Improving Conventional Tag Recommendation System for Software Information Site

Aman A. Patel[1] and Brij P. Patel[2]

[1,2] *Lakehead University, MSc. Computer Science, Canada*

Email:apatel85@lakeheadu.ca, bpatel34@lakeheadu.ca

Abstract— On Community questions and answering sites, tagging has evolved into a huge infrastructure. These sites allow developers to label posted content, referred to as software objects, with short descriptions, known as tags. Tags help to improve the organization and locate, and browsing of the software objects. However, community sites are allowing developers to write the tags by themselves and not restricting if there is a redundancy of the tags. Some new developers tend to post noisy and repeated tags that create the problem to locate the tags. Thus to locate any software objects have become difficult for any algorithm and the question and answering becomes less efficient and accurate. In this paper, we have replicated the FastTagRec model implemented by Jin Liu et al., and additionally also checked the accuracy with different classifiers named one vs rest classifier and also used Latent Dirichlet Allocation (LDA) to evaluate the efficiency of the model. By learning existing software objects and their tags, Fast- TagRec and the baseline classifier can very accurately infer tags for new postings. Further to make the environment effective we also added a field name Ask a question that can predict the tag and which is helpful to the developers while posting new software objects on the community sites. Our results show that FastTagRec implemented with a single hidden layer neural network has given less accuracy but if the multiple hidden layers and different classifiers can be used to improve the accuracy

Keywords—Software objects, Tag Recommendation, community sites, FastTagRec.

I. INTRODUCTION

The recommender system works to predict the anticipation that an individual user will aim for a particular resource. The recommender system uses the past behavior of the user to make a personalized list for the user. The admired domain-like amazon, movie and music services like Netflix and Spotify, and the Community-based question and answering platforms also use the tags recommendation system. Community-based question and answering platforms have come into huge prominence in recent years, these community sites are helpful to the developers and computer-science enthusiasts during the whole software development life-cycle. Many software information sites depend on the tags to classify their software objects from other various software objects thus these sites require great performance and better accuracy to manage software objects. The quality of tags plays a significant role on software information sites. The software objects posted on the software information sites like questions and answers and many open sources projects are increasing with the prominence of the information sites. The number of software objects is gradually are increasing and that makes it difficult for the developer to locate the proper software objects. To resolve this issue Tag system is introduced while posting the software object the objects need to associate with

certain tags. Tags are generally of small words or abbreviations using a semi-colon, underscore, and colon to identify, locate, and manage the software objects on community sites. High-quality tags are expected to be precise and can describe the most important features of the software objects in the software information sites. Tagging software objects by developers is inherently a distributed and uncoordinated process. Most software information sites allow developers to tag their software objects with their own words. Because of the freedom to choose the tags, tags can be idiosyncratic due to developer's understanding of their software objects, English skills, and preferences [1]. So if the information sites allow this practice then there would be a rapid increase in the tags, then the information site will lose its capability to serve the users. Thus the need for better tagging needs to be deployed for more accuracy and efficiency. Hence FastTagRec was introduced by Jin Liu et al.,[1] they proposed the approach that learns from the existing text description and the tags from the existing software tag. To implement their model, they have used a single hidden layer neural network and got better performance compared to the traditional tag recommendation systems. In our study, we have tried to implement a model by replicating the FastTagRec approach and also incorporated Latent Dirichlet Allocation (LDA) which is used for topic modeling to improve the accuracy of the model.

The paper has been organized in the following structure, In section I, we have described the basics of tag recommendations and our approach. Section II, we have given the motivation to pick up the topic of tag recommendation. In section III is a Background study and given a brief of work done and methodology used in past on tag recommendations In section IV, we have collected a dataset from the software information site In section V, we have presented the Research Questions. In section VI, we have represented the Research methodology of our approach. In Section VII, we have displayed the results of our study. In section VIII, we have given a section for discussion where we have given future work and challenges of our study, and last section IX, we have given the conclusion for the proposed research.

II. Motivation

The upward path for the developers and computer science enthusiasts to ask and answer on community-based Question and answer sites has increased. These questions are associated with tags, Tags are a short label that contains a key feature of the posted content which is used to organize and manage questions. Our research is motivated to increase the efficiency and the response time of the developer's probe for software information sites. The work is classified considering two major factors one is community-Based open source and the second is for developers. The information sites have eased the work for developers, these sites work on the tag-based system associated with the questions of the users to identify and classify the questions. The developers or users put the tag which has already been used or it has the same meaning then these sites become less efficient and less precise. In today's progressive world for developer's objective is to work more efficiently at a great pace. Secondly, the word used for the tag is mostly arbitrary. The different tags like some contain full word whereas others is an abbreviated word, hyphens, and underscore which ultimately have the same meaning. The tag list on the information sites is maintained manually which increases the time consumption.

III. Background study

In this section, we have studied the related work, In recent years, different approaches have been proposed for Tag Recommendations in software information sites. Pingyi Zhou et al. [2] proposed TagMulRec to automatically recommend tags and classify software objects in evolving large-scale software information sites. TagMulRec locates the software objects that are semantically similar to the new one and exploit their tags. In this research, two challenges have been addressed, TagMulRec should dynamically adapt to the changes because on software information sites new software objects are getting added continuously every hour. Developers are allowed to modify, adding new tags, and removing existing tags that have already been posted. Secondly, TagMulRec must be efficient which consider the size of the software objects and the tags. TagMulRec performs the algorithm to

rank the tags in a set, and then which tag having higher rank would be recommended to the developer. Pingyi Zhou et al. have contributed automated tag recommendation which helps to solve the problem of the rapid growth of tags by removing the redundant and different tags with inferring the same meaning. Also have proposed the multi-classification algorithm which can handle ample software objects. Creating an index for software objects, constructing target candidate sets, and performing multi-classification algorithm to rank the tags by following these guided steps TagMulRec achieves more efficiency the results given in the paper also profs the same. Deep Learning can be effective on the software information sites as its algorithm can help to recommend and remove the redundant tags with the same meaning.

Jin Liu et al.[3] have proposed four different deep learning approaches like Recurrent Neural Networks(RNN), Convolution Neural Network(CNN), Hireachal Attention Network(HAN), and Region-Based Convolution Neural Network(RCNN) and have compared with the traditional approaches like EnTagRec, TagMulRec, and FastTagRec. The deep learning approach represented in the research were significant and would be beneficial to the software information sites, here the accuracy of the TagCNN and TagRCNN was based on the deep learning module named CNN and RCNN have outperformed the traditional approach of the tag recommendation the results of the deep learning approach were much significant. Additionally, TagRNN and TagHAN based on RNN and HAN were very much underperformed than the traditional methods and the result was not also much effective. Using appropriate deep learning approaches can indeed achieve better performance than traditional approaches in tag recommendation tasks for software information sites.

Reza Gharibi et al. [4] has proposed a Content-Based model for tag recommendation, their approach combines multi-label classification and textual similarity technique to enhance the performance of tag recommendations. When the tag is received the system tries to recommend a series of tags based on a model of multi-label classification component. The textual similarity component is used to complement the multi-label classification component, it suggests tags for new software objects based on the tags similar to existing software objects tags that are already known on the information sites. Lastly, both component's recommended tags are combined to create a ranked list of recommended tags for the new software objects. . Reza Gharibi et al. have evaluated the performance of the system on many software information sites, the result shows a notable improvement over TagCombine, TagMulRec, and FastTagRec. Later they have planned to analyze the semantic meaning of the code-like terms which can help to overcome the textual mismatches between software objects and merging the synonym of tags which can narrow down the label space of the classifier. Chen et., al [5] has experimented on Docker repositories which have contained one or more version of Docker images. It is similar to GitHub. And most of the Docker repositories do not have tags in them. It contains version number in place of tags which has no semantic information. To ease the tagging system, Chen et., al have introduced semi-supervised learning-based tag recommendation which contains components like predictor, extender, evaluator, and integrator. And they have extended this with tag vocabulary for getting high accuracy with the help of the self-adapting iterative model. They have compared the output with D-Tagger and EnTagRec and the result showed that this approach outperforms other approaches.

IV. DATASET

Considering the motivation of the project, we have gone through some papers which is related to our work. And we have decided to use data set from stack exchange which have over one hundered and seventy Question and Answer community [4] and also it is publically available so anyone can access this data set easily and can implement our model by using this site. The main reason to select this data set is that it has all the fields required for the project. It provides data in the form of an XML file. Firstly, we have decided to consider a smaller scale site like Software Engineering data set [6], this has included files like Posts, Tags, Users, PostLInks, PostHistory, Votes, Comments, and Badges in XML format. After that, we will convert it into a .csv file for accessing it easily in

the implementation because we are using jupyter notebook for the implementation and it's hard to read data in XML format in it. In total, it contains two lac and thirty thousand posts in the Posts table and has around seventeen hundred tags in the Tags table with a count of occurrence. We have chosen this dataset for initial bases so that we can check the accuracy of FastTagRec from the small dataset. After the success in the small-scale site, we will aim to add the data of higher scale sites like Askubantu, StackOverflow, AskDifferent, and Unix which have larger and complex data.

V. Research Questions

In order to improve the existing state-of-the-art approaches, we have conducted the experiment to answer the following three question.

1. RQ1: How is the overall performance of the proposed approach compared to methods implemented in FastTagRec?
2. RQ2: Which steps should be considered to extend the approach of FastTagRec? And does it outperform traditional approaches FastTagRec and TagMulRec?
3. RQ3: how the bag of words with the n-gram feature can help in the tag recommendation system?

VI. Methodology

In this section, we have discussed methodology and steps of our approach that have been followed to implement the proposed model.

a. Read data

Considering the first step of our approach, we have gathered data as mentioned in section of Dataset from publically available data set site stack exchange. After that, we have read the data into jupyter notebook for the implementation in which we have considered only 1 file "Posts" because it contains all the fields required for the project. From the posts file, we have used the Body and Tags field which have been given as the text of questions and tags. In the data set vast amount of tags are available and lots of them are repeating several times so as a result we have calculated unique tags from those and we got 1614 unique tags. Due to having a large number of tags, it became hard to evaluate the model so we have considered the top four hundred most common tags we occurring most of the time into questions. And by doing this, the efficiency of the model also got increased.

b. Pre-processing task

The raw data has lots of unnecessary words, HTML tags, punctuations etc. which might affect the accuracy of the model so to increase the accuracy of the model as well as to improve the model for training data. Firstly, we have converted all the original data into the lower case then clean the text by removing the blank line and space. In the data, there are lots of HTML tags available which might skew the result of the model so to remove these types of HTML tags we have used beautifulsoup package. After removing tags from text, we have converted strings into tokens with the help of a tokenizer. We have considered each token at a time and removes the punctuation from the tokens. Natural Language Toolkit has been used to identify the stop-words from the tokens and it also removes the stop-words from the text because stop-words are rare words and they are not appropriate for the model evaluation. So removing stop-words is common practice in machine learning models. We have also used Wordnet which is a big lexical database for the English language that aims to develop organized semantic associations between words. It is free and open to the public. By using this data set we have lemmatized the complex words into its root words. Sometimes words are so similar and also infer to the same meaning so by converting complex words into root words accuracy can be increased.

c. Feature Extraction

Data which we have gathered have lots of different tags available in it so for the evaluation of the tags it's become hard to convert tags into binary to fit into vectorizer. So we have used a multilabel binarized to convert each set of tags into binary form and fit the tags into a multilabel binarizer that returns an array of binarized tags. Then, we have split the data set into training and testing set as eighty percent and twenty percent re-

spectively. After that, we have fitted the training and testing data into a vectorizer that can help to convert data into vectors. We have also considered the bigram approach for TF-IDF vectorizer that can help the model to evaluate two words at the same time which can eliminate the problem of taking only a single word at a time that can decrease the accuracy of the model.

d. Topic Modelling with LDA

Topic modelling stands for an unsupervised machine learning method that can analyze a series of documents, find word and phrase structures within them, and automatically cluster word groups and related expressions that best characterize the set [7]. We have used this concept to aggregate similar questions which might have similar tags for evaluation. LDA implies Latent Dirichlet Allocation which is a popular method of topic modelling. Questions have various words which represent the questions topic and the tag also represents many different questions. LDA is used to find topics from the questions that can help to generate clusters of similar questions that can help to recommend appropriate tags for the questions. We also counted the perplexity of the model by using LDA. And we got the result as shown in figure 1. As

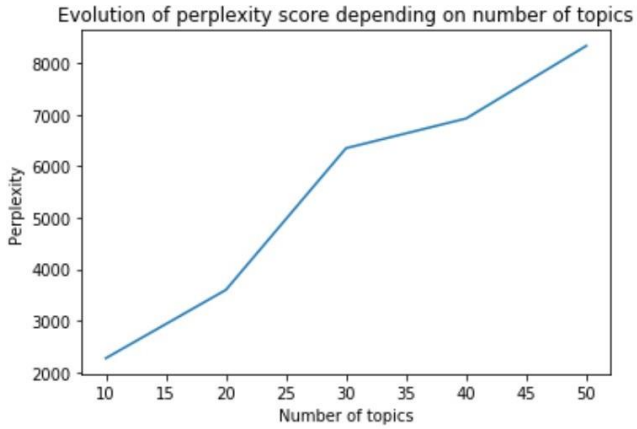

Fig. 1: Perplexity Graph

shown in the figure, the perplexity of the model is increasing as the increasing value of the topic of the questions.

e. Evaluation of Model:

After using the topic modelling method for our model, we have tried many different approaches to evaluate our model. Model evaluation is an important task to find the performance result from the system. We have tried 3 different approaches to evaluate the result of the proposed approach. Firstly, we have evaluated the model by using an approach followed by the FastTagRec model that is based on a single hidden layer neural network model with softmax function that have been used to find probability distribution [8] in the output layer and finds the accuracy of the model proposed in the paper. To fit the data, we have considered data with the bigram tfidf model then train the model and find the accuracy of the method. After that, we have also calculated the Jaccard score for one vs rest classifier as baseline classifier and predict the tags using it and the result has given us the average Jaccard score of the model. Lastly, we have considered the LDA model for classification and calculate the jaccard similarity score using LDA model. By using different classification techniques, we have evaluated the importance of the proposed approach and compare the model with an existing technique of tag recommendation system.

VII. RESULT

In this section, we have reported the results of the replicated model of FastTagRec and also compared the accuracy of the proposed method with Topic modeling. We have used the approach used by Jin Liu et al., that is with a single hidden layer neural network and we have trained and tested the model using a software engineering site dataset as the above methodology stated and for our implementation, we almost got eight percent accuracy that is very low in general terms. This model's accuracy should be improved for a better recommendation of the tags that can be helpful to the developers.

```
Epoch 1/4
23290/23290 [==============================] - 1s 63us/sample - loss: 14.2708 - acc: 0.0330
Epoch 2/4
23290/23290 [==============================] - 1s 41us/sample - loss: 13.9348 - acc: 0.0755
Epoch 3/4
23290/23290 [==============================] - 1s 40us/sample - loss: 13.3907 - acc: 0.0812
Epoch 4/4
23290/23290 [==============================] - 1s 42us/sample - loss: 12.9005 - acc: 0.0745
```

Fig. 2: Accuracy with Neural Network

secondly, replicating the approach used by Jin Liu et al., of a single hidden layer neural network so got low accuracy. So we decided to use the different classifier named one vs est classifier as our baseline classifier to check the accuracy and we got the Jaccard score of thirty-eight percent. After we

performed the LDA model to evaluate the accuracy and use also used multi-label binarization to convert the tags into the binary form and we got a Jaccard score of ninety-three percent.

Additionally, We have created a field regarding tag recommendation as it works like if the user or the developer asks the question then the algorithm will predict the tag according to the question asked. Firstly, the question or text is taken as input then the text is pre-processed and converted into the vector form using TF-IDF vectorizer further it is normalized with the topic modeling named Latent Dirichlet Allocation(LDA).

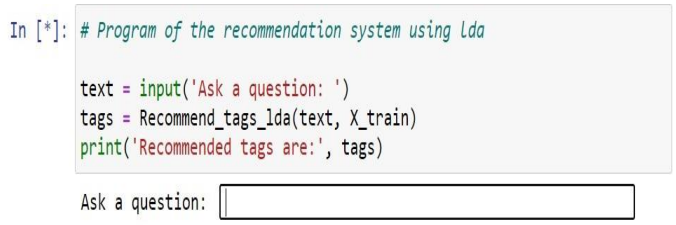

Fig. 3: Input Question

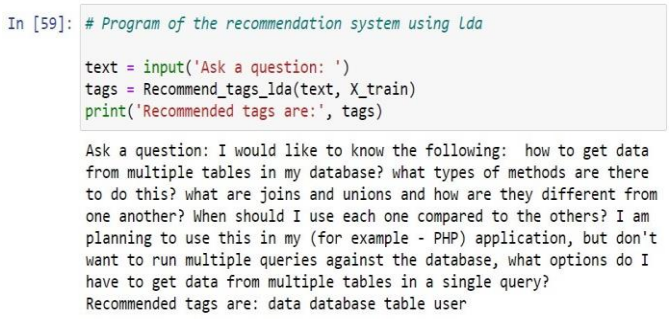

Fig. 4: Recommended Tags

So then the tags are suggested according to the question and it recommends the unique and related tags. The question asked will be answered with up to five similar or matching tags based on the question. As a result, it will help the users and developers to find the appropriate tags regarding their questions and will increase the efficiency of the site.

VIII. Discussion

RQ1: How is the overall performance of our proposed approach compared to the method which replicated (FastTagRec)? **Answer:** : To address this question from implementation, the proposed approach of our replicating the FastTagRec with a single hidden layer of neural network has given the accuracy of almost eight percent, which is accounted as low accuracy. , where we decided to use a different classifier, that is baseline dummy classifier named one vs Rest classifier and we got the Jaccard score of thirty-eight percent. So, as a result, we decided to deviate a little from our implementation and add a new approach named LDA (Latent Dirichlet Allocation) and for that, we calculated the Jaccard score and it is ninety-three percent which is far better than the baseline dummy classifier. The proposed method falls behind in terms of accuracy but when we changed the approach and used the LDA model it has given better performance relative to the baseline classifier. Thus the overall performance of the proposed approach shows a lesser result but if used a different method the accuracy is more.

RQ2: Which steps should be considered to extend the approach of FastTagRec? And does it outperform traditional approaches FastTagRec and TagMulRec? **Answer:** FastTagRec has given excellent results for tag recommendation and we have replicated the model of FastTagRec, we have considered the dataset of the software engineering site and the accuracy of the model was almost eight percent. But to consider improving the accuracy of the model some steps needed to be implemented. To generalize our findings we need to account for more data and as a result, it will generalize the results and will help to improve the accuracy of the model. Secondly, we have checked the accuracy of the model with the LDA classifier and the Jaccard score was ninety-three percent. So if we use a different classifier and the accuracy of the model can be improved and can also discover new findings through the new classifier applied. Code snippet should also be accounted for the tag recommendation so that when the question regarding code is posted the recommended tags will be listed so that the user doesn't post the redundancy.

RQ3: how the bag of words with the n-gram feature can help in the tag recommendation system? **Answer:** The bag of Words model is used to identify the count of occurrence of the word in a particular document it is also used for text processing and analyzing [9]. Bag of Words can be used to define the importance of the words by counting the occurrence of the word in the posts. In our model, we have considered the bigram bag of

word model which considered 2 words at a time that can help to improve the performance of the model because sometimes only consideration of a single word might not be helpful or it might inaccurate for the result for instance if we considered sentence that contains words like “Not Bad” if we evaluate this sentence with a 1-gram feature than it represents negative words but considering 2-gram we can get an appropriate result which means that is a positive sentence.

Considering the shortcoming of our study, the implementation of our project has been done using a single hidden layer neural network, as a result, we got almost eight percent accuracy which has mitigated the accuracy score. Secondly, we have tested the model with only one dataset from the software engineering site, so we were not able to more case studies are required to generalize our findings. Our trained model assumes that all the existing tags from the dataset are accurate and recommends the tags from the previously recommended tags that might not be accurate. We have only considered a description of the textual content of the software information site. This has been narrow down the importance of the model because most of the stack overflow questions contain code snippets and tags are depended upon the code itself so only consideration of text might be improper for evaluation.

In the future, we like to generalize our findings by evaluating more software information sites that will give precise and established accuracy of our model. Our model is based on the multi-classification process so in near future we want to evaluate the model with the multi-label classification approaches. To improve the model’s accuracy, we also like to implement the model with multiple hidden layers in the neural network instead of using a single hidden layer that might improve the accuracy of our model. In the future, we also plan to consider code snippets and screenshots instead of only considering the text, so it will improve the accuracy of our recommendation system. And by doing this the scalability of the model can be increased and a vast amount of different problems can be considered during the tag recommendation system and every aspect of questions maybe get covered.

IX. Conclusion

Tags are widely used by the developers while placing the questions on the StackOverflow site and those tags are crucial for searching the questions in the overtime. In this paper, we have replicated the work of the FastTagRec and checked the accuracy, and also used LDA for tag recommendations. Compared to the association of the FastTagRec, LDA has achieved better accuracy, in particular, recommends more specific tags which would be more helpful for the users and the developers. The evaluation was conducted by randomly selected from around two lacs software objects from the software engineering site. And after the evaluation of the model, we have found better results while using the topic modeling feature with a bigram TF-IDF vectorizer. Certain steps should be taken care to improve the tag recommendation system like consideration of code snippets can help to recommend tags for the questions because StackOverflow mainly contains code snippets into each question.

References

[1] Liu J, Zhou P, Yang Z, Liu X, Grundy J. FastTagRec: fast tag recommendation for software information sites. Automated Software Engineering. 2018 Dec;25(4):675-701.

[2] Zhou P, Liu J, Yang Z, Zhou G. Scalable tag recommendation for software information sites. In2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER) 2017 Feb 20 (pp. 272-282). IEEE.

[3] Zhou P, Liu J, Liu X, Yang Z, Grundy J. Is deep learning better than traditional approaches in tag recommendation for software information sites?. Information and software technology. 2019 May 1;109:1-3.

[4] Gharibi R, Safdel A, Fakhrahmad SM, Sadreddini MH. A Content-Based Model for Tag Recommendation in Software Information Sites. The Computer Journal. 2019 Dec 23.

[5] Chen W, Zhou JH, Zhu JX, Wu GQ, Wei J. Semi-supervised learning based tag recommendation for docker repositories. Journal of Computer Science and Technology. 2019 Sep;34(5):957-71.

[6] Data Set: “Files for stackexchange” Available: https://archive.org/download/stackexchange

[7] Federico Pascual, “Topic Modeling: An Introduction”, MonkeyLearn, September 26th, 2019. [Online] Available: https://monkeylearn.com/blog/introduction-to-topic-modeling/

[8] Bishop CM. Pattern recognition. Machine learning. 2006 Apr;128(9).

[9] Behley J, Steinhage V, Cremers AB. Laser-based segment classification using a mixture of bag-of-words. In2013 IEEE/RSJ International Conference on Intelligent Robots and Systems 2013 Nov 3 (pp. 4195-4200). IEEE.