Question Tag Prediction

Mahi Rahimi
School of EE
KAIST
Daejeon, South Korea
mahi@kaist.ac.kr

Mikhail Pyatykh
School of EE
KAIST
Daejeon, South Korea
torwayafarer@kaist.ac.kr

Yin Song
School of EE
KAIST
Daejeon, South Korea
pineyin@kaist.ac.kr

ABSTRACT

Tag prediction is a trendy topic and can be applied in various fields, like question-answer communities, social media posts, and online shopping items.

Most approaches in question-answer applications use the contents of questions to predict the tag. In this project, we tackle the problem from a different perspective: we predict the tag based only on the questions-answerer-tag relation information.

This approach is more general, meaning that we can transfer the problem to another application and is less computationally expensive since we do not use NLP or Vision. However, after carefully analyzing the data, we claim that the problem is difficult conceptually because there is a weak correlation between similarity in tags and similarity in answerers. We propose and compare several methods in terms of the F1-score.

1. INTRODUCTION

The problem we want to solve is the following:

- GIVEN: a collection of questions with known answerers and tags.
- FIND: tags for target questions only by knowing the answerers.
- to MAXIMIZE: F1-score over validation sets.

We classify our methods into the following groups: baselines, similarity-based, association rules, and graph-based methods. Since the project's primary goal is implementing a novel or advanced algorithm rather than improving the F1-score, we develop a new algorithm - Link Prediction by Auto-Encoder, which we believe can be used for other graph-based applications that might show a higher F1-score.

.

The contributions of this project are the following:

- Our proposed *Link Prediction with Auto-Encoder* is novel, in the sense of modelling the problem in a tripartite graph, find optimized representation for auto-encoder inputs and defining a loss function which leads to link prediction.
- We show analytically and empirically that linear similarities in answerer could not lead to appropriate tag prediction.
- We implement weak and strong baselines for a fair comparison of our methods.

We start by analysis of data and suggest several methods based on intuitions.

First, we implement and compare similarity-based and association rule algorithms, which assign tags to a question using hand-crafted relations and rules in the data.

Next, we implement graph-based algorithms, which can benefit from finding structural dependencies from the question-answerer-tag graph. There is various way to represent the graph assigning nodes to all categories (questions, answerers, tags) or representing some of them as nodes and the other as edges. In addition, we cannot benefit from node features since questions do not have any contents information, so we rely entirely on the graph structure. The credit flow and community detection are among these methods[1].

The final approach is to find a non-linear embedding for questions with neural networks, leading to better representation for tag prediction. This task is done by using an auto-encoder with carefully masking tags in training for improving link prediction performance at test time[2].

2. PROPOSED METHOD

2.1 Baselines

As a baseline for this project, two methods are suggested.

2.1.1 *Random Selection (weak baseline)*

Assigning tags to target question uniformly random over all of the tags is the weak baseline. Instead, we select the tags with probability relates to their frequency in the train set as our first baseline.

2.1.2 *Common Tag Assignment*

In this baseline, we select the top-K most popular tags for all of the queries in the test. This baseline relatively outperforms other approaches. We implement two methods for selecting popular tags.

1) Most Frequent Tags: counting the number of appearances of each tag in questions and select the top-K most frequent.

2) Page Rank over Tags: We start by giving the same constant amount of credit to each tag (e.g. 1) then for every loop, we follow four below steps:

1. Each Tag split the credit between their Questions
2. Each Question split the credit received from Tags between their Answerers
3. Each Answerer split the credit received from Questions between their Questions
4. Each Question split the credit received from Answerer between their Tags.

After every iteration, the difference between the current and previous scores will reduce until a fixed ranking appears.

The ranking from both approaches is similar at first (e.g. till top-15 tags), but after that, they start to diverge, and PageRanking outperform the frequency in both recall and precision.

2.2 Similarity-Based Method

This method follows the intuition that question with similar answerer might have similar tags. This task can be described in three-step:

1) We find a set of similar questions for the target question—the similarity measured by Jaccard Similarities between sets of answerers. Answerers for query q is denoted as A_q.

$$Sim(q_1, q_2) = \frac{A_{q_1} \cap A_{q_2}}{A_{q_1} \cup A_{q_2}} \tag{1}$$

2) We define the tag similarity score for every tag that appears in one of similar questions to our target query as summation of all similarity score of related questions. q^* denotes the target question and Q_t the set of all similar questions which contain tag t.

$$S(t,\ q^*) = \sum_{q \in Q_t} Sim(q^*, q) \tag{2}$$

3) The final score for every tag computed by multiplying the tag probability score in the tag similarity score.

$$Score(t,\ q^*) = Pr(t) * S(t, q^*) \tag{3}$$

4) We select the top-K highest score tag regarding the target question as to the prediction.

$$Tag(q) = Top(K, Score(T, q^*)) \tag{4}$$

2.3 Association Rule Methods

Selecting tags individually for every question can lead to assigning an inconsistent set of tags for one question. To this end, we also measured the *Support* for every set of tags to improve the consistency. Also, we compute the *Confidence* of every set of tags by having the answerer.

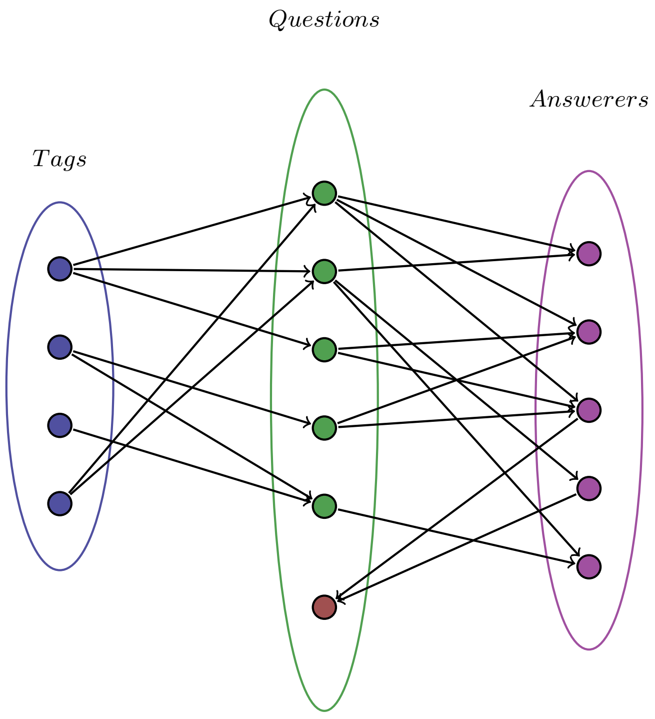


Figure 1: Graph of Tags, Questions and Answerers

$$\begin{aligned} \text{supp}(t) &= \frac{|\{t \subseteq T\}|}{|T|} \\ \text{conf}(a \Rightarrow t) &= \text{supp}(a \cup t)/\text{supp}(a) \end{aligned} \tag{5}$$

The support set plays the role of the prior distribution of every set of tags, and if we consider the answerers as evidence, we can assume the confidence is the posterior distribution of tags. Following this intuition, we can measure the probability of every tag based on our answerers set.

$$P(T|A) = \textbf{Supp}(T) * \textbf{Conf}(A \Rightarrow T) \tag{6}$$

In this method, the final probability considers as the ranking, and we select the most common sets from this score.

Discard that this method has better precision and recall is low and could not show an outperforming F1-Score.

2.4 Graph-Based Methods

2.4.1 *Credit Flow Tag Prediction*

We design the graph by assigning a node for every question, tag, and answer. The edges defined between questions and answerer if the user answered the question and between tag and question if that tag labels question. As a result, the Fig.1, a multi-partite graph is created. The red node in the question part denotes the question in the test.

The edges are directed, and they show the flow of credit through the graphs. In the beginning, we start by giving credit to every tag equal to its popularity. (same as baselines)

Then we split the credit for a related question and after that for a related answerer. We do not mix the credit of different tags and only add up the scores in the same tag.

In the end, every answerer has a score for every tag. Then answerers send these scores for the target question, and the target question picks the top-K tag with the highest score.

Another variation of this algorithm does not split the credit and transfer it directly for all the neighbours.

More results for different variation can be found in Appendix.

2.4.2 Community Detection

The *Common Tag Assignment* baseline outperforms other methods by far, and this can also be reasoned as a result of low similarity correlation between answerer and tags.

Here we try to go in a different direction which is more similar to baseline. This method follows four steps:

- Define communities based on tags.
- Find the community of each answerer.
- Find the community of target question from answerers.
- Assign the most common tag from the target community.

This method proposed is a progress report. Although community detection leads to pruning weak tags and the popular tags remained in the same community; therefore the result became similar to considering all the tags in one community and did not differ from baselines.

The techniques we used for community detection on *mathoverflow* is betweenness, and for *stackoverflow* we first done one step edge pruning with weights and then applied betweenness due to the massive amount of nodes and link the computation of betweenness was expensive from the beginning.

The result did not include in the final graph since they were identical to *top-rank*.

2.4.3 Link Prediction with NN

Another way to face this problem is to link prediction between the graph's target question and tag nodes.

The first idea was to train an Auto-Encoder for encoding the graph in latent representation that when we apply the decoder missing links can be generated.

The whole graph can be represented as the following matrix:

$$\begin{array}{c|ccc} & Q & A & T \\ \hline Q & 0 & \alpha & \beta \\ A & \alpha & 0 & 0 \\ T & \beta & 0 & 0 \end{array}$$

However, since this matrix is sparse even in the case of α, β, we came with the following representation: the input and output of Auto-Encoder.

$$\begin{array}{c|cc} & A & T \\ \hline Q & \alpha & \beta \end{array}$$

The whole process shown is Fig.2.

For every question, we use a one-hot vector of all answerer and tags as feature embeddings. Values in these vector can be one of $\{0, 0.5, 1\}$ which denotes states of *No-Link*, *Unknown*, and *Linked*.

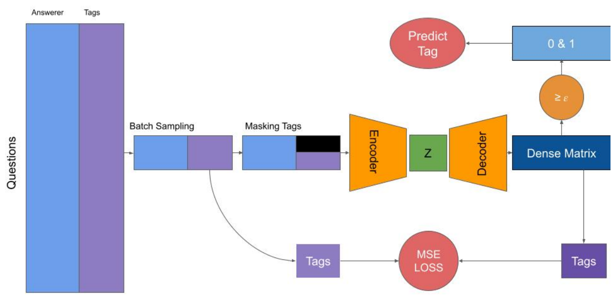


Figure 2: Graph Auto-Encoder Train and Test process

During inference, the question embedding use a one-hot vector for the answerer and put 0.5 in tags embedding, then feed it into Auto-Encoder. We select the link from the generated matrix if the value passed above the threshold.

In the training phase, for every iteration, we sample a batch of question from training. We mask a fraction of the tag embedding with 0.5 as *Unknown* tags. The Auto-Encoder generated a dense matrix. For optimization, we apply gradient descent by measuring the MES loss between sampled batch and the generated one.

Since this training did not converge to the result better than baseline, we start applying hyperparameter tuning and some modification to the training procedure while described below.

- Tuning Hyper-parameters: Size of layers, activation functions, learning rate, batch size, number of epochs
- Scheduling Mask Ratio: Instead of using a random or fix mask ratio, we gradually increase the ratio to create a curriculum for learning from easier task to harder ones.
- Adding Regularization: L1 and L2 regularization added, but since the main loss signal is relatively weak, regularization applied with small coefficients.

In the end, for the *mathoverflow* the representation was good enough to predict the tags as well as *top-rank* baseline, but the training on *stackoverflow* failed to learn proper representation for tag prediction.

This issue can be reasoned since the dimensions in **stackoverflow** are around 17K compared to 7K in *mathoverflow*, but we do not have a significantly larger dataset for training concerning dimensions.

The second time we trained only a NN from the answerers' matrix to tag matrix. The result was almost the same but with fewer parameters, therefore, a faster training rate.

The final result has been compared with strong baseline in Fig.[15, 16, 17, 18] and with all other methods in Fig.[7, 10, 12, 14].

2.5 Evaluation Methods

For evaluation of our methods, we used several metrics. One metric cannot fully understand the weaknesses of our model, so we used a combination of the following.

1. **Accuracy**: is the fraction of correctly predicted data point out of all data points. The problem is that we assign an equal cost to false positives and false negatives when calculating it. So, even though Accuracy is very intuitive, it does not work well for imbalanced classes.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

2. **precision**: is the fraction of relevant instances among the retrieved instances. Intuitively, it means how many of our predicted tags are correct.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

3. **recall**: is the fraction of relevant instances that were retrieved. Intuitively, it means how many of the correct tags we predicted.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

4. **F1 Score**: is the harmonic mean of precision and recall. F1 score conveys the balance between the precision and the recall. In order to get a high F1 score, the model should predict both predict a large fraction of all tags truly corresponding to this question, and a large fraction of the predicted tags should be truly relevant for the question.

$$\begin{aligned} F1\_Score &= 2 * \frac{Precision * Recall}{Precision + Recall} \\ &= \frac{2TP}{2TP + FP + FN} \end{aligned} \tag{10}$$

5. **ROC & AUC**: are the performance measurements for the classification problems at various threshold settings. AUC (area under curve) is the area ROC (probability curve). ROC curve is plotted as True Positive Rate (TPR) on y-axis vs False Positive Rate (FPR) on x-axis. AUC is a good metrics to estimate how different you model prediction is from random prediction.

...

3. EXPERIMENTS

3.1 Data Analysis

3.1.1 Data Histograms

We plotted histograms (Fig.4) to see the different data distributions. We used median instead of mean because the distribution is skewed, and the mean usually shifted to the tail-side. The data below is for **MathOverflow** dataset. Similar distributions are observed for the StackOverflow dataset.

The histograms can be interpreted as follows: how many <y axis label> has <x axis value> number of <x axis label>.

1. **Number of Questions per Number of Answerers (Fig.4.a)**:most of the questions have only 1 answerer (median is 1). However, the distribution is very long-tailed, meaning that some of the questions were answered by many people (max 173).

2. **Number of Answerers per Number of Questions Fig.4.b)**: most of the people answered around 5 questions (median is 5). Some people answered a vast number of questions (max is 1784).

3. **Number of Questions per Number of Tags Fig.4.c)**: we can see that a question has at most five tags and at least one tag assigned to it. Most of the questions have three tags, so we assign this number of most common tags (or random tags to a question) when preparing baselines.

4. **Number of Tags per Number of Questions Fig.4.d)**: there are some trendy tags, which have been assigned to many questions. For example, the most popular tag corresponds to 7931. The median value is 17, meaning that, in general, tags are not very popular.

5. **Number of Answerers per Number of Tags Fig.4.e)**: most of the answerers answered the questions, which were assigned around nine tags (median is 9). However, some answerers answered questions, having a huge number of tags (max is 521).

6. **Number of Tags per Number of Answerers Fig.4.f)**: median value for the number of answerers, which a tag might have, is 23. Distribution is strongly skewed to large values, with the maximum number of answerers per tag equal to 2355.

We also measured **Number of Questions per Number of Candidate Tags**: number of tags, which might be assigned to a question, because this question's answerers answered other question with the corresponding tags. In general, we can see a large number of candidate tags (median is 146) with max 1137 and min 0 - meaning that some answerers have no possible tags because they have not answered other questions.
Main Conclusions on Data Histograms:

- We should assign on average three tags for a question
- Some questions have very a large number of candidate tags
- Some tags are trendy (corresponding to 7000 questions), so it makes sense to assign the most popular tags for questions and achieving a good baseline score

3.1.2 Number of Tag-Answerer Correlation

As shown in Fig.5, there is no strong correlation between the number of tags and the number of answerers, therefor is shows that using answerer quantity does not help for prediction tags quantity.

3.1.3 Jaccardian Similarity

The heatmaps (Fig.6) can be interpreted as follows: the x-axis is the similarity in answerers, and the y-axis is the similarity in questions. The colour represents the total number of questions having similarity in questions and similarity in tags in that range. The data correspond to the validation set, assuming similar distribution to the test set.

Our results show that high similarity in answerers does not lead to high similarity in tags. For StackOverflow (Fig.6.a,b), largest number of questions have high similarity in answerers and low similarity in tags, while for MathOverlow (Fig.6.c,d)

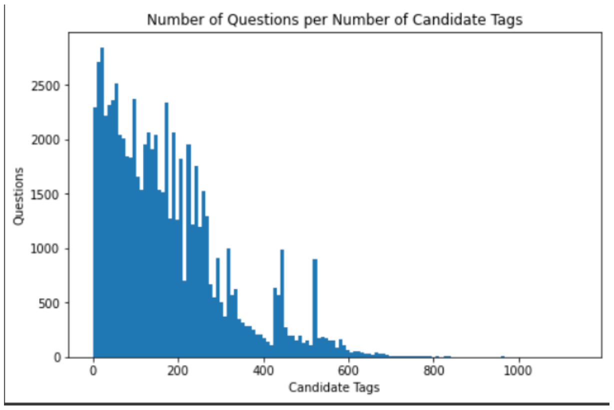


Figure 3: Number of Questions per Number of Candidate Tags

most questions have low similarities in bot answerers and tags.

Conclusion on Jaccard Similarity: high similarity in answerers does not lead to high similarity in tags, which makes this problem difficult for all approaches since it is difficult conceptually.

3.2 Proposed Methods Experiments

According the evaluation methods, the quality of every approach is measured. The Fig.{7,8} show the recall plots for all algorithm and the Precision can be found in Fig.{10,9} for Mathoverflow and Stackoverflow data sets.

Furthermore, the plots for Precision to Recall and F1-Score is shown in Fig.{14, 13, 12, 11}.

The overall results show that **Top-Rank** and **Top-Freq** outperform all other methods in case of a recall, precision and therefore, f1-score.

On the other hand, both approaches for from **Graph Credit Flow** and **Similarity Measure** perform similar, which might be counted as equivalent algorithms.

For the **Associative Rules** we see inferior performance as the number of tags increases, and after passing the maximum tag limits, the algorithm does not suggest any tags, which leads to zero recall and precision.

Selecting more tags for prediction will increase the recall while decreasing the precision.

In Stackoverflow data set, for **Top-Rank** and **Top-Freq** selecting 6 tag shows the highest F1-Score, while for **Graph Credit Flow** and **Similarity Measure** selecting 5, it can.

In the Mathoverflow data set, all approaches show the best F1-Score when three tags are selected.

4. CONCLUSIONS

Ignoring the relationship between answerer and only focusing on tags popularity shows good performance as a baseline.

On the one hand, it is reasonable to expect side information (such as answerers) to improve baseline quality. On the other hand, the low similarity correlation between answerers and tags shows there is not much information that can be found based on one-hot vector embedding and Jaccard Similarity.

This statement also shows empirically by **Similarity** and **Credit Flow** methods.

We worked on non-linear embedding for question with neural networks, which leads to a better representation for tag prediction, and result match the baseline.

Also, as we tried to improve the baseline, by removing **False Positives**. Instead of detecting the tags separately, we try to find several communities and find an appropriate community for the given question. This method could not remove false positive since all popular tags fall inside same community.

5. REFERENCES

[1] Zhangtao Li, Jing Liu, Kai Wu. *A Multiobjective Evolutionary Algorithm Based on Structural and Attribute Similarities for Community Detection in Attributed Network.* in IEEE Transactions on Cybernetics, vol.48, no.7, pp.1963-1976, July 2018, doi: 10.1109/TCYB.2017.2720180.

[2] Tran, Phi Vu. *Learning to Make Predictions on Graphs with Autoencoders.* in 5th IEEE International Conference on Data Science and Advanced Analytics.

APPENDIX

A. APPENDIX

A.1 Labor Division

The team performed the following tasks splitting the roles equally:

- Data Analysis [all]
- Baseline Methods [Yin, Pyatykh]
- Similarity-based Methods [Rahimi, Yin]
- Association Rules-based Methods [Rahimi, Pyatykh]
- Graph-based Methods [Rahimi, Pyatykh]
- Literature Review [all]
- Paper Writing [all]

A.2 Full disclosure wrt dissertations/projects

A.2.0.1 Rahimi:.

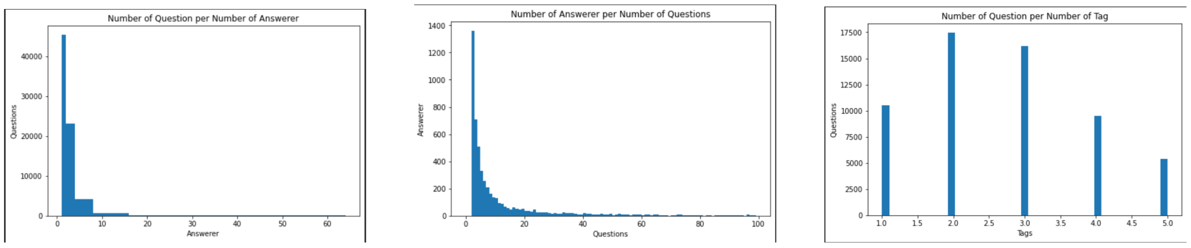
He is not doing any project or dissertation related to this project: his thesis is on reinforcement learning and multi-agent systems.

A.2.0.2 Pyatykh:.

He is not doing any project or dissertation related to this project: his thesis is on super-resolution enhanced video steaming.

A.2.0.3 Yin:.

She is not doing any project or dissertation related to this project. She is an exchange student, and her thesis in the homeland is on-road detection based on point clouds.


(a) Number of Questions per Number of Answerers (b) Number of Answerers per Number of Questions (c) Number of Questions per Number of Tags

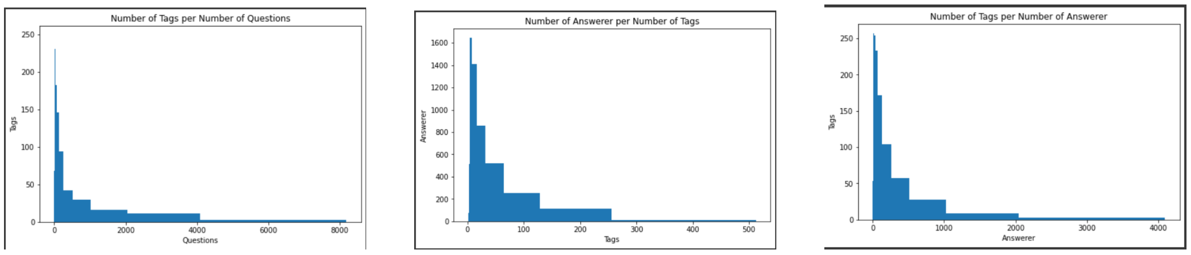

(d) Number of Tags per Number of Questions (e) Number of Answerers per Number of Tags (f) Number of Tags per Number of Answerers

Figure 4: Data Distributions

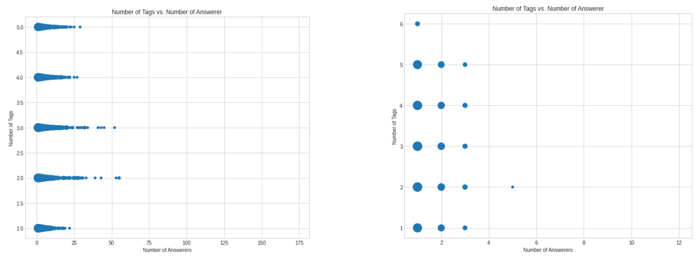
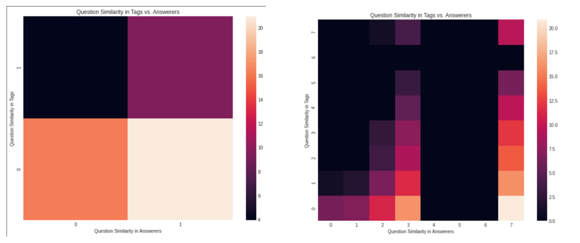

(a) Mathoverflow (b) Stackoverflow

Figure 5: Correlation between number of tag and number of answerer.


(a) StackOverflow Tags Similarities vs Answerers Similarities (b) StackOverflow Tags Similarities vs Answerers Similarities (fine)

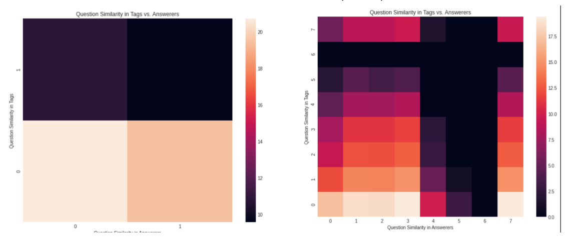

(c) MathOverflow Tags Similarities vs Answerers Similarities (d) MathOverflow Tags Similarities vs Answerers Similarities (fine)

Figure 6: Jaccard Similarities

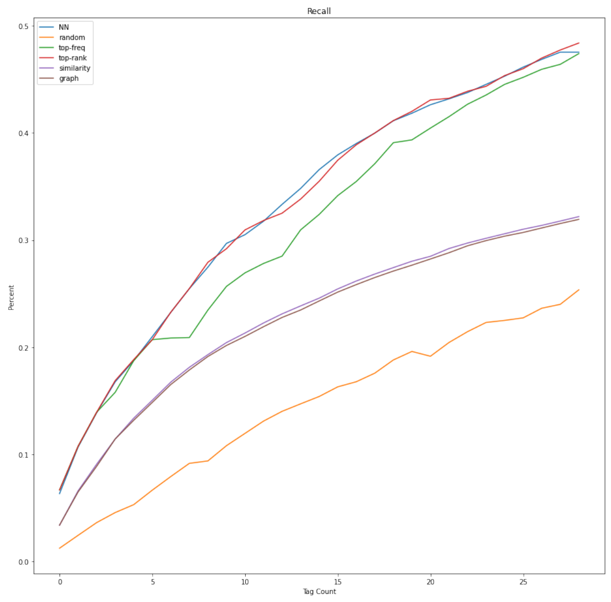


Figure 7: Recall plots for Mathoverflow

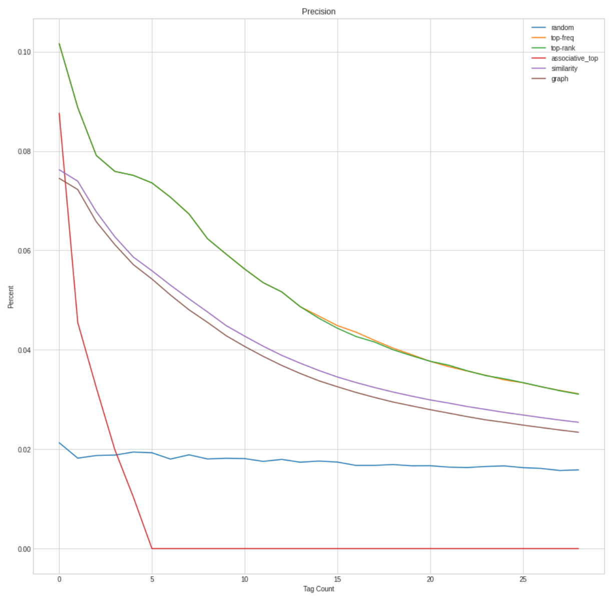


Figure 9: Precision plots for Stackoverflow

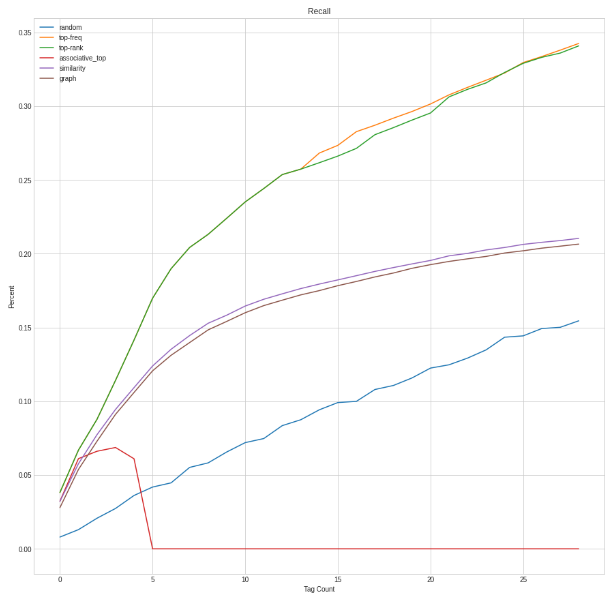


Figure 8: Recall plots for Stackoverflow

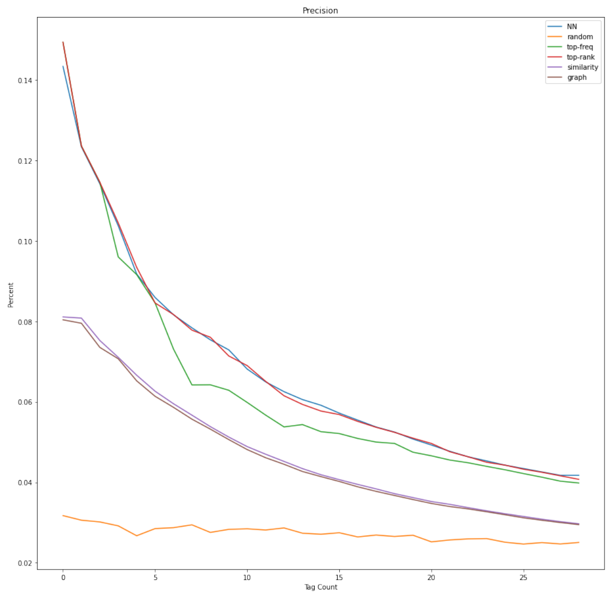


Figure 10: Precision plots for Mathoverflow

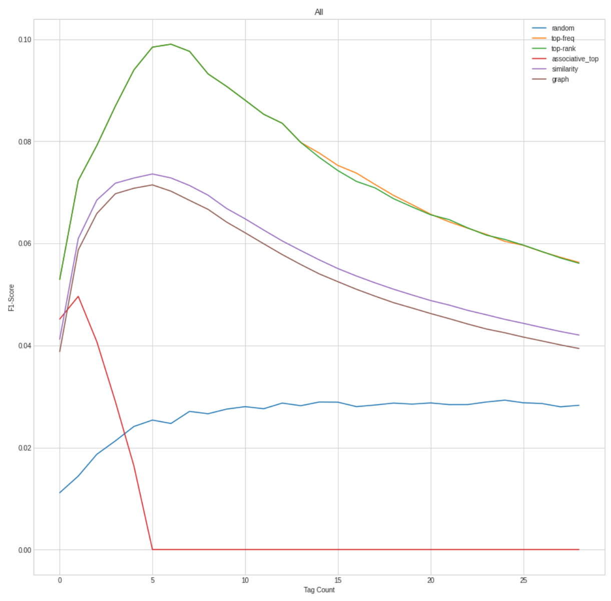


Figure 11: F1-Score plots for Stackoverflow

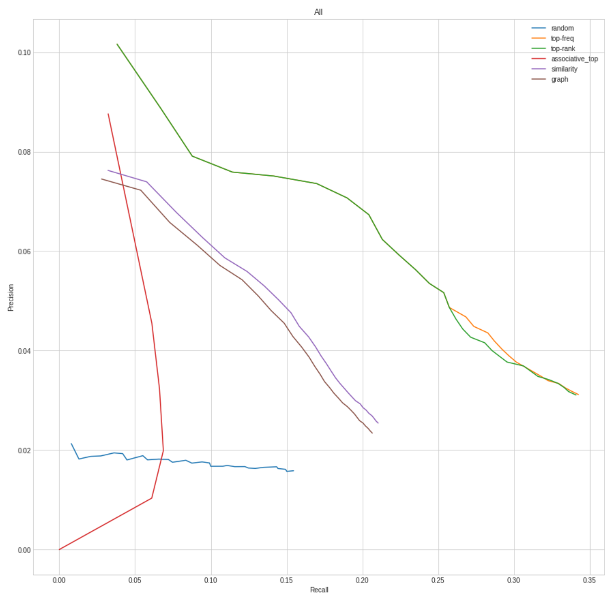


Figure 13: Precision per Recall plots for Stackoverflow

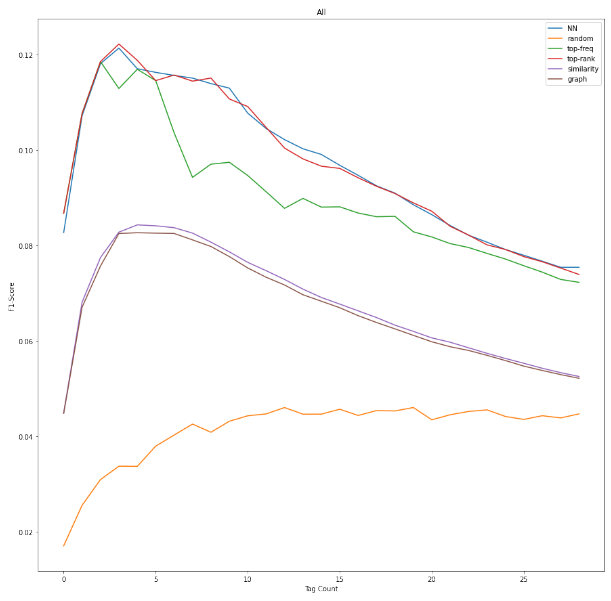


Figure 12: F1-Score plots for Mathoverflow

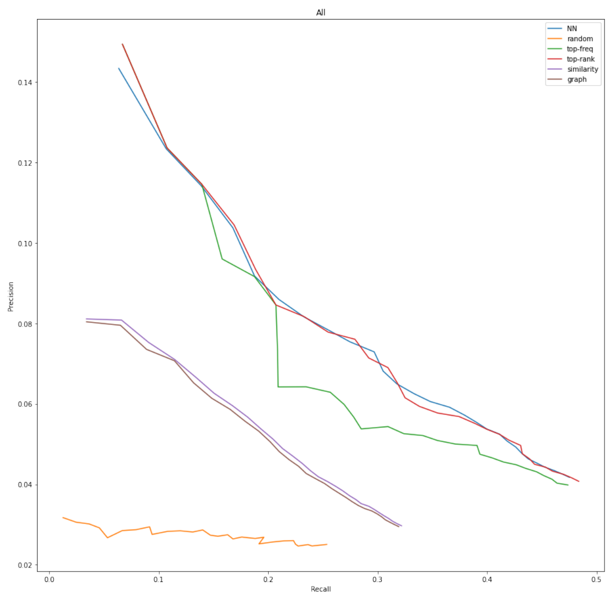


Figure 14: Precision per Recall plots for Mathoverflow

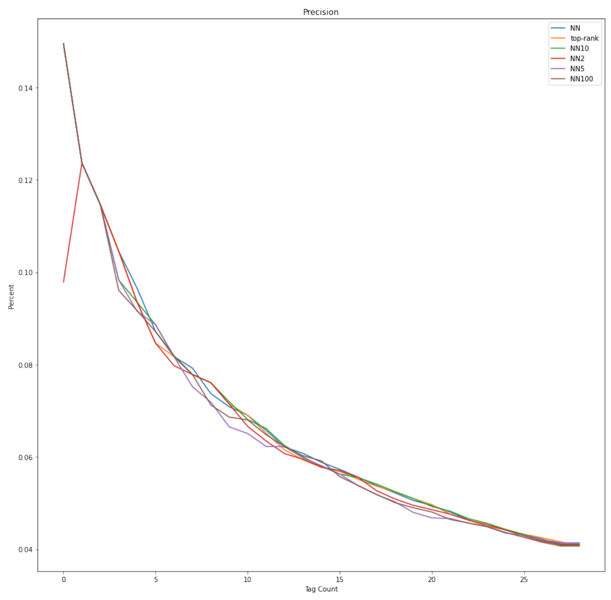


Figure 15: NN Precision plots for Mathoverflow

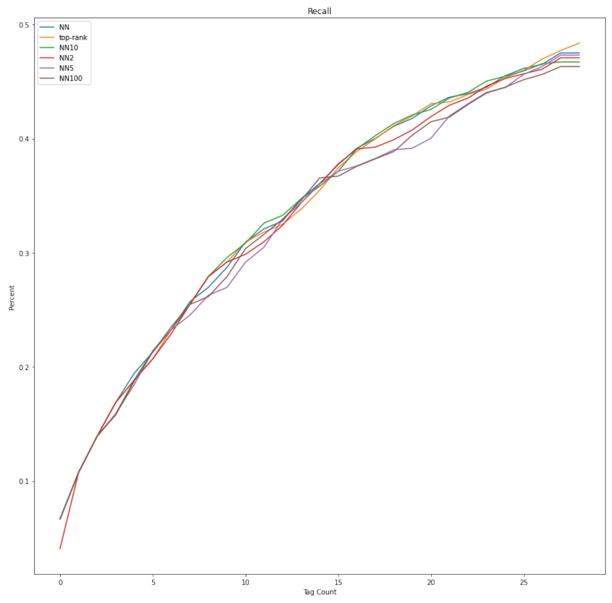


Figure 17: NN Recall plots for Mathoverflow

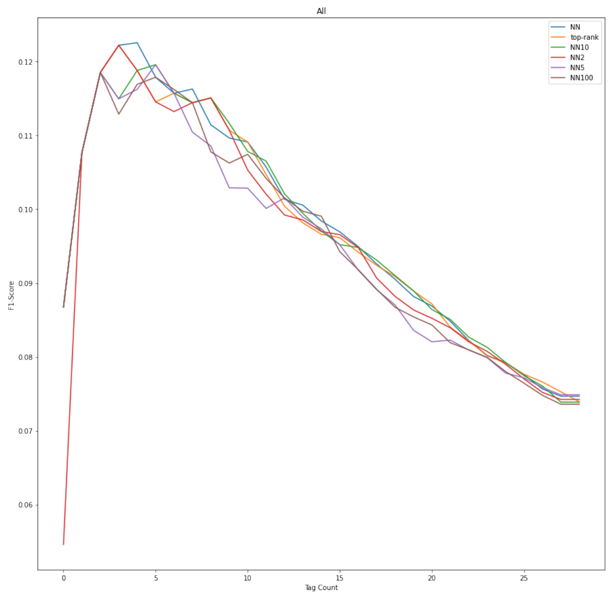


Figure 16: NN F1-Score plots for Mathoverflow

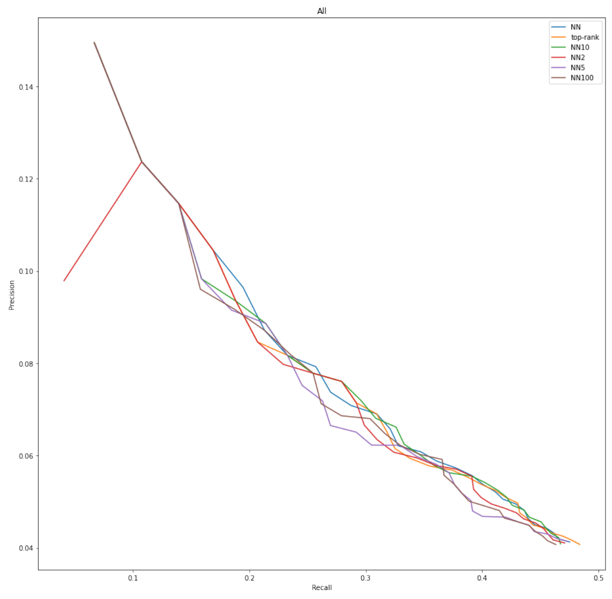


Figure 18: NN Precision per Recall plots for Mathoverflow