Prediction and Correlation-based Analysis of StackOverflow data using Spark & MLlib

JANNAT ARA MEEM, MARJUKA FERDOUSI LAZIN, SAKIB FUAD, SAKSHAR CHAKRAVARTY, and TOMAL MAJUMDER

1 INTRODUCTION

Knowing what technological skills are trending can be useful for pursuing a career in the programming industry. StackOverflow is a popular community platform used by developers and per day over 8000 questions[1] are asked on average about programming problems. Questions are tagged by the author with tags that represent the post best[12]. The volume of the interactions among the developers through questions and replies enables us to dig down some interesting facts. For example: which technologies are trending, how these trends are changing over time, average response time[10], whether the expertise of a developer can be predicted[9], etc. The source code for our project can be found in the following link: https://github.com/sfuad001/BigDataProject-StackOverFlow

1.1 Applications

People interested in learning a new skill can get a better idea about trending technologies which in turn will help them to utilize their time and resources efficiently. Moreover, understanding the expertise level of a developer will allow them to consider improvement in certain areas. Furthermore, Clustering similar tags can help with understanding additional learning domains related to users interest. Being able to visualize the different trends and aspects of the posts and comments related to different technical platforms can be quite intriguing. On another note, this work has potentials of leading us towards the formation of social networks of developers who share common interests.

1.2 The Big Data Problem

The significant challenge in accomplishing our objective is the vastness of our dataset[2]. We will be using the anonymized dump of all user-contributed contents on the stack overflow network from 2014 to 2021. We will be performing query-based statistical analysis on the massively generated posts, comments, badges, tags, users, votes, post history and post links over the years and will also try to generate some models for addressing the predictive tasks. Processing this huge volume of data in a non-scalable platform is highly inefficient. Hence, we need a big data platform to process, store, retrieve, and visualize our large structured dataset.

The whole report is structured in the following manner: in section 2, we define the problem statement and the objectives of this project. After that, in section 3, we try to provide a high-level view of the current literature related to each of the tasks. Next to it, in section 4, we describe the dataset used in this work and subsequent data processing and task-wise implementations using SparkSQL and MLlib. Then in section 5, we report the results of our query analysis and the performances of the prediction models. Finally, in section 6, we summarize our findings and provide some hints into probable future works.

Authors' address: Jannat Ara Meem, jmeem001@ucr.edu; Marjuka Ferdousi Lazin, mlazi003@ucr.edu; Sakib Fuad, sfuad001@ucr.edu; Sakshar Chakravarty, schak026@ucr.edu; Tomal Majumder, tmaju002@ucr.edu.

2 PROBLEM STATEMENT & OBJECTIVES

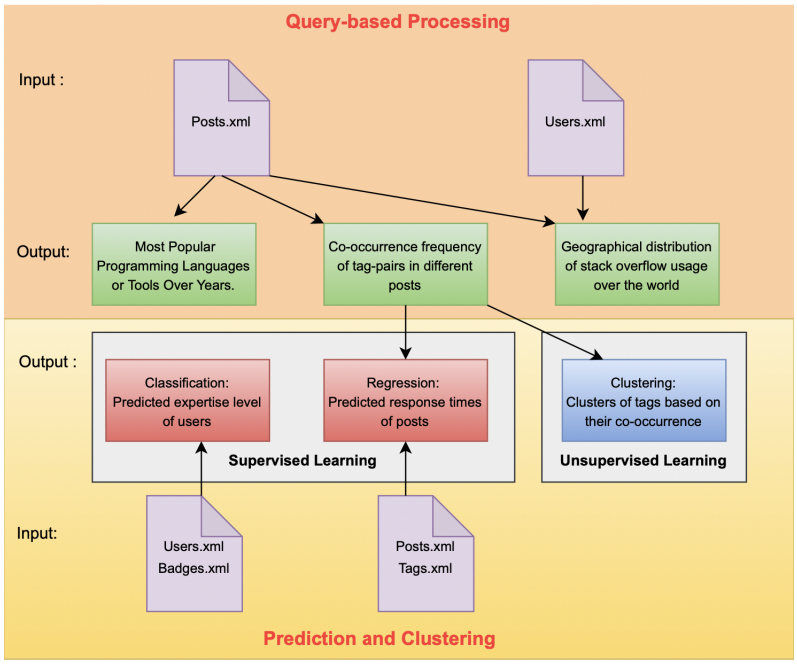


Fig. 1. Project Overview

Our project overview is shown in the above figure 1. The objectives of our project are as follows:

(1) Query-based Processing
   (a) Find out the most popular programming languages or tools over years by analyzing StackOverflow posts referencing that tool or framework.
   (b) Using users' location to find out the distribution of StackOverflow usage over the map.
   (c) Find out the Co-occurrence frequency of tag-pairs in different posts analyzing each post's tags.
(2) Predictive & Clustering based
   (a) Supervised Learning :
      (i) Regression model : Predict response times of posts by extracting certain features from posts and tags, and by analyzing them.
      (ii) Classification model : Predict user expertise level by analyzing user profile and badges assigned by stack overflow.
   (b) Unsupervised Learning :
      (i) Clustering: Cluster keywords based on the frequency of their co-occurrences.

3 LITERATURE SURVEY

Nowadays Community Question Answering (CQA) platforms have become the go-to medium for resolving queries in different programming and technical forums. StackOverflow is the most used CQA among programmers and developers

to get solutions to different problems and issues on a need basis. StackOverflow is preferred by most of them rather than going through the traditional way of acquiring knowledge. This mode of sharing knowledge among peers through Q&A enables a faster and effective learning mechanism. In this work, we have tried to play with the publicly available data [2] of StackOverflow usage to perform a multi-modal analysis. The different modes of our analysis have their own vast literature. In this section, we have tried to sum up the relevant works that have been done in addressing the different objectives of this project.

3.1 The usage trends of Programming Languages and Frameworks: Present and Future

StackOverflow moderates hundreds of thousands of posts each month from developers with a variety of backgrounds. These posts act as a record of the history of the needs and thoughts of developers—a knowledge repository of programmers' needs. Barua et al. proposed a methodology to discover and quantify the topics and trends in StackOverflow by analyzing the textual content of these posts [6]. Their research methodology is based on latent Dirichlet allocation (LDA), which is a statistical topic modeling technique. They used LDA to discover main topics and then analyzed the discovered topics by defining metrics on their usage and inspecting the metrics over time. The major contributions of their work are identifying major discussion topics, identifying closely-coupled topics, analyzing the trend of developer interest over time and investigating topic impact which led to the trend analysis of specific technologies and comparing the impact of a particular technology across multiple topics. Their technique can be extended to help automatically generate contextual tags for posts and also to locate posts of similar content.

3.2 The Distribution of stack overflow usage over the globe

Every year StackOverflow surveys over thousands of developers and finds out the trending languages, frameworks, key territories in the world, demographics of the developers, and so on. In 2021, they surveyed 83,439 software developers from 181 countries around the world [3]. To find out which technologies to include on the survey, StackOveflow looked at the most popular and fastest growing tags. Based on the findings, they prepared the survey questions.

3.3 Clustering keywords based on similarities

Clustering huge amount of data from Stack Overflow based on similar tags can improve responsiveness and overall performance. Daniel et al. described several methods for finding relevant questions based on user queries [8]. They used k-means document clustering and topic modeling to group the questions together on the basis of similar tags or context. Due to misclassification and overlap problem with k-means clustering, topic modeling was also taken into consideration. LDA(Latent Dirichlet Allocation), NMF(Non-Negative Matrix Factorization) and LSA(Latent Semantic Analysis) algorithms were used for topic modeling. Furthermore, they built an ensemble model by combining k-means and topic modeling to retrieve matching questions. Ensemble and NMF both produced a 67% recall whereas k-means produced a recall of 50%. Moreover, comparison of cosine similarity also showed that Ensemble and topic modeling surpassed k-means. With a motivation to answer huge number(6.4 million) of unanswered questions on StackOverflow, Abhishek et al.[11] proposed to create intent-based clusters of questions using cosine similarity of the two vectorized questions. They proposed a new graph-based algorithm using standard BFS algorithm and implemented a tool named SOCluster based on sentence-BERT vectorizer. To evaluate the resultant clusters, they used three metrics; Silhouette coefficient, Calinski-Harabasz Index and Davies-Bouldin Index. One of the inputs of the proposed graph-based algorithm was Threshold Similarity. They used 8 different values for it and applied on 4 different sizes of dataset. Among the different values, 0.9 showed improved result for all sizes of dataset. This experiment created many singleton clusters

(cluster with one entity) which is one of the obstacles for the purpose, i.e. answer unanswered questions using similar answered question.

3.4 The prediction of response time

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 response time. Arunapuram et al. [4] 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. [7]. 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. Finding response time of a question is also related to getting an accepted response. There may be cases when the question is not resolved or not getting any answer. Mohamad et al. [13] worked on finding features contributing to a question not being resolved and validated those features by implementing various predictive models trained on feature values of 18 million questions to predict whether a question will get accepted answer or not.

3.5 The prediction of user expertise

StackOverflow is a common forum for users with expertise levels ranging from beginner to expert. It is important to know the user's level of expertise while answering a question. Also, knowing the expertise of the user who has replied to the post can help assessing the credibility. StackOverflow assigns badges to all the users. But it is not adequate to represent the actual level of expertise and rather misleading in most cases. Baltadzhieva and Chrupala [5] tried to predict the quality of questions that can tell us about the user expertise implicitly. It is evident that recognizing good questions can also help finding out expert users. In their work, they used ridge regression models by incorporating different features like question tags, the length of the question title and body, presence of a code snippet, the user reputation and terms used in the post to predict the number of answers and the question score. They divided the dataset into 60-20-20 partition for training, validation and test. Then four different models were trained by adding features one-by-one. The base model used only the tags, model 2 used question title and body length and code snippet, model 3 added user reputation to it and finally model 4 was fed with the terms. The performance of the models were compared using R^2-value, Mean Squared Error (MSE) and Mean Absolute Error (MAE). For every metrics, model 4 performed the best. Their work concluded that inclusion of terms improved prediction quality. Moreover, they analyzed the influence of different features and stated that larger title and body length, use of code snippet reduce the question quality. The questions with higher quality are dominated by generic terms rather than programming/expertise terms. Also, typos and off-topic nature of the questions penalize the score. They claimed that user reputation has a positive correlation with the quality measure. Diyanati et al.[9] started where the previous discussion ended. They tried to predict the user expertise from two different points of view. First, they considered a linear regression model to evaluate whether the expertise can be inferred from the scores of the questions and answers posted by an individual. Their analysis showed that there is no significant relationship between Q&A scores and user expertise. Later, they performed comment-mining

by scoring positive and negative words in the comments of the posts to figure out the user expertise and this approach came out successful in inferring level of expertise of users. In the comment-mining approach, first they extracted user reputation feature from the dataset and calculated the value based on their own metrics such as positive reputation for up-voted questions and answers, accepted answers and negative reputation for down-voted Q&A's. Afterwards, they filtered out the users whose comments were trivial or generic across different posts as outliers. Then, they combined the reputation and comment scores to get the final user score. Finally, they clustered all the users into five groups namely, *very good, good, medium, low, and very low* using k-means clustering. They evaluated the performance of their model manually by employing five experts in Programming and Computer Engineering, and used precision, recall and F-measure as metrics. This comment-mining approach was better in finding the users with the highest level of expertise and the analysis showed that about 55.9% users have very low expertise.

4 METHODOLOGY

The different tasks that have been completed to implement the project are as follows. First, we have collected from [2] and stored the data. Then we have extracted the query analysis-based results. Next we have built the predictive and clustering models. After that, we have evaluated the performance of the predictive models based on different metrics. Finally, the findings of the analysis have been visualized and implemented on a web interface.

4.1 Data Processing

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

4.1.1 Data 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 the xml files, we have used the following four files:

(1) Posts.xml - This file has 17 attributes. It contains user's posts and all information related to the post.
(2) Users.xml - This file has 12 attributes. It contains user's information.
(3) Badges.xml - This file has 6 attributes. It contains each user's badge type information.
(4) Tags.xml - This has 5 attributes. It contains information of each tag found in the StackOverflow posts.

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.

4.1.2 Data Storage. We have used HDFS to store our raw dataset and generated outputs.

4.2 System Configuration

We have run our experiments on spark-scala platform. We have used scala version 2.12.6 and spark version 3.0.1 for our project. We ran all our experiments locally using local[*] which means spark used as many worker threads as logical cores on our physical machine. For query-based analysis, we have used SparkSQL and for the predictive & clustering tasks, we have used Spark MLlib (for regression model) and python packages scipy & scikit-learn (For classification and clustering). For visualization of our results, we have used reactJS and nodeJS.

4.3 Query-based Processing

In this section, we have provided details about the query-based analysis of our project.

4.3.1 Finding out Most Popular Programming Languages or Tools Over Years. To find out the most trendy tools or languages over the years 2014 to 2021, we require the information of tags from the Posts.xml file. First, we have filtered the posts by year using SparkSQL. Then we have designed a map-reduce function which takes tags as inputs and outputs the total number of times each tag appeared on the posts from that year. The number of times a tag appeared in the posts is proportional to its popularity during that year. The tags that appeared most, are the ones that were most popular in that year.

4.3.2 Visualizing Geographical Distribution of StackOverflow usage. To find out the the total number of posts for a specific location, we require the location from where a post was posted. We have done this by first extracting users' location information from the Users.xml for each post from the Posts.xml file. Then, we have used SparkSQL queries to count the number of posts coming from specific geographic regions. These counts are used to visualize the geographical distribution of StackOverflow usage in Section 5.1.2.

4.3.3 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. Based on this information, we can measure the co-occurrence frequency of each tag-pair. 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. We have further used these co-occurred tag-pairs to create agglomerative hierarchical clusters of tags, , which will be discussed in in Section 4.4.3.

4.4 Prediction & Clustering Tasks

In this section, we will introduce the predictive models used in our project.

4.4.1 Predicting Response Time of a Post: Prediction of response time of a Post has been implemented in two phases. The first phase is preparing feature set and second phase is implementing regression based prediction model to generate the final prediction. Vasudev at el. [7] analyzed the importance of several tag based and non-tag based factors and based on the importance metric they presented we have considered five significant factors to train our model. The list of all features and their detailed preparation steps have been presented as follows:

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. To avoid the overhead of invoking the spark sql query repeatedly to get count for each tag, we have created a HashMap containing <tag,count> pairs as <key,value> by parsing Tags.xml file. So that makes the count query for each tag in O(1) time.

Number of Popular Tags: We have considered a threshold popularity value of 50 to define a tag as popular and non-popular one. For each post (question), we have counted the total number of popular tags.

Tag Specificity: Vasudev at el. [7] defined the tag specificity of a question as the average "togetherness" of all pair of tags. The togetherness of two tags x and y is defined as the following formula:

$$Togetherness(x, y) = \frac{P(x, y)}{P(x)P(y)}$$

Where P(x,y) is probability of x an y occurring together in a question and P(x) is the probability of tag x occurring in a question. For calculating P(x,y) for each question, we iterated over each pair of tags and fetched the number of posts the tags x and y occur together from the result of the query of section 4.3.3.Then we computed P(x,y) of each pair of tags by dividing the co-occurrence frequency of each pair over total number of posts. P(x) and P(y) have been computed in similar and we have used the HashMap created for calculating tag popularity for efficient querying.

Question Title and Body Length: For each question on the Posts.xml, we have computed the length of title and question body using collect() and foreach().

Finally the last stage of data preparation is to calculate response time of each question. To calculate the response time of a question containing PostID *pid*, we found the reply posts of that question by querying over the Posts.xml file, whose parentPostId=pid. Then we calculated the response time of each reply by taking difference between the CreationTime of the parent post and CreationTime of the reply. Then the minimum difference is the considered as the response time of the question. We have implemented a LinearRegression model that takes the feature set as input for training and predict minimum response time (label).

4.4.2 Predicting User Expertise. In this task, we have trained a classification model to predict the user expertise level. The whole pipeline from collecting the data to evaluating the model is shown in figure 2. Among the data files, for this task, we have used Users.xml and Badges.xml. The 2 files are about 5GB each in size. From Badges.xml, we have selected two columns: UserID and Badge that will be the label for our classification model. In StackOverflow, users are assigned three kinds of badges where 1 means Expert , 2 means Intermediate and 3 means Beginner level. For training our model, we have extracted the required features from Users.xml file. From this file, we have selected five columns: UserID, Reputation, Views, UpVotes, and DownVotes. First, we have joined the two xml files on UserID after selecting the relevant columns. We have done this joining and selection steps using SparkSQL in scala. After extracting the features and labels for each data point, we saved them into a csv file. The finalized dataset have about 6 million users' information where about 300K are Experts, about 1M are Intermediate level users, and the rest are Beginners. So, it is evident the dataset is highly skewed and the ratio of expert, intermediate and beginner level users are about 1:3:16. We then inspected the values of different features and we have found that the range of values vary over different features. We then decided to train our model by both normalized and non-normalized data. Next, we split our dataset into 90:10 ratio of training and test. Both training and test data are split in a way that they maintain the ratio of labels.

Since we have three types of labels, we need a multi-nomial classification model. We have first trained our data on a decision tree model using entropy as it is the easiest classification model. However, there are only four training features and it is not enough to get a good result. This proved that our data was not separable using linear decision boundaries. Afterwards, we have looked into a logistic regression model. We have used a 10-fold cross-validation approach to fit the data. We have also tuned the value of the strength of L2-regularization parameter by varying from 0.0001 to 1 on a log-scale and trained a model without any L2-regularization. Besides, we have executed the same steps using non-normalized data to see whether normalization creates any significant difference in the prediction performance. After this cross-validation step, we have evaluated the best models for both normalized and non-normalized data. The performance of the best models are later explained in section 5.2.2.

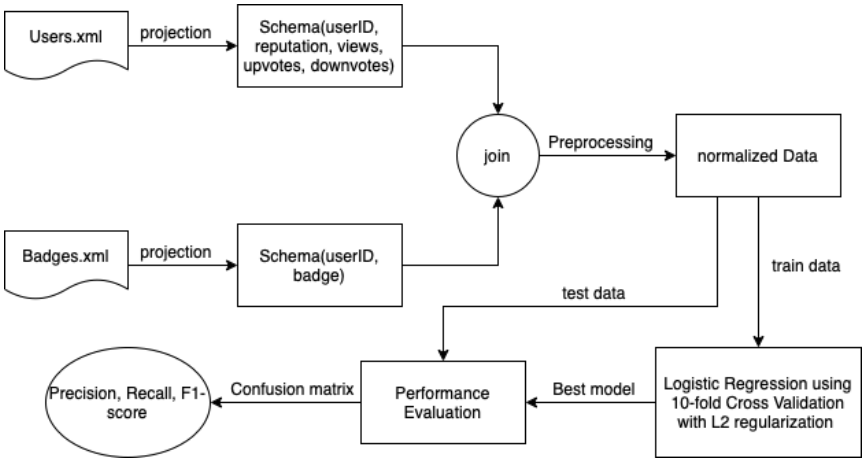


Fig. 2. Pipeline for the Logistic Regression model to predict the user expertise level.

4.4.3 Clustering Co-occurred Tags. In this part, we take the output from section 4.3.3 and develop a similarity matrix from the co-occurrence counts of tag-pairs. The higher the co-occurrence count, the higher the strength of togetherness for those tag-pair. However, the scipy package of python for generating hierarchical clusters uses the distance measures between nodes. We first populate an $N \times N$ symmetric matrix where each entry corresponds to the co-occurrence count of two tags (u, v). 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. Next, we normalize the similarity matrix and subtract each entry from 1.0 to get the distance measure between tag-pairs. Finally, we feed this matrix into a model to generate an agglomerative hierarchical cluster of tags. The analysis of the different clusters obtained in this step is narrated in section 5.2.3.

4.5 Web Application

4.5.1 Back-end. We have used node.js to build the back-end. Here, we have stored the results generated by both query and predictive tasks in csv files. We have written multiple APIs for different tasks. When a request comes from the front-end, using the designated API, the node.js server responds with the corresponding results in json format.

4.5.2 Front-end. We have built the front-end with react.js. This react app communicates with the node-server and shows the project overview and results using different visualization techniques such as column charts, pie charts, dendrograms and boxplots.

4.5.3 Visualization. We are showing the results using different data visualization techniques in our front-end app.

Usage Trends We have projected the results in column charts for a specific language usage vs. years (2014-2021) for top three languages (Figure 4a, Figure 4b, Figure 4c). The top three languages are python, javascript and java. In several pie charts (Figure 3a, 3b, 10a, 10b, 10c, 10d, 10e, 10f), we are showing all languages and frameworks usage per year (2014-2019).

Geo-tagged Distribution We show one column chart (Figure 5) for total posts vs countries(top 20 countries). While calculating the total posts for a country, we have considered the timeline from 2014 to 2021. USA has the

most posts in StackOverflow over this time period, so we are showing the posts' distribution across the different states of USA in a map (Figure 6).

Clustered Tags In dendrograms (Figure 9, Figure 11), we are showing the top pairs of tags based on co-occurrence.

Predictive Tasks We have reported boxplots (Figure 8) for the accuracy of 10-fold cross-validation models trained on non-normalized data with different values of L2-regularization parameter(λ).

5 RESULTS & PERFORMANCE EVALUATION

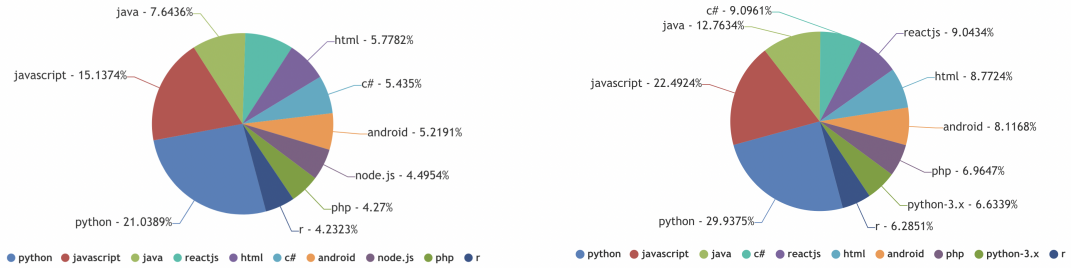
In this section, we have reported the results of the different analytical queries and we have evaluated the performance of the predictive models.

5.1 Query-based Tasks

5.1.1 Visualizing Most Popular Programming Languages or Tools Over Years. We have used two different visualizations in order to represent the usage trends over 2014 to 2019 which we found in Section 4.3.1.

(1) We have created pie charts showing the top 10 popular languages or tools for year 2021 (3a) and year 2020 (3b) separately in Figure 3. The rest of the pie charts for year 2014 till 2019 are included in Section 6. We observe that python and javascript have been the most trendy programming languages for the past few years. We also observe that javascript was trending the most till year 2019, but python surpassed javascript usage from year 2019 and have been trending since then.

(2) We have created bar charts showing usage trends of the three most popular programming languages, python, javascript and java from year 2014 to year 2021 consecutively in Figure 4. We observe that the usage of java has shrank (4a) whereas the usage of python has largely grown over the years(4c. We further observe that the usage of programming languages in general has grown in 2019-2020 period. It seems like this is somehow related to the covid-19 pandemic and people working from home.



(a) 2021 (b) 2020

Fig. 3. Top 10 popular languages or tools

5.1.2 Visualizing Geographical Distribution of StackOverflow usage. First, we have plotted a column chart (figure 5) visualizing the top 20 countries based on the number of StackOverflow posts. We can see that the USA tops the list followed by India, UK, Canada, and so on. Since the USA tops the list, we have then looked into the post count distribution there across states. We have created a choropleth map (Figure 6) to show post distribution over fifty states of the USA for years 2014 to 2021. It can be observed from the visualization that California has the highest StackOverflow

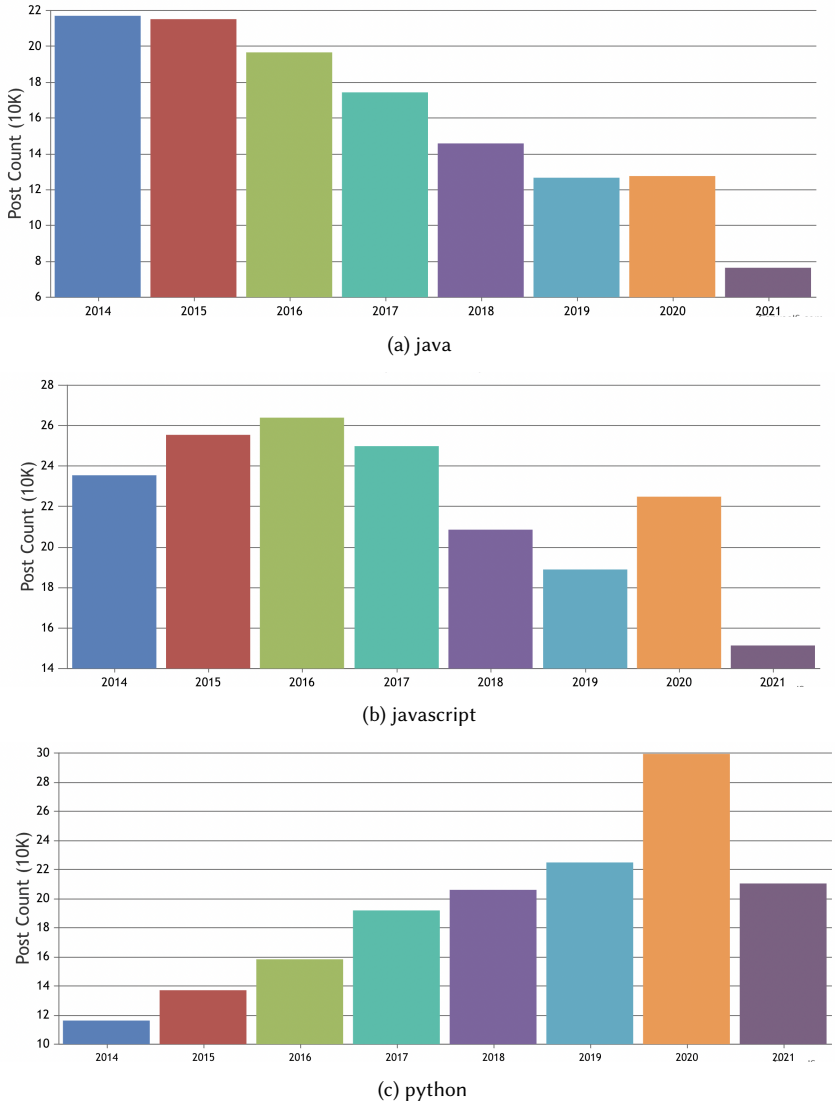


Fig. 4. Usage trends of java, javascript, and python since 2014 (The data of 2021 is incomplete).

usage record followed by Washington, Illinois, Texas and New York. From our analysis, we can say that the geographic distribution actually represents the popularity distribution of StackOverflow over the map. Behind the scene, the popularity can intuitively refer to the extent of software development and computer science and engineering studies taking place at those regions.

5.1.3 Visualizing Co-occurrence frequency of tag-pairs. We have shown the top 10 results found from Section 4.3.3 in Table 1. We observe that related tags appears together such as python and python-3, ruby and ruby-on-rails, html and

StackOverflow

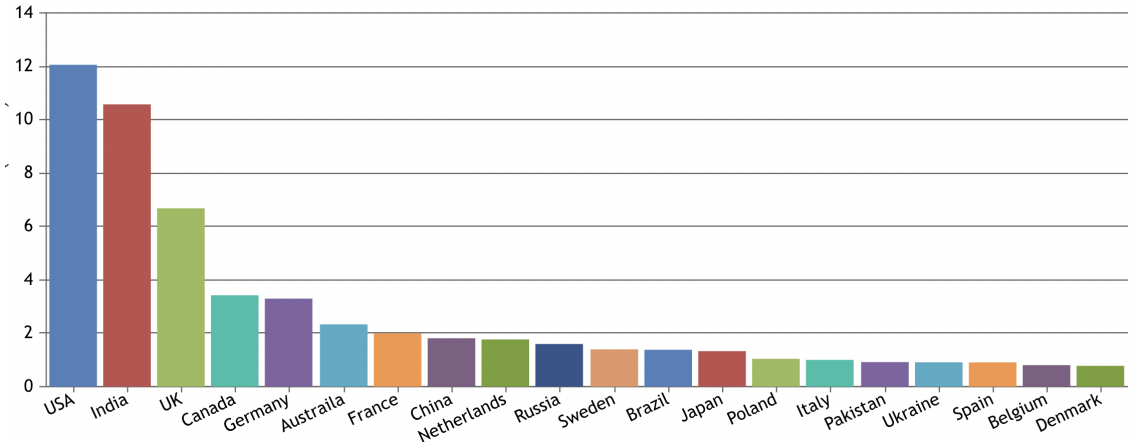


Fig. 5. Country-wise usage of StackOverflow (top 20).

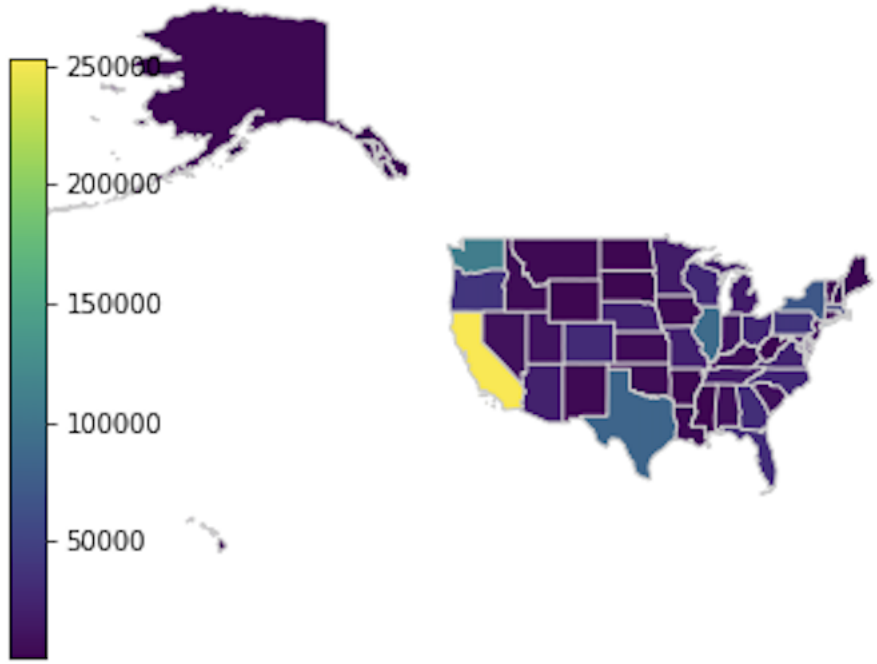


Fig. 6. State-wise usage of StackOverflow in the USA

css etc. From our observation, we can conclude that the co-occurrence frequency of tag-pairs is proportional to how closely they are related to each other.

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.

5.2 Predictive & Clustering Tasks

5.2.1 Evaluation of Response Time Prediction Model. We have used Posts.xml and Tags.xml files to prepare our feature set and label (response time in minutes). The steps of preparing each feature value involves several DAG stages which require very large computing time in such a volume of data. Because of the time constraint, we used a smaller set of Posts.xml (containing 26600 Posts) to prepare the feature set and train the model. We also discarded the post which response time is greater that 10 days. We can get some insights about the correlation of the features with response time from figure 12. The bigger length of body and title of posts tends to have lesser response time due to the fact the the question is well described through title and body. The number of popular tags and 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. Finally the more specificity of tags of a question indicates better context of the question which somehow is helping to get response quickly.

We have used Spark MlLib library to implement the LinearRegression. We have taken RegParam $\in$ (0.3,0.1.0.01) and ElasticParam $\in$ (0.0,0.3,0.8,1.0) and reported RMSE for each combination of these parameters. The RMSE is nearly same in all parameter settings and the value of RMSE is roughly between 1.485 and 1.489. The probable reason for this can be training over of the smaller dataset and RMSE can be improved if we can train over larger sets.

5.2.2 Evaluation of User Expertise Prediction Model. We have first generated the boxplots for the different cross-validation models as mentioned in section 4.4.2. Figures 7 and 8 represent the boxplots for non-normalized and normalized data respectively. For normalized data, we get lower standard deviation than that of non-normalized data. However, the accuracy of all the models are somewhat similar. Afterwards, we evaluate both the models on test data and get the confusion matrix (tables 2 and 4). From the tables, we can see that both models are highly biased towards label Beginner since our dataset is also highly skewed in favor of it. However, if we look into the tables 3 and 5, the model trained on normalized data achieves a little bit of better precision than the one that is trained on non-normalized data.

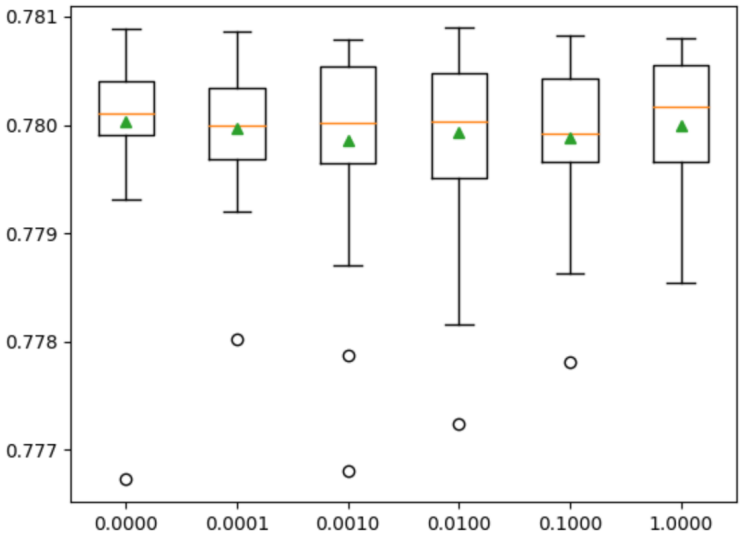


Fig. 7. Boxplots for the accuracy of 10-fold cross-validation models trained on non-normalized data with different values of L2-regularization parameter(λ)

Actual/Predicted	Expert	Intermediate	Beginner
Expert	3	90	30910
Intermediate	8	149	103539
Beginner	16	222	478408

Table 2. Confusion matrix of the non-normalized test data for user-expertise prediction model

Label	Precision	Recall	F1-score	Support
Expert	0.11	0.00	0.00	31003
Intermediate	0.32	0.00	0.00	103696
Beginner	0.78	1.00	0.88	478646

Table 3. Performance metrics of the non-normalized test data for user-expertise prediction model

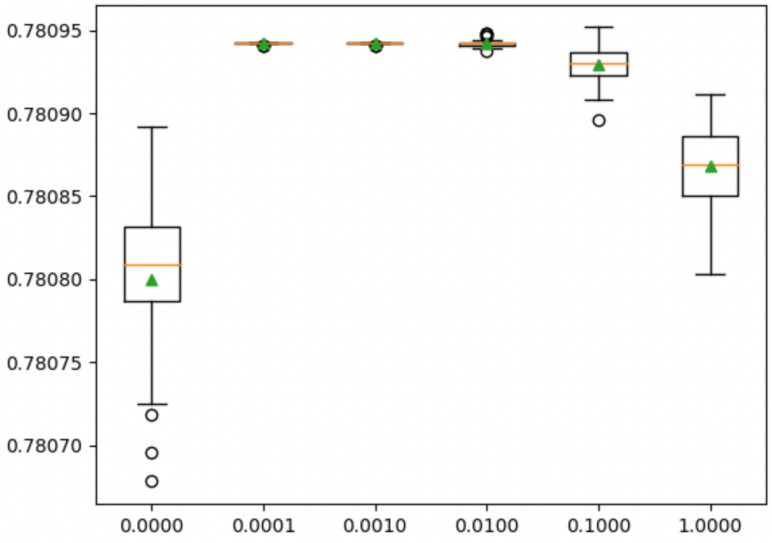


Fig. 8. Boxplots for the accuracy of 10-fold cross-validation models trained on normalized data with different values of L2-regularization parameter(λ)

Actual/Predicted	Expert	Intermediate	Beginner
Expert	7	87	30667
Intermediate	5	174	103044
Beginner	11	212	479138

Table 4. Confusion matrix of the normalized test data for user-expertise prediction model

Label	Precision	Recall	F1-score	Support
Expert	0.30	0.00	0.00	30761
Intermediate	0.37	0.00	0.00	103223
Beginner	0.78	1.00	0.88	479361

Table 5. Performance metrics of the normalized test data for user-expertise prediction model

5.2.3 Analysis of Clusters based on Co-occurrence of Tags. We have generated clusters by taking top 25 and 50 pairs of tags based on their co-occurrence counts. We extracted 10 clusters using the top 25 pairs of tags which contained 30 unique tags. The 10 clusters according to figure 9 are – **C1**: python, python3, **C2**: ruby-on-rails, ruby, **C3**: swift, xcode, ios, **C4**: javascript, jquery, **C5**: html, css, **C6**: pandas, django, numpy, matplotlib, **C7**: objective-c, **C8**: android, java, html5, monaca, onsen-ui, node.js, swift3, **C9**: c#, .net, visual-studio, unity3d, **C10**: php, mysql, laravel. 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. We have also produced a dendrogram (figure 11) for 20 clusters using the top 50 pairs of tags consisting of 51 unique tags. For further details, see section B in the Appendix.

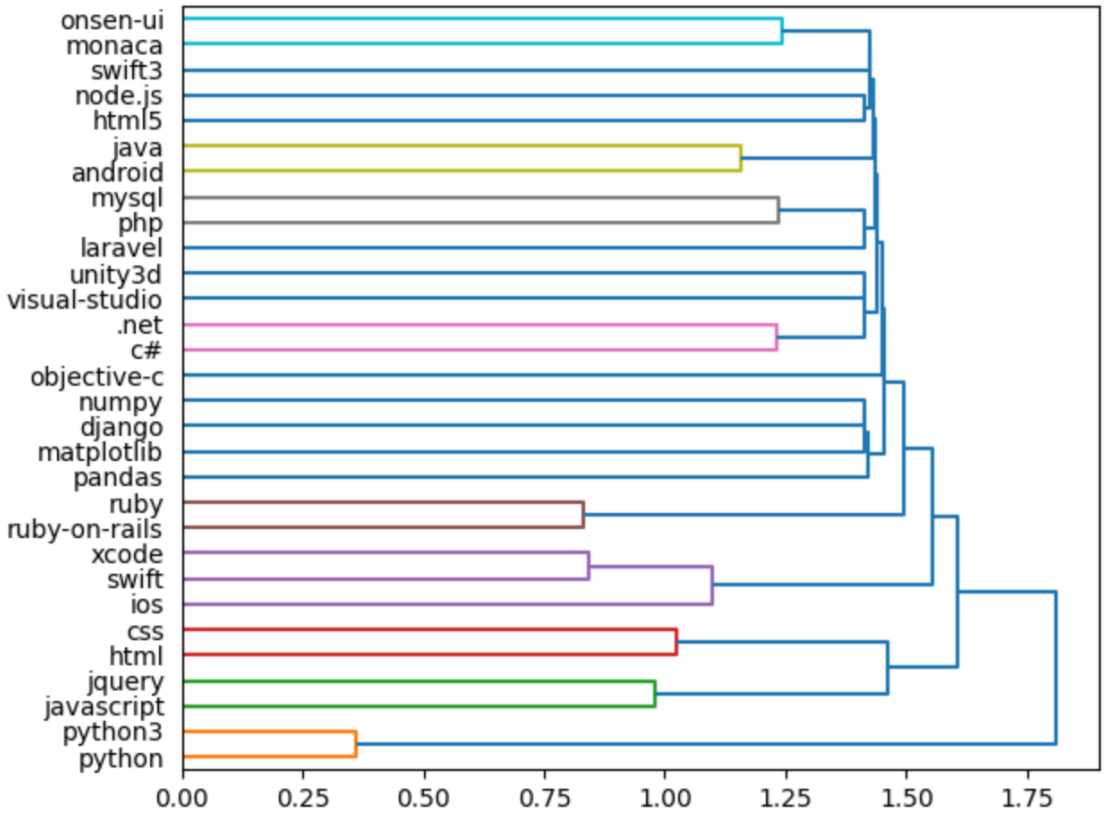


Fig. 9. Dendrogram for top 25 pairs of tags based on co-occurrence

6 CONCLUSION

Our project utilizes 100GB of StackOverflow data and gleans interesting findings based on both query-based and prediction-based analysis. We have shown an overview of usage trends of different programming languages and tools over the years. We have shown the geographic distribution of StackOverflow popularity assuming that the popularity is positively correlated to the number of posts. We have also shown clusters of frequently co-occurred tags using agglomerative hierarchical clustering and found that the tags belonging to a cluster are actually supposed to be together. We also analyzed different contributing factors to predict response time of posted questions and found correlations of response time with tags popularity, tag specificity, question length and title length. We have found that the data is highly skewed in favour of the beginner level users and to be precise, the ratio of expert, intermediate and beginner level users is about 1:3:16. Since, we had only four features for this model, decision tree performed poorly. However,

multinomial logistic regression trained with normalized data could predict the expertise level of users with some level of precision. We actually planned to predict the future trends of programming languages usage but we could not do it because we did not have enough datapoints (8 data points for years 2014-2021) to extrapolate. We believe that our findings address the primary motivations of our work which were to be a guide for an amateur in choosing the next step towards self-development, a data-driven tool for predicting user expertise in different domains and finally, a fun way to get a good grasp over big-data management concepts and frameworks.

REFERENCES

[1] [n. d.]. https://stackoverflow.blog/2017/05/09/introducing-stack-overflow-trends/

[2] [n. d.]. https://archive.org/details/stackexchange

[3] [n. d.]. StackOveflow 2021 Developer Survey. https://insights.stackoverflow.com/survey/2021.

[4] 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.

[5] Antoaneta Baltadzhieva and Grzegorz Chrupała. 2015. Predicting the quality of questions on stackoverflow. In *Proceedings of the international conference recent advances in natural language processing*. 32–40.

[6] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654. http://dblp.uni-trier.de/db/journals/ese/ese19.html#BaruaTH14

[7] 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.

[8] Vishi Cline, Abhishek Dharwadkar, R. C. Goyal, Dan Engels, Raghuram Srinivas, and Sohail Rafiqi. 2018. Stack Overflow Question Retrieval System.

[9] 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.

[10] Felipe Hoffa. [n. d.]. https://towardsdatascience.com/when-will-stack-overflow-reply-how-to-predict-with-bigquery-553c24b546a3

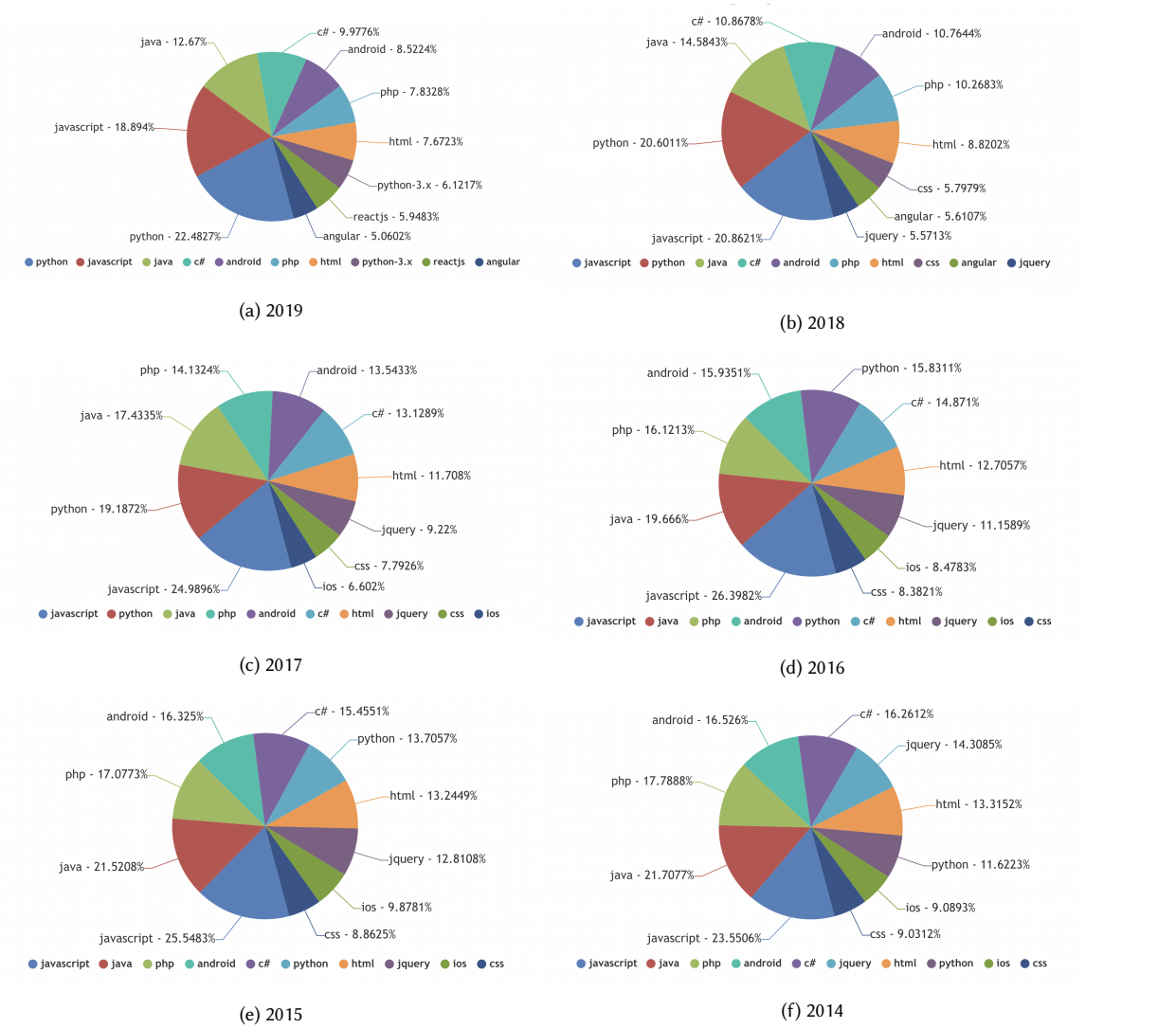
[11] Abhishek Kumar, Deep Ghadiyali, and Sridhar Chimalakonda. 2021. SOCluster- Towards Intent-based Clustering of Stack Overflow Questions using Graph-Based Approach. arXiv:2107.02399 [cs.SE]

[12] 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

[13] Mohamad Yazdaninia, David Lo, and Ashkan Sami. 2021. Characterization and Prediction of Questions without Accepted Answers on Stack Overflow. *arXiv preprint arXiv:2103.11386* (2021).

APPENDIX

A POPULARITY OF LANGUAGES OR TOOLS OVER YEARS



(a) 2019 (b) 2018

(c) 2017 (d) 2016

(e) 2015 (f) 2014

Fig. 10. Top 10 popular languages or tools

B CLUSTERING TAGS BASED ON CO-OCCURRENCES

The 20 clusters according to figure 11 are – **C1**: python, python3, **C2**: ruby-on-rails, ruby, **C3**: swift, xcode, ios, **C4**: javascript, jquery, html, css, **C5**: pandas, **C6**: objective-c, **C7**: android, android-studio, **C8**: java, spring, spring-boot, **C9**: django, numpy, matplotlib, tensorflow, machine-learning, anaconda, **C10**: c#, .net, visual-studio, unity3d, wpf, **C11**:

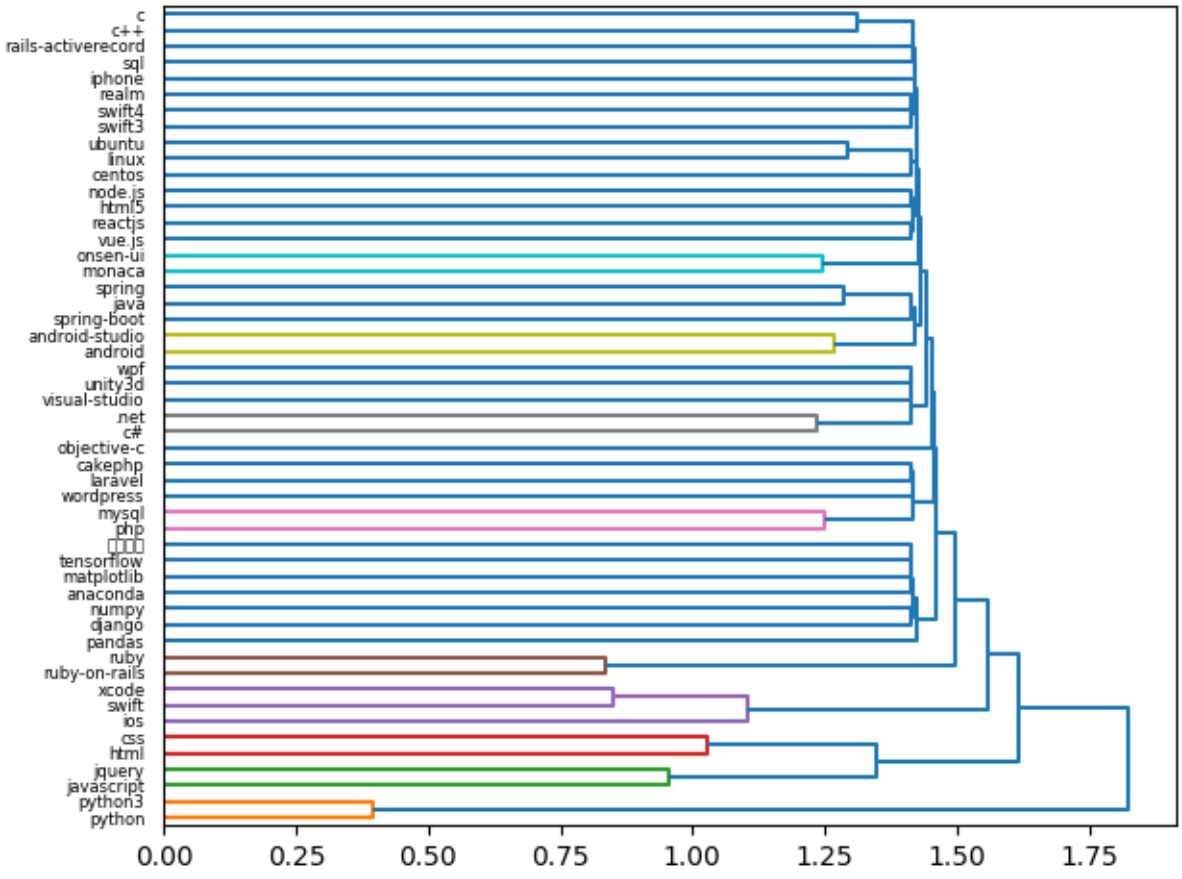
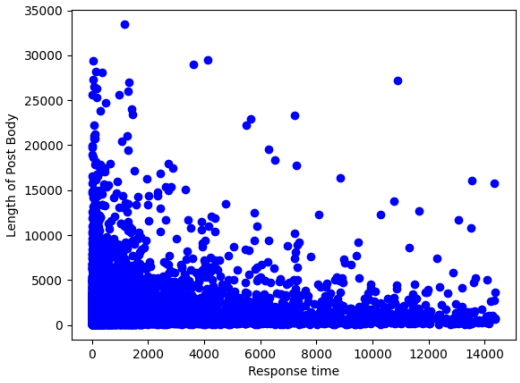


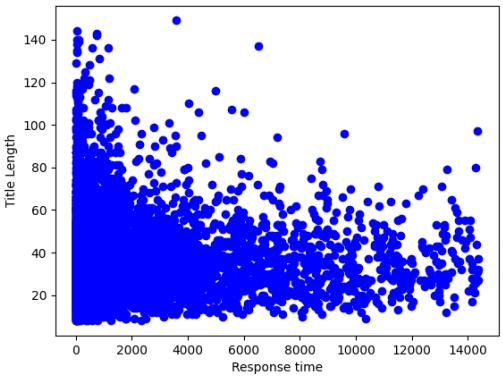
Fig. 11. Dendrogram for top 50 pairs of tags based on co-occurrence

html5, node.js, vue.js, reactjs, **C12**: php, mysql, **C13**: laravel, cakephp, wordpress, **C14**: monaca, onsen-ui, **C15**: swift3, swift4, realm, **C16**: linux, ubuntu, centos, **C17**: iphone, **C18**: sql, **C19**: rails-activerecord, **C20**: c++, c.

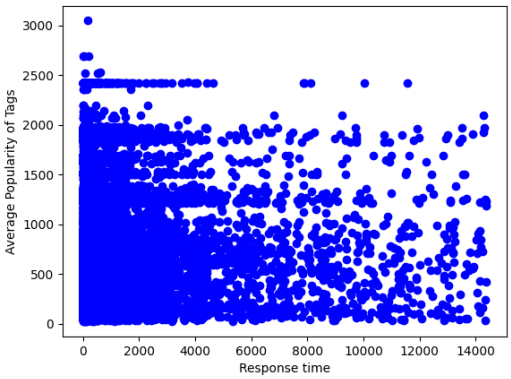
C SCATTER PLOTS FOR DIFFERENT FEATURES VS RESPONSE TIME



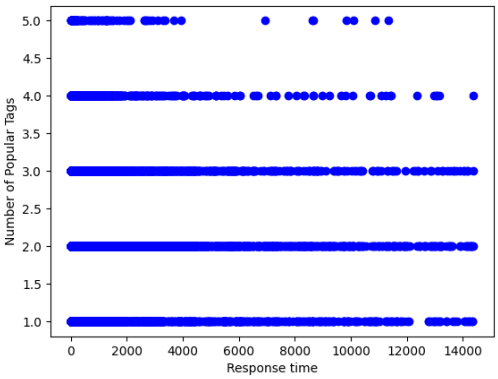
(a) Post body length vs. Response time (minutes)



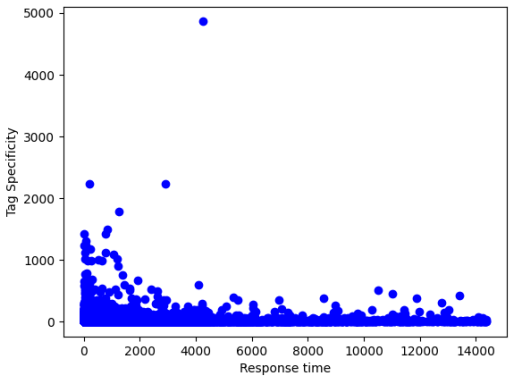
(b) Post title length vs. Response time (minutes)



(c) Average popularity of tags vs. Response time (minutes)



(d) Number of popular tags vs. Response time (minutes)



(e) Tag specificity vs. Response time (minutes)

Fig. 12. Scatter plots for different features vs. Response time