CS299 : Innovative Lab Presentation

"Stackoverflow Answer Predictor"

Team-

1. **Abhinav Siddharth (1601CS01)**
2. **Siddharth Thakur (1601CS46)**

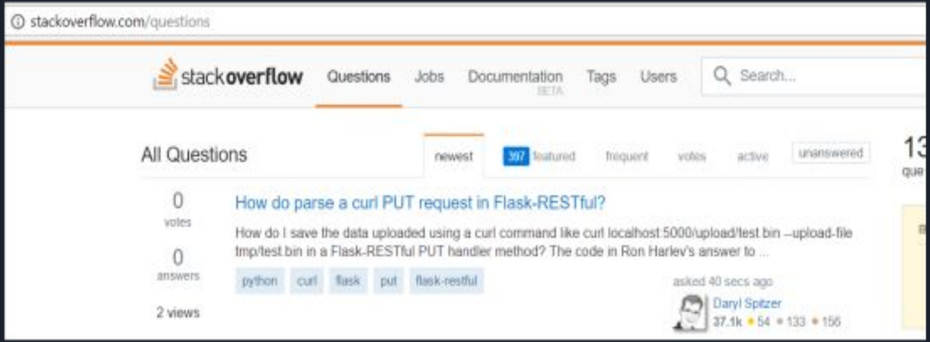
Overview

StackOverflow is an online question-and-answer site for programmers. Started in mid 2008, its rich feature set brought rapid popularity: users can ask and answer questions, collaboratively tag and edit questions, vote on the quality of answers, and post comments on individual questions and answers. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers.

Understanding the problem

01 A shortcoming of Q&A sites is that they provide no indication on what could be potential answer.

02 We also don’t know when our posted question will be answered or ever be answered by the community at all.


stackoverflow.com/questions

All Questions

0 votes

0 answers

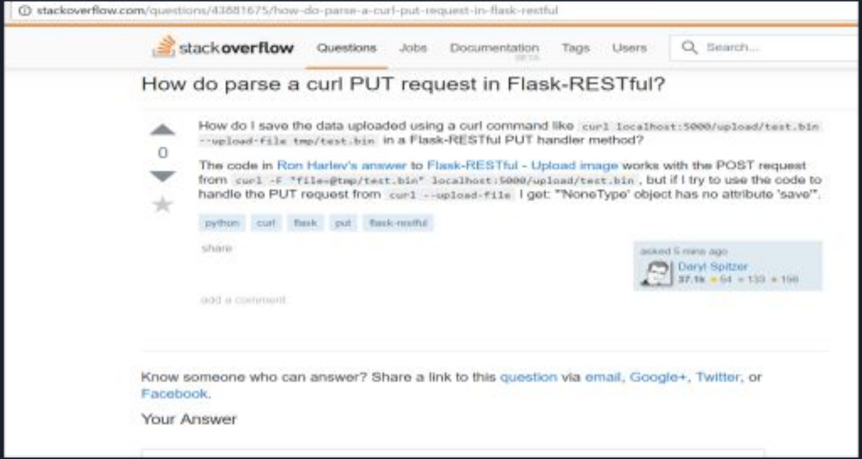
2 views

How do parse a curl PUT request in Flask-RESTful?

How do I save the data uploaded using a curl command like curl localhost:5000/upload/test.bin --upload-file tmp/test.bin in a Flask-RESTful PUT handler method? The code in Ron Harlev's answer to ...

python curl flask put flask-restful

asked 40 secs ago
Daryl Spitzer
37.1k ● 54 ● 133 ● 156


stackoverflow.com/questions/43881675/how-do-parse-a-curl-put-request-in-flask-restful

How do parse a curl PUT request in Flask-RESTful?

0

How do I save the data uploaded using a curl command like `curl localhost:5000/upload/test.bin --upload-file tmp/test.bin` in a Flask-RESTful PUT handler method?

The code in Ron Harlev's answer to Flask-RESTful - Upload image works with the POST request from `curl -F "file=@tmp/test.bin" localhost:5000/upload/test.bin`, but if I try to use the code to handle the PUT request from `curl --upload-file` I get: "'NoneType' object has no attribute 'save'".

python curl flask put flask-restful

share

asked 5 mins ago
Daryl Spitzer
37.1k ● 54 ● 133 ● 156

add a comment

Know someone who can answer? Share a link to this question via email, Google+, Twitter, or Facebook.

Your Answer

Project objective

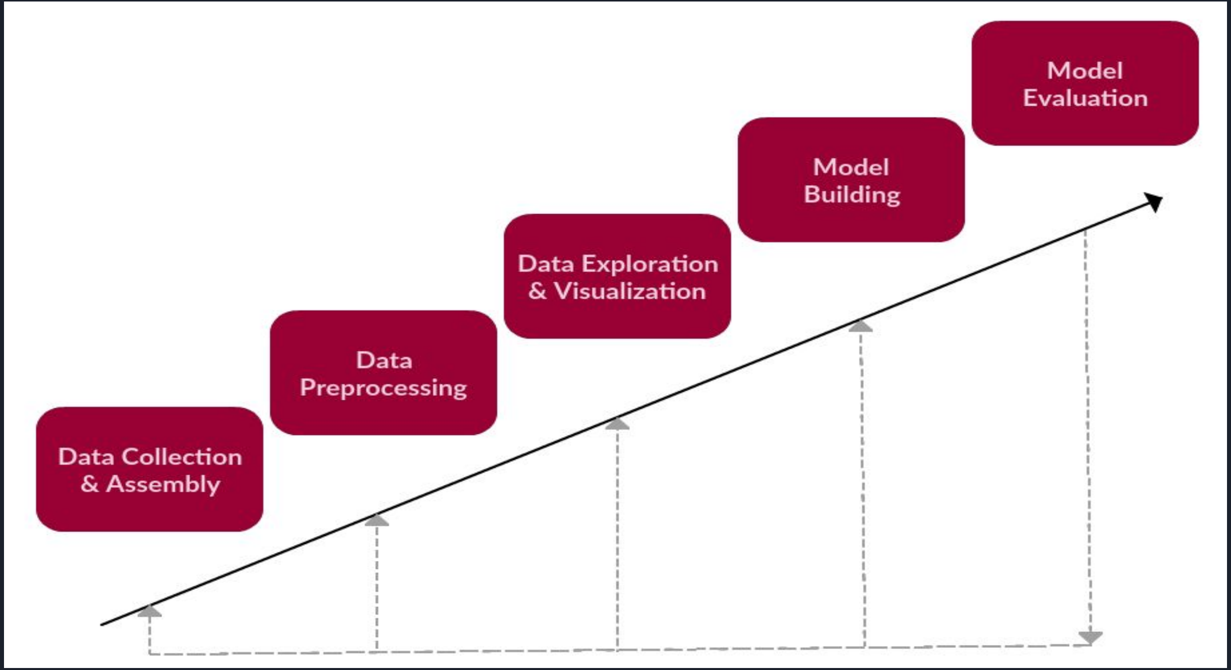
...driving the innovation

In this project, we will try to predict an answer to any question asked by user on StackOverflow.We will also look forward for any scope of improvement in our project. Such advancement in predicting answer to the questions has not been applied by any Q&A website yet.

Target audience

Such an indication would help, for example, the developers who posed the questions in managing their time. It could be very discouraging to young developers if they don’t have their doubts cleared. So, we aim towards helping them through this project.

The Procedure Followed



The Making of the :-
‘StackOverFlow Answer Predictor’

Dataset Generation

The dataset required for the project was generated from https://data.stackexchange.com/stackoverflow/query/new by using SQL query.

Our generated dataset consists of 15,00,000 records that are stored in .csv format that consists of about 1,10,000 distinct questions. Each record has following attributes e.g. UserId, UserUpvotes, UserDownvotes, QuestionBody, AnswerBody, QuestionTags, AcceptedAnswerId, AnswerCommentCount, etc.

Dataset Preprocessing

Unnecessary fields that came along with the dataset were removed. Along with this, we removed the *HTML* tags from the QuestionBody, QuestionTitle and AnswerBody.

Data was cleaned by removing regex(regular expressions), spelling errors were corrected, relevancy and subjectivity of the answer was found by using library textblob’s inbuilt function, etc.
Relevancy was found out with help of Rake library and a comparative value (of 100 or 40) was returned whether a keyword was common to answerBody and (questionBody or questionTag) respectively.

Feature Extraction

The main gist of the text was extracted by tokenizing it, the entropy was calculated to predict the average information produced by the text. (text refers to answerBody).

Readability of the text was then calculated using the various grade levels defined, to come to an outcome of a readability consensus. The readability of the text was calculated using some of the well known grade levels like Gunning fog index, Flesch reading ease, Flesch kincaid grade, etc. A lot of calculations were done in order to calculate the grade levels like character_count, syllable_count, no. of difficult words, avg. sentence length, etc.

Feature vectors were then created on various features e.g. No. of non-stopping words, No. of difficult words, No. of unique words, Answer Count, Answer Subjectivity, Answerer Reputation, Answer Upvotes, Answer Downvotes, Relevancy between Answer Body, Question Body and Question Tags,etc.

The dataset created earlier was then divided into training and testing data upon which ML model will be trained and tested later. *features_test* and *features_train* were created through the previously created feature vectors. *labels_test* and *labels_train* were created by putting labels ‘1’ and ‘0’ if acceptedAnswerID matched with the answerID. These dump files created by pickle library were then tested upon ML algorithms.

Finding the Similarities

A dictionary was 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. *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. NLTK and gensim was used to carry it out.

Finding Out 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.

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.

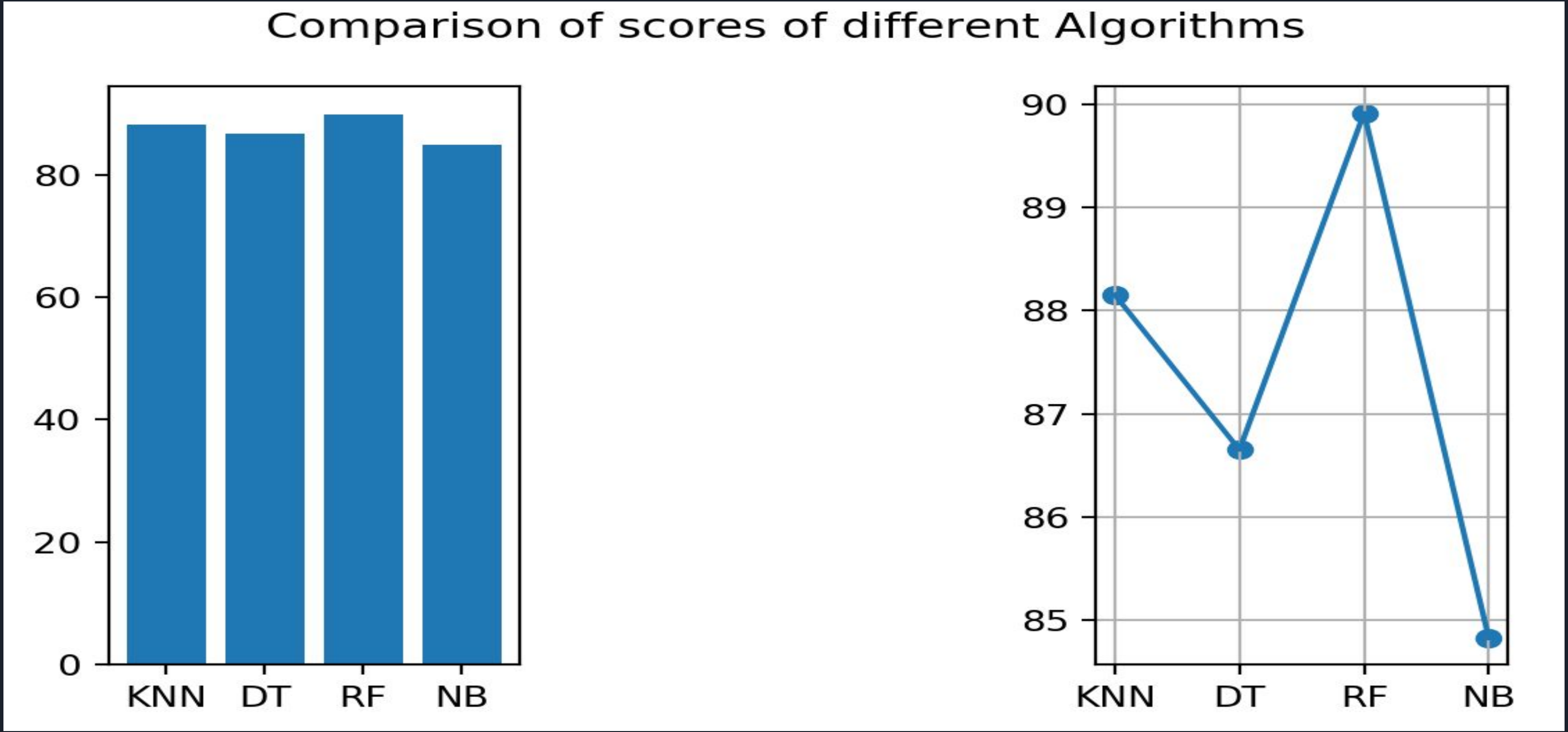
Training and Testing The Dataset

The previously generated feature training files and label training files were used to train the data based on different algorithms used i.e. **Decision Tree** ,**Random Forest Classifiers, kNN, Naive Bayes** separately in different files. Separate dump files were created one for each.

The model was then tested on these dump files and accuracy scores were predicted for each.

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.

Mean accuracy Score comparisons of different ML models



Final 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 whosoever 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**.

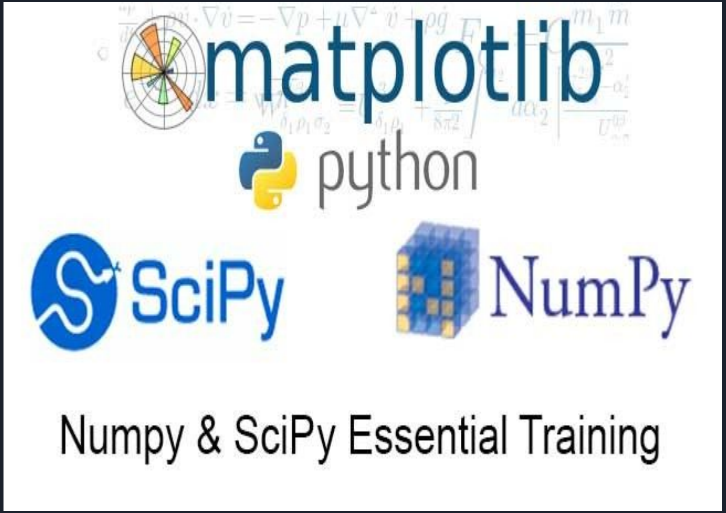
The Challenges Faced and We overcame!

At first, the major challenge was to get the dataset as the required one wasn’t available on Kaggle. So, we generated 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 features we need to incorporate for answer prediction and how we will generate them. After getting more insight through rigorous research we arrived at final consensus of the presently used features.

Which ML model to be used so, we tested on 5 different to choose the best in terms of time and accuracy, (1,500,000 data (~1GB)), it became more challenging to train and test different models to choose the best one.

Tools and Technology Used-



- Python 3.6 with various libraries such as SciPy, CSV, RE, Pickle, NLTK, Matplotlib etc.

- Sublime Text (for writing code)

- SQL for dataset generation.

- 172.16.37.26 cluster server of CSE Dept.

Thank You!