Tag Prediction and Network Analysis of Stack Overflow Forum

Course project for CSE 6240: Web Search and Text Mining, Spring 2021

Neha Milind Pande
Georgia Institute of Technology
Atlanta, GA, USA
npande8@gatech.edu

Seil Kwon
Georgia Institute of Technology
Atlanta, GA, USA
skwon96@gatech.edu

Paul Horton
Georgia Institute of Technology
Atlanta, GA, USA
phorton9@gatech.edu

ABSTRACT

Tag prediction is an important topic with a wide range of applications in social media and online forums. In our research, we predicted at least one relevant tag of a particular post using information extracted from the stack overflow post. We used the following as the baseline of our experiment: 1) ACT-R inspired probabilistic model from Stanley et al. [6]. 2) Self-developed Complement Naïve Bayes model 3) Network prediction based on user post history. As a result, we achieved 58% accuracy with our Complement Naïve Bayes model by only using the title column of the dataset, which was close to 65% accuracy of the ACT-R model. We also propose a novel method of Stack Overflow tag prediction based on a weighted ensemble of a Complement Naïve Bayes model and one-step Markov Chain. After experimenting with three baseline models and two ensemble models, our first baseline method had the highest accuracy at 65%, followed by the second baseline model at 54.4%, two ensemble models at 54%, and the third baseline model at 31%. While finding the limit of Markov Chain where previous user data are insufficient, we test the possibility of applying our ensemble model by controlling its weights and also confirmed the efficiency of Complement Naive Bayes model in the application of network analysis.

KEYWORDS

Stack Overflow, tag prediction, network analysis, text mining, Complement Naive Bayes, Markov Chain

1 INTRODUCTION

For this project, we used approximately 27,000 posts from the Data Science subsection of the Stack Overflow forum to predict tags of the post. The main features used for building our networkx graph and predicting the tags were User ID, Title, Creation data, and Tags. The most frequently used tag was "machine-learning" followed by "python", "neural-network", and "deep-learning". Most of the nodes in the graph had a small degree, but there were few nodes with a large degree (greater than 90). After running TF-IDF on the Title column, "data" ranked the highest score, followed by "model" and then "learning".

Our proposed method is to use a hybrid approach which combines Complement Naive Bayes for feature-based prediction with One-step Markov Chain model. While complement Naive Bayes captures content-based information, One-step Markov Chain captures structural information. However, after experimenting with our ensemble models, we found that the single Complement Naive Bayes model achieved the closest accuracy at 54% to our first baseline. We concluded this was due to the usage of reduced training set of Markov model and insufficient user posts which could not add any value for our case. Despite the fact that Markov model was not as helpful, we predict that such ensemble model could bring more insights when previous user post data are sufficient and subjects are less diverse. Moreover, it is notable that we achieved high accuracy close to our first baseline by using Complement Naive Bayes only on the title column of the posts.

2 LITERATURE SURVEY

Morakot Choetkiertikul et al. [1] proposed two approaches to predict who will answer the questions using network analysis: feature-based prediction and social network-based prediction. They used data from Stackoverflow for prediction and achieved 44% precision, 59% recall and 49% F-measure for predicting who will answer the questions, and 12.8% for candidate identification. While applying their methods as the big framework of our research, we will add other applications and narrow down the scope on data science forum from Stackoverflow to achieve higher results. We will also build a hybrid approach that combines both knowledge from the question and the asker to retrieve more precise candidate lists. A shortcoming of the work is that it makes use of the very limited data available during the time of question creation. This makes it difficult to analyze and predict the user's contributions during the starting of the user's question.

Mark S. Ackerman et al. [7] analyzed Java Forum using social network analysis methods and tested a set of network-based ranking algorithms, including PageRank and HITS, in order to identify users with high expertise. They also found that structural information can be used for evaluating an expertise network in an online setting, and relative expertise can be automatically determined using social network-based algorithms. While applying an algorithm similar to their PageRank method in classifying different levels of users, we will go one step further and modify PageRank algorithm to predict users' next subjects as well. By testing the algorithm on a different help-seeking community, we will compare its results and gain more insights. A shortcoming of the paper is that the authors in the paper have not weighted the tradeoff between algorithm design and the use of networks. Also, combining content and structural information would be useful for building an advanced online community.

Stanley at al. [6] developed an ACT-R inspired Bayesian probabilistic model that can predict the hashtags used by the author of the post. The model was 65% accurate when tasked to predict one tag per post on average. This was achieved by choosing the tag that has the highest log odds of being correct, given the tag's prior log odds of occurrence and adjusting for the log likelihood ratio of the words in the post being associated with the tag. However, by normalizing the top activations for all posts, the model could not differentiate between a post where the top-predicted tag is certain,

and a post where the top-predicted tag is questionable. This is a shortcoming of the work. In addition, post author's past tagging behavior was excluded from their model predictor. In our experiment, we will include other predictors including the post author's past tags to improve our predictability.

3 DATASET DESCRIPTION AND ANALYSIS

3.1 Data Preparation

1.1.1 Source. For this project, we are using data from the Data Science subsection of the Stack Overflow forum. Two tables were retrieved using SQL queries, which are publicly accessible from https://data.stackexchange.com/datascience/query/edit/1363812. The main table (Posts table) that we used contains information about each of the 59,292 posts. It has features such as ID, Post type ID, Accepted answer ID, Creation date, Deletion date, Score, View count, Body, Tags, Owner user ID, Title, and Parent ID. The other table (Users table) includes user data and links between posts. The Users table has information about 95,701 users and includes the account was created, and how much reputation a user has gained through their posts. It has features such as ID, Reputation, Creation date, Display name, Views, Up votes, and Down votes.

1.1.2 Data preprocessing. We have performed the following tasks for preprocessing the data:

- Selected only rows with non-empty tags
- Removed rows without Owner user ID values
- Converted the Creation date column into proper datetime format
- Used regular expression to extract tags
- Removed punctuation
- Converted text to lowercase
- Removed stopwords

These tasks were necessary for our analysis because we had to use the User ID, Title, Creation data, and Tags column for building our networkx graph and predicting the tags. This dataset is sufficient for our project since it has adequate number of posts and captures all the information required and necessary for achieving the goal of our project. The dataset has been finalized and fixed.

3.2 Raw Data Statistics

1.2.1 Properties of the dataset. After preprocessing, the number of rows(posts) in our Posts table were reduced to 27,378. After building an undirected graph with weight on each tag, we obtained a networkx graph with 622 nodes, 15,754 edges, with average degree of 50.65. The date range of the dataset covered the period from May 2014 to February 2021.

3.3 Data Analysis

1.3.1 Data exploration. As it can be seen from Figure 1, "machine-learning" was the most frequently used tag followed by "python", "neural-network", and "deep-learning".

While most of the nodes had a small degree, there were few nodes with large degree that ranged from around 90 to 160 (Figure 2).

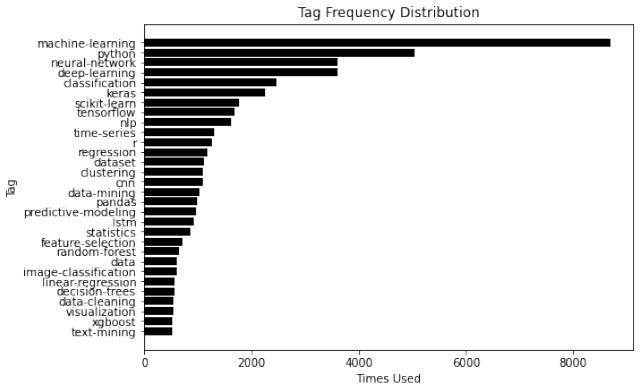

Figure 1: Tag Frequency Distribution

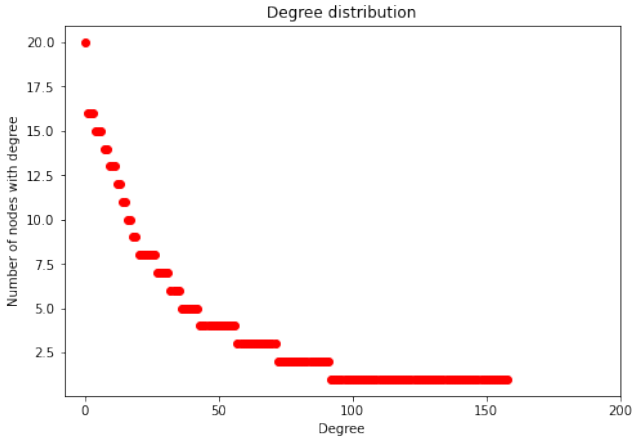

Figure 2: Degree Distribution

After running TF-IDF on the Title column, "data" ranked the highest score with 728.9, followed by "model" with 481.0 and "learning" with 442.4.

4 EXPERIMENT SETTING AND BASELINES

4.1 Experimental Settings

We initially split the whole dataset into training data (80%) and testing data (20%). Grid Search CV was used for performing hyperparameter tuning on the models. The best model was chosen based on which model had the best prediction accuracy. The chosen model was re-trained on the training dataset with selected hyperparameters. It was then evaluated on the test dataset. The task is to predict at least one relevant tag of a particular post using information extracted from related stack overflow posts.

Google Colab platform was used with default RAM and CPU as for the baseline models. Python Environment was used for this project. Numpy and pandas libraries were used for matrix operations and data cleaning, sklearn and nltk for all machine learning tasks and handling textual data, and networkx library was used for creating and manipulating networks.

Accuracy is used as the main indicator to compare between different models. It is appropriate since accuracy was used in the baseline models developed earlier and because this is a multi-class problem where even the naive approach of classifying all observations as the most common tag does not perform well. Accuracy also aligns with the business implications related to tag prediction.

4.2 Baseline results and discussions

Posts can contain multiple tags. So, we will compare baseline models based on accuracy of the prediction being in the list of tags used. This metric will also be referred to as the at-least one accuracy. We have selected three baselines to establish performance benchmarks for predicting tags on Stack Overflow posts:

1) Stanley et al.[6] using the ACT-R inspired probabilistic model.
2) Self-developed Complement Naïve Bayes model.
3) Network prediction based on user post history.

4.2.1 *Baseline 1.*

Stanley et al. [6] takes a Bayesian-based approach for tag prediction using both the words in the title and the body of a Stack Overflow post. There is a component of the model that uses the prior log odds of a post tag which is based on the frequency of use throughout all posts. Two other parts of the activation function consist of the weighted conditional probability of having a tag based on the encountered words in the title and the body. The final component is an offset that subtracts the mean of the top five predictions to scale the output. Instead of removing stop words, they calculate the predictive ability of all words and assign a weighted significance depending on that predictability. They used a bootstrapped logistic regression model to find optimal settings for all initialized hyper-parameters such as weight, offset, and prior log odds. The ACT-R code repository can be found at http://actr.psy.cmu.edu/actr7.x/actr7.x.zip. Bayesian machine learning approach is useful for predicting outcomes on large datasets with several classes and prior observations. Further, the accuracy obtained in Stanley et al. [6] using this approach is higher than other similar works. Therefore, this baseline is suitable for this problem.

4.2.2 *Baseline 2.*

Our internal baseline is a Complement Naïve-Bayes model implemented using the sklearn python module that uses only the words in the post title. As proposed by Rennie et al. [4], this method minimizes the inverse likelihood that a post does not belong to a class and therefore it is suited for unbalanced training data. Therefore, this baseline is suitable for this problem.

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}} \tag{1}$$

$$w_{ci} = \log \hat{\theta}_{ci} \tag{2}$$

$$w_{ci} = \frac{w_{ci}}{\sum_j |w_{cj}|} \tag{3}$$

$$\hat{c} = \arg\min_c \sum_i t_i w_{ci} \tag{4}$$

We removed punctuation with regex and stop words with the nltk python module. Then, converted each word to lowercase to standardize its form. Next, we converted the list of words to the Bag of Words model and then to the Text Frequency Inverse Document Frequency. This latter representation scores words higher if they appear less frequently in the number of overall documents. The output of this model is the probability that a post belongs to a tag. We selected the tag with the highest probability as the classification. We trained this model on 80% of the posts and tested on the remaining 20%. The repository for this algorithm can be found at https://github.com/scikit-learn/scikit-learn/blob/main/doc/modules/naive_bayes.rs

4.2.3 *Baseline 3.*

Our final model uses a one-step Markov Chain to predict the tags of a next post based on the user's post history. To do this, we had to use a Leave-One-Out method to build our test and train sets. We used the most recent post for a user as the testing set and all previous posts as the training set. We built a network based on the training set where each node was a tag and there is a weighted edge between them based on their joint usage frequency. The starting point for each post was determined by the user's weighted tag history. If a user did not have any previous posts, the starting point was distributed equally among all tags. This resulted in the node with the highest in-degree being selected which was "machine-learning". Markov chain is useful for predicting the next state based on the only current state. When used in a graph, it is able to capture all the structural information of the network. The other baselines do not capture this information. Therefore, this baseline is suitable for this problem.

4.2.4 *Baseline result and discussion.*

The ACT-R baseline was the best performing at 65% which was slightly above our internal baseline using Complement Naïve Bayes at 58% and significantly better than our network model at 31%. We are not surprised to see these results since they reflect the amount of relevant information that is being used to train the models.

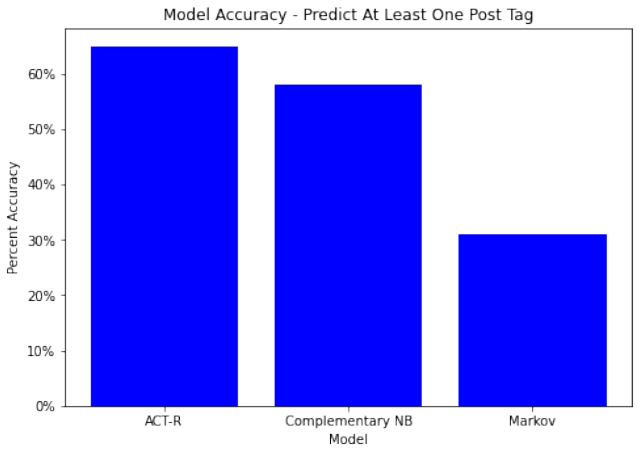

Figure 3: Baseline Performance Comparison

For Stanley et al. [6], they incorporated both the post title and body. There may be relevant information in the body of a post that is not captured in the title which our model does not utilize. They also did not remove stop words from their descriptions instead weighting them relative to a word's entropy or predictability to any tag. Entropy, as shown in equation 5, uses conditional probability

to determine how well a word can predict a tag. Words that appear frequently with different tags will not have much predictive value and will be weighted minimally.

$$Entropy = -\sum_{i=1}^{N} p(tag|words) * \log(p(tag|words)) \quad (5)$$

The Markov Chain performs worst likely because there are many users without any post history. Figure 4 shows that users are primarily concentrated at a low post count which means there is not much relevant historical information for the model to use when predicting the next post. It is also plausible that the relationship between previous posts and the next post is not strong enough to be a good predictor.

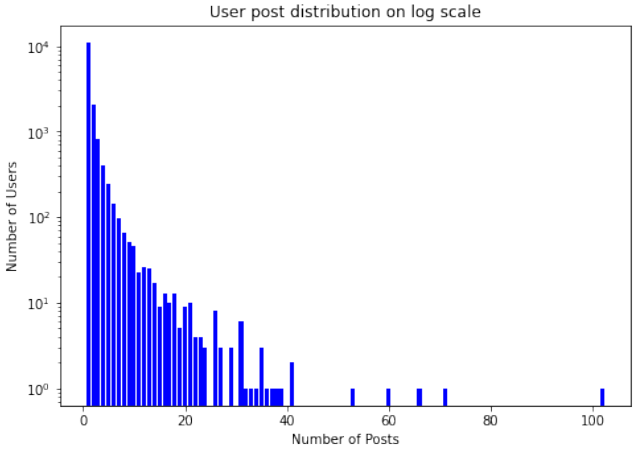

Figure 4: Histogram of post count distribution

Our baselines have similar performance as other models used on the Stack Overflow dataset. Saini et al. [5] created a Support Vector Machine model which had an at-least one accuracy of 65%. No state-of-the-art models have been applied to the Stack Overflow dataset so we will not compare these baseline results to any from novel deep learning approaches.

5 PROPOSED METHOD

From our baselines, we realized that Complement Naive Bayes performs better than other machine learning algorithms on features gathered from Stack Overflow forum. However, it only leverages content-based information. Another baseline one-step Markov chain model uses only information gathered from network graph. Both these independently do not capture complete information about the Stack Overflow forum. So, we proposed a unique hybrid approach which weighs and combines knowledge from feature-based and social-network based models. The novelty of this method is that it both combines content and structural information from two different models. Most of the other methods only use content-based or structural information. With this information, we hypothesizec that better predictions can be made than the existing baselines.

In our hybrid approach, Complement Naïve Bayes was used for feature-based prediction. One-step Markov chain model was used for social-network based prediction. The title and the tags of each post in the preprocessed dataset were extracted. We applied tf-idf transformation on the extracted data. This data was then fitted into the Complement Naive Bayes model. The model was imported from the sklearn library. Using the model, a list of predicted tags for every user were generated. The networkx graph was built using the information in the Posts table to predict the tags of the next post. The creation date field present in the post was used to determine previous tags of a particular user. Previous tags were then used to create a one-step markov chain. Predictions for every user were generated based on the one-step markov model. If a user had post history, these predictions from both models were combined. If the user had no post history, then only the prediction from Complement Naive Bayes was used.

The are shortcomings of the baselines which the proposed method overcomes. The proposed method combines the knowledge gathered from two different baselines methods (Complement Naive Bayes and one-step Markov chain). Complement Naive Bayes captures content-based information and one-step Markov chain captures structural information. The main idea is that a user with the higher number of posts will have more accurate predictions. Such an approach has not been tried before. Further, it works very well on unbalanced training data sets. For evaluation, accuracy was the main metric used similar to our baselines.

6 EXPERIMENTS AND RESULTS

The Naive Bayes and Markov models have output in probabilities per class but we still needed a method for aggregating our results. We evaluated two different logical ensemble processes: excluding the Markov model if the user does not have a post history and using a confidence threshold to find best predictions from each model. Additionally, we chose an approacht that Moreira et al. [2] refer to as forward without exploration for determining model integration. With this technique, we will start with our individual model with highest accuracy and add weight towards our second model. We will continue adding weight until the accuracy starts decreasing.

The first method excluded the one-step Markov model for any users without any post history. Without any user history, the Markov model defaults to the class with the highest prior. This results in an at least one accuracy of 31% which is far below our baseline. The concept behind this method is that, by excluding the Markov model in these cases, we will eliminate using the inaccurate predictions in the ensemble. Our results for this method, seen in Figure 5, show a continuous decrease in accuracy for any weight above zero for the one step Markov model.

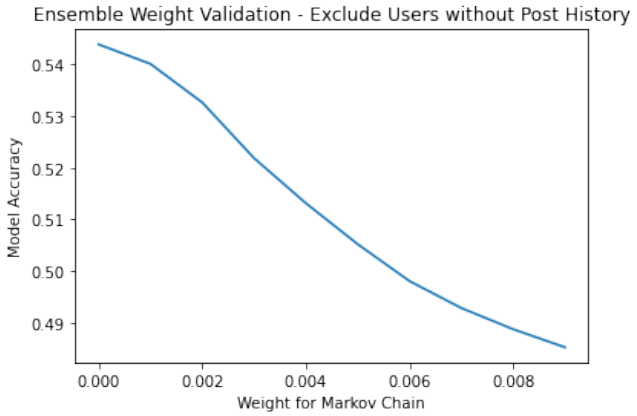

Figure 5: Ensemble model accuracy drops when increasing weight for Markov model

The second method uses a confidence metric based on the ratio of probability from the most likely to second most likely class. The concept was derived from Mikolajczyk et al. [3] who described using ratios of distances as an indicator of prediction quality. There are no established criteria for what make a good threshold so we evaluated values for both models.

The accuracy for the Markov model, seen in Figure 6, does not show the anticipated relationship with the confidence threshold. We expected to see the accuracy increase as we limited our predictions to those with higher confidences. This result indicates a lack of predictive ability for our model independent of other factors.

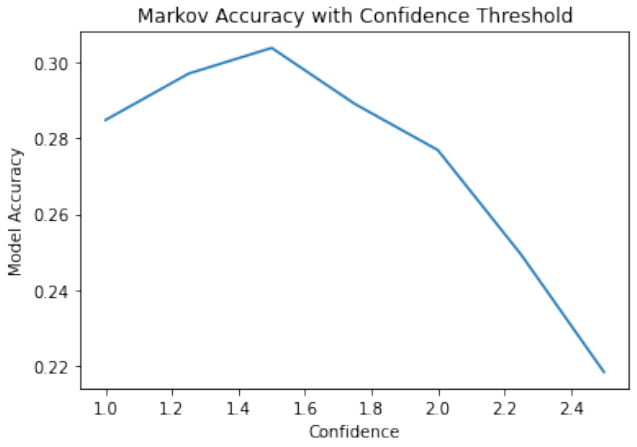

Figure 6: Markov model accuracy when predictions restricted to ones at confidence threshold

The results were opposite when we applied a confidence threshold to the complement Naive Bayes model, shown in Figure 7. There is a gradual improvement in accuracy when we use only more confident predictions. Knowing that implementing a confidence ratio could improve the accuracy for the complement Naive Bayes model, we used cross-validation to find the threshold that had the most impact on our ensemble model.

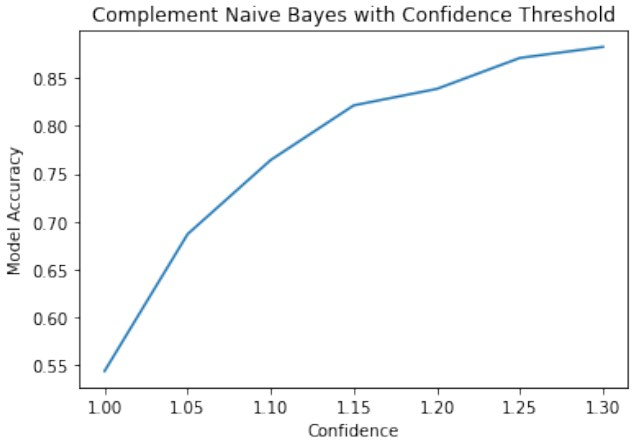

Figure 7: Complement Naive Bayes model accuracy as a function of prediction confidence threshold

With our optimal confidence threshold, the remaining parameter to tune was the weight applied to the prediction from each model. We used cross-validation again to identify the relationship this variable has with the model accuracy. The results in Figure 8 indicate that adding the predictions of the one step Markov chain at any weight reduce the accuracy of the ensemble model. The complement Naive Bayes model is more accurate on its own.

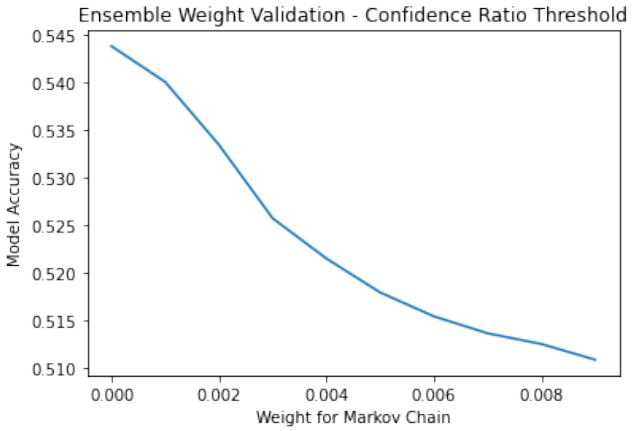

Figure 8: Ensemble model accuracy as a function of the weight of Markov predictions

7 CONCLUSION

Our ensemble method did not have any better performance than our single model which even fell short of our baseline from Stanley et al. [6]. Part of the reason is because we had to retrain our complement Naive Bayes model without each user's most recent post to align the

Method	Test Accuracy
Baseline 1	65%
Baseline 2 - Complement Naive Bayes	54.4%
Baseline 3 - One Step Markov	31.0%
Ensemble - No Post History (Weight 0.01)	54.0%
Ensemble - Confidence Threshold (Weight 0.01)	54.0%

Table 1: Comparison of Test Accuracy for the Different Methods

observations with our Markov model. This reduced our training set to roughly 50% of all observations instead of the 80% we trained on in earlier work. Additionally, we found the Markov model does not add any predictive value beyond what the single complement Naive Bayes offers. We hypothesize that this is because the complexity of the data science field with a diversity of subjects that have distant connections. The frequency of tag usage is not evenly distributed so the Markov model is also biased towards the most commonly used terms.

8 CONTRIBUTION

All members contributed equally to the research and work for this project.

REFERENCES

[1] Morakot Choetkiertikul, Daniel Avery, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. 2015. Who will answer my question on Stack Overflow?. In *2015 24th Australasian Software Engineering Conference*. IEEE, 155–164.

[2] João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. 2012. Ensemble Approaches for Regression: A Survey. *ACM Comput. Surv.* 45, 1, Article 10 (Dec. 2012), 40 pages. https://doi.org/10.1145/2379776.2379786

[3] K. Mikolajczyk and C. Schmid. 2005. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 10 (2005), 1615–1630. https://doi.org/10.1109/TPAMI.2005.188

[4] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*. 616–623.

[5] Taniya Saini and Sachin Tripathi. 2018. Predicting tags for stack overflow questions using different classifiers. In *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 1–5.

[6] Clayton Stanley and Michael D Byrne. 2013. Predicting tags for stackoverflow posts. In *Proceedings of ICCM*, Vol. 2013.

[7] Jun Zhang, Mark S Ackerman, and Lada Adamic. 2007. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*. 221–230.