Flagged Post Analysis

Shantanu Jain
UNI: slj2142
slj2142@columbia.edu

Meghana Joshi
UNI: mmj2169
mmj2169@columbia.edu

Mohammed Aqid Khatkhatay
UNI: mk4427
mk4427@columbia.edu

Abstract

In the past decade, Q&A websites such as stack exchange, quora, reddit etc have become immensely popular. With a stark increase in users, there is also an increase in posts which may be of low quality and can often misguide the users. In order to allow for a well-informed community on Q&A websites, this project aims to analyze leading features that cause a post to be considered as low quality, and to predict if a given post is of good quality or poor quality. In order to accomplish this, the techniques involve rigorous data visualization, feature selection and extraction and making use of a spark pipeline to train a LSTM encoder-decoder model in order to efficiently label the data as having either high or low quality. Finally, a machine learning model that is built using Logistic Regressions is used to classify the posts using user-based, community-based and text-based features from Stack Exchange.
Keywords: Stack Exchange, Spark,LSTM, Data Visualization, Logistic Regressions, Machine Learning

1. Introduction

Stack Exchange is a large network of Q&A websites about topics in several distinct fields such as Computer Science, Data Science, Mathematics, etc. On Stack Exchange, the answers, and users are subject to a reputation award process, which makes it a self-moderated platform. However, with the stark increase in users, there has also been a simultaneous increase in flagged and poor quality posts, which reduce the quality of answers within the community.

From a user point of view, users are interested in finding the highest quality answers to their searched queries and they are looking to avoid answers of low quality which could potentially misdirect them. On the other hand, from a developers perspective, the primary objective is to enhance the experience of users so that they have an efficient experience.

The goal of this project is to analyze and identify the features of poor quality posts, so as to build an efficient and highly reliable machine learning model to classify posts on the basis of its quality. Our project addresses this goal by utilizing techniques of data visualization to observe underlying patterns, feature selection and extraction to strengthen the classification model by feeding it with reliable features. The features that will be given as input to the model are a combination of user-based features, community-based features and textual features extracted from the body of the posts.

The proposed solution consists of two main machine learning models - a semi-supervised LSTM model used to label fresh data and a supervised Logistic regression model used to classify posts. Apart from these models, the solution relies on rigorous data analysis and visualization to understand the features which determine the quality of the post. Additionally, pipelines using PySpark and BigQuery are used to efficiently manage the high volume of data.

2. Related Works

There have been several practicals as well as applied science attempts towards solving the stack overflow question quality problem or to reduce the moderation queue for moderators. Many techniques and advances have been applied. We discuss a few of them below to get inspiration for our project.

2.1. Bidirectional Encoder Representations from Transformers for QA subjective features

The first paper that we study is [1] , where the authors use crowdsourced data from Google Research and applies BERT to predict quality of QA website posts. The paper focuses on how they apply a pre trained BERT using transfer learning in order to achieve this.

2.2. StackOverflow Code Snippets origin & evolution

The second paper that we study is [2], where the authors defined a methodology to collect data easily. This approach also presented a well defined schema design for the entire system which was easily understandable. This paper majorly focussed on the data collection methodology and made the data set obtained open source. The data dump is huge

Related Work	Drawbacks
BERT [1]	1. Requires pre-defined linguistic benchmarks for QA quality. 2. Polarity and subjectivity from sentiment analysis problematic.
SO CodeSnippet [2]	1. Copy and Paste Bugs can originate from genuine users also. 2. BugFixing Edits can be good.
SO PostReconstruction [3]	1. Code/Language Evolution is not predictable. 2. PostQuality not always dependent on Evolution.

Table 1. Summary of related works.

Related Work	Our Approach
BERT [1]	1. We use LSTM with textual features for QA quality. 2. NLP based features can resolve polarity and subjectivity.
SO CodeSnippet [2]	1. UserReputation benchmark solves the problem. 2. We consider only Short Term evolution.
SO PostReconstruction [3]	1. Code or Language Evolution can be derieved from post outline. 2. PostQuality dependent on code based features.

Table 2. Our Approach to the drawbacks of related works.

and hence we uploaded it to BigQuery for the purpose of our project and to handle big data efficiently.

2.3. StackOverflow Post Reconstruction

The second paper that we study is [3], where the authors describe strategies to textually and subjectively analyze the data that was designed in the previous schema. They also presented valuable insights regarding how minor edits that are made right after a post is posted are harmless and indicate that the post is genuine in some way. They also found that the code cell in a block will not be changed in any way unless the text part surrounding it is also changed which was a key insight. GitHub links also play a key role in the way good and bad posts are structured.

2.4. Summary of approaches

In the following table we show how we plan to handle the drawbacks as presented in the above table.

3. Data Overview

There have been several practicals as well as applied science attempts towards solving the stack overflow question quality problem or to reduce the moderation queue for moderators. Many techniques and advances have been applied. We discuss a few of them below to get inspiration for our project.

3.1. Data Collection

Loaded stack overflow data dump from 12 months to Big Query containing 96M posts. Created dataset by extracting posts which were present in December but not in January using temporary relations and views in Google BigQuery.

Figure 1 shows how a StackOverFlow answer looks like. Each component of the answer is well presented on the StackOverFlow website with clearly separate code, userID likes, section.

In figure 2, we show how the answer looks after getting loaded in a dataframe.

In figure 3 we show how the question and answer can be accessed only by using the postID attribute of the data.

3.2. Data Cleaning and Preprocessing

Data cleaning is a fundamental element of any analytics task. It is the process of recognizing and rectifying incorrect, incomplete and inaccurate data for uniformity to the machine learning model. It is important to deal with improper data, by either filling up missing values, updating the inaccurate information or deleting the incomplete records entirely.

The dataset received from stack overflow data dump contained the following abnormalities:

- Missing Attributes such as postID, postBody, etc.
- Heterogeneous post body containing a mixture of code snippets, text and HTML attributes.
- Escape characters, which behave in an abnormal way when parsed by different libraries in pyspark.
- Improper data types for different attributes.

The following decisions were taken on dealing with the above abnormalities:

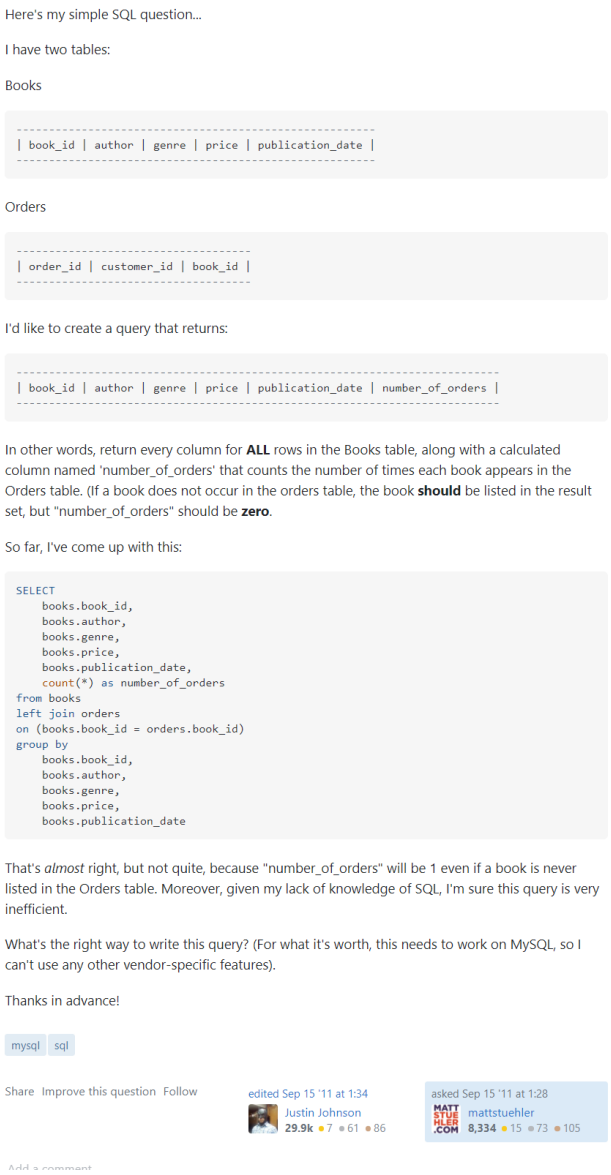


Figure 1. Example of how a StackOverFlow answer looks like.

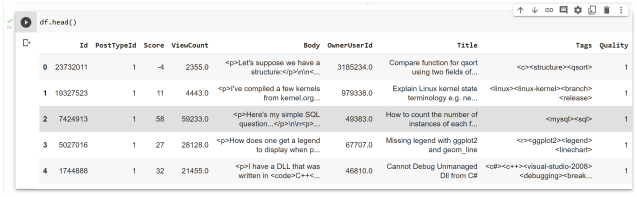


Figure 2. Example of how Fig 1 answer would in a row in our dataframe

- **Dropped posts with missing attribute values:** We identified that records with missing/NULL values had the majority of their attributes as NULL, and hence were rendered useless for our purposes.

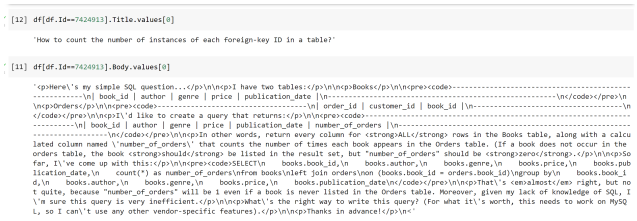


Figure 3. Example of how Fig 1 answer would in a when accessed

```
clean_df.printSchema()
```

```
root
 |-- Id: string (nullable = true)
 |-- PostTypeId: string (nullable = true)
 |-- Score: string (nullable = true)
 |-- ViewCount: string (nullable = true)
 |-- Body: string (nullable = true)
 |-- OwnerUserId: string (nullable = true)
 |-- Title: string (nullable = true)
 |-- Tags: string (nullable = true)
 |-- Quality: string (nullable = true)
 |-- CodeBody: string (nullable = true)
 |-- CleanBody: string (nullable = true)
```

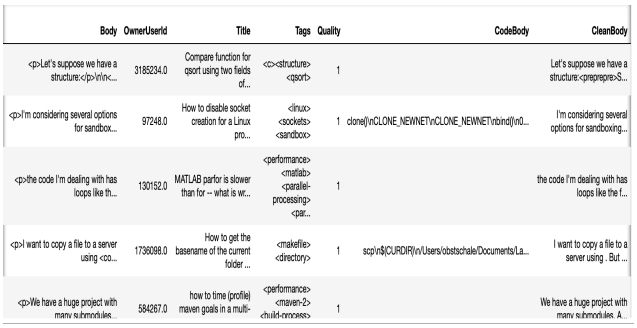
Figure 4. DataSet Schema after Cleaning

- **Dropped posts with invalid/improper data types for postID:** Some posts had postID with alphanumeric characters. postID is numerical, containing 5-9 digits.
- **Code Segregation:** Since the text body was heterogeneous, and we needed to perform separate analysis on code snippets, and text body, we segregated the code snippets from the stack overflow post as a separate feature.
- **Removing HTML tags:** Stackoverflow posts contain HTML formatting, but these tags interfered with the machine learning model. Hence, we removed the tags and obtained both code and textual body as pure characters.

After these steps were performed, we were left with 2.5M posts. This resulted in an approximate reduction of 90% indicating that the dataset obtained from the data dump was messy. Figure 4 and 5 indicate the schema and a sample dataframe after the cleaning steps were performed.

3.3. Feature Extraction

As mentioned, the aim is to analyze the metrics that maximize the performance of our LSTM based model in determining the quality of a stack overflow post.



Body	OwnerUserId	Title	Tags	Quality	CodeBody	CleanBody
Let's suppose we have a structure: \n\n<...	3185234.0	Compare function for qsort using two fields of...	<c><structure><qsort>	1		Let's suppose we have a structure:<prepre>S...
I'm considering several options for sandbox...	97246.0	How to disable socket creation for a Linux pro...	<linux><sockets><sandbox>	1	clone(\nCLONE_NEWNET\nCLONE_NEWNET\rbind(\n0...	I'm considering several options for sandboxing...
the code I'm dealing with has loops like th...	130152.0	MATLAB parfor is slower than for -- what is wr...	<performance><matlab><parallel-processing><par...	1		the code I'm dealing with has loops like the f...
I want to copy a file to a server using <co...	1736098.0	How to get the basename of the current folder ...	<makefile><directory>	1	scp\n$(CURDIR)\n/Users/obstschale/Documents/La...	I want to copy a file to a server using . But ...
We have a huge project with many submodules...	584267.0	how to time (profile) maven goals in a multi-	<performance><maven-2><build-process>	1		We have a huge project with many submodules. A...

Figure 5. Dataframe sample after Cleaning

For this purpose, we analyzed the following featureset which is categorized as follows:

- **StackOverflow Readability Metrics**
  The below mentioned metrics measure the readability of the user using different formulae. Readability is the understanding capability of the user based of a given text. In general, the higher the score, the more the professional expertise required to understand the text [4].
  - **Average Term Entropy:**
    Entropy of terms in a stackoverflow post, essentially calculating the entropy of term distribution with a given set of probabilities.
  - **Automated Readability Index:**
    It is a readability test for calculating the understandability of a text on a US grade level.

$$ARI = 4.71\left(\frac{\text{characters}}{\text{words}}\right) +$$

$$0.5\left(\frac{\text{words}}{\text{sentences}}\right) - 21.43$$

  - **Coleman Liau Index:**
    Similar to ARI, this test is a metric for understanding the readability of the text. Moreover, it also relies on the number of characters per words as opposed to syllables. The better the CLI, the higher the understandability of the post.

$$CLI = 0.0588L - 0.296S - 15.8$$

  - **Flesch Reading Ease score:**
    This metric tells the readability of the text as well. It assigns a score between 1 to 100, with a score of 70-80 being that of a school going grade 8 student.

$$FRE = 206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right)$$

$$-84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

  - **Flesch Kincaid Grade Level:**
    This is a modified version of Flesch Reading Ease score which is easier to use and directly provides the reading grade level.

$$FKG = 0.39\left(\frac{\text{total words}}{\text{total sentences}}\right) +$$

$$11.8\left(\frac{\text{total syllables}}{\text{total words}}\right) - 15$$

  - **Gunning Fox Index:**
    Gunning Fox Index is mainly used because of its simplicity. It assigns a grading to the text between 0 and 20, with a grading of 9 indicating that the text passage can easily be understood by 9^{th} grader and above. Example, it is shown that an average article in New York Times has a GFI of 11-12, whereas academic papers have a grade of 18 and higher.

$$GFI = 0.4\left(\frac{\text{total words}}{\text{total sentences}}\right) +$$

$$100\left(\frac{\text{complex words}}{\text{total words}}\right)$$

    Complex words in the above definition are words that satisfy the following:
    - contains more than three syllables.
    - combined words with a hyphen, eg., "stackoverflow".
    - words beginning with a capital letter.
  - **Metric Entropy:**
    Entropy is the degree of randomness in a system. Similarly, metric entropy indicates the how random/abstract is the stack overflow post. It is calculated as,

$$ME = \left(\frac{\text{average term entropy}}{\text{body length}}\right)$$

  - **Lines of Code Percentage:**
    Lines of code percentage is the percentage of code in the post body. In stackoverflow, code snippets are added between $< code > ... < /code >$ tags. Since, we are restricting our domain to computer science and data science posts, higher LOC percentage would differentiate a specific answer/question from an abstract/verbose one.

$$LOC = \left(\frac{\text{length of code}}{\text{body length}}\right)$$

- **User Community Metrics:**

  These are the metrics that indicate the reputation of the user and the post on stack overflow. Every post on stack overflow is assigned a reputation score. This score depends on a number of factors, but primarily on the number of up votes, down votes, people of who are following the post, and number of viewers [5].

  - **Up Votes:**
    The number of up votes received by the post. Upvotes are indicators of the number of users who agree/follow the post. Higher the number of upvotes generally corresponds to a high quality post.
  - **Down Votes:**
    The number of dislikes received by the post. This is an indicator that the post might be a duplicate, or may contain ambiguous content. The larger the number, the lower the quality of the post.
  - **Favourite Count:**
    This indicates the number of users who have followed the post. Generally, if the question is interesting/curious, users mark it as favourite so that they receive live notifications as to an update on the post. Hence, higher the number of favourite counts to a post, the more popular it is.
  - **View Count**:
    The count indicates the number of views that a post have received. Unlike favourite count, higher number of views does not correlate to a better quality. If the number of views are higher, but there is no activity on the post, then it might indicate a low quality post.

3.4. Exploratory Data Analysis and Visualization

Data Visualization deals with the graphical representation of data and is used to find underlying trends and patterns in data. It is necessary to perform data visualization since it allows for a strong understanding of the data and makes communication more effective.

- **Correlation Heatmap:** A correlation heatmap provides a two dimensional visualization of a correlation matrix between discrete dimensions. The value of correlation lies between -1 and 1, where a value of 1 indicates a high positive correlation, a value of 0 indicates linear correlation and a value of -1 indicates the presence of a negative correlation.

  In Fig. 6, an observation can be drawn that some features are strongly positively correlated to each other, such as - CLI and ATE, GFI and ARI, FKGL and GFI,

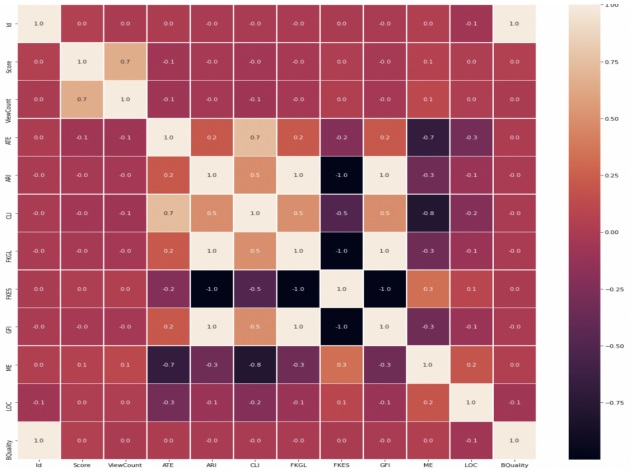


Figure 6. Correlation Heatmap

Viewcount and Score. Apart from a few features that are strongly correlated, it is observable that most of the features do not possess a strong linear correlation.

- **Joint plots:**
  A joint plot is used to visualize the relationship between two features. It consists of three plots within it, which are - variation of the dependent variable(Y) with regard to the independent variable(X), distribution of independent variable, and the distribution of the dependent variable.

  In Fig. 7 there is a strong relationship between metric entropy and average term entropy, average term entropy and Coleman Laiu Index. This analysis van be used to decide upon the features to be included in the prediction model.

- **Distribution Plot:**

  A distribution plot is useful for comparing the range and distribution of groups of data. An observation can be drawn that there is a higher density of views for good quality posts when compared to poor quality posts. Hence, this feature can be relied upon for prediction.

- **Word Clouds:**

  In the word clouds in Fig 11, it can be observed that most of the highly seen words are common, however, there are minor differences. Despite this, since there are no stark visible differences, it is clear that the words alone cannot be used to determine whether any given post is a high quality or a low quality post.

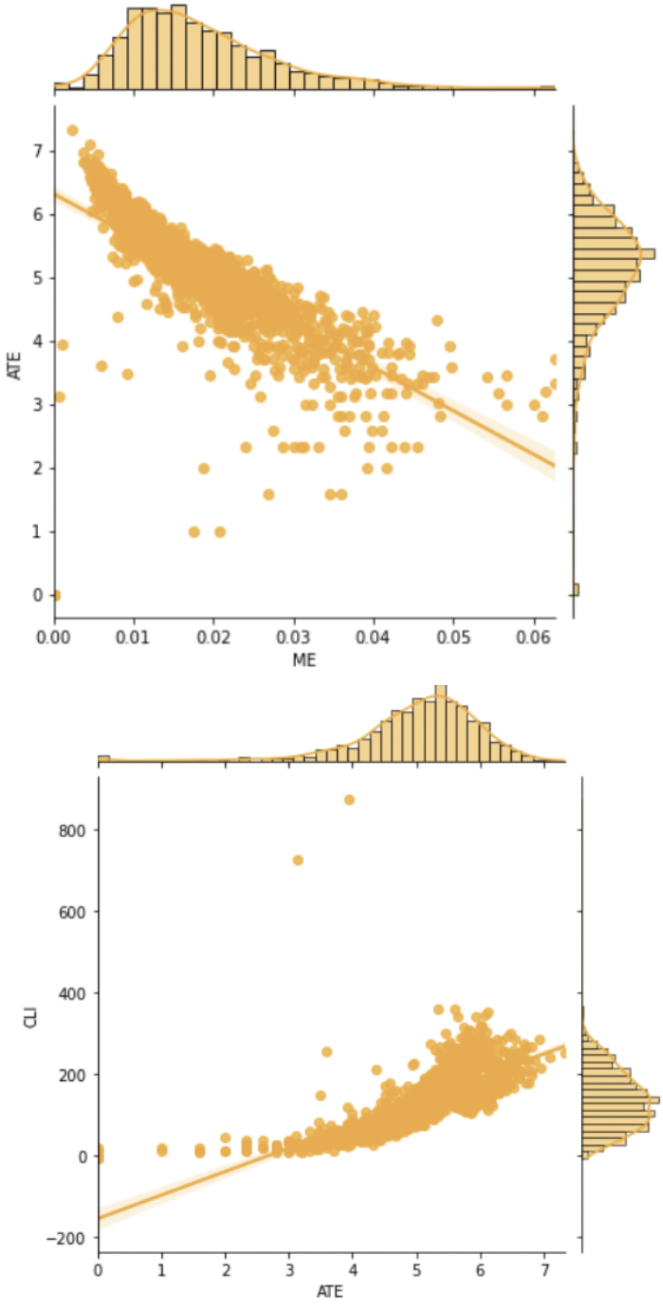


Figure 7. Top right figure shows Joint Plot of MTE vs ATE the bottom right figure shows Joint Plot of ATE vs CLI

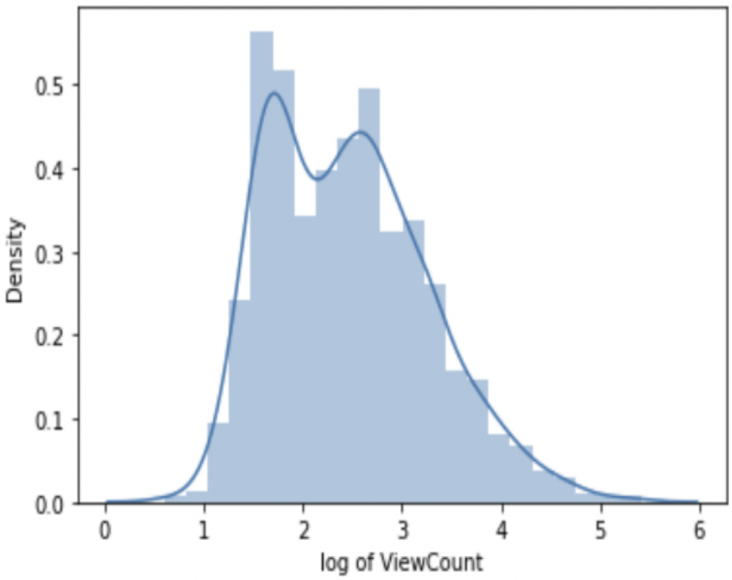


Figure 8. Distribution Plot of Good Quality Posts

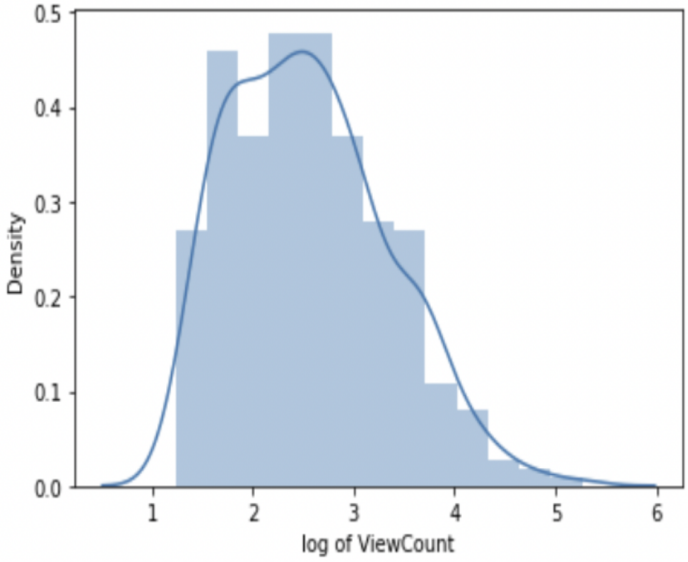


Figure 9. Distribution Plot of Poor Quality Posts

4. Proposed Methodology

4.1. Proposed Architecture

Figure 12 shows the architecture diagram for our project. The following components are utilized for the purposes of analyzing stack-overflow posts:

- **Data Creation:**

The large amount of stackoverflow dump is loaded onto Big Query. From there, we use SQL queries to process the data corresponding to posts that are present in December but not in January. We use joins to merge data from post, answer and user tables to create a dataset containing 24M posts. This is then transferred to cloud storage as a csv file.

- **Data Labelling:**

Figure 10. The first image from top shows word cloud of Good Quality Data and the image below it shows the word cloud of Poor Quality Data.

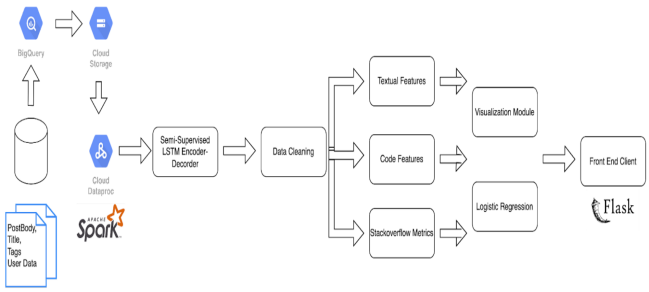


Figure 11. Architecture Diagram

Although we had initially labelled our dataset on the basis of the fact that posts which are not available in December must have been deleted by SO. But there are certain exceptions to this, for example, posts which the users might have deleted. Hence, as a safety check, we used a semi supervised LSTM encoder decoder [6] model for labelling. The model is trained on 60K posts, and was used to label the rest of the data with an accuracy of 84%. The LSTM model uses an SGD optimizer and categorical cross-entropy loss. After training the model on the training data for 25 epochs, accuracy of 84% was achieved. The new data was then labeled using this model.

- **Data Cleaning & Preprocessing:**

  In this module, we used pyspark on a dataproc cluster with 1 master and 2 worker nodes, to perform data cleaning including handling null/missing values, rectifying data types, segregating code and body. Detailed discussion done in section 3.2.

- **Feature Extraction:**

  Extracted and analysed three categories of features: Textual, Code-based and Community based. Utilized pyspark sql's user defined functions along with python libarries for calculating textual features. Detailed discussion under section. 3.3.

- **Visualization Module:**

  Data visualizations guide the feature engineering process, by determining which features are highly correlated and hence can be dropped. Performed extensive analysis on the cleaned data. Visualized word clouds, correlation plots amongst features and distribution of various features. The results and visualizations are discussed in section 3.4.

- **Machine Learning Model:**

  Based on the features obtained, we trained a binomial logistic regression model for predicting the quality of stack overflow posts only on the basis of textual, code and community features. We used a hyperparameter tuned model, and compared the same with other models. The results of the experiments have been highlighted under section 6.

- **UI Client:**

  We hosted our trained LSTM model on a flask server, with a user interface. The user can add a post title, text and the backend calculates the features along with the quality of the posts. This gives visibility of the trained model.

4.2. Software System Overview

We use python flask for our software stack. The main application file contains all the API routes and GET/POST requests which need to be full filled by the front end. Front end and backed are connected through GET/POST request where the header contains the payload of request from the user. The front end is made up of HTML, CSS, JavaScript and Bottstrap along with a few material design elements. The backend contains the trained model loaded with weights, and other python initializations that can be easily deployed to a server.

- **Software Bottlenecks:**

  The model being heavy, takes a while to load completely, in the flask app. This can be overcome in the future by designing an API which loads the model only once.

- **Running the application:**

  The application demo is shown in the youtube video.

- **Software Improvements:**

  The User interface of our software can be made more interpretable and user friendly. More features can be added on the backend side and the model docker image can be deployed using an API.

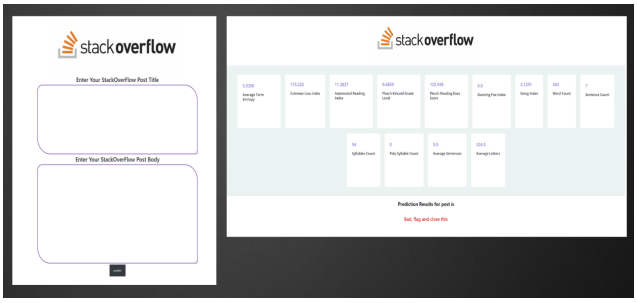


Figure 12. Application UI

4.3. ML Model: Logistic Regression

We trained a binomial logistic regressor model on the features extracted in section 3.3.

Logistic Regression is mainly used for predicting binary labels based on logits (log of odds). We leveraged the pyspark logistic regression library for creating transformers and estimator for fitting our dataset.

Some of the adjustments that we did for our model:

- **Adjusting for class imbalance:**
  - Number of samples with high quality: 22944
  - Number of samples with high quality: 6741

  As clearly visible, the dataset is highly imbalanced. To overcome the same, we adjusted the model to give a high weightage to class 0 (low quality posts) as compared to class 1 (high quality). The weight factor was calculated as follows:

  $$weight = \frac{n-l}{n}$$

  where, n = number of samples,
  l = number of low quality samples

  On our dataset, this gave us a balancing ratio of 0.77 which is good for modelling and machine learning purposes.

- **Dense to Sparse vectors:**

  Dense vectors essentially represent numpy arrays which are used for performing all calculations. After using **VectorAssembler** for combining all the features into a single array, we had a '**features**' column of type *DenseArray*.

  Later, we observed that in most cases our feature vector was empty (due to some features being zero). This meant that our feature vector can be converted to a Sparse representation – sufficiently saving time and space in all subsequent operations. This also optimised our code and other memory requirements which save compute.

- **Grid based search for hyperparameter tuning:**

  We performed automatic hyperparameter tuning, using the grid based search algorithm, iterating through a fixed space of hyperparameters in order to achieve the best hyperparameter combination tuned to suit our problem statement. The algorithm was optimized for f1-score on the validation dataset.

5. Experimentation

For purpose of experimentation, we tried a number of hyper parameters in order to obtain the optimal hyperparameters for the best model and use case. The paper that we survey did not clearly work towards solving our current problem statement, hence we had to do this for scratch. Some of the model parameters we obtained after applying grid search are as follows:

- Train Test split: 90:10
- Train Validation split: 80:20
- Training samples: 25849, Test samples: 2912
- Weight balancing ratio: 0.77
- regParam: 0.01
- elasticNetParam: 0.0
- fitIntercept: True

Final models were trained on these baseline parameters.

6. Evaluation and Results

We use the metrics as given in Table 3 for our experimentation and evaluation.

- **F1 Classification Accuracy**: For training and validating the model.
- **Cohen Kappa Metric:** Dataset imbalanced with more negative samples than positive samples.
- **Matthews Correlation Coefficient:** Model Robustness improved since entire confusion matrix is considered for calculation.

Evaluation Metric	Score
Accuracy	*73%*
Area under ROC	*0.71*
Area under PR	*0.89*
Sensitivity	*76.8*
Specificity	*51.7*
F1-Score	*0.73*
Cohen-Kappa Metric	*0.26*
Mathews Correlation Coefficient	*0.25*

Table 3. Logistic Regression (regParam: 0.01, elasticNetParam: 0.0, fitIntercept: True)

Model	Accuracy	Area under ROC
Logistic Regression	*73%*	*0.72*
Random Forest	*70%*	*0.70*
Decision Tree	*66.4%*	*0.61*
Gradient Boost	*66.5%*	*0.71*
Linear SVM	*63.1%*	*0.71*
MLP	*69%*	*0.59*

Table 4. Comparison of different models

- The results as shown in table 3 and table 4 show great progress towards solving the problem, we can see that our generated features were linearly separable by means of regression.
- We were also able to clearly define a supervised problem from an unsupervised one using very little data and performing extensive feature extraction.
- The area under ROC curve also shows progress, wherein we see Logistic regression is the best model to solve this problem.

7. Business Value

The project has buisness value in three different terminologies. The first one being the monetary Value. In terms of monetary value, Higher number of good quality posts increases the glance per view, thereby generating more advertisement revenue and investments. The second one is the space Value, where flagging low quality posts on StackOverflow will help reduce the search space for good quality posts, thereby generating more page views which will in turn help the website. The last buisness value is the time value, where once a post is flagged by the machine learning model as low quality posts, this will help reduce the moderation queue length which is a cumbersome task for the website and post moderators.

8. Conclusion

Based on the results and analysis undertaken, we believe that simply relying on textual evidence is not enough for classification. Since, we did not have previous flagged post data, that would have been a great feature to improve model performance. All in all, our analysis sheds light on the approaches that can be undertaken for large scale classification of flagged stack overflow posts. We have designed novel approaches to reduce the stackoverflow moderation queue size which will be useful in the long run. .

9. Acknowledgments

We would like to express our sincere gratitude toward Prof Dr. Ching-Yung Lin and the entire team of T.A.s whom have been supportive right from the start and helping us to accomplish this project thereby improving our knowledge and learning experience in the Big Data Domain.

10. References

References

[1] Annamoradnejad I, Fazli M and Habibi J 2020 Predicting Subjective Features from Questions on QA Websites using BERT Proc. of I6th

Int. Conf. on Web Research (Iran) pp. 240-244. https://doi.org/1010.1109/ICWR49608.2020.9122318.

[2] Baltes, S., Treude, C. and Diehl, S., 2019, May. Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR) (pp. 191-194). IEEE.

[3] Baltes, S., Dumani, L., Treude, C. and Diehl, S., 2018, May. Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts. In Proceedings of the 15th international conference on mining software repositories (pp. 319-330).

[4] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza and D. Fullerton, "Improving Low Quality Stack Overflow Post Detection," 2014 IEEE International Conference on Software Maintenance and Evolution, 2014, pp. 541-544, doi: 10.1109ICSME.2014.90

[5] Correa, D. and Sureka, A., 2014, April. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In Proceedings of the 23rd international conference on World wide web pp. 631-642.

[6] Sundermeyer, M., Schlüter, R. and Ney, H., 2012. LSTM neural networks for language modeling. In Thirteenth annual conference of the international speech communication association.

Contribution	slj2142	mmj2169	mk4427
Fraction	1/3	1/3	1/3

Table 5. Contributions