Prediction and Correlation-based Response Time Analysis of StackOverflow data

CS 235: Research Project Report

Jannat Ara Meem
jmeem001@ucr.edu

Sakib Fuad
sfuad001@ucr.edu

Tomal Majumder
tmaju002@ucr.edu

Rituparna Gucchait
rguch001@ucr.edu

CCS Concepts: • **Computing methodologies** → **Supervised learning by classification**; **Neural networks**; **Cluster analysis**; *Cross-validation.*

Keywords: supervised learning, unsupervised learning, regression analysis, neural networks

1 Introduction

StackOverflow is a popular community platform used by developers and per day over 8000 questions are asked [2] on average about programming problems. The volume of the interactions among the developers through questions and replies enables us to dig down some interesting facts. In such a large interactive platform, response time of questions to receive the first answer plays an important role and would largely determine the popularity of the platform. People who post questions would want to know the time by which they can expect a response to their question. Posted questions are tagged by the author with tags that represent the post best[10]. The volume of the interactions among the developers through questions and replies enables us to dig down some interesting facts. Response time analysis is one of the most interesting problems that has never been solved before. Previous work focused on classifying the posts based on whether a post can be answered within a certain timespan or not [7] [8] [4], but no work has been done to predict the actual response time of a posted question. Response time prediction can help engage more users in the platform and facilitate them to improve the choice of tags and the phrasing of their question. This can also help administrators to identify the expertise of a developer [6] who replies to a certain question associated with a certain tag.

Vasudev *et al.* [5] analyzed a long list of factors in the StackOverflow data and identified some factors as significant features that have a direct correlation with the response time of a post. They concluded that tag-based features show notable dominance over non-tag based features while predicting whether a question is answered within 16 minutes(median response time of their data) or not. Based on their conjecture, we propose a novel problem of predicting the actual response time of a question. We use the superior factors identified by them as features to build our regression model. We also propose a novel feature and integrate this with the previous feature set and show evident improvement in the performance of our model.

In our work, given the StackOverflow dataset, we propose to use supervised and unsupervised learning-based as well as correlation-based analysis on the dataset in order to find which technologies/skills are more relevant to each other such that these can be modelled as a topic, which factors contribute to the response time of a question, what's the average response time of a question, whether there is any correlation between the response time of a question and its topic etc .

Following are the main contributions of our work:

- We propose regression-based models to predict the response time of a question posted in StackOverflow. To our best knowledge, this is the first work on response time prediction.
- We propose a novel feature related with question response time and use this along with the significant features proposed by Vasudev *et al.* in [5] to build our prediction models.
- We conduct extensive experiments on our proposed models and present the validity of our proposed feature set.

Our project overview is shown in Figure 1. The rest of the report is structured as follows: thorough description of the related work in Section 2, detailed description of the dataset in Section 3, implementation details of our methodology in Section 4, experimental results and visualizations in Section 5 and finally discussions and concluding remarks in Section 6.

2 Related Work

Response time of questions is an important aspect as it relates to user satisfaction and engagement. Task of predicting response time can be divided into two categories: finding features correlated to response time and prediction of actual

Conference'17, July 2017, Washington, DC, USA
none. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/none

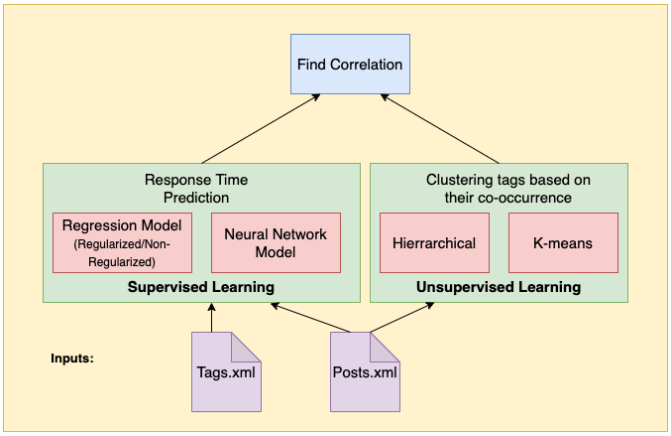


Figure 1. Project Overview

response time. Arunapuram et al. [3] analyzed the distribution and correlation of response times in StackOverflow based on features, such as title length and word usage, and attempted to use them to predict responses. They showed that time of the day, use of punctuation and pronouns in a question did not correlate with response time using PCA analysis and various visualization techniques. Relation of the response time with both tag-based and non-tag-based features in the StackOverflow data was assessed by Bhat et al. [5]. This analysis revealed that tag-related factors like "popularity" and "subscribers" generate significantly stronger evidence than non-tag-based features. They categorized response time prediction problem into two classification tasks and employed the discovered evidentiary features in their learning model. Prediction Task 1 aims to classify questions based on their median response time (16 minutes). Task 2 attempts to distinguish between questions answered within an hour and those answered over a day. Di Wu et al. [11] labeled the data into 4 groups based on the first response time: 0 to 1 hour, 1 to 4 hours, 4 to 12 hours, and 12 hours or more. Later they proposed new features: Tag Count, Answer Count, View Count etc to classify each question into one of the four mentioned groups.

However, all these works focused on whether or not a question will receive an answer within a certain time frame or not which is a classification problem. In our proposed approach, we are trying to predict actual response time of a question which is a regression problem. The overview of our approach and experimental evaluation observed so far have been presented in section 4.2.

3 Dataset Description

We have used a subset of the StackOverflow data from the stack exchange data dump[2]. The dataset can be found here. The dataset is structured and consists of several xml files. Among those, we have used the following two files for our tasks:

1. Posts.xml - This file has 17 attributes. It contains user's posts and all information related to the post. The size of this file is 126.8MB and it has 58969 posts.
2. Tags.xml - This has 5 attributes. It contains information of each tag found in the StackOverflow posts. It has total 1128 tag rows and the file size is 81KB.

The dataset has StackOverflow data from January 2014 to September 2021. One of the most important reasons to choose this dataset is that it comes from the actual source of the data, the parent company stack exchange. Hence, the authenticity of data is guaranteed. The total size of our dataset is approximately 100 GB which is too large to be processed locally. Hence, we need big data processing to process this huge dataset.

3.1 Data Processing

Our data is already cleaned and structured. However, for generating different features to address our tasks, we will require some filtering on the xml files.

4 Proposed Approach

Our approach consists of two stages: first we cluster the co-occurred tags using two unsupervised learning methods: Hierarchical and K-means clustering and build our novel feature associated with each question; then we use our feature set to predict the response time of the questions.

4.1 Unsupervised Learning: Clustering Tags

Clustering tags based on their co-occurrence in the posts involves a preprocessing stage where feature vector for each tag will be created using encoding technique. Then the encoded vectors will be further processed in the clustering algorithms to form the clusters. In our project we have implemented K-means Clustering and Agglomerative Hierarchical Clustering. The preprocessing steps and implementation of clustering methods have been described as follows:

4.1.1 Preprocessing. The preprocessing steps for our implemented clustering task has been described as follows:

Measuring Co-occurrence frequency of tag-pairs: To find out the co-occurrence frequency of tag-pairs, we need to know which pairs of tags appear together in the posts and how many times they do so. To do so, we have designed a map-reduce function which creates a tag-pair whenever two tags appear on the same post and then it provides the total count of the posts where they appeared together i.e. its frequency.

Finding Distance Matrix and Dimension Reduction: After getting co-occurrence frequency of tag-pairs from the previous step, we develop a matrix from the co-occurrence counts of tag-pairs. The higher the co-occurrence count, the higher the probability that the two tags will be on same

cluster. First we populate an $N \times$N matrix where each entry corresponds to co-occurrence count of two tags (u, v) and N is the total number of unique tags found our question dataset. After that, we fill all the diagonal entries with a value larger than the maximum value of occurrence counts so that the diagonal entries capture the maximum similarity measure since they represent a tag-pair with itself and it will have the highest strength of togetherness. Therefore, we have found encoded vectors of N dimensions. In order to visualize the clusters in two-dimensional space and reduce the run time of the implemented clustering algorithms, we have reduced the dimension of the vector to 2 dimensions using **t-SNE**. We have also observed the reduced distance matrix almost shows the similar result in detecting clusters with the original one.

4.1.2 K-means Clustering of Co-occurred Tags. We take the $N \times 2$ embedding matrix from 4.1.1 where N is the number of tags and 2 is the dimension of the embeddings. Then, we find the clusters of tags using K-means clustering algorithm. The K-means algorithm [1] is as follows:

Algorithm 1 K-means Clustering

1: Select number of clusters K
2: Get initial centroids by shuffling and randomly selecting K datapoints without replacement
3: Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
4: Compute the sum of the squared distance between data points and all centroids.
5: Assign each data point to the closest cluster (centroid).
6: Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

We have clustered our data into 95 clusters using K-means clustering. Here 95 is the optimal number of clusters for our data. The finding of optimal cluster number is discussed in 5.4.3.

4.1.3 Agglomerative Hierarchial Clustering. From the results of the embeddings found from 4.1.1, we formulate a pairwise distance matrix of $N \times N$ dimensions using euclidean distance. Then the distance matrix has been sent as an parameter of our hierarchial clustering model. The steps of our implemented Hierarchial Clustering has been shown in Algorithm 2.

In the step-4 of Algorithm 2, we have implemented three ways to compute the distance between two clusters: Single Linkage, Complete Linkage and Average Linkage. In single linkage, the distance of two clusters has been considered as the minimum distance between two datapoints in two clusters. For complete linkage, the maximum distance between two data points of two clusters and for average linkage, the average of distances between all the datapoints in two clusters have been considered as distance between two clusters. The cluster information of each agglomerative step is saved in python dictionary named *clusters* and returned by the algorithm. We have also implemented a method named *cut_tree* where we can set number of clusters as parameter. The method will first calculate the iteration number of the agglomerative steps to get the given number of clusters and return the cluster labels of that iteration from *clusters* dictionary. In the experimental section, we have used **Silhouette Score** to get the optimal number of clusters for our implementation.

Algorithm 2 Agglomerative Hierarchial Clustering

1: Initialize each item to its own cluster.Suppose we have N items, initially there will be N clusters each having exactly one item.
2: Find the minimum distance value entry from the distance matrix, get the row and column index of that entry to get the closest pair of clusters.
3: Merge the cluster pair found from step-2 into one.
4: Compute distances between the new cluster and each of the old clusters and update the distance matrix.
5: Repeat steps 2, 3 and 4 until all items are clustered into a single cluster of size N.

4.2 Supervised Learning: Response Time Prediction

Vasudev at el. [5] analyzed the importance of several tag based and non-tag based factors and identified evidential factors associated with response time. Their experimental results demonstrated the superiority of tag-based features over obvious non-tag based features. So, based on the importance metric they presented we have considered eight significant factors: six tag-based and two non-tag based factors to train our regression model. We have also introduced a new feature named **'topic_relevance'** that we generate from the output clusters provided by Section 4.1. The list of all features and their detailed preparation steps have been presented as follows:

Tag-based Features:

1. **Topic Relevance:** The clusters found in Section 4.1 represent which skills or technologies are closely related to each other. As such, each cluster can be interpreted as a particular topic where each tag in the cluster can be viewed as a reference to that topic. So, a question having more tags from the same cluster is supposed to have more number of references to the same topic. And the more number of references a question has to a particular topic, the more the question is relevant to that topic; which in other words, represents how well the question is tagged by the user. So, we formulated the following equation for a question in

order to capture the essence of the relevance to a topic.

$$topic\_relevance = \sum_{n} avgPopularity(n) * (size(tags_n)/size(n)) \quad (1)$$

where n represents a cluster or a topic and size($tags_n$) represents the number of tags in the question belonging to topic n, size(n) represents the total number of tags present in topic n and avgPopularity(n) represents the average popularity of all the tags in that topic n. The 'topic_relevance' feature is formulated as a weighted sum of the recalls of the tags associated with the question referencing that topic, where size($tags_n$)/size(n) denotes how closely the tags in the question are relevant to the topic represented by cluster n and avgPopularity(n) denotes the weight or importance of that topic in the dataset. The more the number of questions relevant to a topic, the more popular it is i.e. the stronger the impact it has on the dataset. This is why we have picked the average popularity of a cluster as the weight or the measure of how significant a cluster should be.

2. **Number of Popular Tags:** We measure the popularity of a tag by counting the total number of times that tag appeared on the posts in the dataset. The number of times a tag appeared in the posts is proportional to its popularity. The tags that appeared most, are the most popular ones. Here, we set a threshold on the frequency of tags to group them into popular and non-popular ones, and count the number of popular tags each question contains.
3. **Average Popularity of Tags:** We have considered the popularity of a tag t as the number of questions that contain t as one its tags.The Tags.xml file contains this information for each tag in the Count field. For each question from Posts.xml file (whose PostTypeId=1), we have taken the value of the Tags field of the post which contains all the tags of that question. For each tag we fetched its popularity value and finally computed the average popularity of all tags.
4. **num_subs_ans**: Here, we figure out the number of active subscribers to a post. We define a subscriber to be active with respect to a tag if that subscriber has posted a number of replies greater than a predefined threshold (for the experiments, we used 10 as the threshold value) relating to that tag. We determine the number of subscribers for a question by computing the average number of subscribers for all the tag associated with a question.
5. **percent_subs_ans**: We also take which percentage of the subscribers of a particular tag are active into consideration. We take the ratio of the average number of active to all subscribers as a feature for our model.
6. **num_subs_t**: Here, we find out the number of responsive subscribers to a post. We define a subscriber to be responsive with respect to a tag if that subscriber has an average response time less than a predefined threshold (for the experiments, we used 1hr as the threshold value) relating to that tag. To calculate the average response time of a subscriber for a given tag, we have taken all responses of the subscriber for that tag and considered the response time of those. But we observe some extreme values in the response times which badly affect the average value. We have used Interquartile Rule to find those outliers and got better average estimation.
7. **percent_subs_t**: We take which percentage of the subscribers of a particular tag are responsive into consideration. We take the ratio of the average number of responsive to all subscribers as a feature for our model.

Non-tag based Features:

1. **Question Title Length:** For each question on the "Posts.xml", we have computed the length of title of the question.
2. **Question Body Length:** For each question on the "Posts.xml", we have computed the length of question body.

After preparing the feature for our model, we have defined the true label for the datset. We have calculated the response time of each question by taking the difference of the time between the question being posted and the earliest response to that question. We have implemented our regression models that take the feature set as input for training and predict minimum response time (label) for the questions. We have experimented with Linear regression and Multi layer perceptron-based regression. The details of the regression analysis is provided in Section 5.

5 Experiments & Performance Evaluation

5.1 Visualizing Co-occurrence frequency of tag-pairs

We have shown the top 10 results found from Section 4.1.1 in Table 1. We observe that related tags appears together such as python and python-3, ruby and ruby-on-rails, html and css etc. and the co-occurrence frequency of tag-pairs is proportional to how closely they are related to each other.

5.2 Evaluation of K-means Clustering Algorithm

In our experiment, at first we have calculated the silhouette score (fig: 2) for different number of clusters ranging from 40 to 150. From the silhouette score we have found that 95 is the optimal number of clusters. It generates the highest silhouette score that tends to 1. After this finding, we have clustered our data into 95 clusters (fig: 3) using K-means clustering.

Tag-Pair	Co-occurrence
python, python3	1001
ruby, ruby-on-rails	453
swift, xcode	451
javascript, jquery	416
javascript, html	399
swift, ios	366
html, css	319
ios, xcode	307
python, pandas	280
android, java	200

Table 1. Co-occurrence counts for top 10 tag pairs.

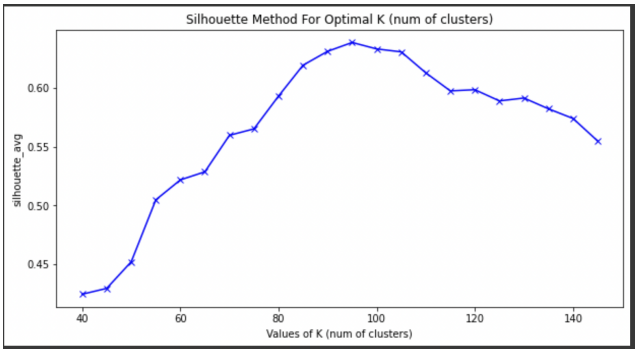


Figure 2. Silhouette Method For Optimal K (number of clusters)

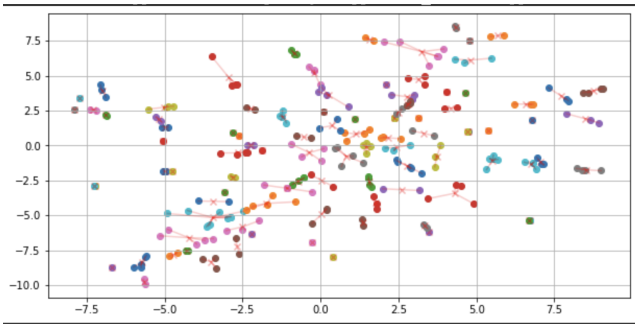


Figure 3. Visual representation of clusters using scatter plots

5.3 Performance Evaluation of K-means Clustering

We have analyzed our K-means clustering results with scikit-learn library's K-means clustering. We have found that our result is very similar to the results of scikit learn K-means. Here is an example: One of our clusters include java8, java-ee, tomcat, swing, javafx, jpa, junit. On the other hand, scikit-learn K-means includes java8, java-ee, tomcat, swing, javafx, jpa, junit, thymeleaf. Another example is both of the K-means method find the same cluster amazon-ec2, amazon-s3, aws-lambda, aws-cli. But there are exceptions also where same tag is clustered into different clusters for the custom K-means and scikit-learn K-means methods.

5.4 Evaluation of Agglomerative Clustering Algorithm

5.4.1 Finding Optimal Parameters. In our algorithm, we have used three options for calculating distances between two clusters: Average, Single, and Complete. We experimented with three options separately and calculated the Silhouette score varying the different number of clusters using *cut_tree* method for each case. The result curves are shown at figure 4.

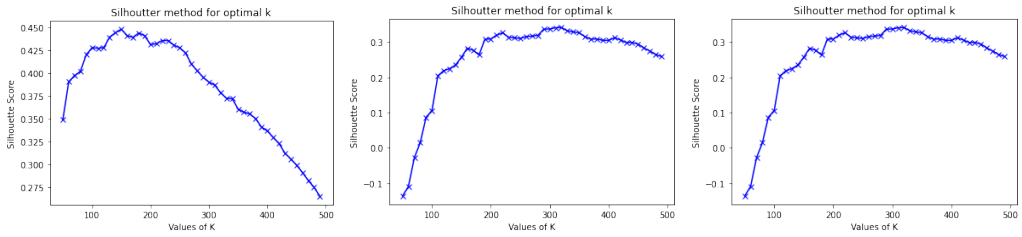
We can see from the curves from figure 4, the maximum Silhouette scores for Average linkage, Single linkage and Complete linkage are 0.454, 0.3265 and 0.35 respectively. So we can see average linkage gives better Silhouette score than other two. That is why we have used Average linkage distance calculation option to generate final clusters. Additionally, we can see the number of clusters that gives maximum score for average linkage is 150. That is why, we set the final number of clusters to 150.

5.4.2 Visualizing Agglomerative Clusters. Dendrogram is a popular visualizing tool for showing hierarchy of clusters. But in our dataset we have over 6000 tags which is difficult to read the labels from the dendogram. That is why in figure 5 and 6 we have generated clusters by taking top 25 and 50 pairs of tags respectively based on their co-occurrence counts. We extracted 10 clusters using the top 25 pairs of tags which contained 30 unique tags. Some of the clusters according to figure 5 are – **C1**: python, python3, **C2**: ruby-on-rails, ruby, **C4**: javascript, jquery, **C5**: html, css, **C6**: pandas, django, numpy, matplotlib etc. From the tags contained in the same clusters, we can see that these align with our intuition and the topics that are actually highly correlated have ended up together.

5.4.3 Comparing Cluster Quality of K-means and Hierarchial. From section , the Silhouette score for the optimal setting of our K-means algorithm is 0.36 and from section , we have found Silhouette score 0.454 for the optimal setting of our agglomerative hierarchial algorithm. The high Silhouette value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. That infers that the more the Silhouette score, the better the quality of clusters. So from that we can conclude that our hierarchial clustering method generates clusters of better quality than K-means in the optimal setting.

5.5 Evaluation of Response Time Analysis

We have implemented two regression models to predict the response time of a question given the feature sets described in Section 4.2 as input to the models. We have experimented with Linear regression model that we implemented from scratch and Multi layer perceptron-based regression model



(a) Average Linkage **(b)** Single Linkage **(c)** Complete Linkage

Figure 4. Silhouette Score vs Number of Clusters (Agglomerative Hierarchial Clustering)

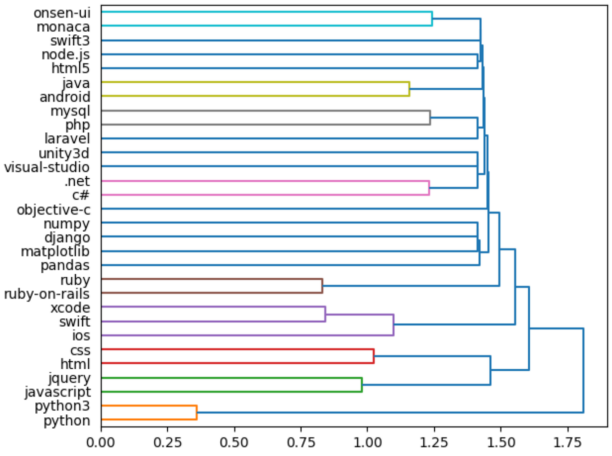


Figure 5. Dendrogram for top 25 pairs of tags based on co-occurrence

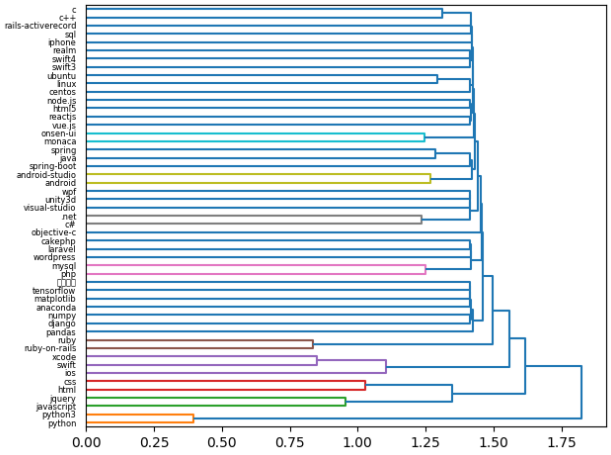


Figure 6. Dendrogram for top 50 pairs of tags based on co-occurrence

from the neural network models available in the scikit-learn library[9]. Here, we have analyzed the response times of questions that have a response time no less than the median of the response times in the dataset which is 3hrs. So, the training set and the test set both included only those questions that have response time $\leq$ 3hrs. The reason behind this selection of train-test set is the non-regular distribution of the response time[5]. We have assumed the time window to be small but effective enough such that the selected distribution contains enough data for the analysis and the number of noisy data points can be brought to as small as possible. Hence, we picked the median value to be the threshold which contains around 50% of the total data.

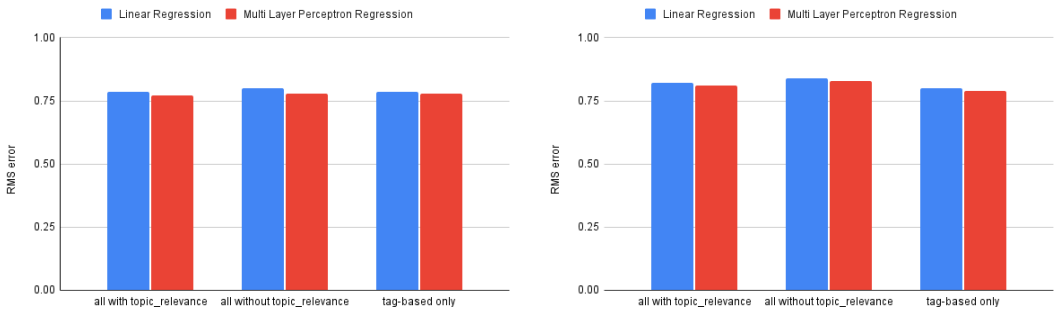
We have used two normalization methods: MinMax Scaling and Z-score method on our feature set. We have shown results for both the normalized data in Figure 7. The results shown here are for models with a train-test split of 80%-20%.

We have experimented with various hyperparatemers but here, we have only included the results for the best hyperparameters. The details of the hyperparameter tuning and cross-validation is described in the implementation correctness report.

5.5.1 Linear Regression Analysis. We have trained our linear regression model with 3 different sets of features. First we train the model using the best features found by Vasudev *et al.* in [5], then we train after integrating our proposed feature 'topic_relevance' with their best features, and finally we train the model with only the tag-based features. The RMS error results of our model are shown as blue bars in Figure 7. This result is found after training the model for 400 iterations and with a learning rate of 1.2.

In the figure, we refer to our feature set as "with topic_ relevance" and feature set proposed by Vasudev [5] as "without topic_relevance". It is observed that the RMS error slightly improves for linear regression when we use our proposed feature 'topic_relevance' along with the features proposed by Vasudev . It is also observed that when we train the model with only the tag-based features the performance is quite similar to the result when trained with all features including 'topic_relevance'. So, the superiority of the tag-based features can clearly be seen here.

5.5.2 Multi-layer Perceptron Analysis. We have used the Multi-layer perceptron regressor as our training model



(a) MinMax Normalization applied on the feature set **(b)** Z-score Normalization applied on the feature set

Figure 7. RMS error found from the Linear Regression model (iteration=400, learning rate=1.2) and The Multi-layer Perceptron model(# hidden layers=250, activation function=ReLu, early stopping = True, max iter=300, 15% data kept for cross validation) for the feature set including 'topic_relevance', excluding 'topic_relevance' and only taking the tag-based features

from sklearn.neural_network.MLPRegressor with 3 different sets of features. We have found the best result when we take the following values as hyperparameters.: no. of hidden layers=250, activation function=ReLu, early stopping = True, max iter=300 and 15% data kept for cross validation. Same as linear regression, we train the model using 3 sets of features: the best features found by Vasudev *et al.* in [5], then integrating our proposed feature 'topic_relevance' with their best features, and finally with only the tag-based features. The RMS error results of our model are shown as red bars in Figure 7.

It is observed that the RMS error shows notable improvement when we use our proposed feature 'topic_relevance' along with the features proposed by Vasudev . Moreover, it shows better results with respect to the results of the Linear regression model as it can capture the non-linear behaviours of the data and can reduce noises better than Linear regression. We can observe the superiority of the tag-based features here as well.

It should also be noted that the MinMax normalized data shows better results than the z-score standardized data.

5.6 Correlation-based Analysis

We have analyzed various factors in the data and show which factors(if any) have direct correlation with the response time. The results are shown in Figure 8. The bigger length of body posts tends to have lesser response time due to the fact the the question is well described. The average popularity of each tag also imply negative correlation with response time. This is due to the fact that the existence of more popular tags in a post alleviates quicker response, as a lot of questions related to these tags are answered. We can see topic relevance also negatively correlated with response time. This is expected as the more the tags of a question are relevant to a topic, the better the question has been tagged. That helps the community to quickly interpret the question and that reduces the response time.

5.7 Result Summary

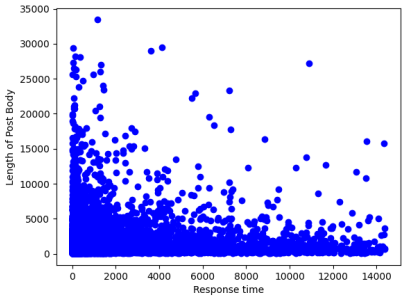
Our tag co-occurrence based clustering approaches have shown promising results as presented in Section 5.1.2. For response time prediction model, we have observed that our Multi-layer perceptron model trained on the MinMax normalized feature set including "topic_relevance" performs better than the feature set proposed by Vasudev *et al.*[5]. Though the improvement is slight, it is consistent for all the methods; hence we can conclude that our proposed approach outperforms the approach proposed by Vasudev *et al.*.

6 Conclusion

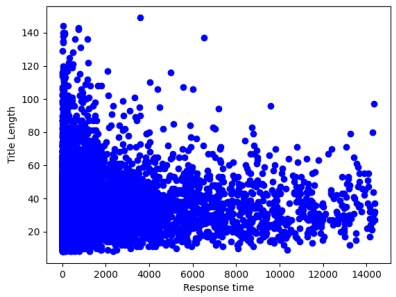
Question response time affects the engagement of users and their interactions. In our work, we have proposed a novel approach of using regression models to predict the response time of a question both using the features proposed by Vasudev *et al.*[5] and using our proposed feature set. To verify our claims, we have thoroughly experimented with our models and validated our results. We have showed that our proposed feature is directly correlated with the response time. We have also included direct comparisons between the results using Vasudev's features and our features. To our best of knowledge, our approach is the first in this domain. However, there are rooms for improvement. In our experiments, we have taken a short snippet of the StackOverflow dataset and we believe that the results can improve if we use a larger dataset. We plan to do extensive experiments on the entire large dataset in future.

References

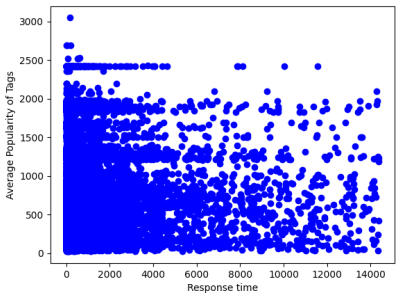
[1] . 2018. K-means algorithm. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.



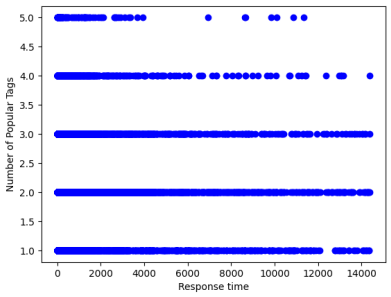
(a) Post body length vs. Response time



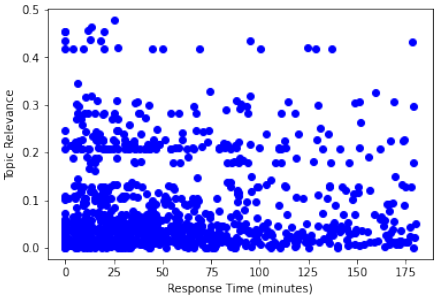
(b) Post title length vs. Response time



(c) Average popularity of tags vs. Response time



(d) Number of popular tags vs. Response time



(e) Topic relevance vs. Response time

Figure 8. Scatter plots for different features vs. Response time

[2] 2021. StackExhange Data. https://archive.org/details/stackexchange.

[3] Preeti Arunapuram, Jacob W Bartel, and Prasun Dewan. 2014. Distribution, correlation and prediction of response times in stack overflow. In *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 378–387.

[4] Daniel Avrahami and Scott Hudson. 2006. Responsiveness in instant messaging: Predictive models supporting inter-personal communication. *Conference on Human Factors in Computing Systems - Proceedings* 2, 731–740. https://doi.org/10.1145/1124772.1124881

[5] Vasudev Bhat, Adheesh Gokhale, Ravi Jadhav, Jagat Pudipeddi, and Leman Akoglu. 2014. Min (e) d your tags: Analysis of question response time in stackoverflow. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. IEEE, 328–335.

[6] Ahmad Diyanati, Behrooz Shahi Sheykhahmadloo, Seyed Mostafa Fakhrahmad, Mohammad Hadi Sadredini, and Mohammad Hassan Diyanati. 2020. A proposed approach to determining expertise level of StackOverflow programmers based on mining of user comments. *Journal of Computer Languages* 61 (2020), 101000.

[7] Felipe Hoffa. 2018. When will Stack Overflow reply: How to predict with BigQuery. https://towardsdatascience.com/when-will-stack-overflow-reply-how-to-predict-with-bigquery-553c24b546a3.

[8] Amit Rechavi and Sheizaf Rafaeli. 2011. Not All Is Gold That Glitters: Response Time amp; Satisfaction Rates in Yahoo! Answers. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 904–909. https://doi.org/10.1109/PASSAT/SocialCom.2011.67

[9] scikit-learn 1.1.1. 2021. sklearn.neural_network.MLPRegressor. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html.

[10] Clayton Stanley and Michael D Byrne. 2013. Predicting Tags for StackOverflow Posts. In *Proceedings of the the 12th International Conference on Cognitive Modelling ICCM 2013*. http://chil.rice.edu/stanley/pdfs/StanleyByrne2013StackOverflow.pdf

[11] Di Wu, Simon Johnson, Chris Foster, Erwin Li, Haytham Elmiligi, and Musfiq Rahman. 2019. Improving Response Time Prediction for Stack Overflow Questions. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 0786–0791. https://doi.org/10.1109/IEMCON.2019.8936252