CS299 : Innovation Lab (Spring Semester, IIT Patna)
Dr. Jimson Mathew, Dr. Arijit Mondal and Dr. Raju Halder

The Final Project Report

"Stackoverflow Answer Predictor"

—

By - Siddharth Thakur (1601CS46), Abhinav Siddharth (1601CS01)

INTRODUCTION

Question and Answer (Q&A) sites help developers dealing with the increasing complexity of software systems and third-party components by providing a platform for exchanging knowledge about programming topics. A shortcoming of Q&A sites is that they provide no indication on what could be potential answer. Such an indication would help, for example, the developers who posed the questions in managing their time. It also could be very discouraging to young developers if they don't have their doubts cleared. We will try to fill this gap by investigating whether and how answer for a question posted on StackOverflow, a prominent example of Q&A website, can be predicted. To fulfill this aim, we will first determine the types of answers to be considered valid answers to the question, after which the best answer answer is predicted.

The *OBJECTIVES*

In this project work, we are trying to predict the answer of a prospective question entered by a user on StackOverflow. We are going to use different data mining algorithms and will analyze by comparing their results. The main tasks to perform for this project are parsing of the questions tags, train the system based on tags and body and test the system with remaining data. We will predict accuracy of an algorithm by comparing the predicted answer to get a response against the existing answer from test data. This way we can decide the accuracy, precision, recall and the other metrics for an algorithm. The overall aim of a project is to have an idea of what could be possible answer on *stackoverflow.com*.

INNOVATION *ACHIEVED*

Currently no community Q&A sites such as Quora or Stack Exchange family of websites provide such facility of predicting answers to users.We attempt to help these users by providing a possible answer. This project has a huge market potential in Q&A sector and it could also be kind of useful in developing automated FAQs system for various organizations.

THE PROJECT TIMELINE

	Time/Date (Year-2018)
Inception/Dataset discussion and topic finalisation	Till 30th January
Project Analysis	31st January to 6th February
Dataset Generation	7th February to 15th February
Feature extraction work (answer prediction)	16th February to 20th February
Model making and training	6th March to 20th March
Accuracy validation and search for scope of improvement	21st March to 27th March
Final touch	28th March to 16th April

THE MAKING

1. Dataset Generation

The dataset required for the project was generated from https://data.stackexchange.com/stackoverflow/query/new by using SQL query. The SQL query was used in the above website to generate 50,000 records at a time. The dataset generated till now consists of about 1500000 records stored in *.csv* format. About 110000 distinct questions are present in the current dataset. Each record has following attributes-

{userid,userReputation,userUpvotes,userDownvotes,LastActivityDate, Answ-eringDate,answerCommentCount,answerBody,answerScore,ans werId,qtag-s,qTitle,qBody,qid,AcceptedAnswerId,etc.}

The above mentioned attributes were generated through this SQL query only.

2. Dataset Preprocessing

We removed unnecessary fields that came along with the dataset. Along with this, we removed the *HTML* tags from the qBody, qTitle and answerBody. Further, tags were processed and a binary dump file of the whole dataset was created for further use.

- **Data Cleaning-**

Dataset was cleaned, regex(regular expressions) were removed, spelling errors were corrected,relevancy and subjectivity of the answer was found by extracting the keywords from text using the following libraries-

- **collections**- to import counter
- **pickle**- to load and unload dump files
- **enchant.checker**- to import spellchecker
- **nltk.corpus**- to import stopwords
- **nltk.tokenize**- to import tokenize
- **textblob**- to import Textblob (subjectivity analysis)
- **rake_nltk**- to import Rake, **re** was also used.

Relevancy was measured with the help of Rake library. Keywords were extracted in order of information provided by them and a comparative value of '100' or '40' was returned if a keyword was common in the answer body and question body or answer body and question tag respectively.

3. Features Extraction

The next level was to extract the important features from the text and create *feature vector*s to train the ML model. Following order of execution was followed to carry out feature extraction process.

- **Extracting the information from text-**

The main gist of the text was extracted by *tokenizing* the text, calculating the *entropy* to predict the average information produced from each letter and also further, *markov model* was used to predict the data delivered by the text i.e. the data that the current text can deliver next. It basically will tell which data comes more often by seeing how much information it can give in the future. Following python libraries were used-

- **collections**- to import defaultdict, deque and counter
- and **re**, **random**, **math** and **textwrap** were also used.

- **Calculating the readability of text-**

Various grading levels like-*Coleman liau index*, *Flesch reading ease*, *Flesch kincaid grade*, *SMOG* index, *Automated Readability index*,etc. were calculated to predict how many years one must study to understand the text, what was the readability of the text, etc. All these data were finally brought together and a readability consensus was formed. A lot of calculations were done in order to find out the above indexes like char_count, avg_sentence_length, lexicon_count, no. of difficult words, syllable_count ,etc.

4. Creating Feature Vectors

Above created codes were imported and basically the following features were given values on which the data similarity were evaluated-

- Non-Stopwords
- No. of occurences of a word in given text.
- Relevance between the answerBody, qBody and qTags.
- Information provided by the data
- Unique words in the text
- Answer Subjectivity
- Answer Score
- Answer Upvotes
- Answer Downvotes
- Answerer Reputation
- Answer Comment Count
- Readability consensus of answer

The values assigned to these features were then used in finally creating, the *feature vectors*. These feature vectors were later used by the model to assign the labels.

5. Generating Feature Vectors from the given data

Finally, binary data dump was unloaded and all the above identified features were extracted. A calculated amount of data was then shifted to testing files and rest to the training files so as to maintain a good ratio between both the types of data. The *features_train* and *features_test* files were created through the previously created feature vectors. Labels were assigned '1' if answer_ID matched with accepted_Answer_ID and else '0'. Thus, we entered the domain of supervised learning. All these files were dumped using **pickle library.** These binary dump files were then used in testing and training the data for many ML models used here i.e. **Decision Tree**, **Random Forest Classifier**, **Naive_Bayes, k_Nearest_Neighbours.** Dump files named *features_train* and *labels_train* were used in training the data while *features_test* and *labels_test* were to used in testing the data.

6. Training the Dataset

The previously generated feature training files and label training files were used to train the data based on the algorithms used i.e. **Decision Tree**, **Random Forest Classifiers, kNN, Naive_Bayes and, Support Vector Classifier.** separately in two different files. Four separate dump files were created by the four different algorithms. Some important libraries were used to carry out the training process. **sklearn** was used to import **tree**, **sklearn.ensemble** to import **random_forest_classifier, sklearn.naive_bayes** to import **GaussianNB** and **sklearn.neighbors** to import **KNeighborsClassifier**. **numpy** and **pickle** were also used in order to size the files into an array and to load and unload the binary dump files.

7. Testing different ML Models

All the 5 models was then tested on the dump files created while training the data. The dump files were then loaded and accuracy scores were predicted to see which algorithm worked better. In our case, all algorithms worked almost similar with Random_forest_classifier being relatively better than all other algorithms both in terms of accuracy and running time. Some important libraries were also used here to carry out the testing process. **sklearn**, **sklearn.ensemble, sklearn.naive_bayes** and **sklearn.neighbors** were used for the same purposes as done while training the data**.** **numpy** and **pickle** were used for similar reasons as while training the dataset.

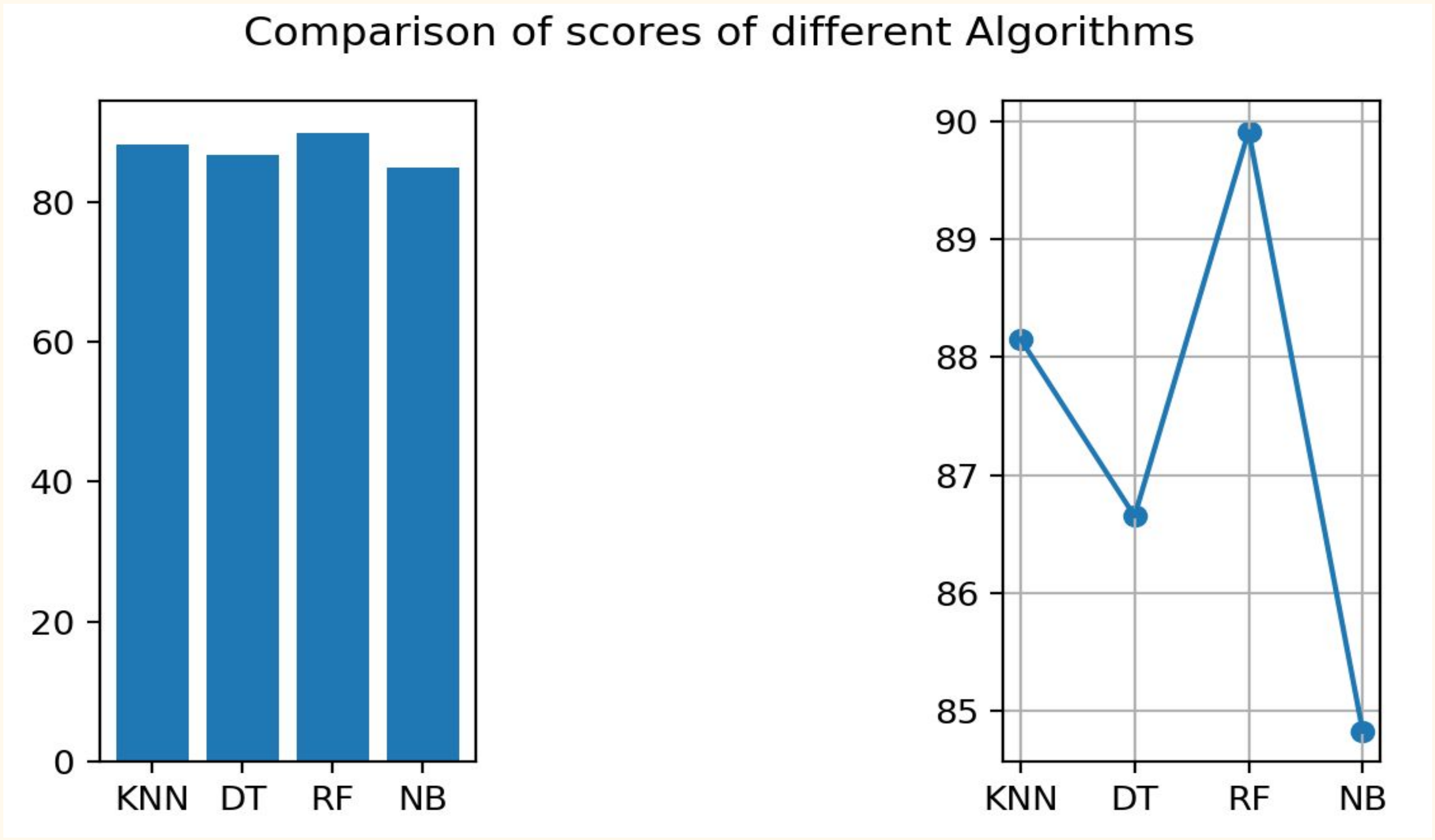
- **Plotting the test results**

Here we plot Mean accuracy (%age)Scores of 4 different algorithms (as a part of our testing processes and best algorithm selection). This was done using *“plotter.py”* code. The graph is here:- (**matplotlib** library used)

Graph Legend -

- KNN : K-nearest Neighbours Classifier
- DT : Decision Tree Classifier
- RF : Random Forest Classifier
- NB : Gaussian Naive-Bayes Classifier

(We ignored Support Vector Classifier because of its very long running time)



8. Finding the Similarities

A library called **gensim** was used to find out the similarities. **NLTK** was then used to tokenize the text. A dictionary was then created from a list of documents to map each word to a number. A corpus (**bag of words**) was then created to find out the number of times each word occurs in a document. TF_IDF model was created then to convert the **bag of words** into vector form. TF_IDF stands for term frequency-inverse document frequency. Term frequency is how often the word shows up in the document and inverse document frequency scales the value by how rare the word is in the corpus. The similarities function of gensim was then used to find out the similarities between the vectors. Separate dump files were created of all the above mentioned steps i.e. tf_idf values, dictionary, and values of similarity between any two documents.

9. Finding out the Relevant Questions

After finding out the similarities with the entered question, our next target was to find the relevant questions with the most similarities. So, we took out the first two questions having the most similarities to the entered question, for each, based on questions title and their body.

The dump files created earlier while finding out the similarities were unpacked and used in finding out the most two relevant questions which were used later.

10. The Final part of Answer prediction

After finding out similarities to the input question and thus the relevant questions, finally answer was predicted and written onto the terminal. For this, first we saw whether question was identical to any question in the dataset or not. If yes, then we automatically printed the answer and terminated the program and if no, we made use of the relevant questions we generated and we stored all their answers. After this, labels were assigned to the answers. If answer_ID matched with accepted_Answer_ID then label '1' was given else '0'. Feature vectors were then extracted of the answers stored using previously written codes and ML model was then used to predict the labels of feature vectors generated. After this, we printed all the answers whose ever label was predicted '1' by the model. If by chance, no label was assigned '1' by the model, then we printed the answer of the most relevant Question whose answer_ID matched with accepted_Answer_ID. Some important libraries used here were **sklearn**, **sklearn.externals** and **sklearn.ensemble**.

Here we present the order following which anybody could reproduce our work-

1. **Use this SQL query to generate the dataset in csv format from** https://data.stackexchange.com/stackoverflow/query/edit/754116

```
select g.*,Users.Id as userId, Users.Reputation, Users.UpVotes as
UserUpvotes, Users.DownVotes as UserDownvotes
from Users join
(select Posts.Id,Posts.AcceptedAnswerId,Posts.ParentId as
ParentQqestID,Posts.Score as qScore,Posts.ViewCount as qViewCount,
Posts.OwnerUserId as qAsker,Posts.Body as qBody,Posts.Title as qTitle,
Posts.tags as qTags,
Posts.CommentCount as qCommentCount, Posts.CreationDate as
qCreationDate, Posts.LastEditDate as qLastEditDate,
Posts.DeletionDate as qDeletionDate,
p.Id as answerId,p.Score as answerScore,p.ViewCount as
answerViewCount,
p.OwnerUserId as answerer,p.Body as answerBody,
p.CommentCount as answerCommentCount, p.CreationDate as
AnsweringDate, p.LastActivityDate
from Posts join Posts as p on Posts.Id = p.ParentId where Posts.Id
between 150001 and 200000)
as g on Users.Id = g.answerer order by Id
```

2. **Run '*dataset.py*' for preprocessing to remove unwanted html tags and regex, etc.**

3. Run '*generate_feature.py*' to generate feature vectors. This code calls following files for attaining its purpose.

- **'*feature_vectors.py*' - it create feature vectors by using following utility files.**
    - **'*info.py*' - it calculates and provide information provided by the 'answerBody' using markov model and entropy.**
    - **'*readability.py*' - this provides the readability consensus.**
    - **'*properties.py*' - this was used to get various properties of the text.**

4. Then run the various training and testing files such as '*Train_1.py & Test_1.py*', '*Train_2.py & Test_2.py*', etc. to be sure of proper working and accuracy of the model.

5. Now run "*TFIDF_util.py*" to prepare dataset for similarity analysis.

6. Finally run the '*final.py*' to get live demonstration of answer prediction, this uses '*rel.py*' to find out relevant questions similar to the user entered question. This now predict labels to answers of the got relevant question.

TOOLS AND TECHNOLOGY USED

Familiarity with Python and the rich library support for data mining, were the two factors that contributed to our choice of Python as programming language. We used sublime text to develop the code. Wikipedia was also used as a reference. Basic problems were sort out with the help of Internet and Professors. CS Dept. server was used to train our model on such large dataset. Newest version of python(version 3.6) was used in order to get high efficiency and rich library support.

CHALLENGES FACED and WE OVERCAME

At first, the major challenge was to get the dataset as the required one wasn't available on Kaggle. So, we took to generating our own having the required fields using SQL query on data.stack exchange website. We ran the same code 15-20 times to get the required amount of data as the api call limits itself in amount supplied at each call.

The second challenge was to decide which features we need to incorporate for answer prediction and how we will generate them. Initially, we decided upon common features of answer body, and after getting more insight through rigorous research we arrived at final consensus of the presently used features.

Moreover, we were puzzled to decide upon which ML model to be used so, we tested on 5 different algorithms to choose the best in terms of time and accuracy. As, our dataset became quite large (1,500,000 data (~1GB)), it became more challenging to train and test different models to choose the best one.

SOURCES AND REFERENCES

- https://www.wikipedia.org/
- https://data.stackexchange.com/stackoverflow/query/edit/754116
- https://stackoverflow.com
- https://www.python.org/

END