PHP and JavaScript: A Comparative Analysis of Common Developer Queries

Diya Burman

Department of Computer Science, University of Waterloo

Abstract— **This paper deals with the analysis of the commonly asked questions and their answers from the Stack Overflow dataset to find out what PHP and JavaScript developers are commonly discussing on such popular question-and-answer (Q&A) platforms. While PHP has been around for quite some time, JavaScript is comparatively newer. This paper takes into account that both PHP and JavaScript have popular frameworks such as Laravel, Cake, Nodejs, Reactjs etc. In such a situation, we try to find answers to questions such as – the trends in questions that are being asked about these languages, who are the people (users of Stack Overflow) most likely to answer questions related to JavaScript and PHP thereby creating a recommendation system of "Answerers", what are the factors that determine whether a question will be answered, and finally, what are the factors that determine whether a question that has been answered will also eventually be "closed" i.e. successfully resolved.**

I. INTRODUCTION

The motivation behind this exercise is manifold. More recently, the web has transitioned from being merely a place for hosting websites to becoming a place for hosting web based applications (web apps). This has consequently led to PHP and JavaScript, two popular programming languages for web development purposes, to move to a more framework-based approach that lets a developer build web applications without having to worry about the nitty gritties of implementation such as underlying data structures, web servers, load balancers, etc. With an ever expanding community of developers around the world connected to one another through the Internet and interacting via Q&A platforms such as Stack Overflow, Quora etc., mining and analyzing the questions and answers on such platforms can yield insightful information that can help the developers streamline their development as well as learning processes.

On most community driven websites, users contribute in the form of questions, answers, comments, criticism and suggestions; thereby allowing other members to learn through collaboration and contribution. As a result, "tags" have become a way for websites to index, categorize and organize the data collected from such discussions. Almost always, it is the responsibility of the users to identify the relevant tags for their content themselves. This, can be a rather confusing process because often, users may not be familiar with the tags provided by the website or may simply not have adequate knowledge in order to identify what tags would be relevant enough to fetch accurate answers from the rest of the community. Add to this the occasion of malicious or "lazy" users simply spamming such platforms with irrelevant or redundant questions. This can lead to improperly tagged posts which in turn can lead to disruption of the organization of the content on the website. One way to help with this problem is to recommend tags for users based on the content they generate.

On a similar note, for a user asking a question on Stack Overflow, it would be of value to have an idea of how likely his question is to get an accurate answer (referred to as "accepted" answer on Stack Overflow). Consequently, how should he phrase the content of his question so that it has a higher chance of inviting an "acceptable" answer?

Furthermore, for users who actively participate in answering questions on sites like Stack Overflow, it would help if there was a system in place that could

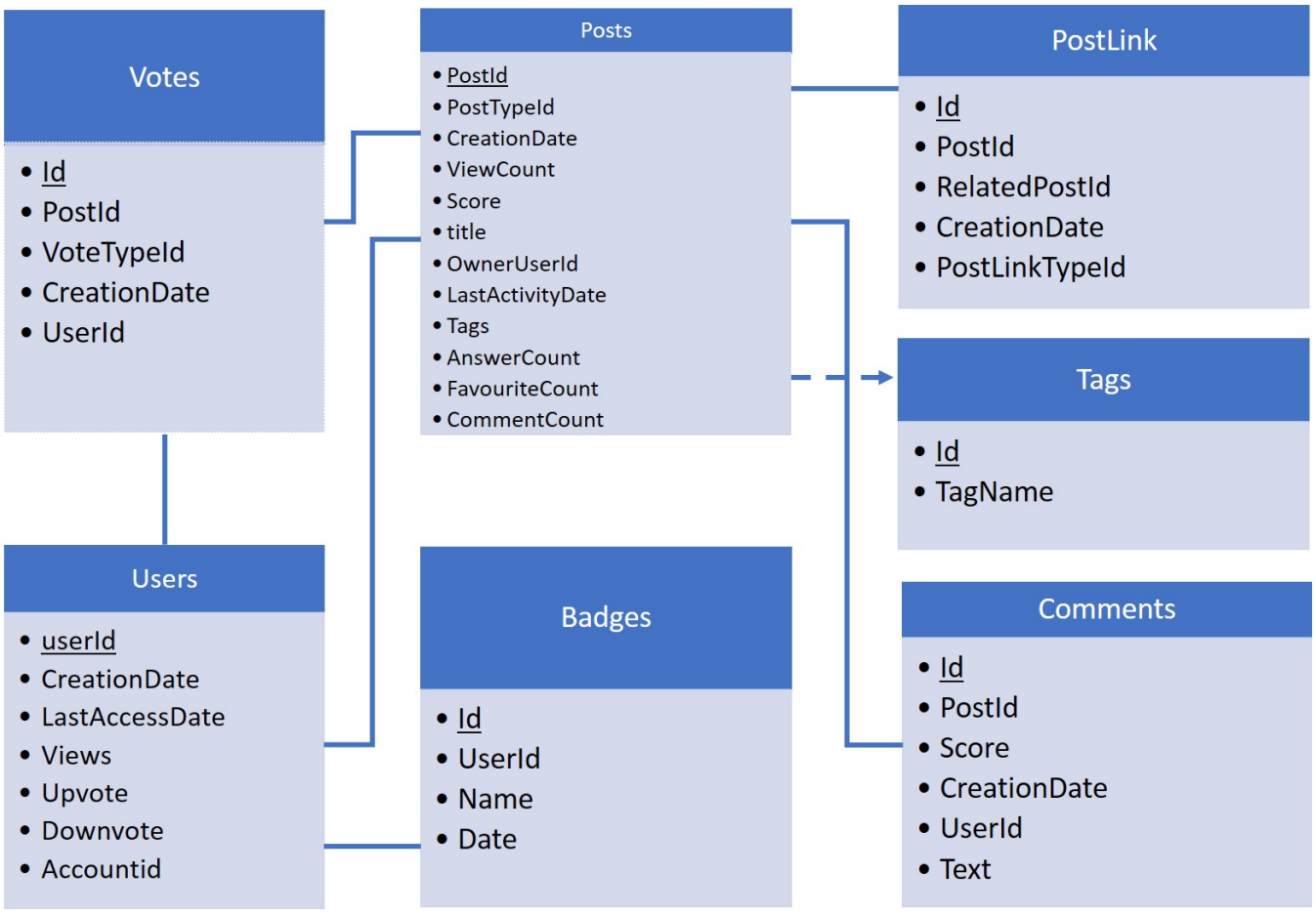

Fig. 1. Stack Overflow Schema

study the content of the user's answer-history and recommend newly posted questions to answer, into the feed; questions which are likely to fall within the prerogative or expertise of the user.

We have also attempted to create a system that automatically suggests additional tags for questions that contain either a PHP tag, or a JavaScript tag, or both. Furthermore, in our analysis of time until answer of both questions, we have attempted to see what are the common features of questions for both PHP and Javascript that assure the least possible time until answer.

II. RELATED WORK

Xia et. all, in their paper "Tag Recommendation in Software Information Sites" have proposed an algorithm called TagCombine in which they have tried to associate the best tags to untagged objects. This algorithm has three distinct parts in its operation. It uses a multi label ranking part to predict tags using a multi label learning algorithm. It also has a similarity based ranking component that uses similar objects to recommend tags, along with a third ranking component that predicts tags based on the association of certain tags with certain words. The TagCombine algorithm calculates the sum of all three models with a mix of weights in order to calculate the best combination of parameters that would yield the correct tag.

Wang et. all, in their paper "EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites" have proposed a tag recommendation system called EnTagRec, which uses a Bayesian Inference system and a Frequentist Inference system separately and then calculates a weighted sum of the resulting probabilities. The Bayesian Inference algorithm in this case uses the text from a Stack Overflow post in order

to calculate if a tag might be associated with it. This approach is possible by converting a post into a bag-of-words model and then training a Labeled Latent Dirichtlet Allocation model that is used to calculate the probability that a tag is associated with a post. The Frequentist Inference method extracts tags from a post based on keyword frequency (or any other type of preprocessing as the implementation might use), and then uses a corpus of tags to select additional tags that are similar to the ones initially chosen. The tag corpus is built by using the initial tags as nodes and the weighted edges between two tags that are contained in two different posts. This is chosen based on the Jaccard similarity between the two posts.

Jeffrey Godrei et al. discuss how most Q&A websites have no indication of when to expect an answer. Such a statistic would help developers who post questions manage their time while expecting answers. To this effect they discuss the prediction of answering time based on similarity of the set of tags associated with questions. To predict the bin of each post and evaluate the outcome, they conducted 10-fold cross validation and use a supervised learning algorithm k-Nearest Neighbors.

III. METHODOLOGY

The data for this exercise was taken from the publicly available Stack Overflow data dump. Stack Overflow is a member site of the Stack Exchange network and is the world's most popular Q&A platform for everything related to software development and computer programming. The data dump is about 20 gigabytes in size, and we downloaded it via bit torrent.

OBTAINING THE DATASET:

The dataset was downloaded directly from the public data dump in the form of zip files. The total size of the dataset was around 20 gigabytes. There are multiple tables that come under the Stack Exchange dataset, but the ones of interest to us are the following tables:

1) Badges
2) Comments
3) PostLinks
4) Posts
5) Tags
6) Users
7) Votes

PREPARING THE DATASET FOR ANALYSIS:

These tables, after unzipping, had multiple attributes within. The schema we used for the analysis is illustrated in Figure 1. The attributes of each table are also illustrated in the same. We wrote a map-reduce job to convert the XML data from these tables into a Hadoop Distributed File System (HDFS) running on an Amazon Web Services (AWS) node (Henceforth referred to as a Hadoop cluster).

EXPLORATORY DATA ANALYSIS (EDA):

Stack Overflow has a data explorer that is publicly available and can be used to query the dataset. We ran some queries on this dataset to obtain some initial metrics to spot trends on the usage of PHP vs JavaScript over time, along with the use of their respective web development frameworks. It turned out in this initial EDA that JavaScript and JavaScript based frameworks were a lot more in demand on stack overflow as compared to PHP. This was further corroborated by the fact that the growth in the total number of JavaScript related queries were much higher than the growth in the total number of PHP related queries. (Refer Fig 2 and Fig 3) We wanted to explore this further by running our own experiments on the dataset.

PREDICTING TIME UNTIL ANSWER:

The objective here was to be able to predict how quickly a question would see activity once posted. We separated our questions by ensuring that each question had either a PHP tag, or a JavaScript tag, but not both. The idea was to compute *exclusively* how much time a question related to JavaScript takes to answer, as compared to PHP.

We decided that in this case we would look at the Posts table, the Tags table, and the Comments table. We decided to create a supervised learning model which would take in a post's title length, creation time, the tags associated with it, and the first comment's time on it as the input feature set, and its output would be a predicted time that a post would see some activity. The challenge here in the dataset was that there are several

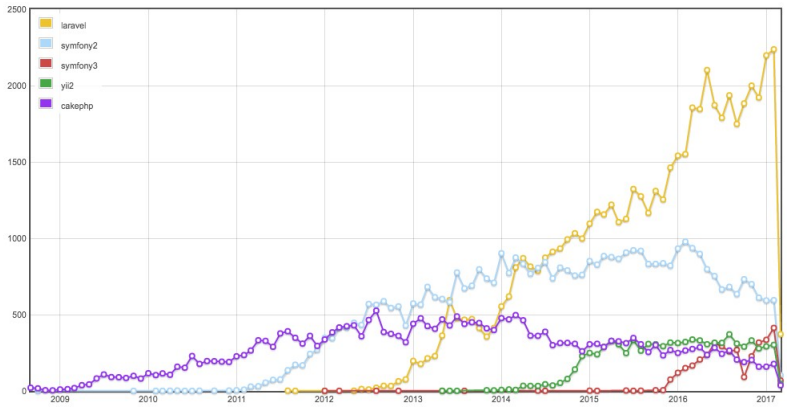

Fig. 2. Queries on PHP frameworks over time.; X: No. Of Queries, Y: Time

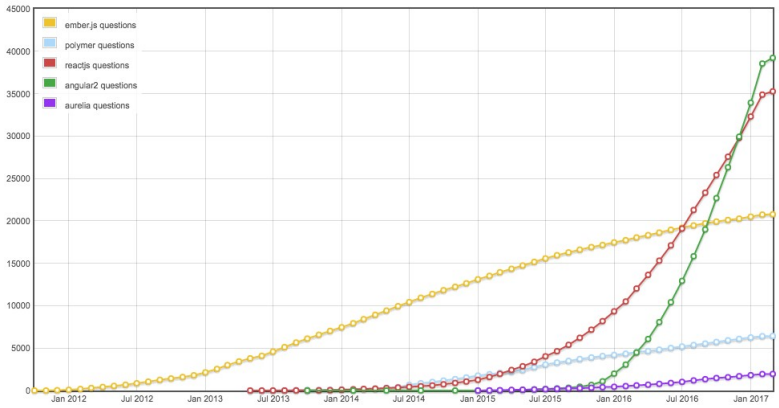

Fig. 3. Queries on JS frameworks over time.; X: No. Of Queries, Y: Time

posts that have gone unanswered, and thus training our model on this dataset would mean that there might be a good possibility of it being skewed. Therefore, we set an upper bound on our time to answer as 12 hours. In other words, we made the assumption that if a question is not answered in 12 hours, then it will not be answered (note than the word "answered" here refers to seeing activity on the post, and not the actual answer itself). We further decided to classify our response into 3 different categories : Questions that are answered within one hour, questions that are answered within one -3 hours, questions that are answered within 3-12 hours. We used 70 percent of our dataset as our training data with the remaining 30 percent as test data. We also used 10 Fold Cross Validation to help train our model. The cross validation process was achieved using Scikit Learn. In this step, we randomly split our training data into 10 different blocks. We used 9 blocks of our training data to train the model and the remaining 1 block to test the same. There were 100 iterations during the training process. The completion of this stage resulted in a fully trained model which was later tested against the remaining 30 percent of our data. We used the random forest classifier in this case, and converted the categorical data of tag names to tag lengths to better fit the classifier. We decided to choose the Random Forest Classifier as our classifier because we felt that an ensemble classifier of this type was typically hard to over-fit as compared to any gradient boosted decision tree. In the end, the question we are trying to answer is, "Given a post on Stack Overflow, which of the answering periods (< 1 hours, >1 but <3 hours, and >3 but <12 hours) would it potentially fall into, and what are the common factors of these questions that make them fall in these different time categories." Using the most influential feature function in Scikit Learn's Random Forest Classifier, it turns out that the most influential attributes of questions(irrespective of their

tags) are their header length and whether a question mark ('?') is present at the end or not. Other important factors were the number of tags associated with the question, and often times the time of day when the question was asked. An understanding of these factors will help a software developer to properly frame his questions such that he can get the fastest answers to his queries.

TAG RECOMMENDATION SYSTEM

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Fig. 4. Perceptron Equation

The goal of this exercise was to come up with a way such that when a user posts on Stack Overflow with a query that is a JavaScript or a PHP question (based on the question header and body), the system should automatically be able to suggest tags to it based on the query. This was achieved by using a neural network based approach using a single perceptron (which is basically a single input single output neural network without any hidden layers). To preprocess the data for this exercise, we first converted the post's header and body completely to lower case, then used the NLTK library in Python to eliminate all stopwords. This was followed by appending both the header and body into a single text. The objective here was to train the perceptron to recommend tags from a corpus of tags against a question. To create the corpus of tags, we queried the Stack Overflow data explorer and picked the top 14 tags (marked as "features") based on usage. Then, we created a supervised learning model where each tag from the corpus was passed into the perceptron and was classified as being associated or not associated with a question. 5 Fold Cross Validation was used during this training phase and the threshold of the perceptron was set at .65 and the bias at .25. Thereafter, the perceptron was subjected to a training set of 70 percent of the data and a test set of 30 percent. Then, for a given question, the perceptron took each tag from the corpus of tags as an input, and classified it as being associated or not associated with a given question. Those that were associated were stored in a list as the recommended tags.

IV. Results and Findings

A. Exploratory Data Analysis

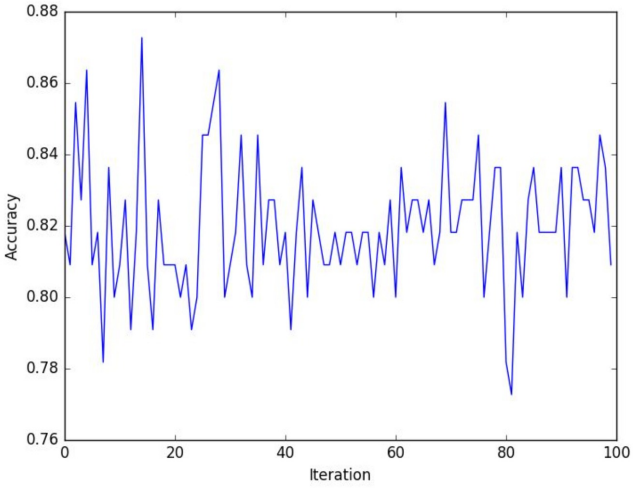

Fig. 5. Iteration vs Accuracy in case of Random Forest Classifier for Time Until Answer

The EDA on the Stack Overflow Dataset has revealed that the popularity of Javascript over PHP has increased exponentially. However, with the advent of frameworks, particularly Laravel, the popularity of PHP has also gone up. But more importantly, both languages are primarily used in frameworks more than anything else.

```
feature  0  has importance  0.0115137065722
feature  1  has importance  0.0858641953377
feature  2  has importance  0.0808103393689
feature  3  has importance  0.0110014044101
feature  4  has importance  0.0157899845058
feature  5  has importance  0.0735822629869
feature  6  has importance  0.0300767385292
feature  7  has importance  0.102459780524
feature  8  has importance  0.201870105617
feature  9  has importance  0.0738996025243
feature  10  has importance  0.101227116193
feature  11  has importance  0.0174719453109
feature  12  has importance  0.00714475689611
feature  13  has importance  0.0874541166576
feature  14  has importance  0.0998339445652
```

Fig. 6. Feature Importance in Tag Recommendation System

B. Time until Answer

The average accuracy for the time until answer was found at around 82 percent as illustrated in Figure 5. We also discovered that more than 120 iterations

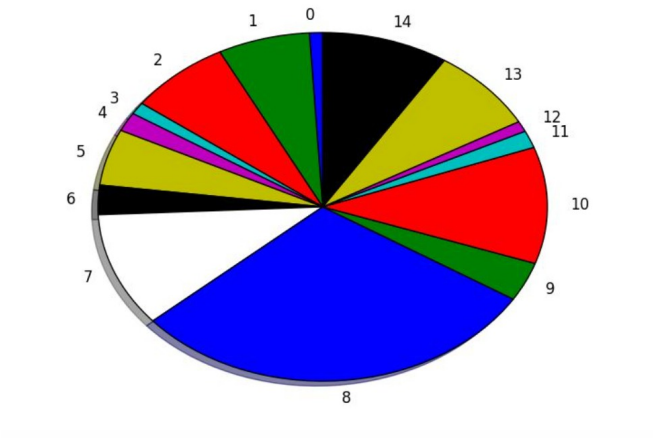

Fig. 7. Feature Importance in Tag Recommendation System

would cause our model to overfit, and hence kept the number of iterations limited to 100. This was also a resource consuming process on AWS. We found that the percentage of Javascript related questions (that is questions not having a PHP tag) answered in less than 1 hour were at 86.35. For PHP that value was at 79.7. Thereafter, we found that questions related to both are answered fastest if they are posted between 10:30am - 2pm EST. We also found that the average length of headers for the questions that were answered the fastest was at 6 words (excluding stopwords), and that 94 percent of questions that ended with a question mark ('?') were likely to receive an answer.

C. Tag Recommendation System

The neural network based tag recommendation system showed an accuracy of 18 percent in the testing phase. As illustrated in Figures 6 and 7 the perceptron takes in a tag and tries to classify its importance and therefore, classify it as being associated or not associated with the question posted. This system was not very accurate and the inference from implementing this technique is that a single perceptron by itself will not suffice to predict tags.

V. FUTURE WORK

This work can be further expanded by exploring several more options in building recommendation systems for tags as well as time until answer. for example, a multi layered deep neural network can be used to predict time until answer, taking every feature of a question into account. Although this is computationally very expensive, it is still worthwhile since all the features associated with a question can be taken into consideration at the same time. Other options include using support vector machines to classify an answer automatically as being acceptable or unacceptable depending on the question. An automatic moderator's tool can also be developed based on tf-idf and bayesian learning to determine if a user's post is offensive in nature or not and whether it should be deleted automatically.

VI. CONCLUSION

In this work, I have explored the Stack Overflow public data dump using the online tools provided. I have also, manually, processed the data on a cloud based system to carry out analysis of the same. I gained experience in using industry standard tools like Apache Hadoop in processing large scale data. I tried to explore what factors in a question can improve its likelihood of being answered, and consequently, what steps a developer must take to ensure that his question gets answered as soon as possible. I also explored a perceptron based system to recommend tags to a given question. Although typically a perceptron based system is not used in such circumstances and more traditional text matching techniques such as cosine similarity (along with other tools of natural language processing) are used, nevertheless I felt that it was quite a new idea to explore.

REFERENCES

[1] Short, Logan, Christopher Wong, and David Zeng. "Tag recommendations in stackoverflow." (2014).

[2] Bajaj, Kartik, Karthik Pattabiraman, and Ali Mesbah. "Mining questions asked by web developers." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.

[3] Blei, D. M., and J. Lafferty. "Text mining: Classification, clustering, and applications." chapter Topic Models, Chapman & Hall/CRC (2009).

[4] Barua, Anton, Stephen W. Thomas, and Ahmed E. Hassan. "What are developers talking about? an analysis of topics and trends in stack overflow." Empirical Software Engineering 19.3 (2014): 619-654.

[5] Allamanis, Miltiadis, and Charles Sutton. "Why, when, and what: analyzing stack overflow questions by topic, type, and code." Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013.

[6] Anderson, Ashton, et al. "Steering user behavior with badges." Proceedings of the 22nd international conference on World Wide Web. ACM, 2013.

[7] Saha, Avigit K., Ripon K. Saha, and Kevin A. Schneider. "A discriminative model approach for suggesting tags automatically for stack overflow questions." Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013.

[8] Sinha, Vibha Singhal, Senthil Mani, and Monika Gupta. "Exploring activeness of users in QA forums." Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013.

[9] Xia, Xin, et al. "Tag recommendation in software information sites." Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on. IEEE, 2013.

[10] Wang, Shaowei, et al. "Entagrec: an enhanced tag recommendation system for software information sites." Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on. IEEE, 2014.

[11] Adamic LA, Zhang J, Bakshy E, Ackerman MS (2008) Knowledge sharing and Yahoo answers: everyone knows something. In: Proceedings of the 17th international conference on World Wide Web, pp 665–674.

[12] Al-Kofahi, Jafar M., et al. "Fuzzy set approach for automatic tagging in evolving software." Software Maintenance (ICSM), 2010 IEEE International Conference on. IEEE, 2010.

[13] Goderie, Jeffrey, et al. "Eta: Estimated time of answer predicting response time in Stack Overflow." Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on. IEEE, 2015.