 + \theta2X2 + \theta3X3 + \theta4X4 + \theta5X5 + \theta6X6 + \theta7X7 +\theta8X8 + \theta9X9 + \theta10X10 + \theta11X11 + \theta12X12 + \theta13X13 + \theta14X14 + \theta15X15 + \theta16X16 + \theta17X17 + \theta18X18 + \theta19X19$

We have then used the gradient descent formula described above to find all the 20 values of the coefficients(theta's). And the values that we have found are-

1.38366460e-03 -2.40695450e-03 -1.63490549e-02 -4.37105229e-03
-1.71520601e-03 1.71751569e-03 -4.08561359e-03 3.16321251e-04
3.57359990e-03 -5.54050994e-03 -6.48889960e-03 -4.22881072e-03
-4.22881072e-03 -4.12866500e-03 -4.37621506e-03 -4.56464118e-03
-7.59560496e-05

Using these values of theta we came across the following noticeable result

The intercept of linear regression in this case is: 3.22319058

The mean squared error of linear regression for this dataset is

0.378462858535

And the Residual sum of squares: 0.38

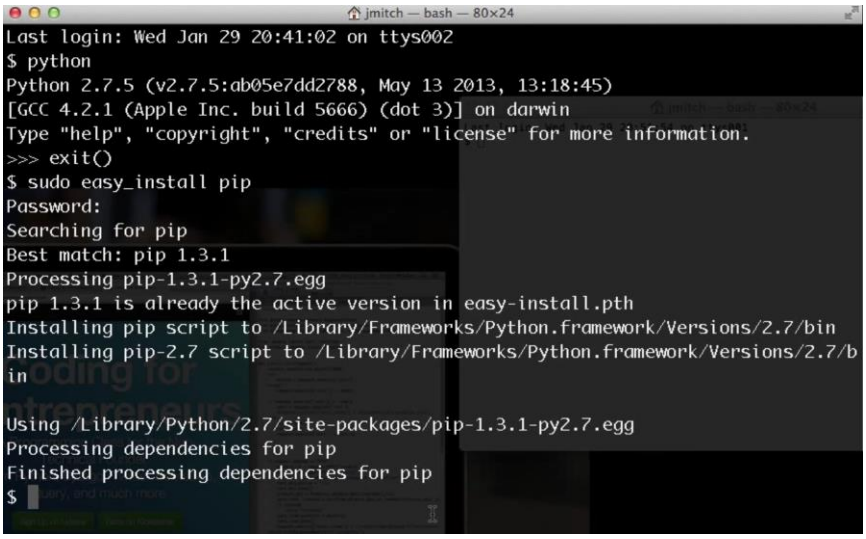
Chapter 7

Implementation

We have started out research with a goal in mind that we would implement our work does not matter how far we have achieved in this given period of time, because with the final implementation of our work others will get the idea and will be able to comprehend, visualized more about our research. In order to make our work into reality we have Django which is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. Along with Octave we have started working with Django and its sqlite database so that we can process large volume of data quickly.

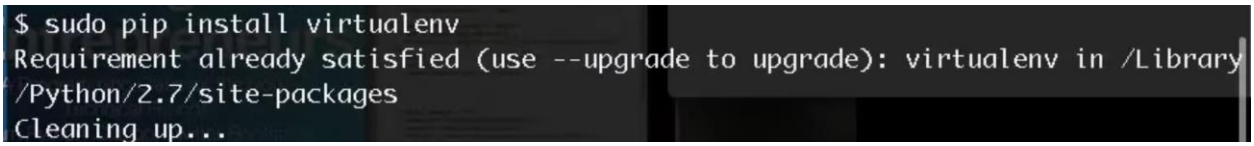
7.2 Installation

Installing Django onto our own machine is very easy, we might have to write few command in our terminals but that is very easy. For installing Django we have to install PIP and VirtualEnv in our system, just by typing "*sudo_easy install pip*" after typing the password we have installed PIP in our system.



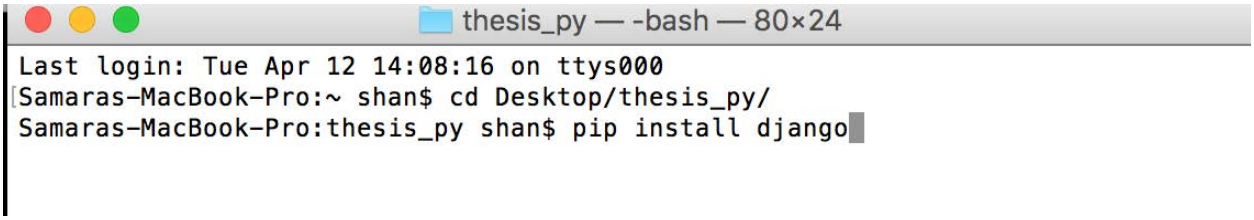
7.1 fig: shows the installation procedure of PIP

Installing VirtualEnv is also easy as it have been with pip all we have to write in the command line is "*sudo pip install virtualenv*".



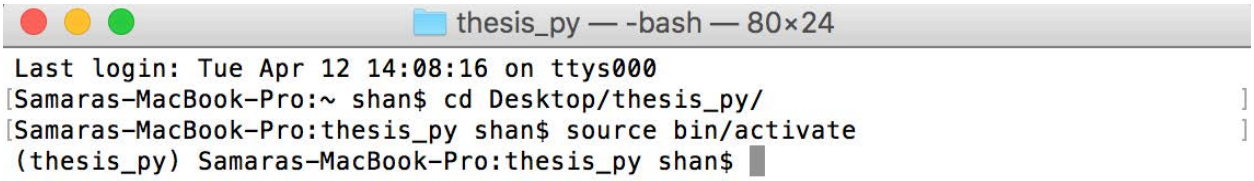
7.2 fig: code for installing virtualenv

After installing virtualenv we have written the command in terminal "*virtualenv thesis_py*" in our desktop, which would create a folder named *thesis_py* . We *cd* (terminal command to enter a folder) into our required folder and after that we typed in the terminal "*pip install django*"



7.3 fig: installing django framework in our research folder

To activate the virtualenv we just have to write "*source bin/activate*" in the terminal.

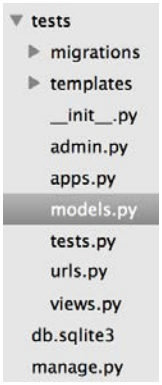


7.4 fig: initiating the virtual environment

7.3 Creating a Django project

Installing Django is just the tip of the iceberg to our research, now we have work with this to build our application. Therefore, we have to create a project in django and for that we typed "*django-admin startproject essay*" in the terminal that is already open. This will create an essay directory in our current

directory. Now that our environment – a "project" – is set up, we're set to start doing work. Each application in Django consists of a Python package that follows a certain convention. Django comes with a utility that automatically generates the basic directory structure of an app, so we can focus on writing code rather than creating directories. To create our app, we typed in the same directory as manage.py and type this command python "*manage.py startapp tests*". That'll create a directory tests, which is laid out like this:

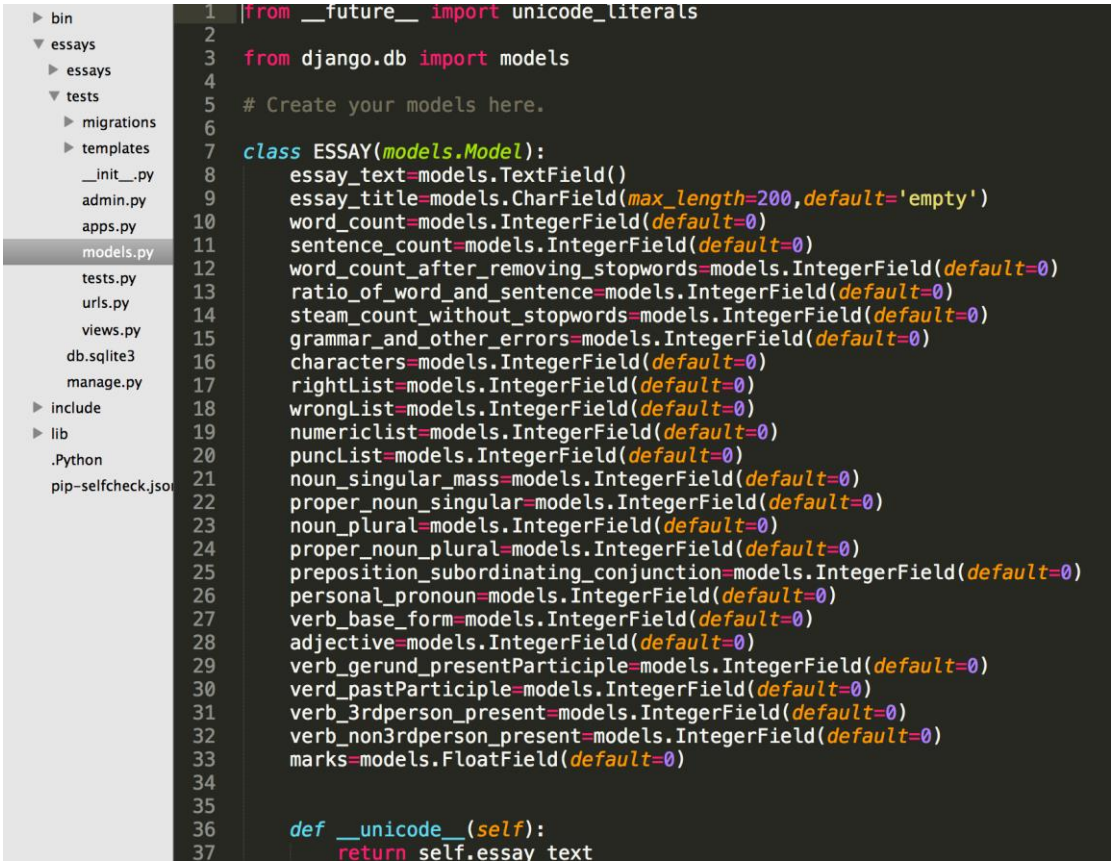


7.5 fig: directory of the project

7.4 Creating Database

As we have collected 1700 essays we needed a place where we could keep them along with their many different features. However, before doing that we have divided our data set into two separate sets, first part contain 1100 essays which would be our training essays and the second part is the test set. We would use the rest 600 essays for our testing and measurement of accuracy and error. Reason for creating a database instead of just parsing them from csv file is the performance, sqlite although it is comparatively small but it is sufficient for our research if I compare the amount of data we have within us. Moreover, SQLite does not need to be "installed" before it is used. There is no "setup" procedure. There is no server process that needs to be started, stopped, or configured. There is no need for an administrator to create a new database instance or assign access permissions to users. SQLite uses no configuration files. Nothing needs to be done to tell the system that SQLite is running. No actions are required to recover after a system crash or power failure and there is nothing to troubleshoot. Probably, due to its dynamic features sqlite comes out of the package when someone installs Django.

Now we'll define our models – essentially, our database layout, with additional metadata. In our simple thesis app, we'll create one model: ESSAY, this ESSAY will contain all the necessary fields which will help us to process our NLP task. These concepts are represented by simple Python classes. Edit the thesis/models.py file so it looks like this:



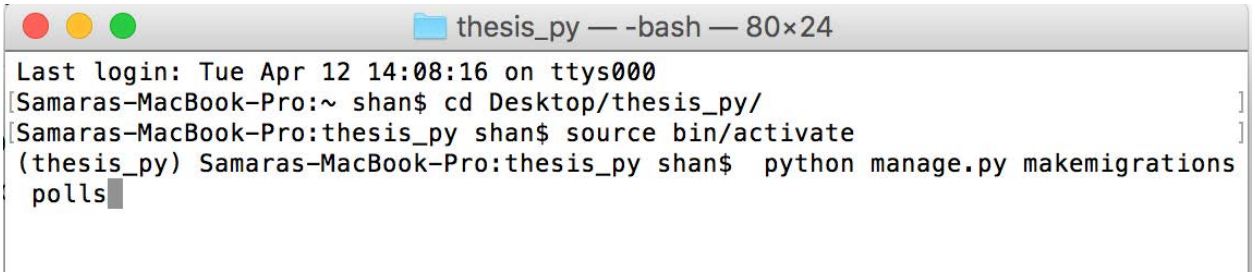
7.5 fig: shows the structure of the database table

The code is straightforward. Each model is represented by a class that subclasses django.db.models.Model. Each model has a number of class variables, each of which represents a database field in the model.Each field is represented by an instance of a Field class – e.g., CharField for character fields and DateTimeField for datetimes. This tells Django what type of data each field holds. After that we have to activate the model that small bit of model code gives Django a lot of information. With it, Django is able to:

*Create a database schema (CREATE TABLE statements) for this app.

*Create a Python database-access API for accessing Question and Choice objects.

Edit the essay/settings.py file again, and change the INSTALLED_APPS setting to include the string "*tests.apps.TestsConfig*''. Now Django knows to include the essay app. The next command we have written is:



7.6 fig: shows migration of the table

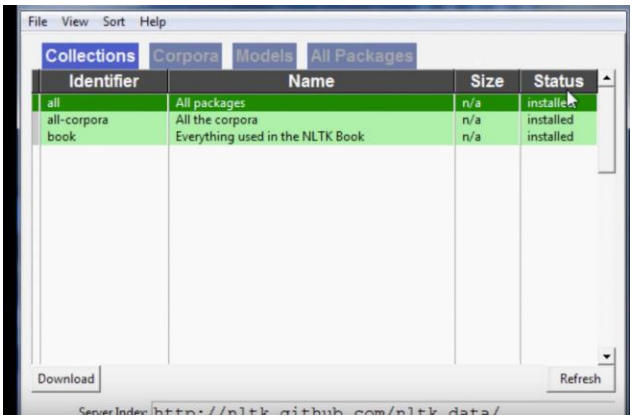
Now, we have to run migrate again to create those model tables in our database " *python manage.py migrate*".

7.5 Install NLTK

Python's NLTK package is one of the most important components to our research and overhear we will show how it is being installed. Start python from terminal by just typing "*python*" on mac's terminal after that write "*import nltk"* then write *nltk.download()*.

7.7 fig: showing NLTK installation

After this a new window would pop up when that happen just click the download option to download the NLTK libraries.

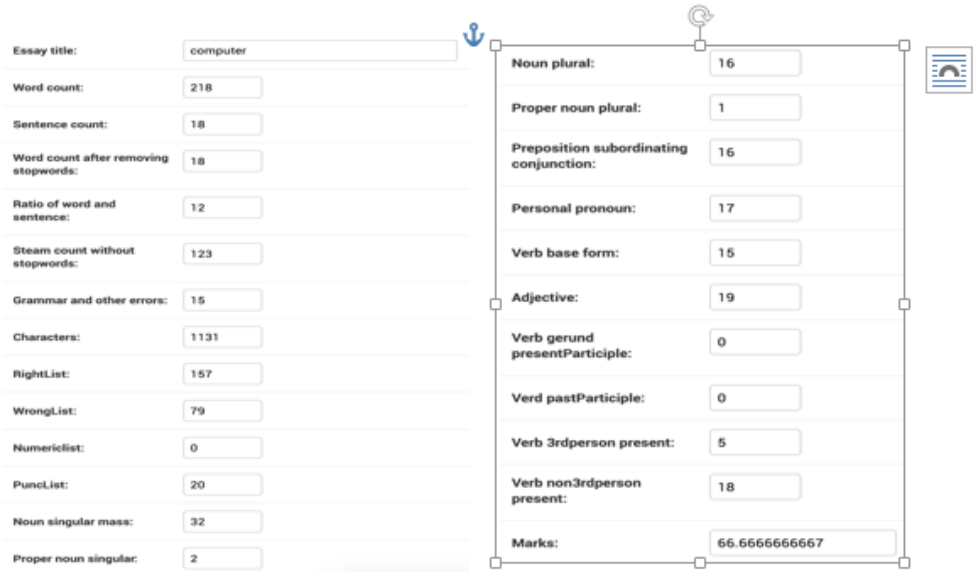


7.8 fig: download options that comes after writing that code.

Well, we are doing with installing NLTK to our development environment, now its time for the processing part.

7.6 Extracting the features, Scoring and Suggesting

We have extracted all the necessary features from the essays that we are going to trained and later stored them into a database that we have already created above. Some of the features maybe for important than the others but we left no stone unturned in terms of features which might give us better result.



7.9 fig: shows all the features which could have been used for grading

The numbers in the above figure are irrelevant to the subject we are talking now however, we will deal with them shortly. Moreover, anyone can understand about the features that is been shown above because we have carefully designed the database in such a way that anyone who is new could easily understand the thing we are doing just by have a look into our database.

Now in this section we will look into the codes and how we have managed to put all these things into one place. In some cases there are comments on the codes which have been deliberately planted to make better understanding of the code and some would be discussed here. The first we would discuss is how we have managed to extract features from essays by using those components that we have installed and pushing those new information into the database.

```
def index(request):
    file_locaiton="/Users/shan/Downloads/training_set_rel3.xlsx"

    workbook=xlrd.open_workbook(file_locaiton)
    sheet=workbook.sheet_by_index(0)

    tokenizer = RegexpTokenizer(r'\w+')

    tool = language_check.LanguageTool('en-US')

    array=[]
    stop_words = set(stopwords.words('english'))
    stop_words.update(['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}'])
    punctuation=['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}']


    col=2
    mark=3
    word_count=""
    text=""
    sentence_count=""
    word_count_after_removing_stopwords=""
    ratio=""
    steam_count=""
    grammar_error=0
    counter=0
    characters=0
    rightlist=0
    wronglist=0
    numericlist=0
    punclist=0

    NN= 0
    NNP=0
    NNS=0
    NNPS=0
    IN=0
    PRP=0
    VB=0
    JJ=0
```

7.10 fig: fetching the essay and initializing the variables

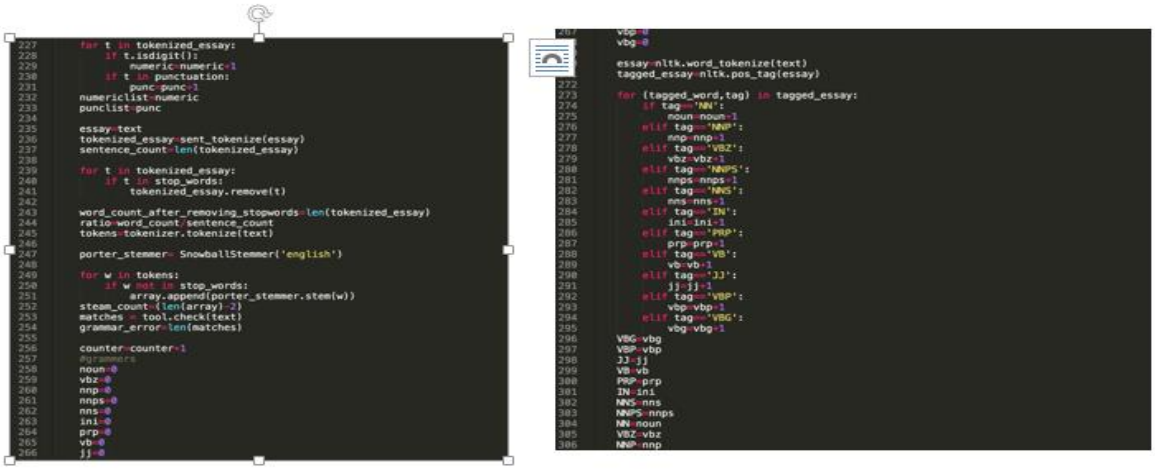
In the above figure we are fetching the essays that we have collected from Kaggle website and parsing them into our python program. We have initialized all the variables that we need for extracting the features and as most of them are integer type we have initialized them with 0s' however, some are string type so we just left them empty or null. A tokenizer that divides a string into substrings by splitting on the specified string. As we're trying to process a user entered text by removing stopwords using nltk toolkit, stopword function removes the words like 'and', 'or', 'not' or in other sense words which have no meaning or humans uses them to construct sentences. We have also update the stop_word collection by using the auction stop_word.update just the remove the sings from the text and by doing that we think it would help us to process.

The next thing we have created a for loop which would go for index 1 to 1100 and fetch those essays along with its' marks given by a human rater. We kept the original marks into one of the variables then counted the number of words in an essay by using the NLTK function tokenizer and we have also counted the number of characters in those essays by using another NLTK function FreqDist. NLTK has large sets of corpus and by the help of those corpus we have documented the numbers of right_words and wrong_words.

```
VBG=0
VBN=0
VBZ=0
VBP=0

#1784
for row in range(1,1784):
    text=str(sheet.cell(row,col))
    temp_mark=str(sheet.cell(row,mark))
    num=int(re.search(r'\d+', temp_mark).group())
    #converting number into percentage
    t_num=float(num)/6*100
    print(num)
    tokens = tokenizer.tokenize(text)
    word_count=len(tokens)

    #counting the characters
    fdist = nltk.FreqDist(text)
    characters=fdist.N()

    #counting rightlist and wronglist
    rightcount=0
    wrongcount=0

    tokenized_essay=word_tokenize(text)

    for t in tokenized_essay:
        if not wordnet.synsets(t):
            wrongcount=wrongcount+1
        else:
            rightcount=rightcount+1

    rightlist=rightcount #contains all the right feasible words
    wronglist=wrongcount #non feasible words

    #numericlist and punctuation list
    numeric=0
    punc=0
```

7.11 fig: started the for loop to fetch essays and doing different operations

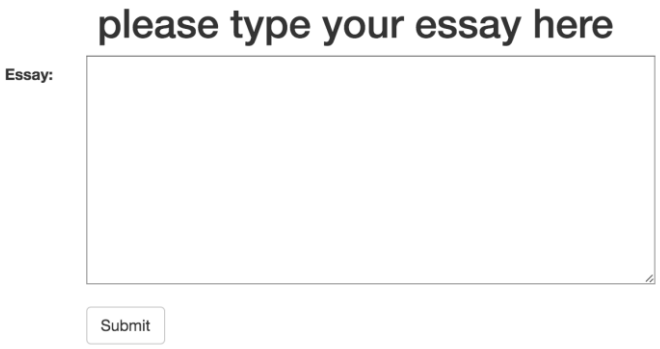
In the next fugue we have calculated the ratio of words and sentences which after removing the stop_words. We have also enumerated stemmer which basically removes morphological affixes from words, leaving only the word stem. Later in that part we have also counted the number of grammatical mistakes that we have found from an essay. And finally we have stored all this information that we could gather into our database.



```
for t in tokenized_essay:
    if t.isdigit():
        numeric=numeric+1
    if t in punctuation:
        punc=punc+1
numericlist=numeric
punclist=punc

essay=text
tokenized_essay=sent_tokenize(essay)
sentence_count=len(tokenized_essay)

for t in tokenized_essay:
    if t in stop_words:
        tokenized_essay.remove(t)

word_count_after_removing_stopwords=len(tokenized_essay)
ratio=word_count/sentence_count
tokens=tokenizer.tokenize(text)

porter_stemmer= SnowballStemmer('english')

for w in tokens:
    if w not in stop_words:
        array.append(porter_stemmer.stem(w))
steam_count=(len(array)-2)
matches = tool.check(text)
grammar_error=len(matches)

counter=counter+1
#grammers
noun=0
vbz=0
nnp=0
nnps=0
nns=0
ini=0
prp=0
vb=0
jj=0
vbp=0
vbg=0

essay=nltk.word_tokenize(text)
tagged_essay=nltk.pos_tag(essay)

for (tagged_word,tag) in tagged_essay:
    if tag=='NN':
        noun=noun+1
    elif tag=='NNP':
        nnp=nnp+1
    elif tag=='VBZ':
        vbz=vbz+1
    elif tag=='NNPS':
        nnps=nnps+1
    elif tag=='NNS':
        nns=nns+1
    elif tag=='IN':
        ini=ini+1
    elif tag=='PRP':
        prp=prp+1
    elif tag=='VB':
        vb=vb+1
    elif tag=='JJ':
        jj=jj+1
    elif tag=='VBP':
        vbp=vbp+1
    elif tag=='VBG':
        vbg=vbg+1
VBG=vbg
VBP=vbp
JJ=jj
VB=vb
PRP=prp
IN=ini
NNS=nns
NNPS=nnps
NN=noun
VBZ=vbz
NNP=nnp
```

7.12 fig: starting from top left to right then coming down to bottom showing all the process that we have used for feature extraction

Now then we are done with the feature extraction part we would now move on the machine learning part. In this section we have borrowed the values of theta, mu and sigma from our actave code by using these same features and use them to grade our essays.



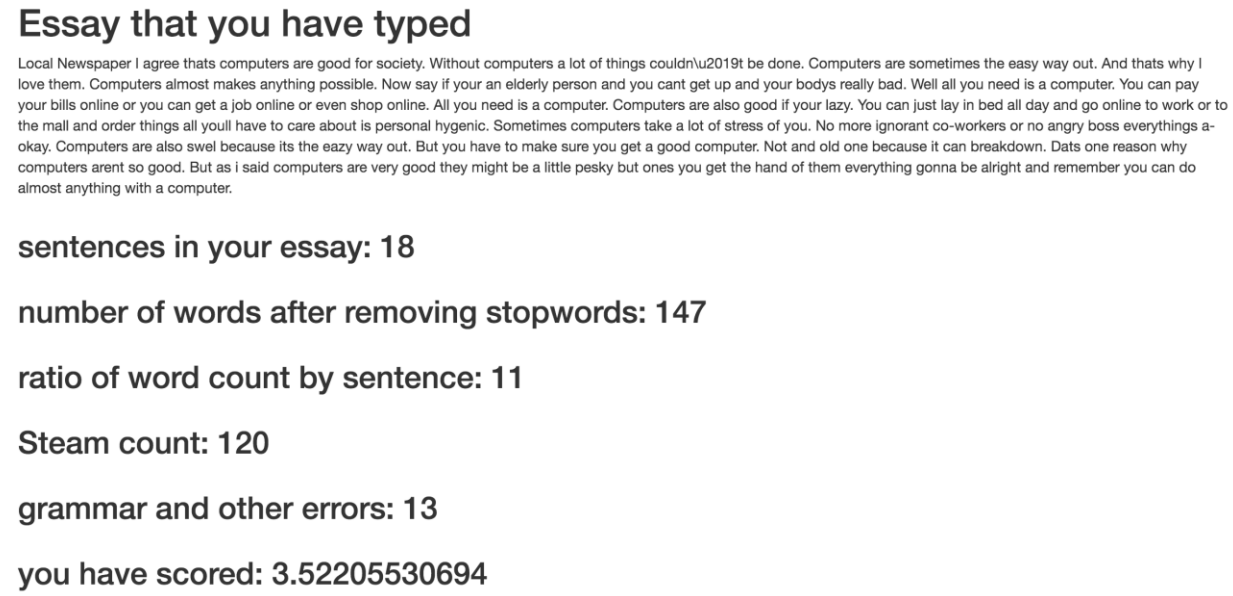
7.13 fig: this the part here the writer has to paste an essay and click the submit option.

```
def candidate(request):

    tokenizer = RegexpTokenizer(r'\w+')
    stop_words = set(stopwords.words('english'))
    stop_words.update(['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}'])
    tool = language_check.LanguageTool('en-US')

    essay=str(request.POST['essay'])
    num=len(essay)
    template = loader.get_template('polls/result.html')

    tokens = tokenizer.tokenize(essay)
    word_count=len(tokens)


    tokenized_essay=sent_tokenize(essay)
    sentence_count=len(tokenized_essay)
    temp_sentence_count=sentence_count

    tokenized_essay=word_tokenize(essay)
    for t in tokenized_essay:
        if t in stop_words:
            tokenized_essay.remove(t)

    word_count_after_removing_stopwords=len(tokenized_essay)
    temp_word_count=word_count_after_removing_stopwords
    ratio=word_count/sentence_count
    temp_ratio=ratio

    array=[]

    porter_stemmer= SnowballStemmer('english')
    for w in tokens:
        if w not in stop_words:
            array.append(porter_stemmer.stem(w))

    steam_count=(len(array)-2)
    temp_steam_count=steam_count
    matches = tool.check(essay)
    grammar_error=len(matches)
    temp_grammer_error=grammar_error
```

```
    mu=[259.427,21.961,13.186,18.169,202.148]

    sigma=[99.6241,9.5542,5.1484,8.6915,68.7314]

    theta=[4.2607964,0.0341061, -0.0093306, -0.0273228,-0.0265270,0.6120277]

    word_count_after_removing_stopwords=word_count_after_removing_stopwords-mu[0]
    sentence_count=sentence_count-mu[1]
    ratio=ratio-mu[2]
    grammar_error=grammar_error-mu[3]
    steam_count=steam_count-mu[4]

    word_count_after_removing_stopwords=word_count_after_removing_stopwords/sigma[0]
    sentence_count=sentence_count/sigma[1]
    ratio=ratio/sigma[2]
    grammar_error=grammar_error/sigma[3]
    steam_count=steam_count/sigma[4]

    result=(1*theta[0])+(word_count_after_removing_stopwords*theta[1])+(sentence_count*theta[2])+(ratio*theta[3])+
    (grammar_error*theta[4])+(steam_count*theta[5])

    context = {
        'essay': essay,
        'sentence_count':sentence_count,
        'word_count_after_removing_stopwords':word_count_after_removing_stopwords,
        'ratio':ratio,
        'steam_count':steam_count,
        'grammar_error':grammar_error,
        'result':result,
        'temp_sentence_count':temp_sentence_count,
        'temp_word_count':temp_word_count,
        'temp_ratio':temp_ratio,
        'temp_steam_count':temp_steam_count,
        'temp_grammer_error':temp_grammer_error,
        'matches':matches,
    }
    return HttpResponse(template.render(context,request))
```

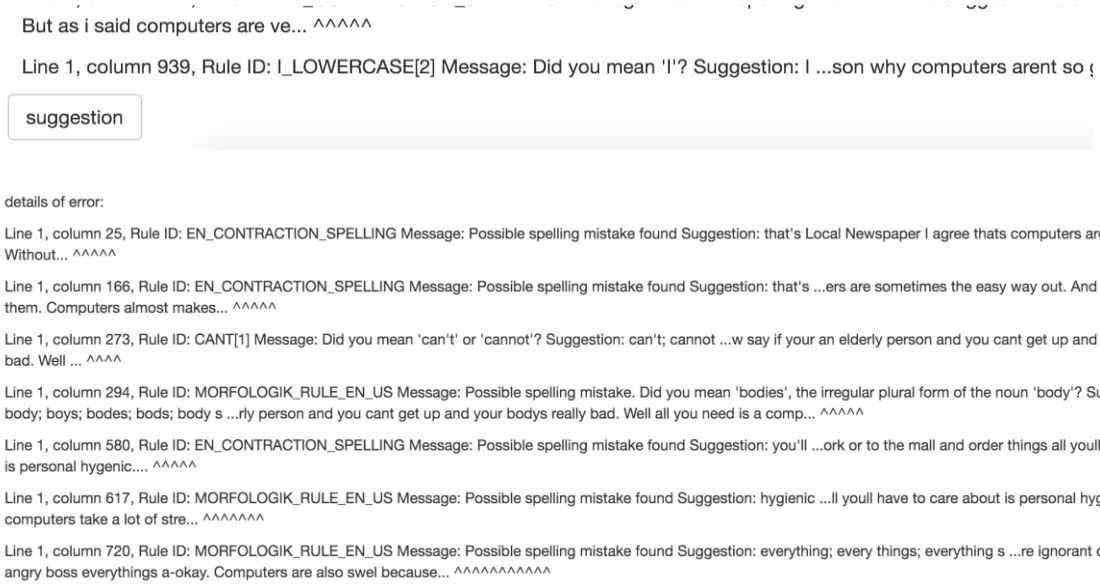
7.14 fig: the input is being processed by this code

Overhear, we have done the same thing as we did when extracting features from those essays. We have extracted those features that we think would give us as accurate result as possible. Some part of the linear regression we have already done in octave programming language to save some time which we could utilize to build this system. Finally, we put all these things together to get the predicted marks.

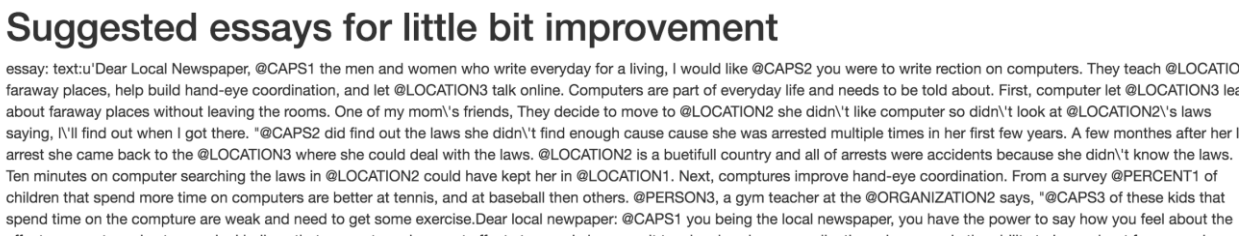


7.15 fig: shows the predicted score along with its features that has been used for calculation

We have also implemented a suggestion system in order to help an essay writer to improve his writing skills along with the predicted score. However, we have divided this system into two parts error suggestion and essay suggestion. For each of these different suggestions we have chosen different methods for implementing them. We have taken the help of language-check tool for find the grammatical mistakes as well as the suggestions. This tool that we have used is a beta version and requires lot of improvement especially when catching complex grammatical mistakes. We believe by the passes of time this tool will improve its accuracy which be very beneficial for future text processing research.

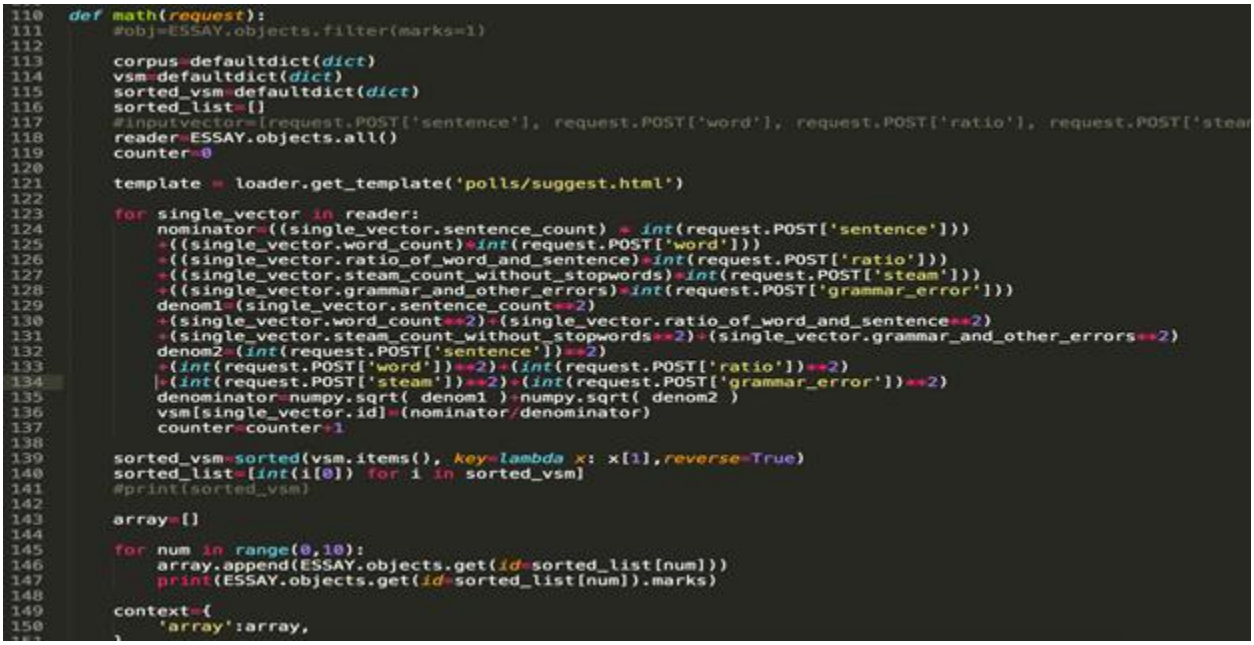


7.16 fig: showing errors and suggestion



7.17 fig: shows the suggested essay for better writing

Not only we have predicted a score for an essay we tried our best to give some sort suggestion both in the error department and display some better essays than the given essay to a writer. Hence, for suggesting some better essay we have introduced the cosine distance in our system. We calculated the cosine distance for each against the given essay then we have sort them in an reverse order which would give us the maximum distance and this maximum distance turned out to be closest distance from the actual given essay.



```
def math(request):
    #obj=ESSAY.objects.filter(marks=1)

    corpus=defaultdict(dict)
    vsm=defaultdict(dict)
    sorted_vsm=defaultdict(dict)
    sorted_list=[]
    #inputvector=[request.POST['sentence'], request.POST['word'], request.POST['ratio'], request.POST['stea
    reader=ESSAY.objects.all()
    counter=0

    template = loader.get_template('polls/suggest.html')

    for single_vector in reader:
        nominator=((single_vector.sentence_count) * int(request.POST['sentence']))
        +((single_vector.word_count)*int(request.POST['word']))
        +((single_vector.ratio_of_word_and_sentence)*int(request.POST['ratio']))
        +((single_vector.steam_count_without_stopwords)*int(request.POST['steam']))
        +((single_vector.grammar_and_other_errors)*int(request.POST['grammar_error']))
        denom1=(single_vector.sentence_count**2)
        +(single_vector.word_count**2)+(single_vector.ratio_of_word_and_sentence**2)
        +(single_vector.steam_count_without_stopwords**2)+(single_vector.grammar_and_other_errors**2)
        denom2=(int(request.POST['sentence'])**2)
        +(int(request.POST['word'])**2)+(int(request.POST['ratio'])**2)
        +(int(request.POST['steam'])**2)+(int(request.POST['grammar_error'])**2)
        denominator=numpy.sqrt( denom1 )+numpy.sqrt( denom2 )
        vsm[single_vector.id]=(nominator/denominator)
        counter=counter+1

    sorted_vsm=sorted(vsm.items(), key=lambda x: x[1],reverse=True)
    sorted_list=[int(i[0]) for i in sorted_vsm]
    #print(sorted_vsm)

    array=[]

    for num in range(0,10):
        array.append(ESSAY.objects.get(id=sorted_list[num]))
        print(ESSAY.objects.get(id=sorted_list[num]).marks)

    context={
        'array':array,
```

7.18 fig: showing the code for calculating the cosine distance

Chapter 8

8.1 Making Essay recommendation

As mentioned above our research goal is not confined to scoring the essays only, rather we wanted to suggest the user other essays based on his or her current essay writing skill. That is let's say a user wrote an essay that got a score of 2.5 out of 6 by our automatic system. Now how the user know what are the necessary changes needed to aim for the target 3. Again we believe a user with an essay writing skill worth score 2.5 mustn't aim for the highest 6 mark the very next time. Rather he or she should work to improve his writing skill to score 3 or 3.5. Then once he or she gets that result he or she should then aim for next milestone which could be 4 or 5. And then ultimately 6. So that is we want our system to find some relevant essays that helps us improve our writing skills slightly. With an aim to do so we then focused our research in the field of information retrieval and information filtering.

Information retrieval starts with a user query. The retrieval engine then finds necessary documents from its corpus or database related to that queries and sends it back to the user.
In our case we have a corpus full of essays and the user gives us input as a query is another essay. So if we want to find relevant essays from our dataset we have to consider the input essay as query and process it further to make the suggestions.

Before describing what we have done to achieve our aim to recommend relevant essays to the user let us describe you something from where we have found our insight. Vector Space Model or VSM. VSM is widely spread in Information retrieval, NLP and other research. In Information Retrieval research every document is considered as a point in multidimensional vector model. The query also placed in the model too based on some proper weight. Then we find the nearest documents of that query. The approach in finding the nearest documents could be several. It may be based on the linear distance, Euclidean distance, Cosine Distance many more. The picture describes it better. In the picture we see a few documents are placed on the three dimensional vector space model. And then the query is placed. We then find the nearest

document based on their cosine distance. Usually the VSM in Information Retrieval uses the Bag of Words Model. That is it makes a list of all the common word in the corpus as well as make a list of words present in each documents. And then when it gets the query it looks the words inside it and finds the document that has most of its words.

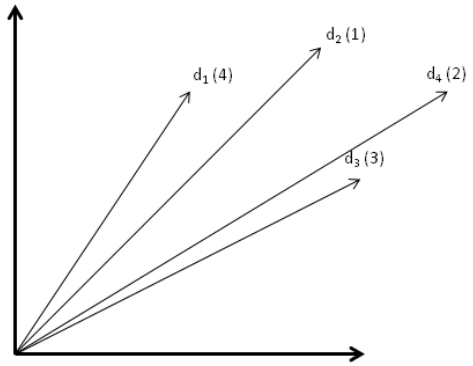


Figure 8.1: Vector Space model relation of Document and Query

In our case as we don't have queries rather than input essay given by the user. And in the database or corpus we have essays and all the features that we have extracted from those essays. So to recommend the user other essays based on the input essay we slightly manipulated the Bag of words and the Vector space model. Firstly we have extracted all the necessary features from the input essay, secondly we compared the input essay's features with other essays feature. We tried to learn how similar the other essays are with this one. We made a list of distance from each essays to the input essays. We have used the principles of the cosine distance to do so. Cosine similarity is a measure for vector space model that finds the distance between two vectors. The following formula is used in finding the cosine distance between two points.

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Or the value of Theta could be

$$\theta = \arccos\frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Here a and b are the two vectors that may have many more features we have to calculate the dot products of these vectors to get the value of theta. And then again we have to sort them to know which are the closest

or farthest essay to the input essay. In the following figure we have shown a screenshot that provides the code which we have written to calculate the cosine distance of one input essay with the rest of the essays in the corpus and sorts them and finally creates a list of the essay id that based on the distance from the input essay.

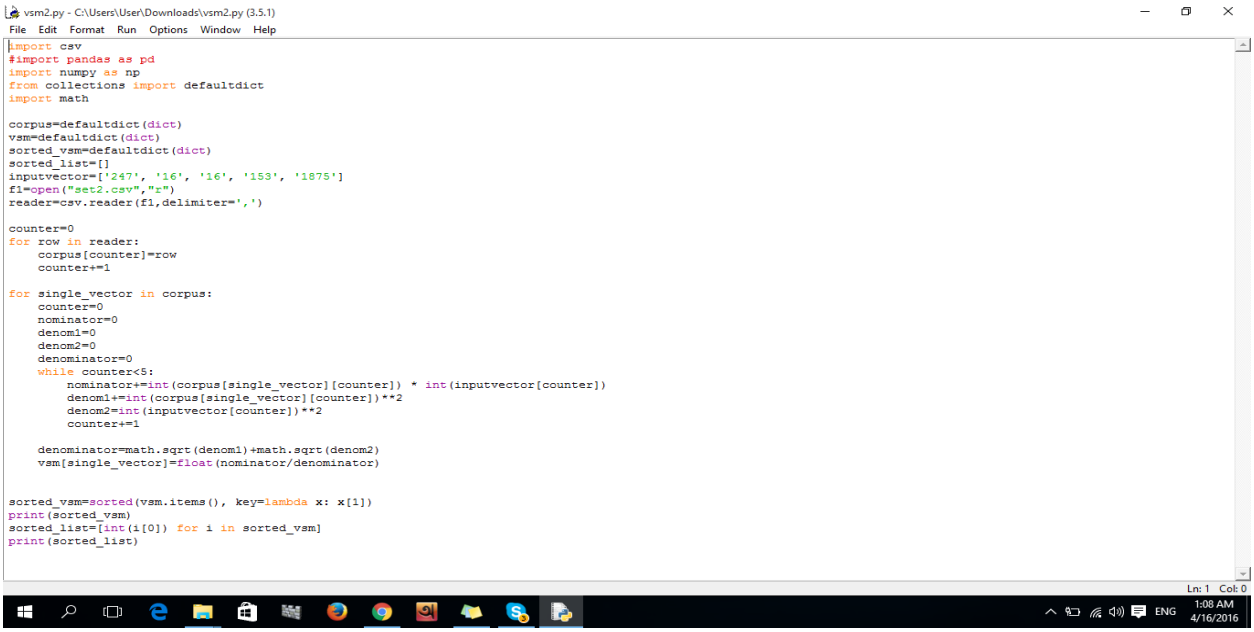


Figure 8.2: code to create a list of relevant essay id

After watching the response of the above mentioned code. We then once again added the module in our Django implementation. And ultimately we tested the system whether it gives right input or not. We have shown the response in the following few figures. In figure 8.3 we have shown what the essay input that we have given as an input to the system.And in figure 8.4 we have shown the title list of the essays recommended by the system.

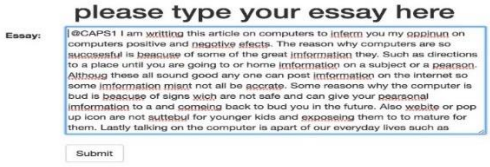


Figure 8.3: User interaction with the system

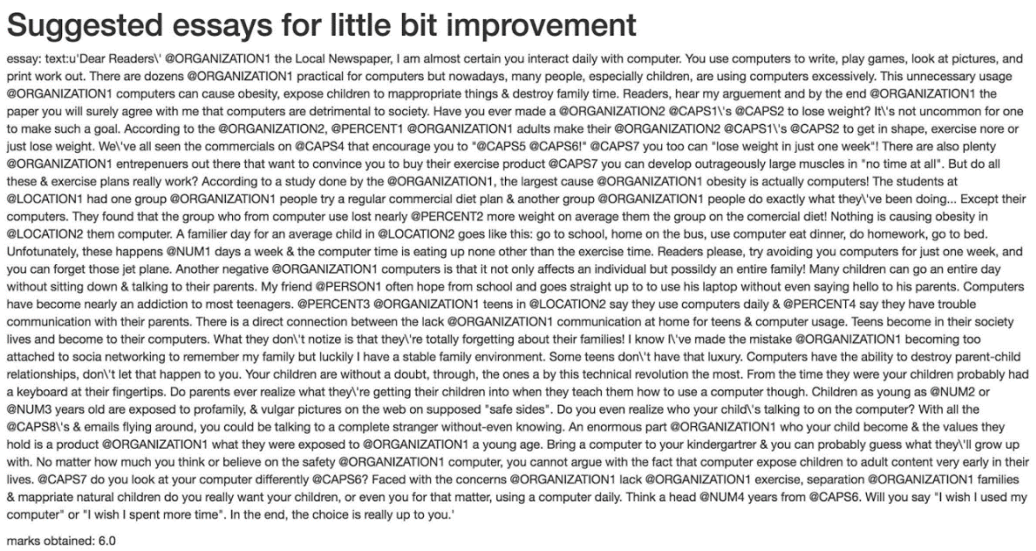


Figure 8.4: Suggested essays

8.2 Making Inline Change Recommendation

As we have accomplished suggesting essays with slightly better marks than the submitted essay to make a writer better at his writing we also wanted to take a step further. The goal that we have set for this higher purpose is to point out the mistakes in an essays does not matter whether a mistake is grammatical or simple spelling or just a misplaced of a punctuation symbol . On top of that we have liked the idea of suggesting some if not all on how to work around those mistakes so a writer can also learn what is right and not to repeat similar kind of mistakes in the next when he is going to writer. For this kind of important tasks we have started looking into several things which could help us to accomplish this thing. We have tested our idea with several different techniques some of them were promising and some of them were not however, we have managed to find a python's package called Language Check to server our purpose. Albeit, this package itself is a beta version which will improve its accuracy over the passes of time and the advancement in natural language processing but for the time being this beta version servers pretty well. The detail explanation is being shown below starting with the installation of this package into our system.

Installation

To install via pip:

```
$ pip install --upgrade language-check
```

If you are using Python 2, you'll need to install 3to2 beforehand:

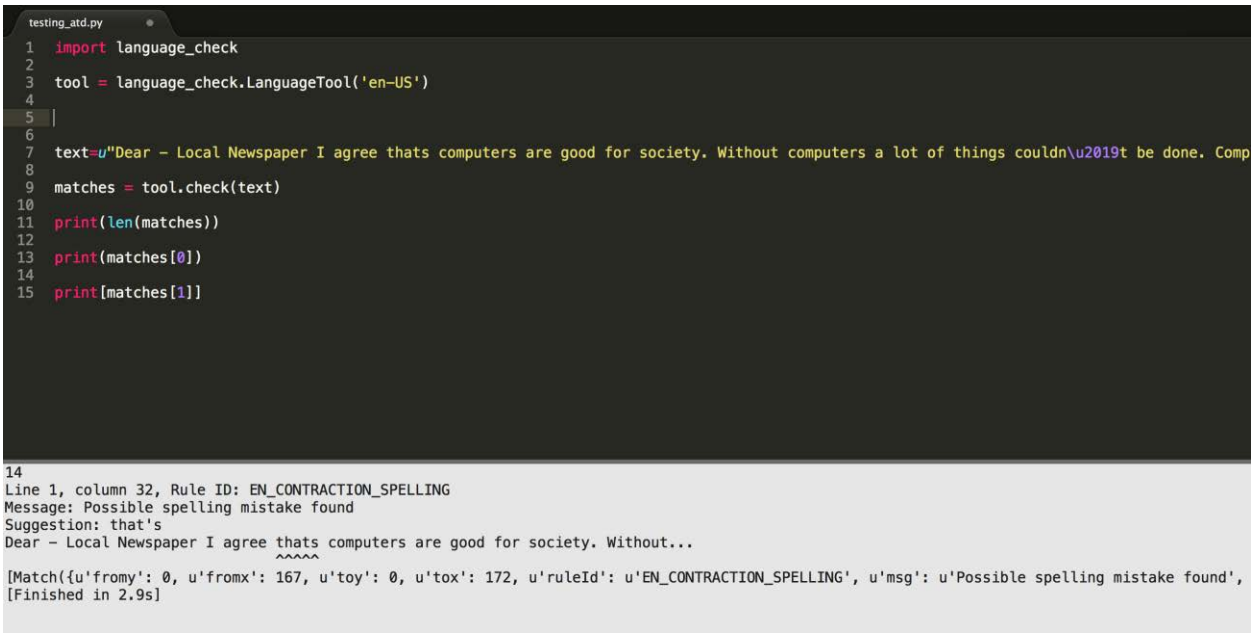
```
$ pip install --upgrade 3to2
```

Prerequisites

- Python 3.3+ (or 2.7)
- lib3to2 (if installing for Python 2)
- LanguageTool (Java 6.0+)

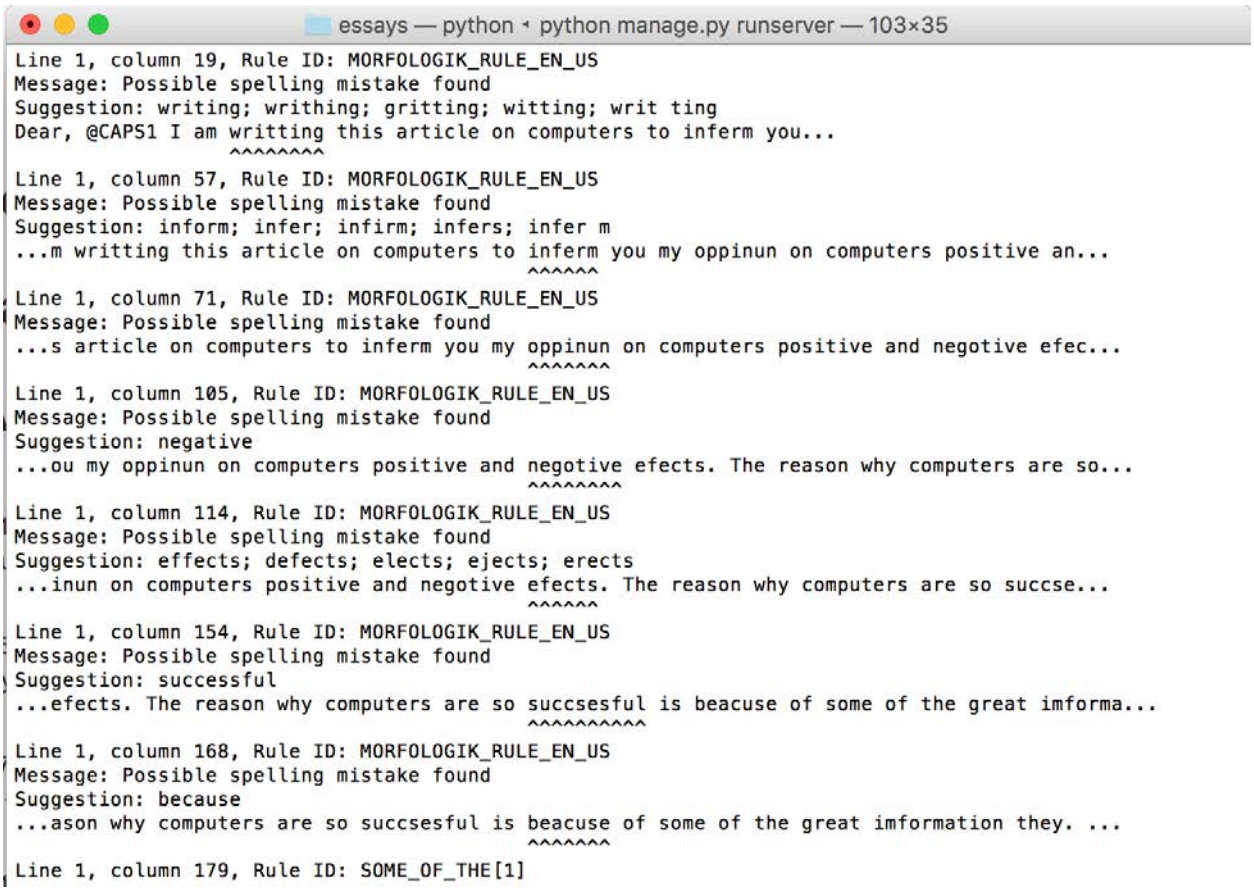
8.5 fig: shows how to install the package in the system.

Using the python package is easy all we have to do is to import language check into our environment after the installation and rest of things are as easy as it could be. We are somewhat surprise at the convenient way of using it and fairly accurate results.



8.6 fig: shows the process of utilizing the python package

As it is shown in the above figure, it not only identify the mistakes it also put an underline to show where something has gone wrong. Moreover, in the figure below it is also shown the suggestions that this tool is suggesting to more comprehendible to a writer.



8.7 fig: shows the suggestions as well as the mistakes together.

Chapter 9

9.1 Actual and Predicted Result Comparison

Using Linear regression and Random forest on the dataset we calculated various results, errors which is briefly discussed in chapter 6. Though the result from Linear regression and random Forest were pretty much the same that's why during our system implementation we chose Linear Regression as a training algorithm. We have then tested nearly 1200 essays. The result predicted by the system we developed was very similar to the actual predicted score. Here is a screenshot of the actual score and the score we have predicted.

Actual Score	Predicted Score
4	4.011599592
5	4.559157611
4	3.747726443
5	5.266436858
4	4.69384768
4	3.506443724
5	4.946921026
5	5.092648671
4	4.643369818
5	4.675042796
4	4.242688728
4	4.213801945
4	3.243604929
3	4.057257978
3	3.351218914
6	5.34390826
4	3.774274597
4	4.04385526
2	2.770476849
3	3.327597643
4	4.191007267
2	2.643873671
5	4.995420655
6	5.500359084
4	3.881584348
5	4.175066501
2	2.986861575

Figure 9.1: Actual score vs predicted score

To understand the relation between the actual and predicted score in a better way we have drawn the following graph that represents the comparison of actual score and the predicted score. Here in the picture the blue lines are the actual score graded by the human grader and the predicted score is shown in the orange color.

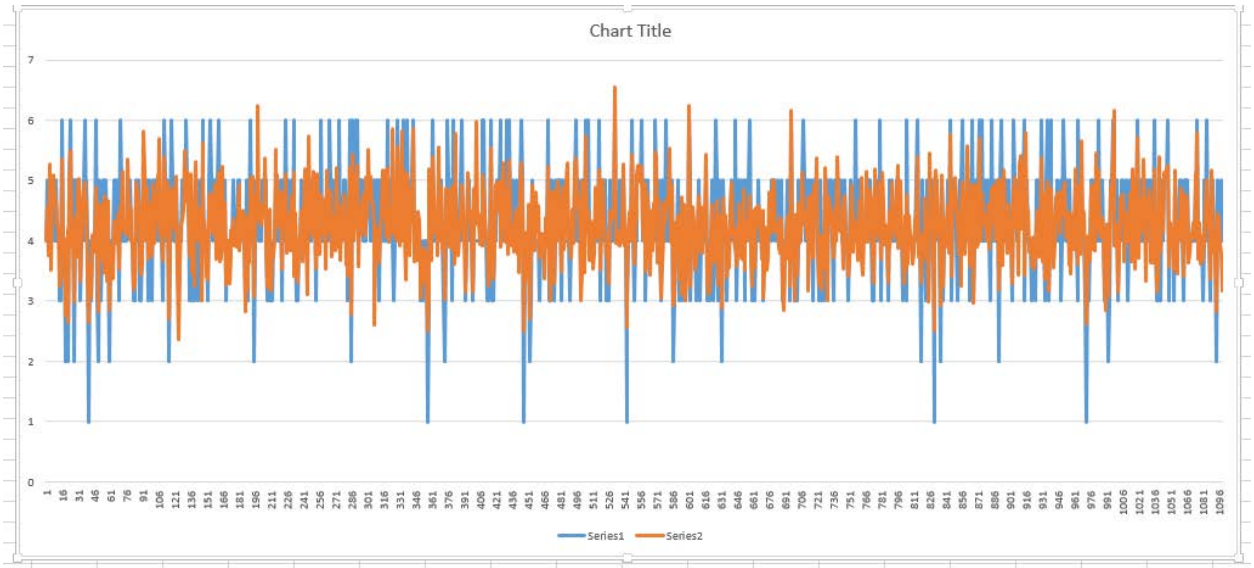


Figure 9.2: Graph of Actual Score vs predicted score

9.2 Accuracy and Precision

Our research would not be fulfilled if we do not calculate the accuracy and the precision of the results that we have obtained. Accuracy and precision speaks for itself for any kind of research, does not matter how good an accuracy or precision be these two are vital components when doing any kind of research. However, people often get confused with these two terms but in reality each of the term has its own meaning. **Accuracy** is how close to "true" measurements are. Measuring devices or techniques can easily be inaccurate and lead to false measurements, and no matter how accurate a device or technique maybe, there is still a tolerance for error. Therefore, no measurement is perfect that is why accuracy must be accounted for our own results. **Precision** is how consistent our results are for the same phenomena over several measurements or how predictable a device's or algorithm's performance can be made. Precision is a measure of variation, must be accounted for our own results and calculation.

In statistics, the coefficient of determination, denoted R2 or r2 and pronounced R squared, is a number that indicates how well data fit a statistical model – sometimes simply a line or a curve. An R2 of 1 indicates that the regression line perfectly fits the data, while an R2 of 0 indicates that the line does not fit the data at all. This latter can be because the data is more non-linear than the curve allows, or because it is random.

If y is the mean of the observed data:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

then the variability of the data set can be measured using three sums of squares formulas:

- The total sum of squares (proportional to the variance of the data):

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

- The regression sum of squares, also called the explained sum of squares:

$$SS_{\text{reg}} = \sum_i (f_i - \bar{y})^2,$$

- The sum of squares of residuals, also called the residual sum of squares:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

The most general definition of the coefficient of determination is

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

The result that we have obtained by using the above formula is **0.75** or in other words **75%** linear regression accuracy.

9.3 Variance/Standard Deviation (Precision)

The variance of a set of values, which we denote by σ2, is defined as –

$$\sigma^2 = \frac{\sum(x - \bar{x})^2}{n}$$

where ¯x is the mean, n is the number of data values, and x stands for each data value in turn. Recall that x, for example, means add up all the values of x. Similarly, (x−x¯)2 means subtract the mean from each data value, square, and finally add up the resulting values. (If necessary revise

the leaflet Sigma Notation). An alternative, yet equivalent formula, which is often easier to use is

$$\sigma^2 = \frac{\sum x^2}{n} - \bar{x}^2$$

Therefore the standard that we have obtained is **0.52**, standard deviation is a measure of how widely the results of experiment (or observations) are spread. For a mound shaped distribution usually most about 95% of the data points are within 2 std dev units of the mean. Precision is a general term regarding the spread of a distribution, things with a high precision have a low standard deviation, but this says nothing of the accuracy of the measurements. Accuracy means that on average the data points are true, but that does not exclude the possibility of widely variation in the measurements. To quote, "precision of measurements has to do with getting essentially the same values every time a particular measurement is done".

Chapter 10

10.1 Conclusion

Our research shows that automated essay grading along with the recommendation system will help writers to know their own level of competency and their mistakes as well along with suggestion of some better essays to increase their potential of writing. It also tells us that natural language processing has come a long way from where it started its journey and maybe because of those earlier ground breaking research we have accomplished the our goal. However, there are many areas of improvement specialy with the complex for languages around the world even in our system some of the critical grammatical errors are quite impossible to register into our system and therefore they have gone unnoticed. Nonetheless, it is quite evident what machine learning can do if the agent can be trained properly with large set of relevant data and reasonably good algorithm will even make the learning easier.

10.2 Future Work

As we have already established a server type system by Django's help with the limited amount of data that we could gather in this reasonably constrained time which can stably predict the score of a given essay as well as suggesting some improvements. We think there is more scope to work on making the system more dynamic and widely scattered in the Internet not just confined server. If this system can reach out to the internet and to the billions of internet users around the world it would be able to train itself with different categories of essays and in different languages also. We strongly think that with the machine leaning algorithm that we have used in our research and if this system could reach out the billions of intent users it would be an ultimate system where essays of any kind, any category could be graded and recommended also. Moreover, the Language check tool that we have used to check for grammatical errors is adequate for this research but to make this research into a larger scale this tool have to be revised to more recent versions so that it could also detect more complex grammatical errors. If Machine learning agents can be trained with more efficient algorithms than the results obtained would be more accurate and robust.

However, we firmly believe that this research is just a drop in the ocean for what is about to come in the further in terms of natural language and teaching writers on how to improve their creative writing.

GLOSSARY

Covariance: Measurement of the statistical change in values of two random variables together after each individual event.

Regression: Most widely used process for forecasting where the statistical relation of two or more variables are estimated according to the best fit line of plotted points.

Standard Deviation: The dispersion or spread of a given attribute values of a data frame can be perceived by the result of standard deviation.

Truncation error: Round off error made by truncating the outcome of a calculation to estimate a value in the finite range of set.

Stemming: A processing interface for removing morphological affixes from words. This process is known as stemming.

R Square Error: R Square is the proportion of variance in the dependent variable explained by the model.

Reference

1) Abu-Mustafa, Y. S., Magdon-Ismail, M., & Lin, H.-T. (2012). *Learning from Data* (1 ed.). AMLbook.
2) An Overview of Current Research on Automated Essay Grading", S. Valenti, F. Neri and A. Cucchiarelli,
3) After The Deadline - Spell, Style, And Grammar Checker For Wordpress, Firefox, Tinymce, Jquery, And Ckeditor". *Afterthedeadline.com*. N.p., 2016. Web. 20 Apr. 2016.
4) Automated Essay Grading" P. Reddy and G. Jambagi, University of Illinois, Chicago, USA, 2003
5) Blood, Ian. "Automated Essay Scoring: A Literature Review". *Working Papers in TESOL and Applied Linguistics* 11.(2011) (2016): n. pag. Web. 16 Apr. 2016.
6) Burstein,J.,Wolff,S.,Lu,C.,& Kaplan,R. (1997) . An Automatic Scoring System for Advanced Placement Biology Essays. Proceedings of Fifth Conference on Applied Natural Language Processing(pp.174-181). Washington D.C . : Association for Computational Linguistics
7) Coursera". *Coursera*. N.p., 2016. Web. 20 Apr. 2016.
8) "Language-Check 0.8: Python Package Index". *Pypi.python.org*. N.p., 2016. Web. 17 Apr. 2016.
9) Liaw ,Andy.Wiener,Matthew (2002) Classification and Regression by randomForest
10) "Linear Regression". *Wikipedia*. N.p., 2016. Web. 17 Apr. 2016.
11) "Machine Learning: What It Is and Why It Matters". *Sas.com*. N.p., 2016. Web. 17 Apr. 2016.
12) Mitchell, Thomas M. "Machine Learning". *McGraw-Hill, Inc.* (1997): n. pag. Web. 20 Apr. 2016.
13) Ng, Andrew.(2012). CS229 Lecture notes [portable document format].Retrieved from http:// cs229.stanford.edu/notes/cs229-ntes1.pdf
14) "Pandas: Powerful Python Data Analysis Toolkit — Pandas 0.18.0 Documentation". *Pandas.pydata.org*. N.p., 2016. Web. 17 Apr. 2016.
15) Perone, Christian. "Machine Learning :: Cosine Similarity For Vector Space Models (Part III) | Terra Incognita". *Blog.christianperone.com*. N.p., 2013. Web. 20 Apr. 2016.
16) Random Forests - Classification Description". *Stat.berkeley.edu*. N.p., 2016. Web. 15 Apr. 2016.

17) "Software - The Stanford Natural Language Processing Group". *Nlp.stanford.edu*. N.p., 2016. Web. 20 Apr. 2016.

18) The Hewlett Foundation: Automated Essay Scoring | Kaggle". *Kaggle.com*. N.p., 2016. Web. 16 Apr. 2016.

19) "The Stanford Natural Language Processing Group". *Nlp.stanford.edu*. N.p., 2016. Web. 20 Apr. 2016.

20) "Types of Machine Learning Algorithms | @Thedatayou". *Datayou.org*. N.p., 2016. Web. 17 Apr. 2016.

21) Wu, Xindong et al. "Top 10 Algorithms in Data Mining". *Knowledge and Information Systems* 14.1 (2007): 1-37. Web. 17 Apr. 2016.