Does Feature Mining Help Neural Networks?: In Perspective of Tag Prediction

Miyoung Ko
Graduate School of AI
KAIST
Seoul, South Korea
miyoungko@kaist.ac.kr

Soyoung Yoon
Graduate School of AI
KAIST
Seoul, South Korea
lovelife@kaist.ac.kr

Kyumin Park
School of Computing
KAIST
Daejeon, South Korea
pkm9403@kaist.ac.kr

ABSTRACT

In the rise of online question-answering platforms, question tag prediction has been an important problem in order to reduce redundant questions and to provide related information to users. Many recent works utilize content information such as question text, to improve tag prediction. However, without having additional content information other than graph structures, most methods do not effectively work for question tag prediction. In this environment, we test whether a combination of feature mining approach and neural network-based approach to overcome this shortcomings. Through our experiments, we conclude that concatenating additional features such as answerer importance, and relational tag clustering does not help on the neural-network based question tag prediction.

1. INTRODUCTION

Since questions and answers are one of the most fundamental methods to gather and share information, arise of online platforms[1][2] that enables share questions and answers in each domain has been natural in the era of internet. In order to reduce redundant questions and provide related information to users, the question answering platforms introduce tag attached to each query. Representing characteristic of each query as a set of tags, platforms categorize and cluster similar questions and use these characteristics for several services including recommendation. Since incorrect tag or too high delay would be not accepted for users, accurate and efficient tag prediction is needed for applying tag system to service.

Studies to predict tag efficiently and accurately already exist in several domains [4, 2]. However, many of the previous works mainly focus on the content, i.e., question and answer text [1, 6]. Even these methods successfully predict tags of each query, models have been taught relation between query characters and tag, not relation between queries. This does not fit our environment, content-related information is not provided.

In our environment, no content-related information including question or answer text is provided. We need to utilize limited information to predict appropriate tag. Given information is: the set of query index, tag of some queries (train/dev), and answerer ids with query indices that each answerer responds. Using these information, we need to predict tag of queries whose ground truth is not provided. Therefore, strategy different from previous text-based tag prediction is required to solve this problem.

In order to predict appropriate tag with restricted information, we use graph structure to represent relationship among queries, tags and answerers. Since given information consists of relation between some subjects (e.g., answerer-query, query-tag), we can build several graph structures with various node-edge configuration. However, expected problem of graph construction exists that the number of tags to predict is too big to map from single graph. To resolve this large-label problem, we extract available feature from existing data then add features within graph, then use this feature as an input for graph neural network, e.g., node2vec [3] or graph convolution network [5]. Also, we vary prediction scheme rather than computing probability of each tag, including hierarchical prediction or tag embedding.

Summing up, we solve tag prediction problem with several graph representation, feature engineering for graph, and various tag prediction strategy. We argue that the contribution from our proposed method is:

- We propose two different graph representation to compute varied possible tag-related features.
- We add engineered feature and graph neural network methods to utilize limited amount of data.
- We propose various feature engineering and mining to find appropriate characteristics from dataset.

2. PROPOSED METHOD

To predict the tag of each question, we first construct two types of graphs as shown in Figure 2. The first structure is the homogeneous graph with only one type of node, which represents the answerer. When the pair of answerers answer the same question, corresponding nodes are connected by the edge. The second is a bipartite graph with answerer nodes and question nodes. Each edge between answerer and question represents a relationship that question answered by

[1]https://stackoverflow.com/
[2]https://mathoverflow.net/

.

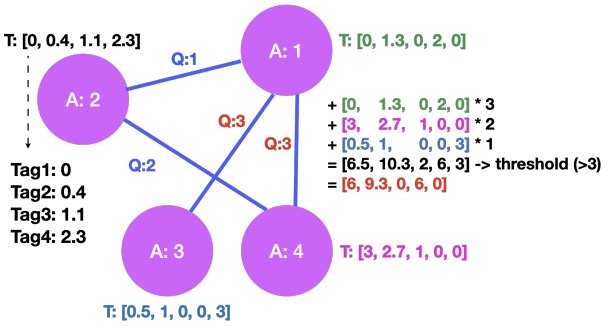


Figure 1: Illustration of TagRank. Nodes indiate the Answerers, and edges indicate the questions that each answerer answered. Tag information are stored in each node (Answerer), weighted in importance. Given that question 3(red) is from the valid dataset, we can calculate the tag information by aggregating the weighted answerer tag information by the adjacent nodes and thresholding it. The weights for each answerer are defined by the number of questions each answerer answered. Normalization function is omitted for clear explanation.

the answerer. We call each structure as *Answerer graph* and *Question-Answerer graph*.

With two types of graph structures, we propose four methods: (i) TagRank, (ii) Answerer GCN (Ans GCN), (iii) Question-Answerer GCN (Q-Ans GCN), and (iv) GCN with TagRank. TagRank is the data mining approach where it aggregates the answerer information to predict the question tag. The second and third methods utilize GCN (graph convolutional network) structure on each graph structure. Lastly, we propose the combined method to overcome the limitations of TagRank and GCN-based methods.

2.1 TagRank

2.1.1 Motivation

We start with the simple intuition that tag information with respect to answerers whom answered the target question should be aggregated to predict tags from an unknown question. We believe that this approach is novel in that we didn't convert question information and tag information into nodes, and rather converted it to **relational features** with respect to answerers. However, we additionally thought that answerers who answered more questions are likely to be experts in the field, therefore the tags that those answerer answered are more likely to be trustworthy. Therefore, we multiply importance scores from each answerers by the number of questions each answerer answered to give a weighted sum of tags when aggregating tag information from related answerers. We also list the hyperparameters that we tuned for each dataset on Table 1.

2.1.2 Calculating tag id predictions

Fig 1 illustrates how TagRank are calculated. We will now describe in detail with examples about how we implemented our TagRank method.

Tagweight assignment to answerers. We first construct an answerer tagweight. Suppose Q_i is the set of question ids that answerer number i (A_i) answered. Than for all question id q $\{\forall q \in Q_i\}$ that are in Q_i, we sum all the related tag ids t, which is $\{\forall t \in q\}$ related to question q for all q, *with* repetition. This means that tag ids that appear multiple times in multiple questions related to a particular answerer can have a higher weight compared to tag ids that appeared only once throughout the summation. The output of summed tag ids for each answerer i are then represented as bag-of-words. For example, if the total of all tags is 5 and answerer 1 (A_1) had summation of 2 t_1 and 1 t_3, the tag weight vector of answerer 1 can be represented as vectors with values as frequencies, where each index corresponds to the tag id. Therefore, the tag vector becomes [2, 0, 1, 0, 0], meaning answerer 1 has 2 occations of t_1 and 1 occation of t_3.

Hyperparameters	Tested values	Meaning
Value threshold	0.01∼0.03	Threshold for tagweights
Minimum length	1∼2	minimum number of tag ids in each question.
Maximum length	1∼10	maximum number of tag ids in each question.
Frequent tag length	1∼5	number of most frequently appeared tags
Frequent tag threshold	1∼5	Threshold value to determine frequent tags
Consider importance	TRUE or FALSE	Whether to consider the answerer importance.

Table 1: Explanation of experimented hyperparameters for TagRank.

Normalization. For fair comparison, we then normalize the values to make the sum be always 1. In this example, the tagweight vector will become [2/3, 0, 1/3, 0, 0].

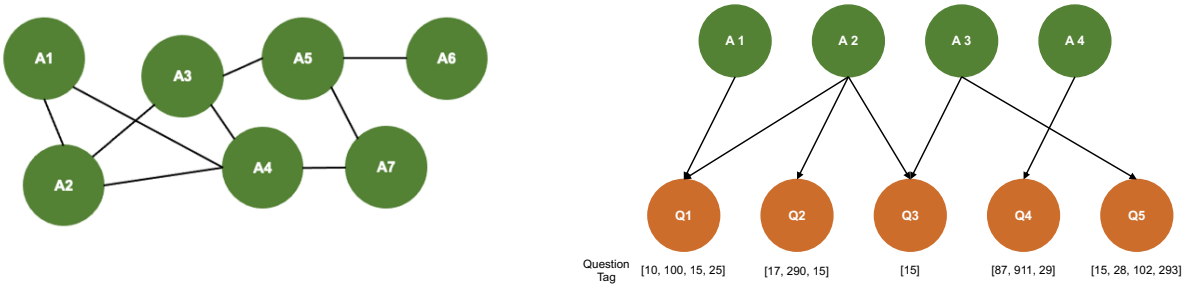
Thresholding. To get rid of unncessary information and only leave out important information, we threshold the tag values. Suppose the tagweight vector for answerer i is v_i, the output vector becomes $F(v_i)$. Given the thresholding value k, the thresholding function $F(t)$ works as follows:

$$F(t) = \begin{cases} 0 & \text{if } t < k \\ 1 & \text{if } t \geq k \end{cases}$$

In our example, if k=0.5, the output tagweight vector then becomes [2/3, 0, 0, 0, 0]. We are now done calculating the tagweight vector v_i for answerer i.

Tagweight assignment to question ids. We now calculate the tagweight vector for unknown question k q_k. For all answerers that answered question q_k - that is, $\forall A_v \in (\exists q_k \in Q_v)$, we sum the tagweight vectors of all A_v. When we consider importance, we do a weighted sum of each tagweight vectors by the number of questions each answerer answered. Than, we normalize the weighted sum of tagweights from all related answerers.

Argsort & output. We then sort the values and pick up the indexes. During this process, we first select minimum length amount tag ids no matter the values are (minumum length). Than, we discard indexes that have 0 values, or have lower values than value threshold (explained at Table 1). We then select the rest until we reach maximum length. If the total number of selected tag ids are smaller than the frequent tag threshold, we replace the selected tag ids to top-k frequent tag ids (selected by frequent tag length). The final output becomes the predicted tag ids for unknown question k.



(a) Answerer Graph Structure

(b) Question-Answerer Graph Structure

Figure 2: Illustration of two graph structures. (a) Answerer graph structure only has answerer nodes, and a question is represented as an edge. Connection between two nodes indicates that a question is answered by the answerers. (b) Question-answerer graph is a bipartite graph that each node represents answerers and questions. The edge between a answerer and a question node reflects whether each answerer answers each question.

2.2 Answerer GCN

Using same graph structure as TagRank method, we can introduce graph convolutional network. Manual algorithm to compute answerer feature has limitation that non-perceptible or indirect features are not considered, and this limitation can be alleviated with neural network-based methods.

Given answerer graph in Fig 2-(a), let j-th node has d-dimension embedding $e^{(j)}$. Here, the goal becomes to find best embedding so that the combination of embeddings best predicts tags of each query. Randomly initialized embeddings are computed to contain answerer information with graph convolutional network. Next, we aggregate answerer embedding for each query and compute summation of aggregated embedding. Regarding the summed vector as latent vector of query, we predict the tag of the query using fully connected layer: $h^{(i)} = \sum_{j \in q^{(i)}} e^{(j)}$ where $q^{(i)}$ denotes i-th query and $h^{(i)}$ is representation vector of $q^{(i)}$. As an output of the model, logit vector that each space represent single tag is obtained. We use binary cross entropy as a loss calculation on aggregated embeddings, since there is no restriction in the number of tag for each query.

2.3 Question-Answerer GCN

Unlike the previous approach, with a question-answerer graph, the tag prediction problem formulates as the node classification. Question-answerer graph consists of two types of nodes; question nodes (q) and answerer nodes (a). For i-th question node $q^{(i)}$, the corresponding tag of the question is now represented as a node label, $y^{(i)} = [y_1^{(i)}, y_2^{(i)}, ..., y_T^{(i)}]$ where $y_k^{(i)} \in \{0, 1\}$ and T is number of tags. Our goal is to obtain a node representation of each question node and predict a tag label using the representation.

We utilize a similar GCN structure as in Method 2. Both question and answerer nodes are first embedded to d- dimensional vectors. As explicit information on each question and answerer is not provided, we use a randomly initialized vector as the initial node embedding. We stack two layers of GCN and one fully connected layer with sigmoid activation on the embedding layer to predict labels. Additional residual connection from the embedding layer to the GCN output improves the performance. The model is optimized to minimize weighted binary cross-entropy loss on all train labeled question nodes:

$$L = \sum_{i=1}^{N} \sum_{k=1}^{T} \alpha y_k^{(i)} log(\sigma(h_k^{(i)})) + (1 - y_k^{(i)}) log(1 - \sigma(h_k^{(i)})) \quad (1)$$

where $h^{(i)}$ is the output of the last layer, α is the weight of positive examples, and $\sigma(\cdot)$ is the sigmoid function.

As the number of positive tags is much less than negatives, models easily converge to local minimum whose predictions are all negatives. To control this issue, we give additional weight α to positive classes.

2.4 GCN with TagRank

For GCN-based models, we initialize the node embedding as a random vector. In our problem setting, we only have connection information between questions and nodes without explicit feature information. However, a deep neural network like GCN is effective when it incorporates rich feature information of each node with connection information. To address this lack of information, we combine the data mining approach and the deep-learning approach.

We obtain answerer features and importance scores of answerers in TagRank. A set of tags answered by the answerer represent the answerer node and the importance of each answerer is calculated by simple statistics. This information can be used as the feature information of each node or edge in a graph. By combining this feature information in node embedding, GCN-based models have more opportunities to encode important information. We call this method GCN with TagRank.

We utilize the same graph and GCN structure as Question-Answerer GCN (Section 2.3). We combine TagRank results of answerer and question as the node feature. $T_n^{(i)}$ is the answerer feature of i-th answerer and $T_q^{(j)}$ is the question feature of j-th question. Both T_n and T_q are obtained by the equation (1). We first project T_n and T_q into d-dimensional vectors, using two fully connected layers for each question and answerer node. Input for the GCN layer is now converted as the sum of projected TagRank vectors and randomly initialized embedding vectors, calculated as below.

$$\begin{aligned} \hat{x_q} &= x_q + g_1(T_q) \\ \hat{x_n} &= x_n + g_2(T_n) \end{aligned} \quad (2)$$

	Mathoverflow			Stackoverflow		
	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)
Random (Bernoulli)	0.1818	49.9239	0.3619	0.0498	49.8710	0.0996
Random (Fixed # tags)	0.1846	0.3469	0.2303	0.0651	0.0145	0.0857
Frequent (Fixed # tags)	9.3216	18.8664	11.8599	7.3364	16.9640	9.8214
TagRank	34.7935	30.0350	30.3047	10.2872	12.9211	10.8066
Ans GCN	9.7021	8.4078	8.4194	5.2035	14.0905	7.1029
Q-Ans GCN	35.2126	33.5538	30.6928	10.1036	12.4742	10.1788
GCN with TagRank	32.6831	35.2241	30.1862	9.4689	12.4056	9.8705

Table 2: Results on different methods for tag prediction. Random bernoulli indicate random binary tag prediction output results (which can be treated as baselines), and Random fixed tags indicate random tag predictions on a fixed number of tag size. The two methods are values averaged over 3 different seeds. For mathoverflow, mathoverflow has average of about 5 tags per question, and stackoverflow has about 6 tags per question. Considering the statistics, we experiment with those baselines to show the effectiveness of our proposed methods. We report our best accuracy of each of our methods.

where x_q and x_n are the node embedding vectors, and g_1, g_2 represents linear layers for TagRank vectors. The combined inputs, $\hat{x_q}$ and $\hat{x_n}$, are feed into the same GCN model as in Question-Answerer GCN to predict the node labels.

3. EXPERIMENTS

For the entire experiment, we use query ids, tag ids for each query, and answerer ids for each query as inputs of all models. We report precision, recall, and F1 score of prediction inferred with two different datasets: Stackoverflow and Mathoverflow.

3.1 Baselines

We first experiment with naive baselines to see the effectiveness of our methods compared with random and simple methods. We experiment with three baselines: (1) random-bernouii, (2) random-fixed-ntags, and (3) frequent-fixed-ntags. For random-bernouii, for each question, we assign both (1) number of tags and (2) tag ids randomly. After inspection, we found out that the average number of tags per each question is 5 for Mathoverflow, and 6 for Stackoverflow. Therefore, we conduct a more precise approach by random-fixed-tags. Compared with random-bernouii, (1) number of tags is fixed to 5 for Mathoverflow and 6 for Stackoverflow. only (2) tag ids are defined randomly. Lastly, we conduct frequent-fixed-ntags, where (1) number of tags is fixed to the same as random-fixed-tags and (2) tag ids are chosen as top-k tag ids that appeared most frequently in the train query. random-bernouii and random-fixed-ntags are experimented with 3 different seeds and averaged.

3.2 Result & Discussion

Table 2 shows our results, including all methods proposed in Section 2 and the baselines. For the TagRank, we report implementation with importance rate, since importance rate is revealed to have insignificant influence to the performance. Hyperparameter for TagRank is reported in Appendix. For GCN with TagRank, we use Question-Answerer GCN model for GCN as described in Section 2.4.

It is clear that our proposed methods outperform random method baselines, which record 0.36 and 0.23 for Mathoverflow and 0.085, 0.099 for Stackoverflow. Since the task is selecting maximum 5 appropriate tags from 1429/6790 tags, bernoulli baselines should report low accuracy. Recall of bernoulli baseline (around 49%) proves that the tags are selected randomly, since the half of correct tags are predicted as query's tags. Random method with the fixed number of tags reports lower F1 score than the baseline without tag number restriction, since the number of predicted tags gradually decreases.

Compared to two random method baselines, our methods report higher precision, recall and F1 score. This proves effectiveness of tag prediction based on previous query-tag information and query-answerer information. Also, our assumption - one answerer may answer to similarly-tagged query, and answerers who answer to same query will appear spontaneously in another query in similar tags - is proved through the results, which record 20 to 100 times higher F1 score than random method baselines.

Among the proposed methods from TagRank (Section 2.1) to Quention-Answerer GCN (Section 2.3), Answerer GCN (Section 2.2) records lower overall score than the other methods and frequent method baseline. We suspect that this low score results from shortage of information. For Answerer GCN, we only construct answerer graph, and represent query and tag information by summation of some nodes in answerer graph. This method to utilize given information is insufficient for deep learning-based method, which needs larger amount of data for training. TagRank method (Section 2.1) record higher score than Answerer GCN since it uses explicit feature mining rather than deep learning, where feature mining provides clearer characteristic than deep learning method for each query to predict the tags. Question-Answerer GCN (Section 2.3) utilizes query and tag information when constructing graph, which provides larger information during training.

When merging TagRank method and GCN-based method as described in Section 2.4, we use Question-Answerer GCN as a GCN-based method since Question-Answerer GCN records higher F1 score than Answerer GCN. The result shows that applying TagRank into Question-Answerer GCN does not increase the F1 score in either dataset. We suspect two reasons that applying TagRank does not help GCN performance: increased latent space and computed TagRank features in GCN training procedure. First, TagRank has larger feature space in oreder to represent all tag information in each node. This means, there exists no linear layer to expand latent vector to tag prediction space (size is equal to the number of tags), so that the TagRank nodes need to have dimension whose size is at least the number of the tags. Therefore, merging TagRank features into GCN may increase the dimension of latent vector, which makes feature

extraction more difficult than lower dimension with small size of data. Next, GCN training might already extract features defined in TagRank. During GCN, node vectors may already represent TagRank features including tag importance or answerer relativity. In this situation, adding TagRank features could be redundant for GCN model, which result in similar score with GCN-only model.

4. CONCLUSIONS

In this project, we explore the influence of feature mining toward the GCN-based method in tag prediction task. Throughout the experiments, we prove that both feature mining approach and GCN-based approach outperforms random baselines, and tested whether applying feature mining improves performance of GCN-based method. Since adding feature mining and GCN-based method does not improve performance compared to each individual model, it is concluded that adding feature mining does not help GCN-based method. According to our conclusion, tag prediction with additional information might be effective for more precise result.

5. REFERENCES

[1] José R. Cedeño González, Juan J. Flores Romero, Mario Graff Guerrero, and Felix Calderón. Multi-class multi-tag classifier system for stackoverflow questions. In *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–6, 2015.

[2] Francisco Charte, Antonio J. Rivera, María J. del Jesus, and Francisco Herrera. Quinta: A question tagging assistant to improve the answering ratio in electronic forums. In *IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON)*, pages 1–6, 2015.

[3] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[4] Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. Social tag prediction. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, page 531–538, New York, NY, USA, 2008. Association for Computing Machinery.

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[6] Prabhnoor Singh, Rajkanwar Chopra, Ojasvi Sharma, and Rekha Singla. Stackoverflow tag prediction using tag associations and code analysis. *Journal of Discrete Mathematical Sciences and Cryptography*, 23(1):35–43, 2020.

APPENDIX

A. APPENDIX

A.1 Implementation Details

A.1.0.1 TagRank.

We use the networkx library [3] for implementation. You can fully reproduce the results for TagRank on this ***tagrank colab repository***. The hyperparamters are also shown on this repository. Implementations on baseline random experiments are shown on this ***random baseline colab repository*** (google drive with datasets should be mounted).

Experiments with full log of different hyperparameters are shown in Table 3 and Table 4. In Table 3 and Table 4, best results that are reported in the main table is highlighted in bold. Meanings of each row are described in Table 1. Replaced percent indicates the proportion of valid queries which had prediction numbers smaller than the frequent tag threshold and thus the prediction outputs replaced to predefined frequent tags. The computation time of TagRank is extremely faster, compared with other GCN based methods. For each experiment, it takes *less than a minute* to finish the question tag prediction for TagRank.

A.1.0.2 Answerer GCN.

This method is implemented by pytorch-geometric [4]. The model is optimized using an AdamW optimizer with a learning rate of 0.0005. We set embedding and hidden dimensions as 1024 for both StackOverflow and Mathoverflow. For Mathoverflow, the model is trained by batch size 51200 with $\alpha = 10$. For Stackoverflow, the batch size is 25600 and α is 10.

A.1.0.3 Question-answerer GCN.

This method is implemented by DGL [5]. The model is optimized using an AdamW optimizer with a learning rate of 0.004. We set embedding and hidden dimensions as 512 for both StackOverflow and Mathoverflow. For Mathoverflow, the model is trained by batch size 20000 with $\alpha = 10$. For Stackoverflow, the batch size is 10000 and α is 13.

A.1.0.4 GCN with TagRank.

GCN with TagRank follows a similar implementation as Question-answerer GCN. The batch size is 20000 with $\alpha = 12$ for Mathoverflow and 10000 with $\alpha = 12$ for Stackoverflow.

A.2 Labor Division

The team performed the following tasks

- Implementation of TagRank, conduct baseline experiments [Soyoung]
- Implementation of question-answerer GCN and GCN with TagRank, wrote the corresponding method [Miyoung]
- Implemetation of answerer GCN, wrote corresponding method and introduction [Kyumin]
- Wrote abstract and overall [All]

[3] https://networkx.org/
[4] https://pytorch-geometric.readthedocs.io/en/latest/
[5] https://www.dgl.ai/

value threshold	minimum length	maximum length	frequent tag length	frequent tag threshold	consider importance	f1	prec	rec	replaced percent
0.02	1	1	1	1	TRUE	26.92%	45.84%	20.72%	0.24%
0.02	1	2	1	2	TRUE	29.66%	34.01%	29.43%	0.47%
0.02	2	3	1	3	TRUE	29.13%	27.61%	35.20%	0.97%
0.02	2	5	1	3	TRUE	26.26%	20.58%	42.32%	0.97%
0.02	2	6	1	3	TRUE	25.02%	18.60%	45.01%	0.97%
0.02	2	1	1	3	TRUE	8.64%	14.90%	6.65%	100.00%
0.02	2	1	3	3	TRUE	11.83%	11.43%	13.94%	100.00%
0.02	1	3	1	3	TRUE	29.13%	27.61%	35.20%	0.97%
0.02	1	5	1	5	TRUE	26.02%	20.56%	41.86%	2.92%
0.02	1	4	1	4	TRUE	27.58%	23.38%	38.94%	1.76%
0.02	1	2	1	1	TRUE	29.72%	34.10%	29.49%	0.24%
0.02	1	3	1	1	TRUE	29.29%	27.77%	35.36%	0.24%
0.02	1	5	1	1	TRUE	26.41%	20.74%	42.48%	0.24%
0.02	2	3	1	1	TRUE	29.29%	27.77%	35.36%	0.24%
0.02	2	4	1	1	TRUE	27.82%	23.56%	39.25%	0.24%
0.01	2	4	1	1	TRUE	27.82%	23.56%	39.25%	0.24%
0.01	1	2	1	1	TRUE	29.72%	34.10%	29.49	0.24%
0.03	2	3	1	1	TRUE	29.29%	27.83%	35.30%	0.24%
0.03	2	10	1	1	TRUE	25.83%	19.99%	45.64%	0.24%
0.03	**1**	**2**	**1**	**1**	**FALSE**	**30.30%**	**34.79%**	**30.03%**	**0.24%**
0.03	**1**	**2**	**1**	**1**	**TRUE**	**29.72%**	**34.10%**	**29.49%**	**0.24%**
0.01	1	2	1	1	FALSE	30.30%	34.79%	30.03%	0.24%
0.01	1	3	1	1	FALSE	29.75%	28.23%	35.84%	0.24%
0.01	1	3	1	1	TRUE	29.29%	27.77%	35.36%	0.24%
0.02	2	6	1	5	FALSE	24.53%	18.63%	43.90%	4.05%
0.02	2	5	1	5	FALSE	25.87%	20.65%	41.49%	4.05%
0.02	2	5	1	4	FALSE	26.28%	20.87%	42.21%	2.40%

Table 3: Full hyperparameter experiment results on TagRank for **mathoverflow**.

value threshold	minimum length	maximum length	frequent tag length	frequent tag threshold	consider importance	f1	prec	rec	replaced percent
0.02	1	1	1	1	TRUE	8.81%	15.76%	6.58%	2.55%
0.02	1	2	1	2	TRUE	10.31%	12.27%	9.88%	3.45%
0.02	2	3	1	3	TRUE	10.57%	10.46%	12.29%	5.80%
0.02	1	2	1	1	TRUE	10.29%	12.22%	9.87%	2.55%
0.01	1	2	1	1	TRUE	10.29%	12.22%	9.87%	2.55%
0.02	1	3	1	1	TRUE	10.52%	10.27%	12.30%	2.55%
0.02	2	4	1	2	TRUE	10.25%	8.98%	13.96%	3.43%
0.02	2	4	1	2	FALSE	10.25%	8.98%	13.96%	3.43%
0.02	2	4	1	1	TRUE	10.23%	8.93%	13.95%	2.55%
0.02	1	4	1	1	TRUE	10.23%	8.93%	13.95%	2.55%
0.02	1	3	1	1	FALSE	10.52%	10.27%	12.30%	2.55%
0.02	2	5	3	1	FALSE	9.99%	8.02%	15.46%	2.55%
0.02	2	5	3	3	FALSE	10.12%	8.11%	15.61%	5.80%
0.02	2	6	3	3	FALSE	9.79%	7.53%	16.51%	5.80%
0.02	2	6	3	4	FALSE	9.86%	7.60%	16.60%	9.55%
0.02	2	6	4	4	FALSE	9.92%	7.56%	16.85%	9.55%
0.02	2	5	4	4	FALSE	10.26%	8.15%	15.95%	9.55%
0.02	2	4	4	4	FALSE	10.59%	9.02%	14.62%	9.55%
0.02	2	3	3	3	FALSE	10.74%	10.32%	12.62%	5.80%
0.02	1	2	2	2	FALSE	10.38%	12.21%	10.00%	3.45%
0.02	1	3	3	3	TRUE	10.74%	10.32%	12.62%	5.80%
0.02	1	3	3	3	FALSE	10.74%	10.32%	12.62%	5.80%
0.02	1	3	5	3	TRUE	10.80%	10.28%	12.93%	5.80%
0.03	1	3	5	3	TRUE	9.47%	8.67%	12.11%	25.32%
0.01	**1**	**3**	**5**	**3**	**TRUE**	**10.80%**	**10.28%**	**12.92%**	**5.62%**
0.02	1	4	5	3	TRUE	10.51%	8.95%	14.58%	5.80%
0.01	1	4	5	3	TRUE	10.50%	8.92%	14.60%	5.62%
0.01	**1**	**3**	**5**	**3**	**FALSE**	**10.80%**	**10.28%**	**12.92%**	**5.62%**

Table 4: Full hyperparameter experiment results on TagRank for **stackoverflow**.