BUDT758X- Data Processing and Analysis in Python

Analysis and Classification of Stack Overflow Questions

A Project By

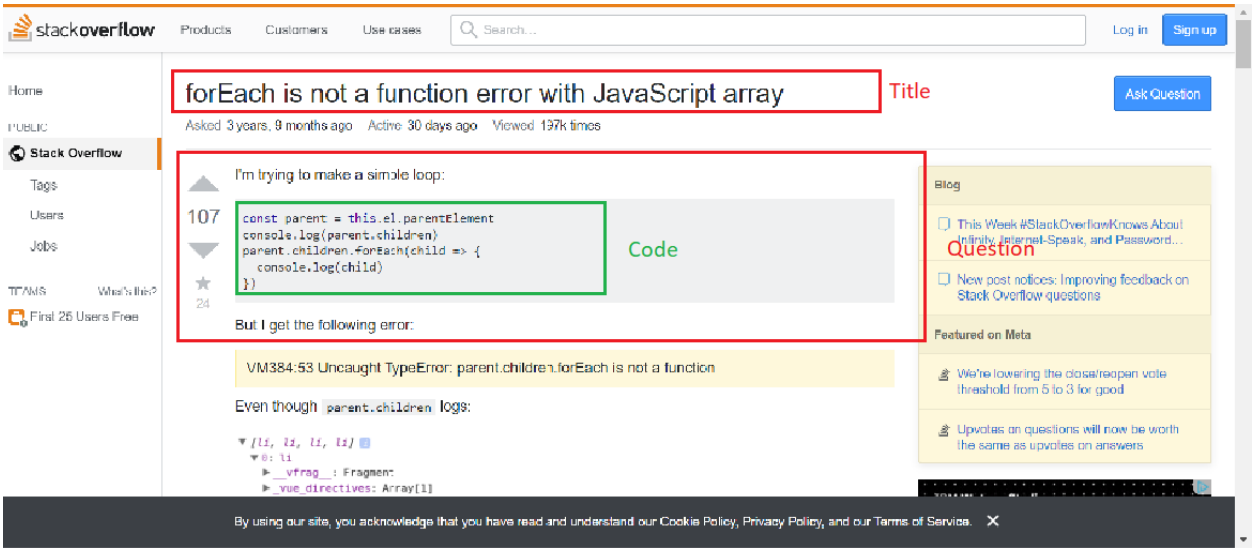
Anuj Doshi

Table of Contents

1. Project Overview

Stack Overflow has grown to become one of the most popular question and answer website among programmers and software developers. These questions posted on the platform usually contain code snippets. Stack Overflow relies on the fact that the users have properly tagged the questions and assumes that the programming language of the code snippets is the same as the tag of the question itself. The goal of this project is to classify questions retrieved from the Stack Overflow database available on BigQuery based on the programming languages they are related to. We analyzed about 7500 records in order to understand how to leverage the attributes in the stack overflow database to accurately classify the questions into five different programming languages (android, java, javascript, asp.net, ruby).

An example of a Stack Overflow post:



2. Dataset Description

The data set was obtained from BigQuery (Stack Overflow database). The Stack Overflow database contains multiple tables like badges, comments etc. Our primary data set includes columns - title, body, tags and view_count from the post_questions table. The table originally contains 18154493 records, but for this project we have chosen 7500 records due to memory and processing constraints.

3. Feature Engineering

The dataset that we considered for our project was the Post Questions Table which consists of four attributes- Title, Question, Labels, Views. Before training the predictive models to come up with predictions for classification of questions, we decided to first explore the data to determine what features were to be considered as inputs to the models.

The first features that we thought of analyzing for their usefulness were the number of views each question received as well as the length of each question. Maybe due to the complexity of the language, questions belonging to that language might have been longer in length. Or that due to the popularity of a language over another, questions related to that language might have more views. Since these features were numeric, we thought of analysing them first. To analyze the length of questions, we created a new column by applying the length function on the question column. After plotting graphs which are explained in exploratory data analysis, we realized that the languages are evenly distributed across the number of views, and that the length of the questions is fairly same for all of the languages.

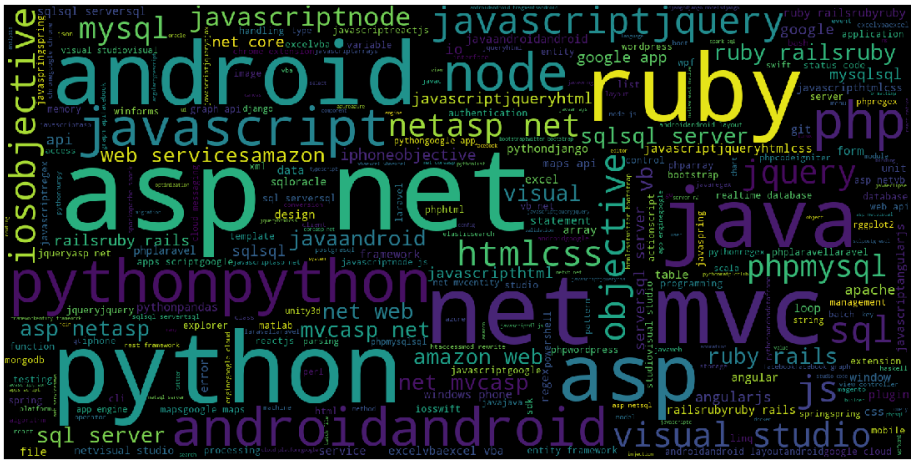
We decided to find different features that would be useful for classifying the questions. In stack overflow questions, questions are asked by first summarizing the problem, and then describing the problem which most of the time includes code blocks. We thought of extracting the code blocks from the questions to hone onto the syntactic differences between each language. For example, in JavaScript, quotes are used a lot since the language is mostly used in conjunction with HTML, or in Java, the dot operator is used to access members of a package or a class. We also decided to extract features based on the number of brackets used, the number of arithmetic operators, and punctuations. Thus, we had features like colon count(:), semicolon count(;), slash count(/), cbracket count({}), sbracket count([]), quote count(“ and ‘), operator count(<,>,-,+) and period count(.). Since we considered only 7500 records for our analysis, we decided to scale the features to remove the effects of uneven retrieval of SQL querying. For this, we used the MinMaxScalar package provided by Scikit Learn library. From the Exploratory Data Analysis we noticed that these features were indeed useful in classifying the questions.

We also analyzed the content from the title and questions columns to find useful features. For this, we first used bs4 package from Beautiful Soup library to parse the html and get the actual text. This was followed by converting the text into tokens for which we used Countvectorizer provided by Scikit Learn library. Stop words and punctuations were excluded from the text.

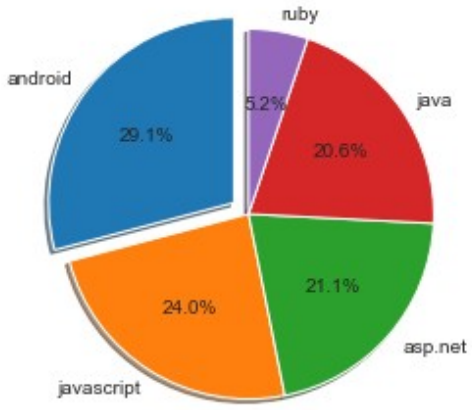
Since the length of questions and number of views columns have far greater values than the other columns, we scaled these two columns using the MinMaxScalar from Scikit Learn. Our final data frame consisted of the following features: vectorized_title_df, vectorized_question_df, scaled_views_and_length_df, scaled_code_features_df.

4. Exploratory Data Analysis

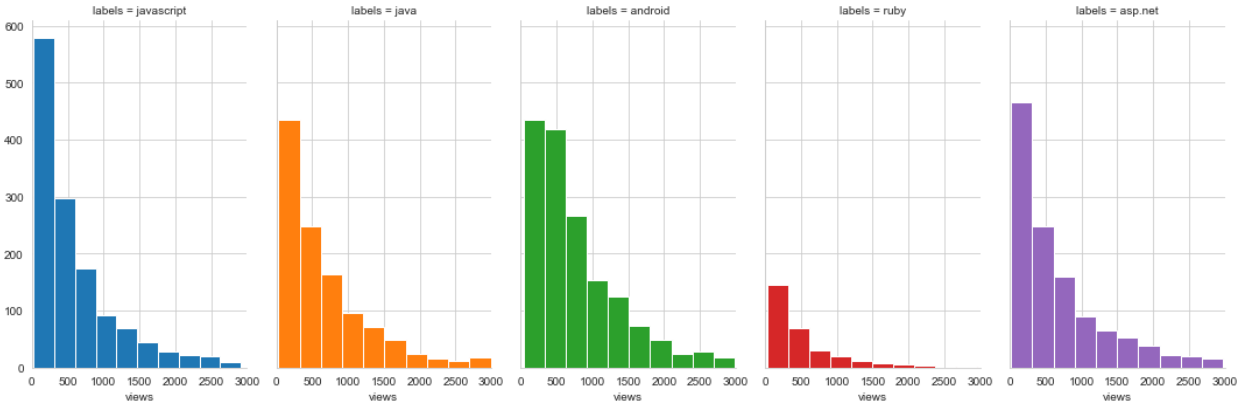
We created a word cloud to choose the most popular tags there by choosing those languages as our labels for classification purpose.



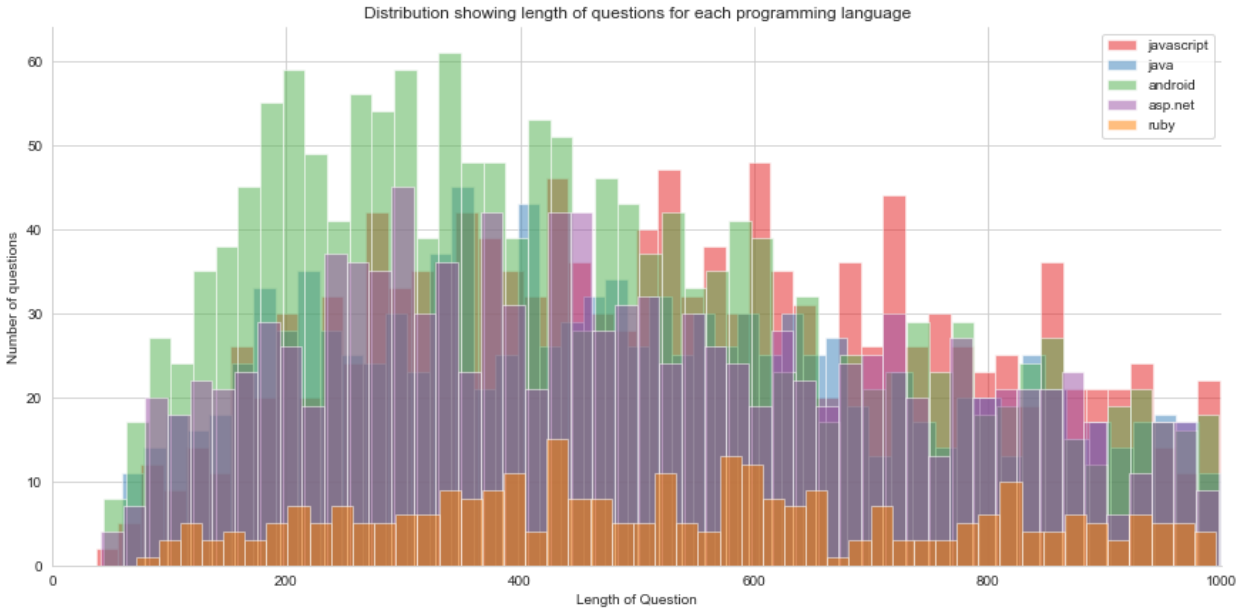
From the above created word cloud, the most used tags are - android, asp.net ,java ,javascript and ruby. We also created a pie chart to see percentage of each tag in our primary data set.



The aim behind performing data visualization was deciding which features were needed to be extracted to train deep learning model and Naive Bayes model. We decided to explore the distribution of number of views of each questions since depending upon the popularity of the language the views will differ. We also explored length of the questions since depending upon the complexity of the language, the length of questions will differ.

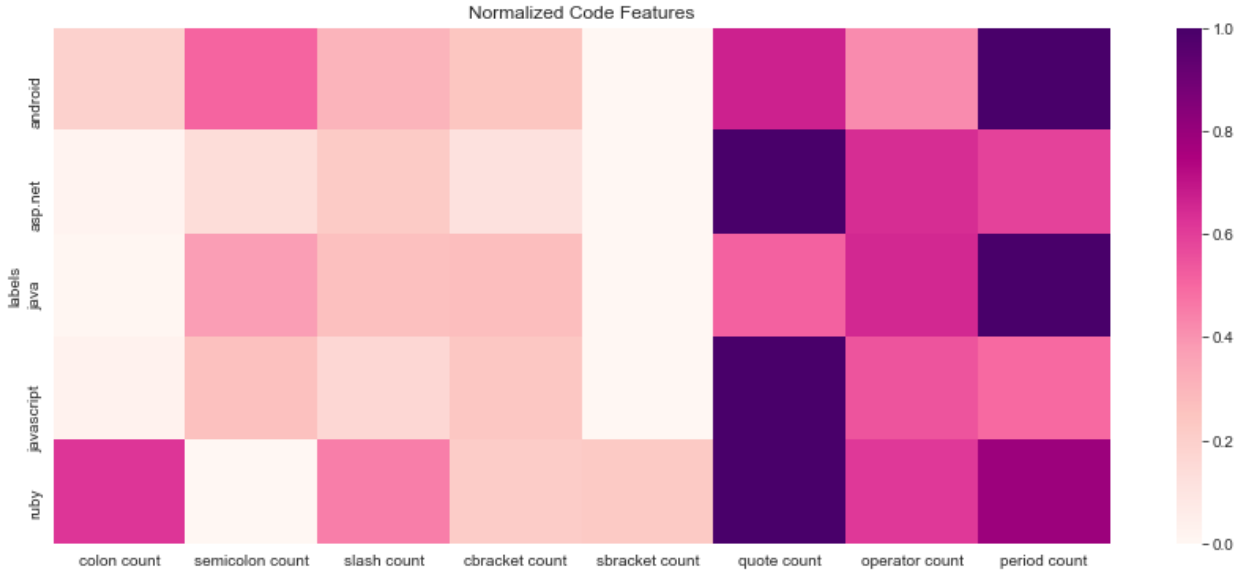


It appears from the above graph that most languages have evenly distributed number of views.



The above figure showing the distribution showing length if questions for each programming, there are no trends visible. This means that the questions are fairly similar in length.

We can conclude that the length of the questions and number of views are not that significant, we decide to extract features from the code mentioned in the questions. Each programming language has a syntactic difference as well as differences in the number of brackets used, number of arithmetic operators used etc.
After extracted the code and performing feature engineering, we created a heatmap to see which syntactic features are most prominent in the respective programming languages.



The above heatmap is skewed since it completely depends on how many of each type of question was retrieved by the SQL query. Some observations we made are:-

- The quote count for javascript is high. This is because javascript is used in conjunction with HTML and quotes occur frequently in HTML to denote text.
- The period count of java is high since Java is an object-oriented programming language and methods in java are denoted by periods.

5. Predictive Analytics

We implemented Naive Bayes and Deep Learning model for classifying the questions into five languages: asp.net, java, JavaScript, ruby, android.

A summary of the predictions is as follows:

Model	Accuracy	Precision	Recall	F1-Score	Support
Neural Network	0.82	0.82	0.82	0.82	1752
Naïve Bayes	0.70	0.79	0.70	0.64	1273

Deep Learning Model:

Since our final data frame consisted of huge number of features, the first model that came to our mind is deep learning model- neural network. We implemented deep learning using Keras, which is a framework for developed for building and training deep learning models.

We have built a sequential model containing 5 dense layers. The layers have 6000, 2000, 500, 50 and 5 neurons at each of the five layers(starting from the first layer) respectively. The activation function for the hidden layers is Rectified Linear Unit (ReLU) to avoid and rectify the vanishing gradient problem. Since this was a multiclass classification problem, it made sense to have Softmax as the output activation function. We chose Adam optimizer so that the learning rate for the parameters could be adjusted. Our loss function is Categorical Cross-Entropy. We also added a dropout of 0.25 at each layer to avoid overfitting.

Results:

We achieved an accuracy of 0.815 using the deep learning model. The metrics that we considered to assess our model were mainly precision and recall.

	precision	recall	f1-score	support
0	0.90	0.87	0.88	580
1	0.73	0.76	0.75	232
2	0.77	0.81	0.79	368
3	0.81	0.78	0.80	464
4	0.74	0.80	0.77	108
micro avg	0.82	0.82	0.82	1752
macro avg	0.79	0.80	0.80	1752
weighted avg	0.82	0.82	0.82	1752
samples avg	0.82	0.82	0.82	1752

0.815068493150685

Naive Bayes:
This project helps demonstrate the power of the naive-bayes algorithm. We have made use of the model for the following reasons:

- The assumption made by Naive bayes is that the features are independent within classes. This model could be used for our problem as the conditional probabilities between words are so small, they are almost negligible.
- For text classification problems, the number of features are very high and the observations are relatively fewer. Naive bayes model works well in such situations.

accuracy			0.70	1273
macro avg	0.88	0.47	0.46	1273
weighted avg	0.79	0.70	0.64	1273

By using the Naive Bayes model, we get an accuracy of 70%

Naive-Bayes with TFIDF :

	precision	recall	f1-score	support
android	0.95	0.60	0.74	264
asp.net	1.00	0.01	0.02	102
java	0.83	0.73	0.78	347
javascript	0.59	0.99	0.74	476
ruby	1.00	0.01	0.02	84

For Stack Overflow, having high precision and recall values is business essential. If we want to predict any of the tags we should have a high precision value i.e, we have to be really sure that the predicted tag belongs to the given question. Also, we want to have a high Recall rate, which means If the tag actually supposed to be present, we want it to be present most number of times.

6. Challenges

There were quite a few challenges that we faced during our analysis and classification of questions. One of the main challenges was to retrieve data from Google Bigquery. Due to memory limitations, we were unable to retrieve and analyze all the records from the dataset and had to limit our rows to 7500. We tried to retrieve the records randomly by using Rand() in SQL query and limiting the records to 7500. Due to this, the size of vectorized questions and titles was getting changed according to the words in each of them. Because of this we were unable to fit data on our predictive models. Hence we decided to include the first 500 rows in our dataset. Another challenge was to parse the content of title and questions properly using lxml and beautiful soup parsers.

7. Future Scope

In this project, we have made use of syntactic feature like characters. In the future, we would like to develop a way to use other syntactic features like spacing and stylistic tendencies used by users in different coding languages.

Since this project was on a BigQuery dataset and involved text classification, we were unable to use random sampling on the data. By using greater computational power, we would like to use random sampling or use a larger portion of the dataset.

We have made use of multiclass classification and didn't select data having multiple tags assigned to them. If you wish to implement an auto-tagging feature, then using multilabel classification is very crucial, i.e, each question could be assigned multiple tags. This would require us to decompose the multilabel problem into multiple independent binary classification problems.

8. Conclusion

Using feature engineering and natural language processing techniques: We were able to classify the questions from Stack Overflow based on the titles and their content with an accuracy of 82%. These results were achieved using Neural Network in conjunction with the extracted features from the code blocks of the text.

This auto-tagging feature is extremely business critical for Stack Overflow. The more accurately Stack Overflow can predict these tags the better it can create an Ecosystem to send the right question to the right set of people.

9. Appendix

- Research Paper: Predicting the Programming Language of Questions and Snippets of StackOverflow Using Natural Language Processing https://www.researchgate.net/publication/327835784_Predicting_the_Programming_Language_of_Questions_and_Snippets_of_StackOverflow_Using_Natural_Language_Processing

- BigQuery public dataset stackoverflow Q&A

https://cloud.google.com/blog/products/gcp/google-bigquery-public-datasets-now-include-stack-overflow-q-a

- Auto Tagging of Stack Overflow Questions
  https://towardsdatascience.com/auto-tagging-stack-overflow-questions-5426af692904