University of California, Davis

STA 160 - Practice in Statistical Data Science

Routing Collaborative Activity in Community Answering Forums

A look into the Stack Overflow Network

Authors
Dan Maksimovich
Esmond Chu
Seungmi Oh
John Nguyen

Instructors
Prof. Duncan Temple Lang
Nick Ulle

June 12, 2018

Git Repository: https://github.com/MaksimDan/160-Stackoverflow
Website: http://maksimdan.github.io/160-Stackoverflow

Contents

1 Executive Summary

Community question and answer services (CQA) like the Stack Exchange platform enable users to crowd source knowledge in the form of questions and answers. Thousands of new questions are posted each day, but approximately 22% of non-closed questions remain unanswered, and 72% do not receive an accepted answer. With a large volume of incomplete answers, new questions generated everyday and considerable interest in users to become able to identify these questions [3, 4], it can be useful to route questions to potential answers at a user's convenience and interest. In this paper, we present a user-question routing scheme that is focused on the context of unanswered questions. Our experiments show that by selectively targeting 15% of a user population, we can expect to capture 90% of the activities in unanswered questions in the form of answering, commenting or editing. We also show that this can be achieved through a simple selection of engineered features.

2 Introduction

Stack Overflow is the flagship of the Stack Exchange platform, that focuses solely on specific questions related to programming. In this online community, developers share their skills, build their careers, and collaborate together to learn. Since 2009, the question and answer (Q&A) platform has accumulated millions of questions relating to virtually every aspect of programming. Users can ask, answer or vote on the quality of posts and responses. Over time, users can earn reputation points and badges for their valued contributions, which unlocks more administrative privileges for the user. The platform as a result, is almost completely self-regulated by the community. Unlike other Q&A alternatives like Yahoo Answers, questions can be closed, or put on hold, in order to prevent the flow of low quality or irrelevant questions to develop.

Work in 2012 [3] has shown that CQA sites like Stack Overflow and Quora consist of a set of dedicated domain experts who are interested in adding lasting value to a wider audience and providing feedback to satisfy a user's query. Our aim is to build a program that utilizes a users availability, expertise and interest to recommend answers, editors, or commenters to a question. In addition to reducing the response time for questions, we aim to direct respondents to questions where they can make a valuable contribution.

3 Stack Overflow Data Set

Data from Stack Overflow is publicly available on archive.org [2]. We decided to subset on Java related questions in the months of January to June of the year 2012. This decision was made for two key reasons. First, due to Java's versatility, the language contains a large ecosystem of questions that already comprises a large subset of programming related topics. Secondly, due to the questions being 6 years old at the time of query, responses to these questions have stabilized in terms of obtaining new responses.

4 Recommender System

4.1 Loss Function

We define our loss as the inverse ratio of observed activities within a chosen threshold (t^*) of the user population. In other words, given a limited number of users to choose from, we are interested in maximizing the identification of individuals who have responded to a question. This loss function reflects the philosophy that there are many users who are

equally qualified in answering a question. We are more interested in adequately identifying these users, than quantifying a measurable metric that distinguishes one user better than the other.

Our choice of threshold selection is ambiguous because it requires taking into account information such as the available pool of unanswered questions, the time available to answer these questions, and the rate of incoming new questions in order to be within reason of not over spamming the user base. We decided that $t^* = .15$ or 15% of the user population was a fair approximation of this.

One advantage of this loss function is that it can be tweaked to directly target different user populations on the basis of an activity. For example, a variable set of weights can be optimized to exclusively target commenters, answers, or editors to a question by application of a contrast vector. Our current implementation weights each type of user activity (comment, answer, or edit) equally $[1/3, 1/3, 1/3]$, but with different weight vector for example, can be optimized to ignore commentators and attract twice the amount of editors than answerers $[0, 1/3, 2/3]$.

The mathematical equation for the loss function, along with its caveats can be found in the appendix of this report.

4.2 Model Features

4.2.1 Indicator Variables

Some users may have not seen a question simply because they are not available at the time the question posted. One option is to use the context of a post and information about the user to try and distinguish between some individuals who have seen the question and those that have not. However, we do not have the data to 100%accurately distinguish these users. Indicator variables are our attempt to eliminate those that we know for sure did not see the question.

Utilizing a users history, our model applies two features as indicators. The first identifies whether a user created an account after the question was posted, and the second identifies whether the user was inactive before a question was made. On average, this has helped eliminate 31% of the user population per question ($\pm$9% as standard deviation).

4.2.2 User Availability (Feature 1)

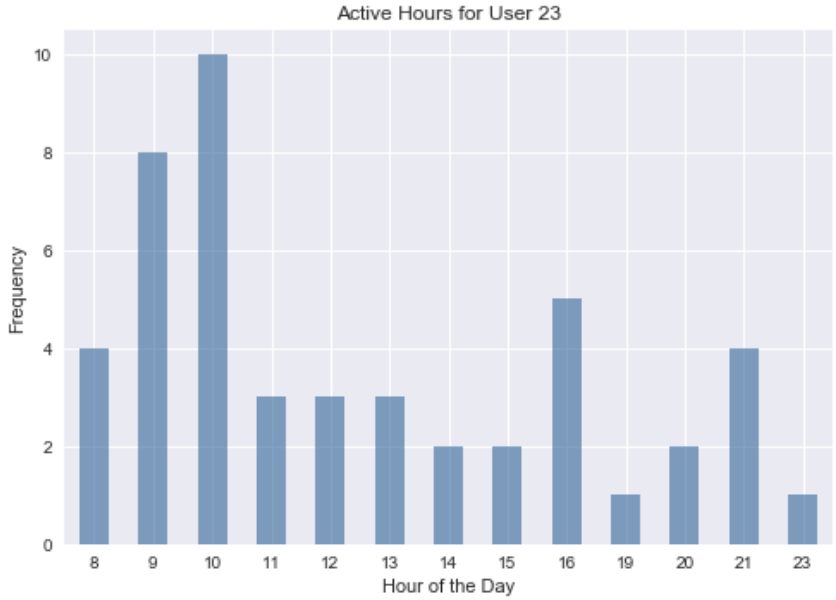
As shown in figure 1, we noticed that users generally stick to an hourly schedule to respond to questions. In particular, of the questions that eventually receive a response, 50% will do so within 18 minutes and 95% within the hour. Therefore, matching a question to a user who is more likely to be active when the question is posted would increase model accuracy.

One flaw with this feature, is that it does not account for responses that are later updated due to being incorrect or outdated. The majority of questions however, tend to receive a response within minutes of being posted. Ergo, given the activity history of a user, we define feature one as the probability that a user will respond to a post.

4.2.3 User Expertise (Feature 2 to 4)

Users who tend to be more experienced with a particular tag, tend to comment or answer more questions with that tag. We naively approximate of "user expertise" to be proportional to the number of times a user as contributed to a post with the tag. We define three separate features associated with "user expertise": the frequency of comments, questions and answers in relation to the tags of a question.

Figure 1: Hourly activity distribution of a randomly selected user



4.2.4 User Tag Similarity Expertise (Features 5 to 7)

Tags can be related to one another. For example, a user might not directly identify with the tag eclipse3.4 (a Java IDE), but they still might be able to identify with it because they answered questions related to an earlier version of eclipse.

We define the cosine similarity between two tags as

$$\text{cosine}(tag1, tag2) = \frac{\text{cooccur}(tag1, tag2)}{\sqrt{\#(tag1) * \#(tag2)}}$$

To avoid overlap with features 2-4, the next three features only considers the tags within a post that a user is not directly familiar with (like in the example above). To compute these features, the idea is similar to the features described previously.

Given a user and a question, features 5-7 aggregates the tag frequencies (with respect to comments, answers, and questions) that the user is directly experienced in and relates those tags that the user is indirectly familiar with in the question.

4.2.5 Question Familiarity (Features 8 to 11)

The final four features of our model takes into account the content of a question, and relates it to the historical record of a user. Our goal here was to try to loosely understand the query from a question, and use it to quantify how familiar a user might be in terms of begin able to contribute in some way. This feature goes somewhat hand-in-hand with user expertise, but instead of raw frequency, we relied on TF-IDF feature matrices.

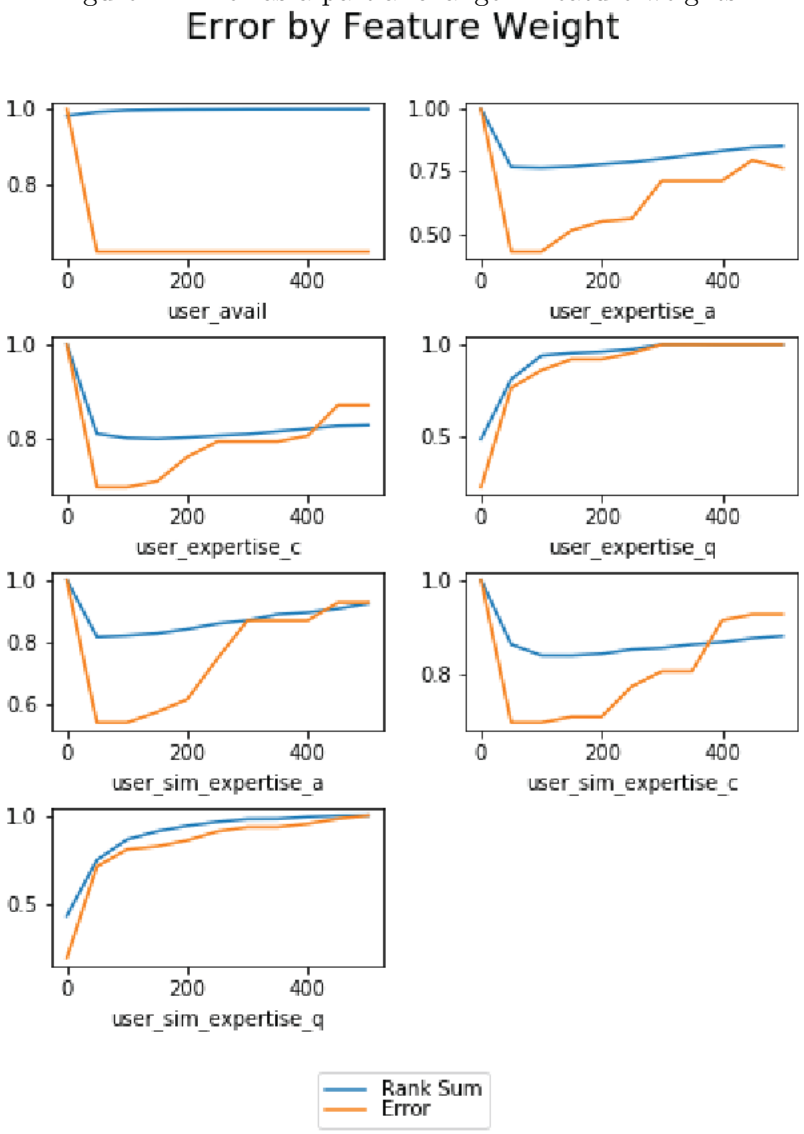
A summary of the mathematical details of each feature can be found in the appendix of this report.

4.3 Fitting the Model

4.3.1 Technique

Due to the nature of user ranks in our loss function, our objective function is nonlinear and the derivatives cannot be known. In addition, because computing a single error for a random sample of 30 questions can take up to an hour of processing time on a modern CPU, our procedure was very limited in terms of the feature space and density that could be explored.

Figure 2: Error as a partial change in feature weights



Given the inefficient time complexity of our program, we narrowed down our feature space using the information modeled in figure 2. This plot describes each feature error depending on the value of the feature's weight. This gave us a general idea of what happens when a target weight was kept as variable, while all other weights were kept constant. The subtle irregularities are present because a random sample of questions was taken on a per error basis. The sum of the ranks were additionally shown because we wanted to show how it correlates with the loss function error.

We found the following weights to be optimal using exhaustive search in a very sparse and limited feature space. Take these results with a grain of salt. Interestingly two

features in similar contexts were found to have zeroed out weights (features 4 and 7). We suspect this to be the case because users who ask questions are there to gain information that they do not have, and so contributing to posts with the information they have gained would be duplicated information (which does not happen very often).

Sub-optimal Weights:

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
50	100	130	0	75	60	0	60	75	80	29

5 Results

The weights discussed previously were obtained of a training set of consisting of 1000 questions. The results that will be discussed in this section was the application of these weights on a test set of 800 questions.

5.1 Predicted Ranks

Our first objective was to try and identify where and in what context have users been ranked. In other words, we needed a way to illustrate an overall summary of the classifications. This information is captured in figure 3.

Figure 3: Position of user ranks who were active on a per question basis

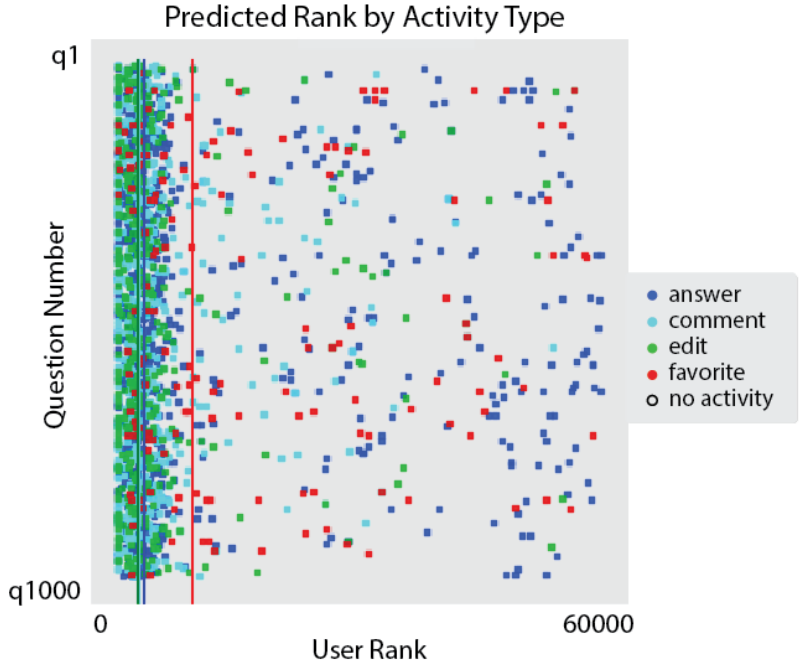


Figure 3 illustrates the overall performance of our model on a sample of 1000 questions. The x-axis is the user rank as determined by the recommender system and the y-axis denotes each questions from 1 to 1000 respectively. Finally, the vertical lines represent the average rank with respect to the activity type.

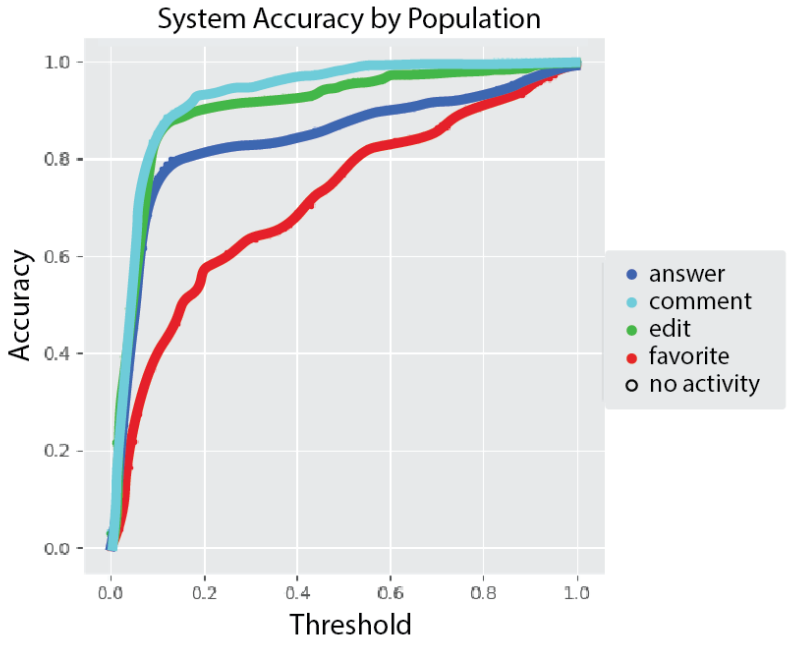
In the context of our project, a highly recommended individual has a small rank, where as a unrecommended individual has a large rank. Therefore a performant system would be illustrative of active users who are clustered towards the left, and inactive users who are clustered towards the right. For the most part, this seems to be the case. We also observe a heavy cluster of activities towards the left, and a sparsely uniform distribution of users to the right of that cluster, rather than a gradually fading one. This is to be expected

because our loss function is not designed to penalize "highly incorrect" misclassifications anymore than borderline incorrect ones. The appearance as to why the average ranks for user favorites are unlike the rest will be analyzed in the next plot.

5.2 Accuracy by Threshold

Because our loss function is dependent on selecting a threshold, we needed a way to understand how well the recommender performed as a greater pool of users were accepted. In other words, how many users did the recommender system get right in the top 10, or top 100 recommended users? More generally, given some threshold t^*, we were interested in the ratio of observed user activities under t^* to the total observed user activities.

Figure 4: Error as a function of t^*, which denotes the available proportion of a user population



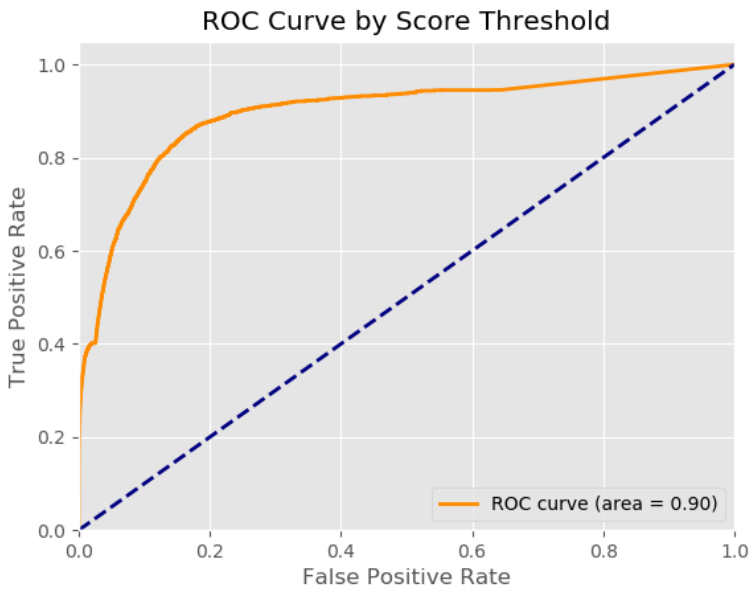
Although favorites were not not formally considered into our loss function, we have included these activities into figure 4 because we wanted to illustrate the ambiguity that comes along with favoriting an activity (anyone can favorite, and do nothing else) in a feature framework that on a high level is designed by domain-expertise.

5.3 ROC Curve

Next, we were interested in formally evaluating the results of classifications. Unlike the previous plot, we put combined all level of activities (comments, answers, and edits) into a single binary classification (active or nonactive).

Using user scores as a classification metric, our true positive rate is defined as the ratio of a over a+b, where a is the number of correctly labeled active users and b is the number of incorrectly labeled inactive users for all questions. Steepness for early thresholds, is an critical property that was achieved within the ROC curve. This is because we are looking to maximize the true positive rate while minimizing the false positive rate without having to resort to high thresholds, or scores. Doing so would otherwise be akin to contacting a needlessly large majority of the user population to look into a question. More information about how the ROC curve was generated will be in the appendix.

Figure 5: ROC evaluation diagnostic for the models binary classification of inactive and inactive users per question



5.4 System Entropy

As a final model evaluation metric, we were interested to see if the recommender system held any bias by rank. In other words, are the same users recommended over and over again? To do this, we observed each column (or rank) in the recommended user matrix. In this matrix, each row maps to a question, and each column represents the associated rank of a user. Figure 6 measures the level of Shannon's entropy $H(x)$, per column of users.

$$H(x) = -\sum_{x_i} p(x_i) \log_2 p(x_i)$$

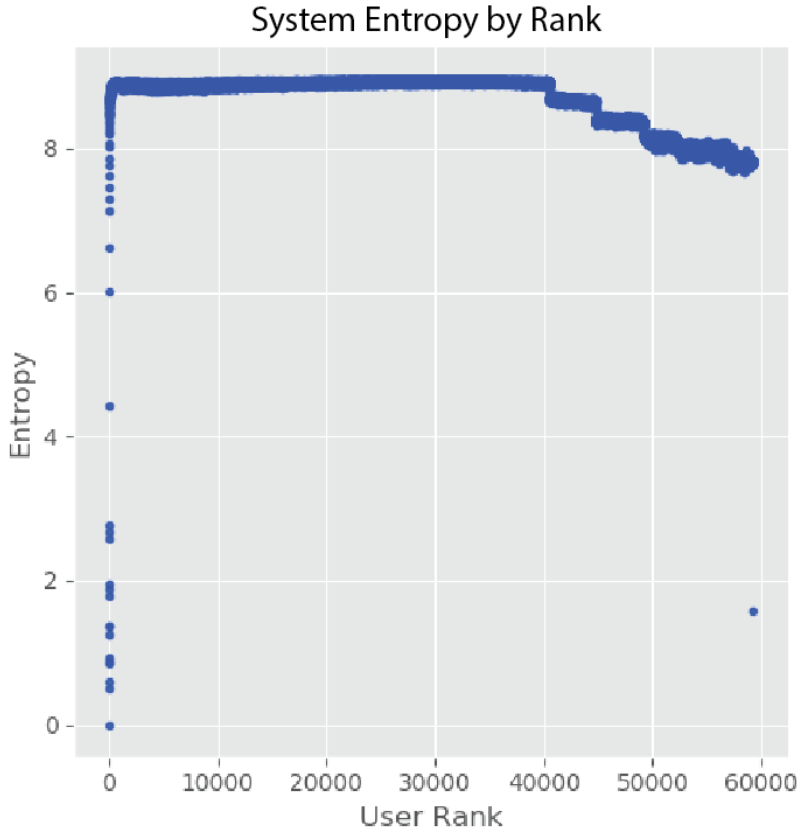
Figure 6 illustrates that there is heavy bias in recommending the same users at early ranks and less so towards the end. With an exception to early ranks, entropy reaches the maximum level rapidly, suggesting that the features sourced from each question identify uniquely with a variety of different users. In other words, the variability between users as measured from the features, resonates uniquely with the content of the question and the history of the user.

5.5 Conclusions

With a threshold of $t^* = .15$ (which accounts for 10063 out of 59196 total users), the system is able to capture 91.5416% of observed activities for all questions. Breaking this down even further, from the identified 4292 total user activities derived from 1000 random questions, a threshold of .15 would correctly identify 3928 observed activities and misidentify 364. The remaining 5771 users, did not contribute to these 1000 questions in terms of comments, answering, or editing.

One question that is worth considering is whether or not it is fair to contact the users who have no activity to a question but were classified as such (false positives). One argument is that because all users were ranked, the users who were correctly classified are a best approximated by those who did not. In other words, if we were to cluster the

Figure 6: Shannon's entropy per column rank in the recommended users matrix, where each row is a question and each column designates the rank in ascending order



users under $t^* \leq .15$, using the same features to classify them, the users who actively contributed and those that did not would identify into the same cluster group.

6 Residual Analysis

6.1 Missed Users

Our goal in this final section is to identify why active users between $t^* < x < t^{**}$ are ranked as they are. In other words, we want to understand the common characteristics shared between the users within this threshold in order to possibly engineer more robust features in the system.

To study the characteristics between the classified group $0 < x < t^*$ (A), against misclassified user group $t^{**} < x < 60000$ (B), we look over features that dominated the overall scores of the user with respect to the kind of user activity. Moreover, we wanted to identify how the distributions in group B compared when lined up against group A.

In figure 8, what is immediately obvious is that there are 3 key features that dominate the overall feature score. These are features user availability, user expertise (by answers), and question familiarity (by question title). This plot helps us identity which features are important in terms of accurate classification in our model. Interestingly, each of these features will uniquely map to their own respective groups.

Another observation that is common between both groups is that a exponentially decaying feature mean by ascending rank. This is to be expected, as a the system naturally accepts a few individuals who are a good match for a particular question, and more

Figure 7: Position of user ranks who were active on a per question basis, with emphasis on three groups: $x \leq t^*$, $t^* < x < t^*$ and $\geq t^{**}$

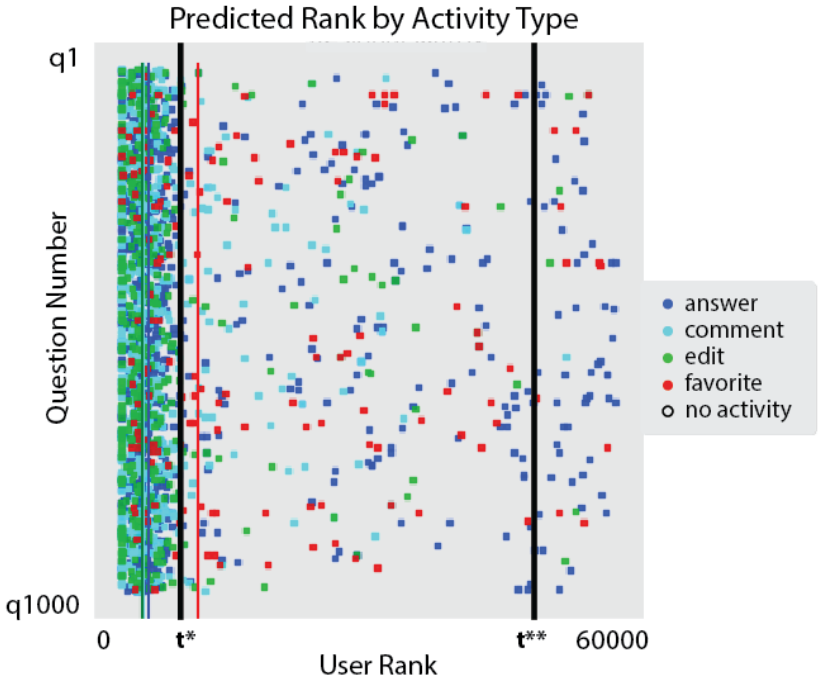
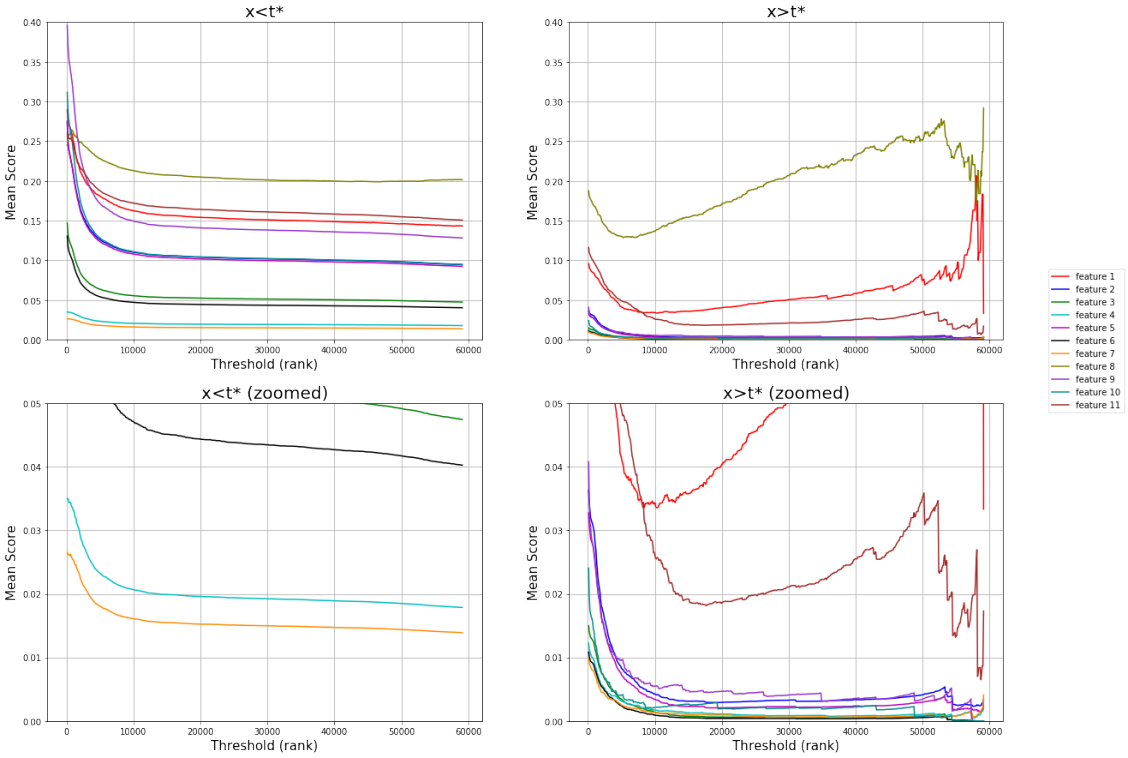


Figure 8: How mean feature scores change as a function of rank



individuals who are more or less are on the same playing field. In other words, this plot highlights the cluster of individuals spread across $x < t^*$ as shown in figure 7, and more

or less uniform spread of individuals spread across $x > t^*$. Another conclusion is titles from questions are more powerful indicators in terms being capable to summarizing a user coherency to a question. This makes sense because questions naturally produce the most important elements of a text.

7 Future Directions

As stated before, the information gained in each features have been chosen to be limited to the context of an unanswered question. However, if there is interest to also target question with answers, such as questions without an accepted answer, we recommend taking in account user compatibility as a feature. This feature would measure the degree of similarity between the users who have answered against all remaining users. Similarly, user co-occurrences between questions could also be taken into account.

Defining user expertise to be proportional to the frequency of posts as we have done is a naive way of measuring expertise in a topical area. To improve this, we recommend further subsetting on features such as length of response, or the number of upvotes associated under a particular post.

For the first feature - user availability, rather the observing the global history of activities for a given user as we have done, in a real system, we recommend observing user activity for the last few months, in order to maintain dynamic scheduling.

Spectral Clustering in [1] has been proven to be very effective. We recommend clustering together questions, and then use a voting algorithm to correct for user rankings. Note that this will affect entropy by rank, but with a weight parameter can be proven to always reduce or maintain the same level or error. For reference a sample implementation of the voting algorithm has been included under source/misc in the GitHub repository.

As discussed earlier, a large disadvantage is that user history logs are not publicly available. Within real world infrastructure, user history logs can be valuable in narrowing down an identifiable set of users who have been shown the question. Adopting this information into our system would make the results more concrete.

Using Mean Average Precision and Average Precision as evaluation metric as versus to our current error function. As we increase our threshold the precision would increase but recall would decrease.

8 Afterthoughts

During the first few weeks, even though progress seemed to be smooth, the reality was that we were building unnecessary work that ended up not being incorporated into the final program. As we learned more about our data, our goals and motivations became more refined, and so a lot of what we felt was solid progression actually ended up becoming unused.

Together we learned about information preprocessing and how to select and weight features at scale using numerical optimization techniques.It was generally accepted that reading code is more difficult than writing it, and so in addition to the scale our project, we made the extra push as group to document our code moving forward. Additionally, in several weeks we had focused on the computational implementations, missing statistical basis which leads to a huge lack of our model. However, after having conversations with Duncan we realized the importance of statistical setups and were able to improve our model in a more persuasive way. Lastly, as for project management, we used Slack and Trello, and found these tools to be very effective to discuss and distribute tasks.

9 Appendix

9.1 Data Schema

Below, we briefly summarize the database schema relevant to our goals.

1. Comments: Users have the option to comment under any post. This facilitates a discussion or clarification under a new thread.

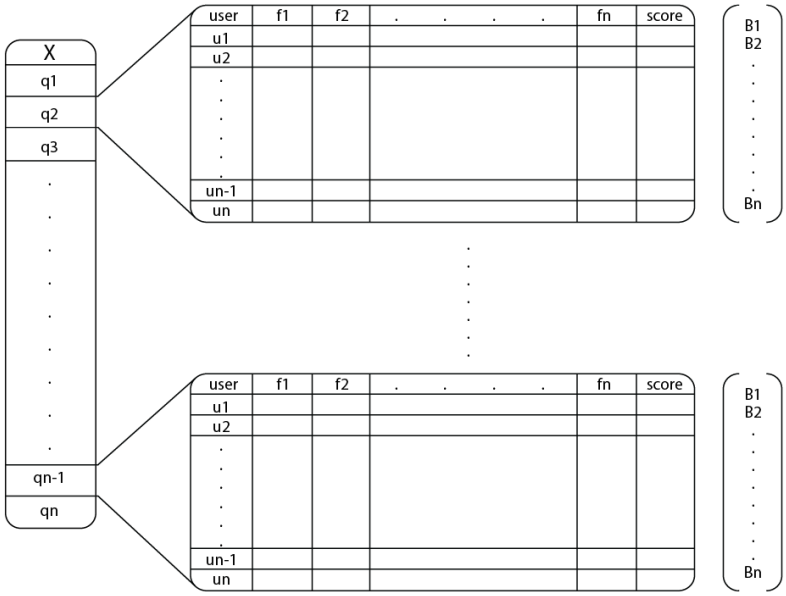
2. Posts: Contain information of all posts, and its associated meta data such as whether or not is was closed, who posted it, and the content of the post.

3. Users: Contains profile information such as their reputation, id, age, description, total number of upvotes and downvotes.

9.2 System Architecture

Our recommender system architecture is structured on three components: the recommender, features and weights. For each question, a feature matrix is computed using the features discussed in the previous section.

Figure 9: Basic system architecture: for each question a feature matrix is built by extracting details from question and user context



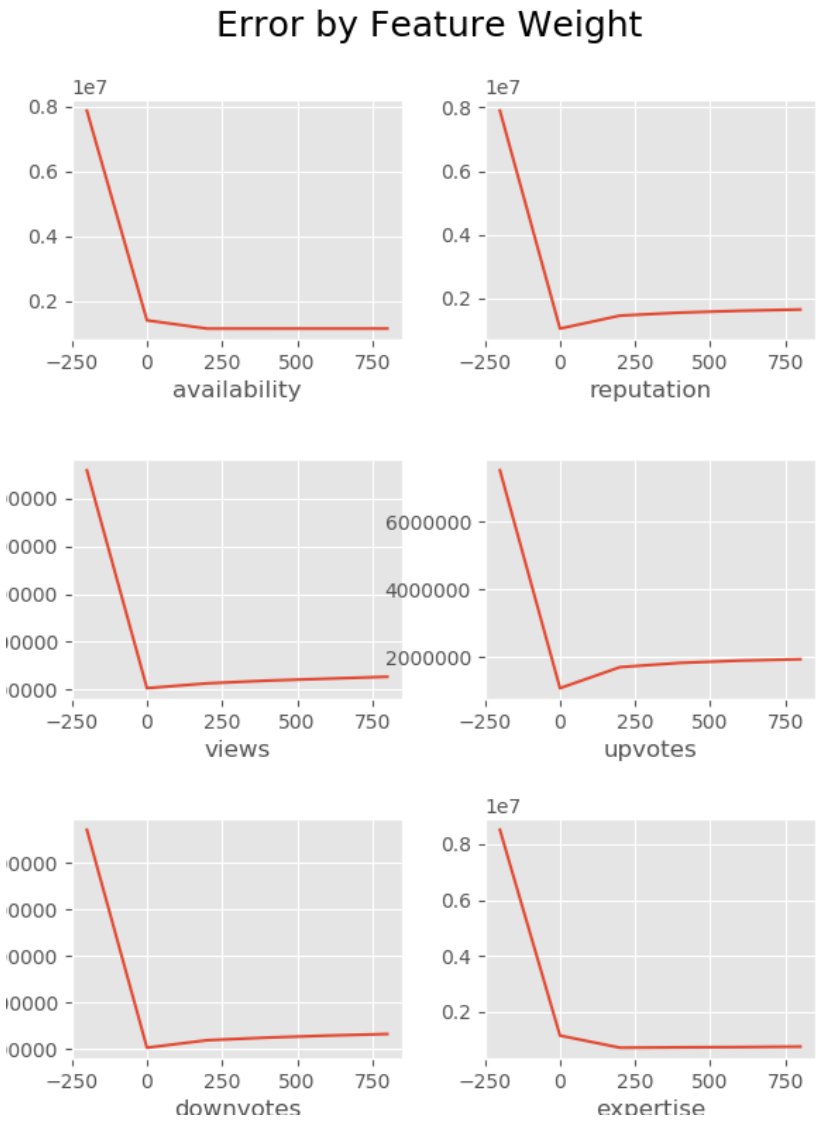
In order for the weights to have a fair affect on each feature, once computed, all features are scaled using a MinMaxScaler. Finally, the scores and subsequent ranks are computed as the sorted sum of these features in descending order.

9.3 Features

9.3.1 Deprecated Features

Our model originally worked with six features obtained from either the question or user. After some testing, however, we found that error was reduced when four of these features are ignored completely; namely user reputation, views, up-votes, or down-votes. These are all static attributes associated with each user, that will not change depending

Figure 10: Error as a partial change in feature weights with insights to how static user meta data are not a good features



on the question at hand. On the other hand, features engineered to correlate between the question and a user dynamically have proven to be effective. Figure 10 shows the relationship between weight and error within each of the original six features. To compute the error, we kept a target weight as a variable, while keeping all other weights constant.

9.3.2 Feature Summary

9.3.3 Loss Function

$$J(\beta) = \sum_{q \in Q} \sum_{obs \in q} \begin{cases} 0 & \text{if } \|obs_q\| = 0 \\ 1 - \frac{1(\text{obs.in}(\hat{rank}[0,t]))}{\|obs_q\|} & \text{otherwise} \end{cases}$$

$$\hat{rank} = sorted(scores_q(\beta))$$

$$scores_q(\beta) = \{\sum_{i=0}^{F} \beta_i f_{i,u} \{foreachuseru\}$$

- U = number of users

- F = number of features
- $||obs_q||$ = is the total number of observed user activities under a particular question
- t = threshold $\leq U$

There are a few caveats about this loss function. For one, each question, regardless on the number of activities in that question, is equally weighted in terms of error. For example, if a one question has 4 res ponders and we captured 2, then this would equivalent to missing 5 res ponders from a question that contains 10 answers. Secondly, this loss function does not capture a measurable distance of how close this system was in identifying a target group. For example, a user ranked at position 0 is equivalent to a user ranked at t-1, where t is a chosen threshold. This decision was made because while we can predict rank, we do not have access to observed ranks.

9.4 Model Run Time Performance

To improve the run time performance of our model, all features were preprocessed into a dictionary (pickle) file. This was in exception to the indicator variables, which scaled in size very quickly with the number of questions involved. To combat this, we decided to cache and preprocess this file dynamically depending on the number of questions analyzed during run time.

9.5 Generating the ROC Curve

Figure 11: Illustration of stacked label and score matrices that were used to generate the ROC curve

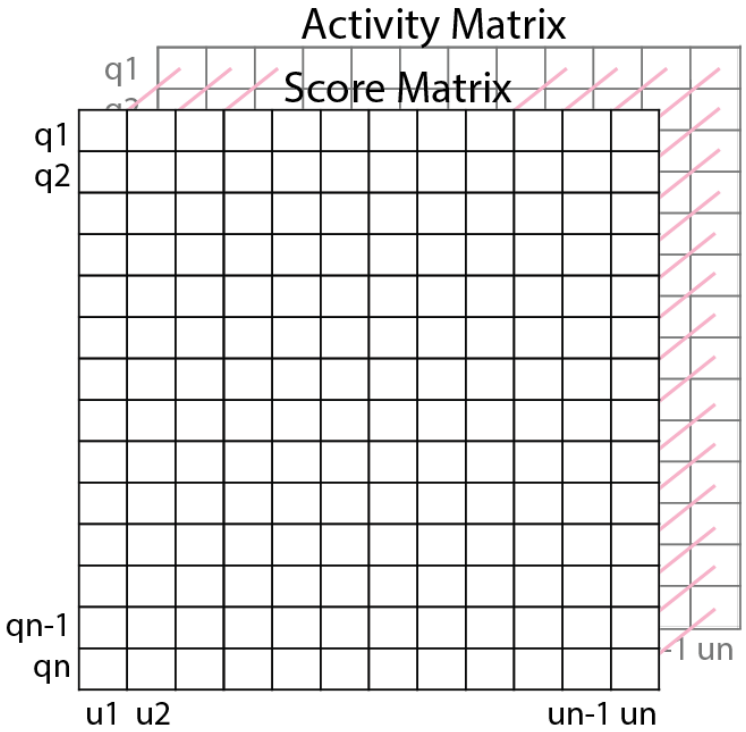


Figure 11 illustrates how the ROC curve was computed. The score matrix unravels as threshold in order classify users who were and were not active on a particular question. For each question, we map each cell of the score matrix to the activity matrix, such that there

is a direct one to one mapping between the two matrices. Finally, using the true labels of 0 for no activity and 1 as activity, along with the target scores as the non-threshold measure of decisions, we can build the ROC curve.

9.6 Summary of Author Contributions

Dan

- Managed overall procedures and initialized ideas, assigning task to everyone
- Data Preprocessing: convert the XML to csv and created test and train sets, subsetting posts with Java tags in 2012
- System Variables; constructed a similarity matrices of users and a tag association network, implementing a model to predict missing tags by the context
- Streamlined a model architecture and integrated features into the final model
- Model Evaluation; visualizations for the error by feature weight, accuracy by threshold and system entropy to evaluate our model

Esmond

- Data Preprocessing: created and refined a subset, posts with Java tags in 2012
- Exploratory Data Analysis; inspected characteristics of users and answered questions such as probability of postings being responded, a distribution of responses and ratio of answers to questions
- Residual analysis; designed plots to detect each feature's performance on computing scores of well-classified and misclassified users

John

- Exploratory Data Analysis; figured out the proportion of questions left unanswered and response time
- System Variables; implemented cosine similarity to identify relevant tags for a tag association network
- Defined the loss function for our data in order to solidate our goal to reduce this error

Seungmi

- System Variables; devised user availability, user topical strength and indicator variables which are detrimental factors in user's activity for an assigned post
- Model Evaluation; plotting a residual matrix of predicted user ranks to identify the overall performance of our model
- Residual Analysis; inspected manually for outlier users

9.7 Log file

In order to maintain a historical record of how our system performed, per system run, a log file is generated that summarizes relative details about its performance. This allowed us to quickly review the changes and employ changes if necessary. The log file shown here is our final and most performant run.

```
INFO:root:Computing ranks using weight vector: [ 50 100 130 0 75 60 0 60 75  80
    20]
INFO:root:Ranking all questions computation finished in 1144.6596206585566
    minutes.
With an average time of 85.84947154939175 seconds per question.
INFO:root:Residual Summary Statistics

----- META -----
Total Questions Analyzed: 800
Total Users Ranked: 59196

----- RANK -----
Summation of Ranks: 16741793
Error per User Activity:
{
    "answer": 10951649,
    "comment": 3382779,
    "edit": 2407365
}
Average Percent Error per User Activity:
{
    "answer": 0.2312582142374485,
    "comment": 0.07143174792215691,
    "edit": 0.05083462142712346
}
Threshold Accuracy per User Activity:
{
    "i_answer": {
        "0.0000%": 0.005964214711729623,
        "10.0000%": 0.7813121272365805,
        "20.0000%": 0.8694499668654739,
        "30.0000%": 0.9012591119946985,
        "40.0000%": 0.9058979456593771,
        "50.0000%": 0.923790589794566,
        "60.0000%": 0.9410205434062293,
        "70.0000%": 0.970178926441352,
        "80.0000%": 1.0
    },
    "i_comment": {
        "0.0000%": 0.026490066225165563,
        "10.0000%": 0.8391674550614948,
        "20.0000%": 0.9385052034058656,
        "30.0000%": 0.9839167455061494,
        "40.0000%": 0.9905392620624409,
        "50.0000%": 0.9962157048249763,
        "60.0000%": 0.9981078524124881,
        "70.0000%": 1.0,
        "80.0000%": 1.0
    },
    "i_editor": {
        "0.0000%": 0.03241491085899514,
        "10.0000%": 0.833063209076175,
```

```
        "20.0000%": 0.9076175040518639,
        "30.0000%": 0.9611021069692058,
        "40.0000%": 0.9675850891410049,
        "50.0000%": 0.9854132901134522,
        "60.0000%": 0.9886547811993518,
        "70.0000%": 0.9967585089141006,
        "80.0000%": 1.0
    }
}
Average Error Per Question: 20927.24125

----- LOSS FUNCTION -----
Threshold Choice: 10063
Total Error: 90.99289321789331
Average ratio of missed users per question: 0.11402618197730979
Standard Deviation of missed users per question: 0.21092206768671673

If we send out an email to 10063 out of 59196 total users,
then we can expect to meet 88.5974% of our target responders.
From the identified 3186 total user activities, we can expect to hear back
from 2822 such users.
```

References

[1] Chang, Shuo, and Aditya Pal. "Routing questions for collaborative answering in community question answering." Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ACM, 2013.

[2] "Stack Exchange Data Dump : Stack Exchange, Inc. : Free Download, Borrow, and Streaming." Internet Archive, The Library Shelf, archive.org/details/stackexchange.

[3] Anderson, Ashton, et al. "Discovering value from community activity on focused question answering sites: a case study of stack overflow." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

[4] "Find Interesting Unanswered Questions." Stack Exchange Data Explorer, data.stackexchange.com/stackoverflow/query/4038/find-interesting-unanswered-questions.