User Recommendation for Stack Exchange

Priyank Mathur
priyankm@stanford.edu

Siamak Shakeri
siamaks@stanford.edu

Dibyajyoti Ghosh
dibghosh@stanford.edu

Abstract

Recommendation systems have become very common and can be found in some form in most of the services we use. These range from movie recommendation on Netflix, product recommendation on Amazon, pin recommendation on Pinterest to expert user recommendation on Quora, job recommendation on LinkedIn and so on. Traditionally, these systems are built using some form of collaborative filtering combined with content analysis and latent factor models. In this project, we aim to utilize the underlying graph structure and network properties of the dataset to identify features for machine learning based user recommendation system. In particular, we envision using node2vec [5] to provide node embeddings that are expected to capture some of the important network features. Our objective is to recommend the most appropriate user when a question is asked in Stack Exchange forums.

1 Introduction

Q&A sites like Quora, Stack overflow, Stack exchange, Yahoo Answers have been around since early days of online communities. Collectively these forums represent a wide spectrum of crowd sourced and curated contents in various domains of interest e.g. mathematics, physics, computer science to sociology, finance, gardening, origami, arts, cuisines etc. However a major shortcoming for these communities is that at any given instant there are more users who ask questions than those who have sufficient domain knowledge to correctly and satisfactorily answer questions. Some of these forums have come up with clever ways of incentivizing experts through reputation points and badges but it leaves a lot more to be desired. Discoverability of open questions is a daunting challenge.

We would like to address discoverability of open questions for expert users in these forums. By recommending right group of users for a question we can maximize chances of questions being answered satisfactorily. User's expertise when used with deep learning based vector representation of users has not been previously explored. We believe that we can capture a more complete picture by extracting features from node representations in a network model that embeds social and categorical context.

2 Related Work

There has been a lot of interesting research in the area of user recommendation in Q&A communities. Lada Adamic et al [1] analyzed Yahoo Answers characteristics across multiple categories. They used Motif Analysis, SCC and Ego Networks to analyze network structure, and to gain insight. In their network model each node is a user. A link exists from user A to B, if A has answered a question from B. Although this approach delivered good insights, it failed to embed quality of the answer provided. For example it ignores if an answer has been selected as best answer or not. They also ignored the number of the votes that an answer has received or features that embed social behavior of the network e.g. answerer to asker interactions were not considered.

Gideon et al [2] proposed a recommendation system for Yahoo Answers and they heavily relied on feature extraction. They used content signals extracted from question text, and social signals, extracted from user-user interactions such as asking, answering, voting, etc. Addition of bias features to both questions and users in order to grasp state of the question and overall mood of the user is a novel aspect of this work.

Jure et al [3] focused on a less highlighted aspect of Q&A communities, namely, defining its future value in terms of how quickly and accurately answer-seeker is directed to correct content. For domain-experts value of communities is contingent upon connecting them to less answered but important questions. Their work formulated processes to define future relevance of Stack Overflow. This paper claims that answer arrival dynamics within as little as an hour of posting a question can be an effective predictor of long term page views of the given question. They also claim that number of answers to a given question is a powerful latent feature for future page view trend prediction. Something they haven't explored in this paper is the effect of cross-domain answering dynamics. E.g. a Java programming language expert answering Python programming questions. Or for that matter, how question type can affect its longevity and chance of it being answered sufficiently.

Szpektor et al. [4] have explored question recommendations in Yahoo Answers Q&A community by advocating three guidelines for better chances of getting questions answered - remove user bias i.e. treat all users as equal, recommend questions outside user's active area of interest and serve questions fast and fresh. Authors have used adaptive topic sampling algorithm to generate recommendations of questions to users. They claim that such an approach can maintain diversity of questions apart from the ones generated based on personalized user profile. Diversity of recommended questions based on user's geolocation would be a curious extension of this work.

3 Data

3.1 Data collection

We are using question and reply data set from Stack Exchange website [6]. The data has been divided into sub domains like math, physics, computer science, data science etc. For each sub domain, information is provided about posts, votes, users, post relations etc. More details about the data set and schema can be found at [7].

For this project, we mainly worked with the data for Stack Exchange's math community. This data set contains *1,584,856* posts that include *652,128* questions. Most questions in this data set have an accepted answer which is an answer that has been deemed by the question asker as being sufficient and correct. We use this field to determine the target user for each question. We also experimented with physics dataset. This dataset is relatively smaller with *205,134* posts that include *91,962* questions. Unless mentioned otherwise, all results and graphs used in this report are derived from math's dataset.

3.2 Data preparation

We modeled user recommendation in Stack Exchange as a classification and a ranking problem. The dataset is split into a training, validation and a test set for best results. For each question, we try to predict the user who will provide the accepted answer. The candidate selection process for each question looks at the following features for each user and scores them:

1. If the user did provide the accepted answer for the question under consideration
2. Number of votes the user's answer, if any, received for that question
3. Jaccard similarity between the tags for the question and all the tags that the user has provided an answer for

We then combine these scores to come up with the top 5 users as the candidates for each question. This process ensures that the correct user is always one of the five shortlisted users. Ideally, this problem could be solved using a 2 level classification scheme. At the first level, we choose a subset of the users using a simple and an efficient method. At the second level, we identify individual users to recommend. However, to reduce the scope of the problem, we pre-select the candidates and assume that in the real world scenario the shortlisting procedure would be good enough to zoom in to the set of relevant users.

After candidate selection process, we divide the dataset into a training and a test set based on the creation date of the question. All the questions before *'2015-06-01'* form the training set and all the latter ones are in the test set. This process gives us *15796* training data points and *4146* test data points. The full training data set is then further divided into *11000* final training examples and *4796* validation examples. Once we have this split, we use various methods discussed further in the report to generate representations for each question and its candidate users. These representations are finally fed into a machine learning model that will make predictions about the likeliness of a user giving the accepted answer.

3.3 Evaluation

The final recommendation model would output a score or a probability for each candidate user for a given question. These scores can be interpreted, and therefore, evaluated in two different ways. We can use these scores to rank users to come up with the order of recommendation. In this case, a metric like mean reciprocal rank would be used. In addition, if we consider the top ranked user as the final prediction, then this becomes a classification problem. A common way to evaluate classification results is through accuracy values. These metrics can determine how accurately can we rank the users such that the ones at the top of the list are most likely to supply the best answers for a given question. In this project, we will report both these metrics. However, since mean reciprocal rank depicts how the model performs on average and not just at the top of the result list, we will use it as our primary performance indicator.

4 Networks

As discussed earlier, we focused on only using network properties and node representations as features for our user recommendation algorithm. Most previous approaches [2] have focused on content analysis combined with some form of collaborative filtering to develop recommendation systems. However, they failed to utilize the implicit information stored in the structure of the underlying networks.

We constructed the following 2 networks from the training data set's time frame and performed analysis on them.

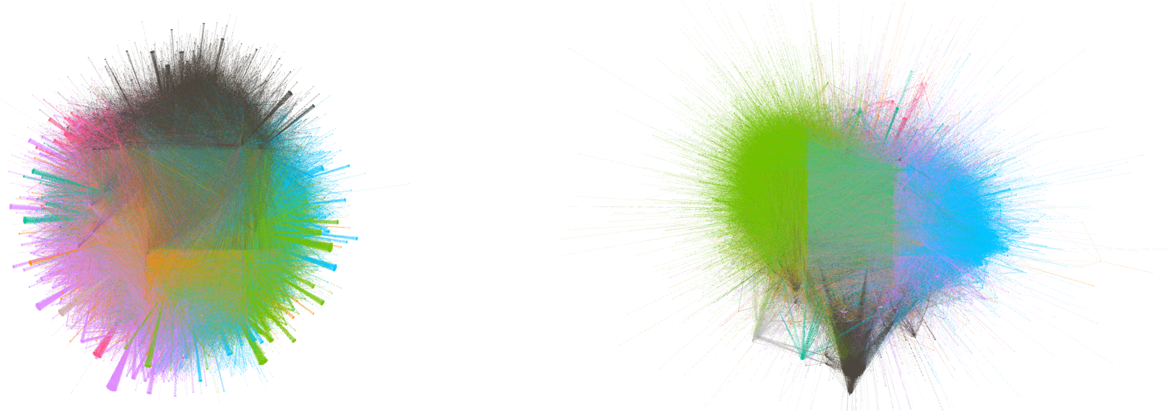
1. **User user network through posts**
   This is an undirected and unweighted network consisting of all the users in the training data set as nodes. Users that answer the same question have edges between them. This network has *31834* nodes and *321993* edges.
2. **User tag network**
   This network is also undirected and unweighted where a node can be a tag or a user. An edge exists between a user and a tag if the user answers a question marked with that tag. This network has *33605* nodes and *368506* edges

4.1 Network study

We performed detailed investigation of the above mentioned networks to identify features that might help us during user recommendation. In addition, we also performed some qualitative analysis to determine if these metrics actually captured meaningful information. We used Gephi[12] for network analysis on the two networks.

4.1.1 Community Detection

Figures 1b and 1a show the community detection done on both networks using clustering algorithm in [13]. Each color represents a community.



(a) user-tag

(b) user-user

Figure 1: community detection

In the user-user network, a total of 4880 communities where detected. However only 8 of them have higher than 1% of the total nodes. As it can be seen from the figure, this network is not highly modular, and communities are highly connected. This stems from the fact that answerers usually have a wide range of technical skills, thus they answer questions from various domains of expertise.

In user-tag graph, a total of 8 communities were detected, and node count ranges from 3% to 23% in each community. Similar structure as user-user network can be seen here as well. They have distinct communities with strong interconnection.

We used community membership of each user in both graphs as an input feature to the classifier for user recommendation.

The same community detection approach was applied to the physics community dataset, and similar results were seen. In user-tag network, 7 communities were detected with each community comprising 5% to 22% of the users. Similar to math community dataset, for user-user network in physics dataset we found a larger number of communities, 1101 in total with only 13 comprising more than 1% of the whole population. This shows similar community structure in math and physics communities.

4.1.2 Pagerank and HITS

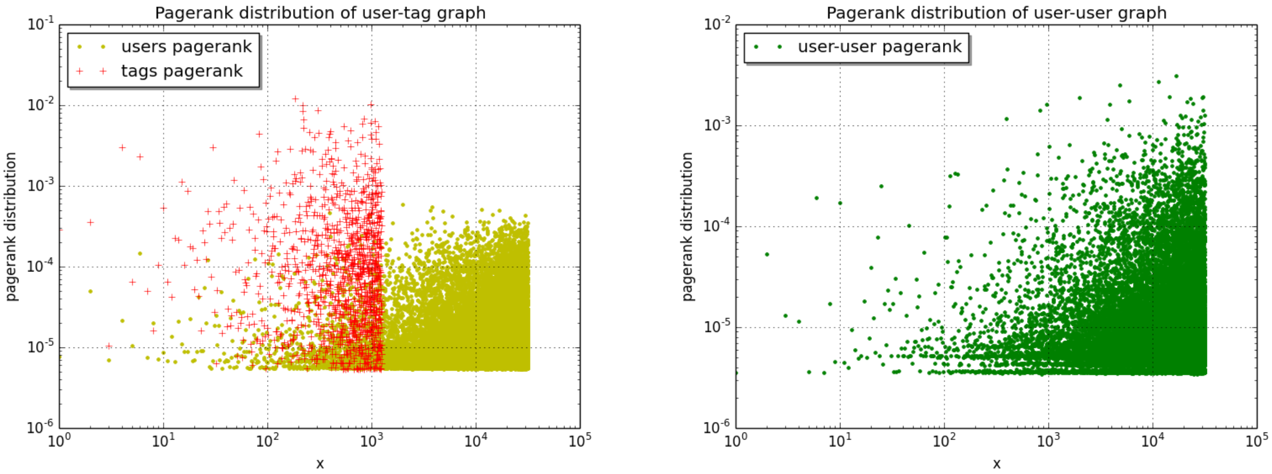


Figure 2: Pagerank distribution for user-tag and user-user graphs in log-log scale

We explored link structure of both networks to assign weights to nodes. Because of the nature of these networks, these weights can be interpreted as expertise or authoritativeness of the user in their respective domains. These properties are important factors in our recommendation algorithm. Kleinberg's HITS algorithm[14] and PageRank algorithm by Brin and Page[15] are two very well known approaches in this space. We ran page-rank analysis on both user-user and user-tag graph with $\epsilon = 1.0E - 4$ & probability = 0.85 and used *(node, pagerank)* tuple to generate page-rank distribution. These plots are presented in figure 2. For user-tag graph pagerank values are more widely distributed as opposed to user-user graph. Y-values ranges from

$$y \in [1.0E - 6, 1.0E - 1] \tag{1}$$

in user-tag graph and

$$y \in [1.0E-6, 1.0E-2] \tag{2}$$

in user-user graph. Similarly, we ran HITS for both graphs with $\epsilon = 1.0E-4$. Hub and authority distribution data for both networks are presented in appendix section in figure 9.

4.1.3 Structural analysis

In our analysis, we asked a few structural questions about both user-tag and user-user networks. These include - who are these top 10 users in the both graphs by pagerank, HITS or eigencentrality score; what are top 10 tags in user-tag graph and how they compare across various scoring systems; do the top users in graphs form communities among themselves; how many of the top 10 users by pagerank or HITS score are common between the graphs; how does the last measure change if we consider top 10 percentile users instead; and finally, if and how these measures will help us validate our final recommendations?

We found that {'calculus', 'linear-algebra', 'real-analysis'} are the three most central tags by pagerank, HITS and eigencentrality measure while {'calculus', 'probability', 'integration', 'sequences-and-series', 'linear-algebra', 'algebra-precalculus', 'real-analysis'} makes it to top 10 tags list by all four measures we ran on user-tag graph, namely pagerank, authority score, hub score and eigencentrality. Table 1 shows the top 10 tags using the four measures.

Table 1: Top 10 tags by structural statistics

Tag, Pagerank	Tag, Authority score	Tag, Hub score	Tag, Eigen-centrality score
calculus, 0.0122	calculus, 0.1848	calculus, 0.0382	calculus, 1.0
probability, 0.0101	linear-algebra, 0.1663	linear-algebra, 0.0343	linear-algebra, 0.8775
linear-algebra, 0.0100	real-analysis, 0.1660	real-analysis, 0.0343	real-analysis, 0.8612
real-analysis, 0.0086	algebra-precalculus, 0.1509	algebra-precalculus, 0.0312	algebra-precalculus, 0.7779
algebra-precalculus, 0.0085	sequences-and-series, 0.1445	sequences-and-series, 0.0298	sequences-and-series, 0.7171
combinatorics, 0.0068	integration, 0.1411	integration, 0.0298	integration, 0.7120
geometry, 0.0067	limits, 0.1346	limits, 0.0278	probability, 0.7039
integration, 0.0063	analysis, 0.1325	analysis, 0.0274	limits, 0.6566
soft-question, 0.0060	functions, 0.1319	functions, 0.0272	geometry, 0.6455
sequences-and-series, 0.0058	probability, 0.1299	probability, 0.0268	analysis, 0.6445

In user-tag graph, top 10 users belong to three communities. These communities also hold majority of the top 10 percentile users as well. For top 10 users in user-user graph, community formation was more evenly spread out with a $30:20:20:20:10$ percent split. However, top 10 percentile users overwhelmingly belong to two communities with 36% and 31% split. Higher pagerank in user-user graph implies either a given user answered a large number of questions by volume or answered a fewer number of popular questions. Thus community distribution in this graph for top 10 or top 10 percentile users (by pagerank score or authority score) should implicitly capture expert users. One may assume that correlation between the 2 graphs is inconsequential. However, we argue that an expert user is one who has higher number of edges with popular tags and also has answered more questions in general or answered more popular questions leading to more tags to user connection in reverse. We found that about 40% of top 10 percentile users are common in both user-user and user-tag graphs across all the scoring measures. In particular, users with id's {1827, 39174, 6312, 12042, 11667, 8508} feature in the top 10 list for both graphs.

We started our analysis with an assumption that patterns discovered by classical approach of central actor detection by structural analysis of graphs should also be reflected in the learning based expert detection. However, as described further in the report, classical metrics do not perform well when compared to node2Vec.

4.2 Node2Vec

Emergence of deep learning as one of the most efficient machine learning techniques has mostly benefited NLP and computer vision tasks. Node2vec [5] algorithm transfers some of the techniques in those domains to network analysis. This method allows us to learn vector representation of nodes that capture the inherent structure of connections within the network. The training process involves two steps -

1. In the first step, we generate multiple random walks for each node in the network. Consider that a random walk traversed an edge (t, v) and v is the current node. The transition probabilities of the nodes x that can be considered next is then given by -

   $$\pi_{vx} = w_{vx} \times \begin{bmatrix} \frac{1}{p} \text{ if } d_{tx} = 0 \\ 1 \text{ if } d_{tx} = 1 \\ \frac{1}{q} \text{ if } d_{tx} = 2 \end{bmatrix}$$

   Where w_{vx} is the edge weight for the edge v to x and d_{tx} is the shortest path between t and x. These walks are governed by following parameters:

   (a) Return parameter p - This parameter controls the probability of returning to a node in a random walk. High value of p will give a lower likelihood to returning to a node.

   (b) In-out parameter q - This parameter controls how far is the random walk allowed to go from the current node. Higher values of q reduce the likelihood of the random walk from going too far out.

2. The second step involves running a model like GloVe[8], CBOW[9] or SkipGram[11] on the random walks generated above to learn the distributed representation of the nodes.

Intuitively, one can imagine the representations generated from the user-user network to capture information about user similarity or user communities that answers similar questions. Similarly, the embeddings from the user-tag network should capture information about user expertise and tag similarities.

Node2vec on networks

To generate these vectors from the networks, we ran a grid search over the parameters p and q to find the best combination. We set the other hyper parameters to their (snap) default as - vector dimension: 128, walk length: 80 and number of walks per node: 10.

In this experiment, for each question, we used the average vector of its tags as the question's representation. Each candidate user representation was generated by the concatenation of their vectors from the 2 networks. Once we had this data, we ran a logistic regression classifier [10] to train a model on it. We used the validation set's mean reciprocal rank to evaluate the quality of the vectors. The following plots show the results from the grid search.

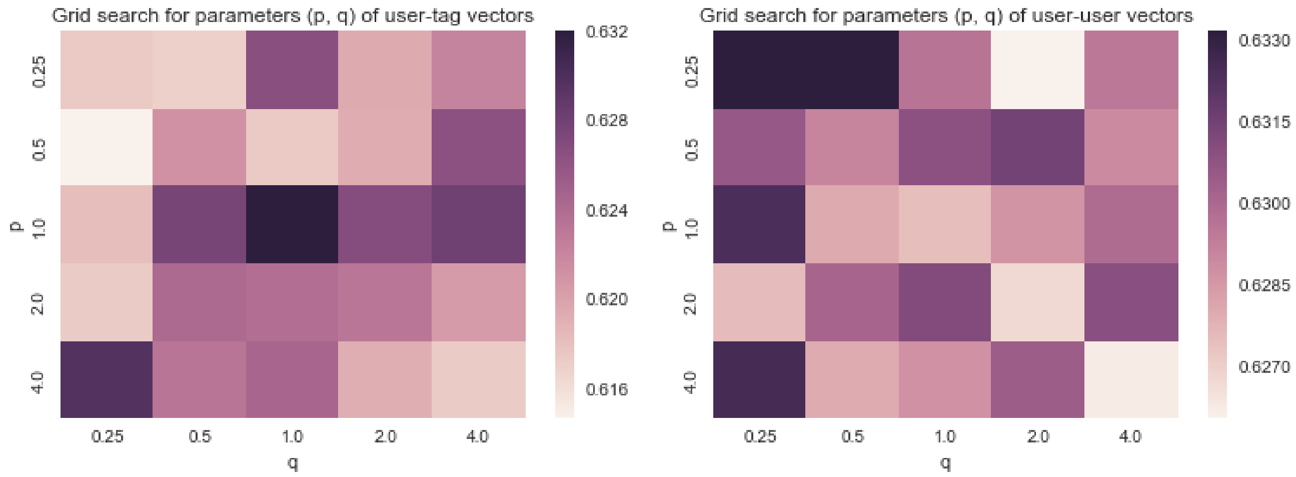


Figure 3: Grid search for user-tag (left) and user-user (right) network vectors

For the user-tag graph, we get $p = q = 1$ and for user-user graph, we get $p = q = 0.25$. In the user-tag graph, a relatively high value of p implies that same nodes would be rarely sampled during random walks, while high value of q will ensure the walks do not go out too far. This implies structural equivalence, where nodes that have similar roles lie close to each other in vector space, is preferred for this graph. This is expected as user-tag graph is a bipartite graph. Borrowing some terminology, tags here play the roles of hubs and users play the roles of authorities. In case of user-user graph, low values for both p and q will allow reasonably deep explorations with restarts of the network. This implies that vector representation which places interconnected nodes closer together is preferred for this graph. Since an edge in this network means that users answered same questions, such a vector representation will capture user communities that are likely to answer similar questions.

To analyze the quality of the vectors generated from these graphs, we performed k-means clustering on the resultant node vectors. To identify appropriate number of clusters(k) for each network, we searched for the best k by plotting a cost vs. k plot and choosing k corresponding to the inflection point. The visualizations of the clusters can be seen in figure 4. We used PCA to reduce the dimentionality of the vectors. For both the graphs, we can clearly notice different, though sometimes overlapping, communities that exist in each graph.

Table 2: Examples of tags in each cluster

Machine Learning/Optimization	Statistics	Geometry	Calculus	Networks
regularization	probability-distributions	spheres	limits	trees
hadamard-product	law-of-large-numbers	rectangles	integration	coloring
convex-analysis	stochastic-analysis	3d	calculus	graph-theory
lagrange-multiplier	statistics	geometric-construction	derivatives	hamiltonian-path
constraints	bayesian-network	computational-geometry	definite-integrals	clustering
machine-learning	markov-chains	circle		random-graphs
regression	covariance	convex-hulls		network
convex-optimization	expectation	euclidean-geometry		network-flow
numerical-optimization	probability-theory	triangle		graph-isomorphism
game-theory	monty-hall	quadrilateral		planar-graph

In addition, from some of the tag clusters, we can clearly identify the "topic" for that cluster as depicted in table 2. This indicates that the vectors from node2vec can successfully capture useful information about not only the network structure, but also some implicit information like similarity of tags.

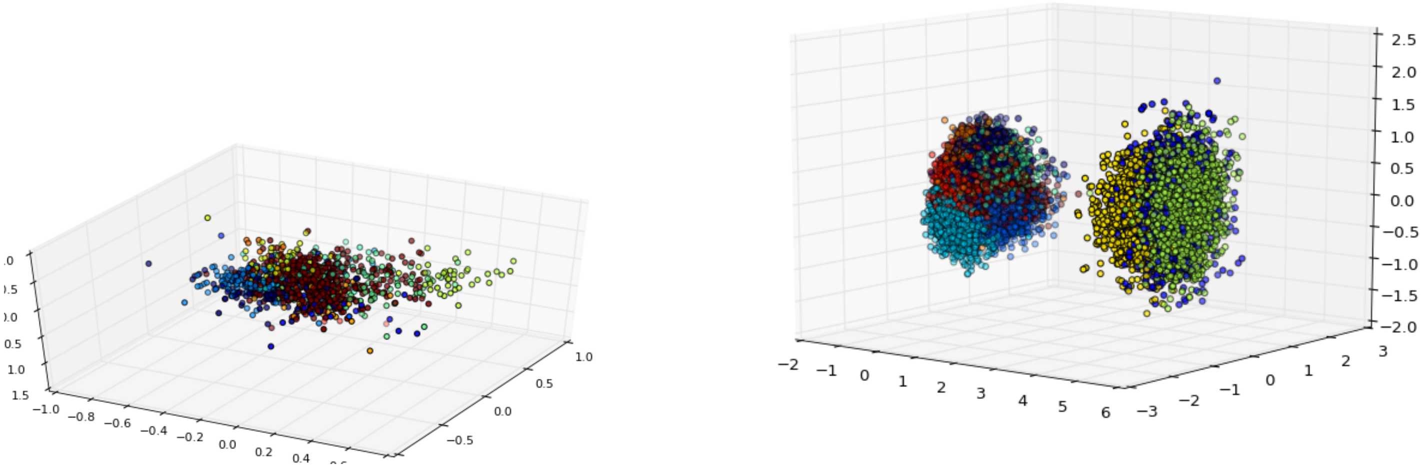


Figure 4: Clusters for tag (left) and user (right) vectors

5 Methodology and experiments

Our objective is to explore the importance of network based features for users and tags when it comes to recommending users. These features are fed into a machine learning model to predict the appropriate user for each question. Figure 5 below describes the process at a high level. We conducted several experiments, each with a different feature set or model to achieve the best performance. Each experimental setting is further described below. All the results are tabulated in table 3.

We used validation set to determine which experiments performed well based on the training set, and measured performance on the test data only for the best models. In addition, to test performance on the test data, we trained the model on the entire training plus validation set (*15796* examples).

The choice of machine learning models for our experiments included logistic regression and gradient boosted trees. Logistic regression is a popular and fairly good linear model that works well when we have relatively large data sets, as in our case. However, being a linear model, it does not identify inter feature interaction patterns. Gradient boosted decision tree is an ensemble model that allowed us to learn non-linear decision boundaries. It also helped us reduce the variance by combining many different smaller decision tree models to provide the final prediction. In order to ensure we are studying the effects of features and not of the classifier, we used default settings for both the models as documented at Logistic Regression and Gradient Boosting Classifier.

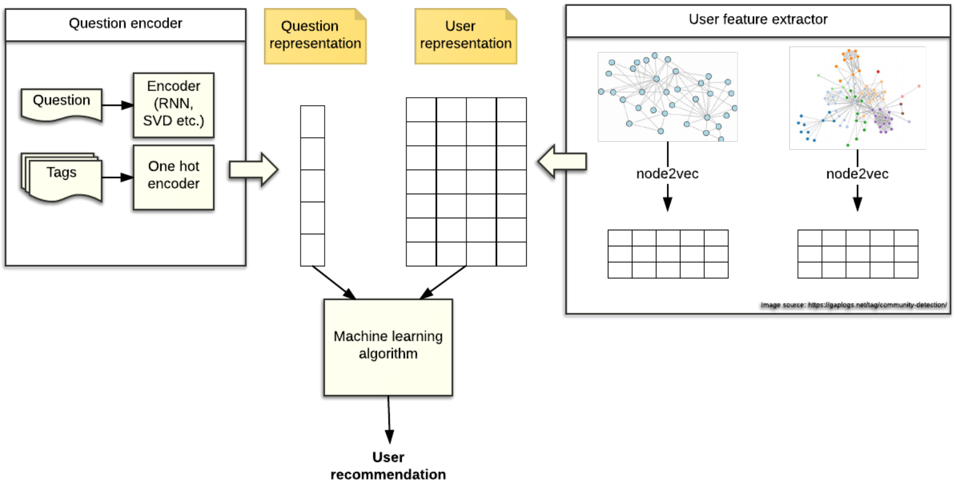


Image source: https://gaplogs.net/tag/community-detection/

Figure 5: Recommendation pipeline

We experimented with several different feature sets and models as detailed below -

1. **Baseline** - We created a reference baseline model for the task using only a single feature, namely Jaccard similarity. Jaccard similarity is a measure of set overlap. In this model, we computed Jaccard similarity between the tags assigned to the question and tags for which each candidate user has provided an answer for. Candidate users were then ranked in order of their decreasing similarity with the question and the top ranked user was predicted as the recommended user. As expected, being very simple, this model did not perform well.

2. **Node2Vec** - As mentioned in section 4.2 , this set of experiments only used node embeddings generated by running node2vec on the two graphs. The hyper parameters p and q were chosen as the best performing pairs from the grid search done in 3. For each question, we used the average vector of its tags as the question's representation. Each candidate user's representation was generated by the concatenation of their vectors from the 2 networks. We experimented with both, a logistic regression and a gradient boosted tree model with this feature set.

3. **Graph features** - We created several graph based and other features for questions and users for this experiment. For the question, features consisted of

   - Tf-Idf features of the question title followed by SVD to retain 50 most relevant components
   - Tf-Idf features of the question body followed by SVD to retain 50 most relevant components
   - Favorite count of the question, which represents number of times someone followed the question

   For the users, the representation consisted of one-hot-vector encoded community ids, page rank and HITS scores from section 4.1.

4. **Combined feature set** - Here we combined all the graph based features discussed above to form representations for the users and questions.

The results from the experiments above are documented in table 3 and are further analyzed below.

6 Results and analysis

As can be seen from the results, we can achieve much better performance than our baseline in all of the experiments. Looking at the results from node2vec (experiment 2), we see that node embeddings alone were able to provide very good performance. In comparison, even though graph features (experiment 3) consisted of features from many different analysis, were not able to reach a similar level of performance. Surprisingly, even when using all the features combined (experiment 4) and a powerful model like gradient boosted trees, we could only get as good as the node2vec experiment alone. This indicates that node2vec captured information like node importance, node connectivity, similar clusters better than individual traditional graph analysis methods.

Our best model from the experiments used node2vec vectors as features and gradient boosted decision trees classifier. We further analyzed the predictions we made on the test set by this model. For each question, we used the score from the model to rank the candidate users. Below is the distribution of the ranks of the correct answer for each question.

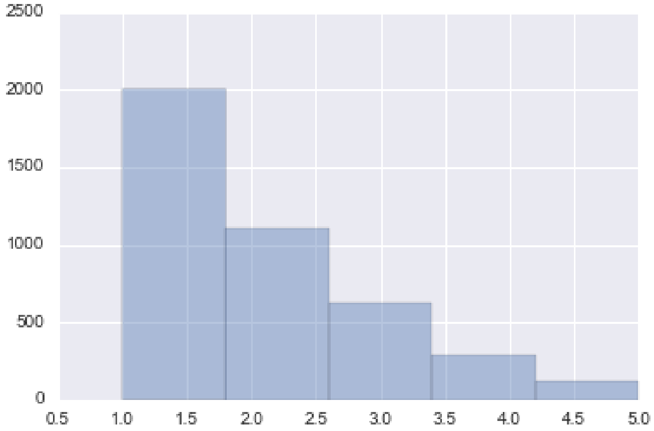


Figure 6: Rank distribution for test set

We notice that more than 75% of the time, the correct answer will be suggested in the top 2 results. Given that we only considered graph features to generate representations, we believe we were able to successfully demonstrate the importance of network features, in particular of node2vec embeddings, while predictions.

6.1 Error investigation

We performed additional investigation of the test set questions where the correct answer appeared at rank 4 or 5, which we call the bad questions set and compared it to questions where correct answer appeared at ranks 1 or 2, called the good questions set. Since we only used graph based features, we wanted to identify if the bad performance can be explained by some network characteristics or not. We only look at the user-tag graph since question representation can not be generated using only user-user graph for experiment 2.

First, we take a look at the tags for the 2 sets. We find that there is very little overlap between the tags from the two sets, as indicated by jaccard similarity of only 0.39. This may imply that there are certain tags for which it is easy to identify best users in the vector space, either due to fewer neighbors or due to closeness to them. Diving further down, we compared some statistics for the tag sets as shown in figure 7. We found that scores like pagerank, hub, node degree etc. are, on average, lower for the good question's tags than for the bad question's tags. This may appear counter-intuitive at first, but we attempt to explain it as follows. Higher degree, hub score etc. indicate that a node will have more neighbors, which in this case means users. More users may indicate that we will have lesser confidence when we try and identify expert users

Table 3: Performance summary

Model	Mean reciprocal rank			Accuracy		
	Training set	Validation set	Test set	Training set	Validation set	Test set
Baseline	0.3027	0.3025	-	0.0365	0.0342	-
Node2Vec + LR	0.6536	0.6332	-	0.4298	0.3959	-
Node2Vec + GBT	0.6852	0.631	**0.6907**	0.476	0.3969	**0.4835**
GF + LR	0.5015	0.4914	-	0.4589	0.4825	-
GF + GBT	0.5239	0.4878	-	0.5182	0.4908	-
Node2Vec + GF + LR	0.6586	0.6309	-	0.4369	0.3961	-
Node2Vec + GF + GBT	0.6856	0.6357	0.6899	0.4762	0.4051	0.4807

LR - Logistic regression, GBT - Gradient boosted trees, GF - Graph features

for that topic. Another explanation might point to our training procedure, where we chose 10 random walks per node, which might not be sufficient for nodes with higher degrees.

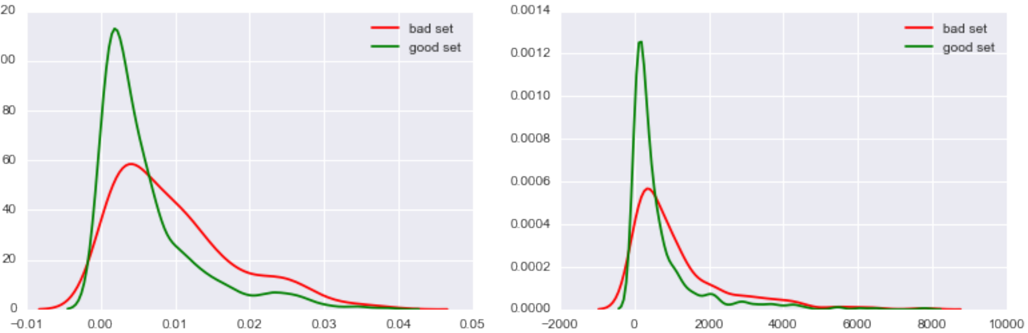


Figure 7: Comparison of tag statistics - hub score (left) and degree (right)

A similar analysis of the metrics for user nodes gives more expected results, as shown in figure 8. We again created 2 sets - users that gave accepted answers for good questions set and the bad questions set. Looking at the authority score and degree distributions, we notice that users in the good set have higher scores and degrees. In contrast to our argument above, even if higher degree for a user results in lesser confidence in its expertise, we feel that the information from the vectors from the user-user network overcomes this.

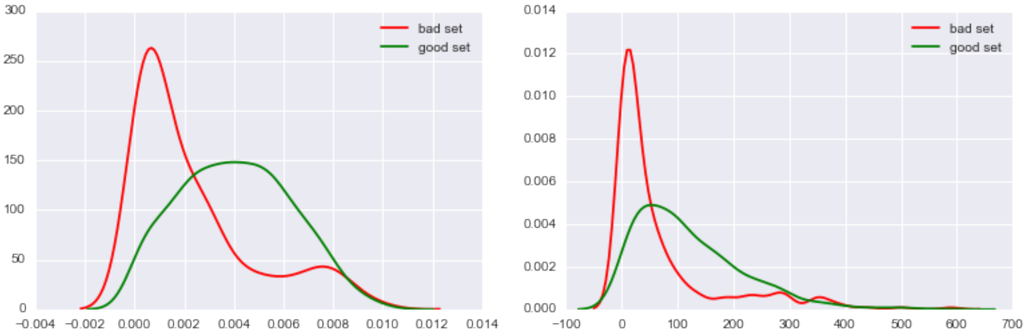


Figure 8: Comparison of user statistics - authority score (left) and degree (right)

6.2 Physics Community

To study if our approach was generalizable, we experimented with applying our recommendation pipleline to the physics data set. Table 4 shows the results. As it can be seen, the values are very close to the results from the math community (3) with performance being slightly better. As with the math community, there is more than 10x improvement in the performance when using node2vec, compared to the baseline.

Having similar performance results in math and physics community datasets proves that our approach is not specific to the math community, and can be applied accross various question answering social networks.

7 Conclusion and future work

Our objective in this project was to recommend the most appropriate user when a question is asked in Stack Exchange forums. In addition, we wanted to investigate a new method wherein we only considered features learned from the underlying networks in the data set. Based on our

Table 4: Performance summary - Physics Community

Model	Mean reciprocal rank			Accuracy		
	Training set	Validation set	Test set	Training set	Validation set	Test set
Baseline	0.302	0.296	0.351	0.049	0.042	0.065
Node2Vec + LR	0.705	0.669	-	0.505	0.440	-
Node2Vec + GBT	0.772	0.665	**0.764**	0.603	0.430	**0.599**

experiments, we see that node2vec features performed much better than other hand crafted features computed using traditional graph analysis methods. Even though we only considered this small feature set, we were successfully able to rank the candidate users such that the correct answer was present in the top 2 results 75% of the time. We believe we can now successfully assert the importance of doing graph analysis for this task. Having similar performance results in both math and physics communities proved that our approach is rather generic.

The results achieved in this project can be improved upon in several ways. Building more and different kinds of networks like directed, weighted graphs from the data can help improve the current model. Using other features in the data set like answer scores, text, user rating etc. can provide additional useful information to the learning model. Of course, combining this method with traditional recommendation systems would be another great exploration.

Acknowledgments

We are grateful to the following people for resources, discussions and suggestions: Prof. Jure Lescovic and Ben Ulmer.

Individual contribution

1. Priyank Mathur - Data curation and training/validation/test set preparation; node2vec training, exploration and analysis; machine learning setup and experiments; result/error analysis; report writing.
2. Siamak Shakeri - Related work study, community detection analysis, question feature extraction, experimentation and analysis of the physics data set, report writing.
3. Dibyajyoti Ghosh - Related works study, Gephi and network analysis tools exploration, graph data and plot generation, graph data analysis, community detection analysis, report writing.

References

[1] Lada Adamic et al. *Knowledge Sharing and Yahoo Answers: Everyone Knows Something*. WWW ’08. `http://www-personal.umich.edu/~ladamic/papers/yahooanswers/fp840-adamic.pdf`.

[2] Gideon Dror et al. *I want to answer; who has a question?: Yahoo! answers recommender system*. KDD ’11. `http://dl.acm.org/citation.cfm?id=2020582`.

[3] Jure Leskovec et al. *Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow*. KDD ’12. `https://www.cs.cornell.edu/home/kleinber/kdd12-qa.pdf`.

[4] Szpektor et al.. *When Relevance is not Enough: Promoting Diversity and Freshness in Personalized Question Recommendation*. WWW ’08. `http://www2013.w3c.br/proceedings/p1249.pdf`.

[5] A. Grover, J. Leskovec. *node2vec: Scalable Feature Learning for Networks*. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[6] Stack Exchange `https://archive.org/details/stackexchange`

[7] Stack Exchange Dataset `https://ia800500.us.archive.org/22/items/stackexchange/readme.txt`

[8] Jeffrey Pennington, Richard Socher, Christopher D. Manning *GloVe: Global Vectors for Word Representation* `http://nlp.stanford.edu/projects/glove/`

[9] Kenter, Tom, Alexey Borisov, and Maarten de Rijke. *"Siamese CBOW: optimizing word embeddings for sentence representations." arXiv preprint arXiv:1606.04640 (2016).* `https://arxiv.org/abs/1606.04640`

[10] *Logistic Regression* `https://en.wikipedia.org/wiki/Logistic_regression`

[11] Guthrie, David, et al. *"A closer look at skip-gram modelling." Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006). 2006.* `http://www.cs.brandeis.edu/~marc/misc/proceedings/lrec-2006/pdf/357_pdf.pdf`

[12] Gephi: The Open Graph Viz Platform `https://gephi.org/`

[13] Blondel V, Guillaume J, Lambiotte R, Mech E *Fast unfolding of communities in large networks.J Stat Mech: Theory Exp 2008:P10008.*

[14] Kleinberg, Jon M., et al. *"The web as a graph: measurements, models, and methods." International Computing and Combinatorics Conference. Springer Berlin Heidelberg, 1999.*

[15] Page, Lawrence, et al. *"The PageRank citation ranking: bringing order to the web." (1999).*

Appendix

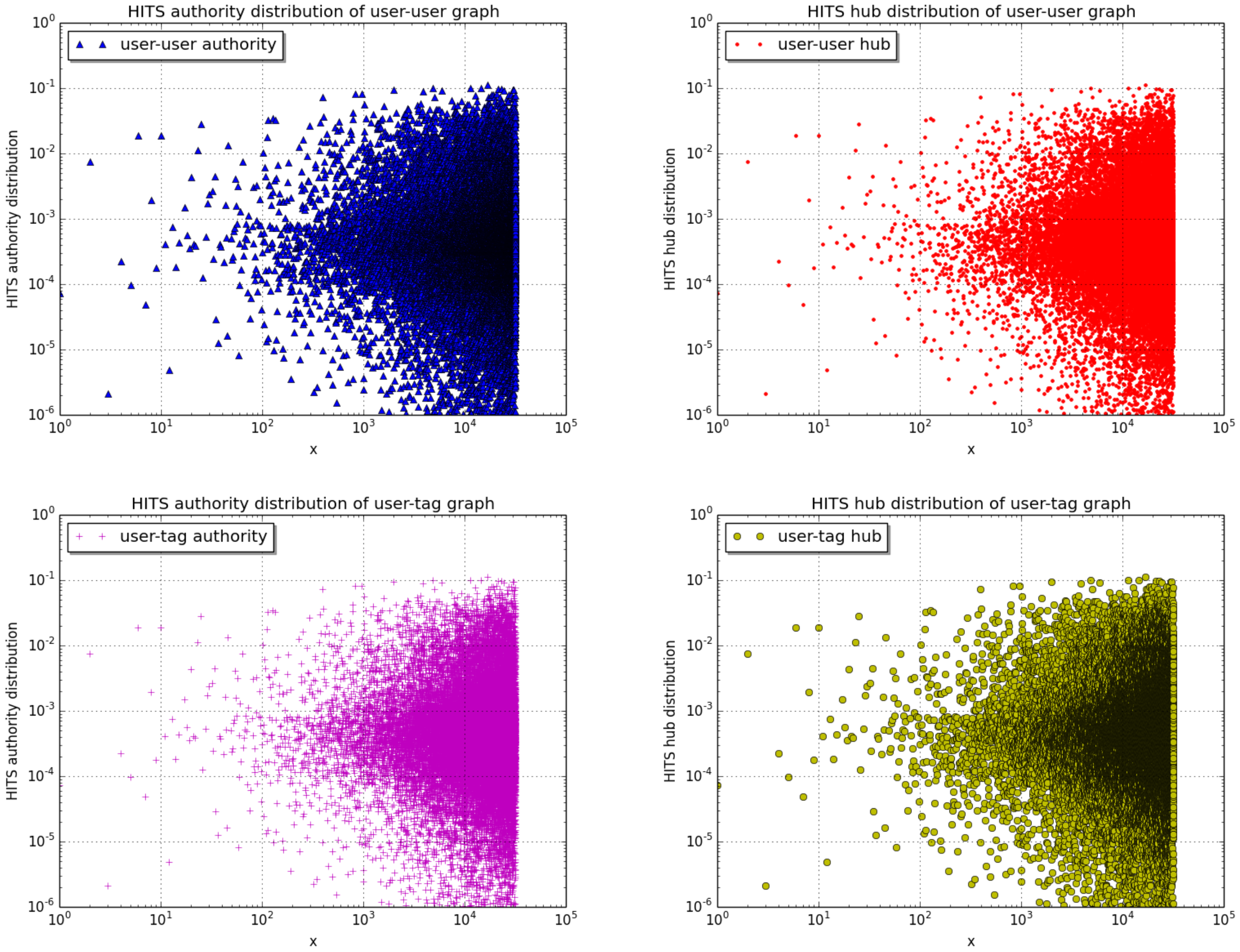


Figure 9: HITS hub and authority distribution for user user and user tag graph in log-log scale