Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TRITAG RECOMMENDER: A HYBRID APPROACH TO STACK OVERFLOW TAG PREDICTION USING TRANSFORMERS AND KEYWORD EXTRACTION

MORARJEE KOLLA[1], SRINIVASA RAO BURAGA[2], SUBHASH BHAGAVAN KOMMINA[3], J KAVITHA[4], NITYA RAMIREDDY[5], SREEMUKHI SAVALGE[6]

[1]Associate Professor, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad-500075, Telangana, India
[2]Professor, Department of Information Technology, Lakireddy Bali Reddy College of Engineering (Autonomous), Mylavaram, NTR District, Andhra Pradesh-521230, India
[3] Professor, Department of Information Technology, Sasi Institute of Technology & Engineering, Tadepalligudem, Andhra Pradesh-534101, India
[4]Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Bowrampet, Hyderabad-500043, Telangana, India
[5,6] B.E Student, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad-500075, Telangana, India
E-mail: [1]morarjeek_cse@cbit.ac.in, [2]buragasrinivasarao@gmail.com, [3]subhash@sasi.ac.in ,[4] j.kavitha5555@gmail.com, [5] ramireddynitya@gmail.com, [6]savalgesreemukhi@gmail.com

ABSTRACT

Community question-answering (CQA) platforms offer new opportunities for users to share knowledge online. Tags are added to data on these platforms to define, classify, and discover the information. Accurate tags help find users to answer the question. However, tags may be inaccurate or improper since users tag questions based on their understanding of the question's content and other tags that are on the site. Existing methods often fail to fully capture the semantic context of questions, especially those containing code snippets, leading to suboptimal tag suggestions. This highlights a gap in the literature for models that can effectively handle both natural language and programming content. To address this issue, we propose a novel deep learning approach combining advanced transformer-based architectures with keyword extraction techniques to understand the context in text and code snippets and produce relevant tags. Our proposed method will be able to suggest relevant and appropriate tags and demonstrate superior performance compared to state-of-the-art techniques, thus contributing new insights into context-aware tag recommendations for CQA systems.

Keywords: *Tag Recommendation, Community Question-Answering, Transformer, Keyword Extraction, Tritag Recommender*

1. INTRODUCTION

Community Question Answering (CQA) sites, such as Stack Overflow and AskUbuntu have emerged as popular venues for users to ask questions and exchange knowledge on a broad range of subjects\cite{b1}. To cope with the enormous and ever-increasing amount of user-provided content, these sites heavily depend on tags—user-defined keywords or phrases that classify questions and enable content organization, searchability, and topic-based browsing. A sample question posted on Stack Overflow is shown in Figure 1.

Appropriate tagging is important for improving user interaction, making content more discoverable, and optimizing future processes such as expert suggestions. Tagging manually, however, is typically irregular, incomplete, or inaccurate due to the user's varied levels of expertise and the inherent subjectivity in assigning tags. Automated tag recommendation systems have thus become critical components of contemporary CQA sites to address these issues.

Thus, by using proper and precise tags reflecting the content of the question, many aspects of site engagement such as connecting experts across

different communities and directing the question to the appropriate people with appropriate knowledge can be enhanced. Stack Overflow does not limit user's freedom to tag creation and posting; they can create and select whatever tags they wish. However, the quality of tags is substantially influenced by the writing habits of the users, their English proficiency, and their experience level; thus, the tags used by different users tend to become inconsistent, which is a problem that contributes to tags. These issues highlight how desirable it is to have a tag recommendation system to automatically forecast tags for future questions posted on the site.

Figure 1:A Sample Question on Stackoverflow

This study focuses specifically on building an automated tag recommendation model using deep learning techniques for Stack Overflow questions. It does not cover non-technical CQA platforms or manual tagging behavior analysis. It assumes the availability of clean and labeled textual data and does not account for user intent or external metadata.

The scope is limited to English-language questions and predefined tag sets within the dataset. Limitations include reliance on textual input quality, and exclusion of multimodal content such as images or code snippets not represented as text.

Automated tagging systems offer many benefits aside from simply improving tagging accuracy and efficiency[3]. They are also crucial to maintaining tag consistency by collecting similar questions in standardized topics, greatly improving content organization and search capabilities. These systems are particularly beneficial for new or novice users by reducing the effort required to choose suitable tags through intelligent recommendations. Additionally, by reducing tagging mistakes and redundancy, automated tagging decreases the burden on moderators, allowing them to attend to higher-level activities such as community interaction and platform creation. Because of this, there has been a need to build automated tagging systems that can suggest appropriate and relevant tags[4].

2. RELATED WORKS

In Community Question Answering (CQA) systems, the Tag Recommendation involves aiding users by suggesting relevant tags that can categorize their questions effectively. It presents a set of potential tags based on the question's content, from which users can select multiple tags—typically between one and five—to accurately label their questions. The tags improve the organization and discoverability of questions, enabling more efficient retrieval and better alignment with the user's search intentions. To achieve this, various models have been developed and refined to make tag recommendations more accurate, relevant, and contextually appropriate.

He et al.[5] proposed a complex system called PTM4Tag+ for automatic tag recommendation in Stack Overflow. The approach, which is based on pre-trained language models (PTMs), models the Title, Description, and Code of a post using a triplet architecture. PTMs are utilized to independently encode each component to generate feature representations. These are used to predict relevant tags in a multi-label classification task after concatenation. PTM4Tag+ was developed using a variety of encoder-only and encoder-decoder PTMs, including BERT, RoBERTa, CodeBERT, and CodeT5. In measures like F1-score@k, Precision@k, and Recall@k, CodeT5 under the PTM4Tag+ framework outperformed all the other variations, significantly outperforming previous state-of-the-art techniques like Post2Vec (CNN-based). The authors also investigated smaller models, such as DistilBERT and CodeT5-small, which improved inference time by more than 47.2% while maintaining over 93.96% of the performance, to solve latency and deployment issues. PTM4Tag+ used rich semantic representations from pre-trained models to provide strong tag recommendations.

Amsa-nguan et al. [6] have a practical AI-based approach for automatically topic tagging web content in Thai and English-translated formats. Their strategy involved gathering a sizable dataset of news articles from prominent Thai sources arranged into six categories: technology, politics, economics, entertainment, sport, and crime. To classify content, the study investigated several machine learning models, including Random Forest (RF), K-Nearest Neighbors (KNN) with TF-IDF representations, and an LSTM model with word embeddings. Depending

on the model, TF-IDF, and sequence vectorization were used for feature representation. The outcomes demonstrate that while all models perform well, Random Forest continuously beats the others on both Thai and English datasets. This model was selected to be implemented in an online topic tagging program. The final system architecture incorporates asynchronous processing for efficiency and supports user interaction via the web and API.

An automated question-tagging system was proposed by Raja Ram et al. [7] to increase the precision and effectiveness of tagging in Community Question Answering (CQA) platforms. The method used preprocessed question data to predict tags using machine learning algorithms, namely Random Forest (RF) and k-Nearest Neighbors (KNN). To prevent model bias, feature extraction was then carried out using the top 100 tags, selecting a balanced set of positive and negative examples. The preprocessed text is transformed into numerical feature vectors appropriate for classification using TF-IDF vectorization. Using Euclidean distance, KNN finds the closest neighbors in the feature space and then tags them according to the majority class. RF aggregates the predictions of several decision trees using tuned parameters (max_depth = 5, n_estimators = 150). According to experimental results, KNN outperforms Random Forest (72%) by achieving a slightly higher accuracy of 75% across the top five tags. It utilized only the text content of posts thereby reducing its efficiency by not considering the code.

To overcome the limitation of not utilizing code and to improve semantic understanding through code-mixed deep representation learning, Li et al. [8] presented CDR4Tag, a novel method for tag recommendation in software information sites. The approach addressed two major challenges: leveraging the semantic richness of tags and integrating both text descriptions and code snippets of software objects. It does this by treating tag recommendation in a unique way as a sentence-pair matching task. A dual-interactive fusion approach comprising the Connective Semantic Interaction (CSI) and Separative Semantic Interaction (SSI) modules was proposed to accomplish this. CSI uses heuristic matching to calculate semantic similarity and BERT to encode software object descriptions and tags. In contrast, SSI encodes code snippets using GraphCodeBERT and uses hierarchy-aware multi-layer attention to capture deep semantic correlations between text and code. By combining representations from both modules and applying a fully connected layer and a softmax function, the system forecasts the likelihood of relevance for every tag. In terms of top-k recall, precision, and F1-score, tests performed on four Stack Exchange datasets (StackOverflow, CodeReview, DBA, and WordPress) show that CDR4Tag performs better than cutting-edge baselines.

However, while the method [8] relies on supervised learning and annotated data, Devine et al. [9] address this limitation by introducing an unsupervised tag recommendation approach. This approach uses pre-trained text embedding models to recommend tags rather than supervised training data. Models like MPNet and MiniLM, which are trained on natural language text pairs, are used to embed every Stack Overflow post and every tag that could be used. The cosine similarity between post embeddings and tag embeddings is then used to recommend tags. With an emphasis on “unpopular” tags, which are infrequent, the method is assessed on a 0.1% sample of Stack Overflow posts. Models that were trained on domain-specific data, such as title-body pairs from Stack Exchange posts, performed noticeably better than others. Interestingly, the MPNet (QA) model performed the best, with an R@1 of 0.161, showing that about 16% of its top-1 tag recommendations matched the actual post tags.

A deep learning-based method for automatically recommending tags on Stack Overflow posts is presented by Subramani et al. [10]. The suggested approach predicts multiple tags for a given post based on its textual description by using Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN). The tag prediction system uses a pipeline that includes feature extraction, data cleansing, and multi-label classification with deep learning models. Cleaning the StackSample dataset, limiting it to the top 50 most frequent tags, and eliminating extraneous components like HTML tags, stop words, and punctuation are the first steps in the process. Bag-of-Words and GloVe word embeddings are used for feature extraction, and multi-label binarization is used for tag encoding. A comparison is made between the Gated Recurrent Unit (GRU), LSTM, and Multi-Layer Perceptron (MLP) models. The highest F1 score (54.3%) and Jaccard score (49.6%) are attained by GRU, while LSTM provides a good balance between accuracy and training efficiency.

A specific BERT-based model called LegalBERT-th was created by Saengwaree et al. [11] for automatic question tagging in Thai legal Q&A forums. In this method, a Thai legal domain-specific dataset is created, and pre-trained language models are used

for tag prediction. The dataset comprises 10,000 legal Q&A pairs that were manually tagged with legal topics like labor law, civil law, and criminal law after being gathered from online Thai legal discussion forums. Pre-processing, LegalBERT fine-tuning, and classification are all part of the process. While the tagging task is modeled as a multi-class classification problem, the pre-processing phase includes tokenization and language cleaning specific to Thai. The researchers train and assess the tagging system using BERT-based models, such as Multilingual BERT (mBERT) and LegalBERT-th. The findings highlight the significance of domain-specific pre-training by showing that LegalBERT-th performs noticeably better than mBERT, with a test accuracy of 90.2% as opposed to 76.4% by mBERT.

The existing recommendation systems show significant improvements by utilizing deep learning models, but they also have a few limitations. These include ignoring the semantic details provided by code snippets of the questions by removing them from the data during training. The identified gap leads to the research question: How can we improve tag recommendation by leveraging both text and code data without losing semantic information? Our proposed model aims to address the limitations using advanced domain-specific transformer models and Keyword extraction models by utilizing both text and code data. This new approach will enhance the relevance
of recommended tags.

3. TRITAG RECOMMENDER

The TriTag Recommender we are proposing uses two components namely Keyword extractor and Tag Classifier which uses the title, body text and code snippets to recommend tags for a question. The high-level architecture of TriTag Recommender's is displayed in Figure 2.

3.1 Dataset Preprocessing

Out of all the features, we considered only the Title, Body, and Tags of stackoverflow questions because these keep the model simple and help in capturing the semantics or context of the questions. We performed data cleaning to increase the effectiveness of the dataset. The data cleaning was applied to the Title and Body of the questions. The operations applied are

- Removed all HTML tags from the Title and Body of the questions using the BeautifulSoup library while extracting the code snippets from the Body of the questions. The code snippets will be present between two HTML tags <code></code> and these will be extracted to create a new feature named code for the questions which will be used for training the model.
- We checked the language of the questions and identified that the majority of the questions were in english and removed questions from our dataset in languages other than english.

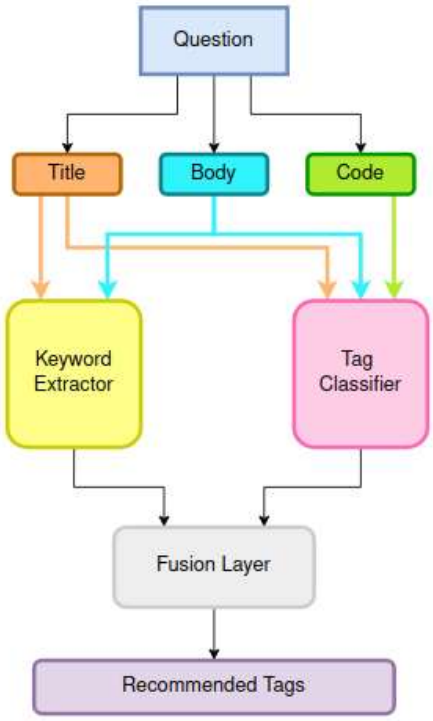

Figure 2: Architecture of the TriTag Recommender

- The text content of the title and body were converted into lowercase characters.
- All the Unicode characters like emojis were removed from the text content.
- Extra spaces were removed from the text content.
- Punctuation marks were removed from the text content.
- All the links and numbers were removed from the text content.

Finally, We had our required features - Title, Body, and Code. These are cleaned and can be used for recommending the tags.

3.2 Keyword Extractor

The Keyword Extractor is used for obtaining significant terms or words from the text which are treated as tags for the question. KeyBERT is a keyword extraction model that works using BERT embeddings to identify keywords similar to a document by using cosine similarity. It also supports many embedding models. Since our data is English, we utilize "all-MiniLM-L6-v2" from Sentence Transformers, a pre-trained transformer model to compute embeddings. The KeyBERT gives important keywords in the increasing order of their relevance score. These keywords are then filtered by using a threshold of 0.5. This makes sure that only the keywords which are important and semantically, contextually relevant to the text are retained.

3.3 Tag Classifier

The architecture of the tag classifier is shown in Figure 3. For the processing of the various components of the question namely title, body text and code we are utilizing two separate text encoders and one code encoder. These encoders are utilized to obtain feature representations or embeddings for each individual component of the question. To get accurate embeddings for our data, we will be finetuning the encoder models.

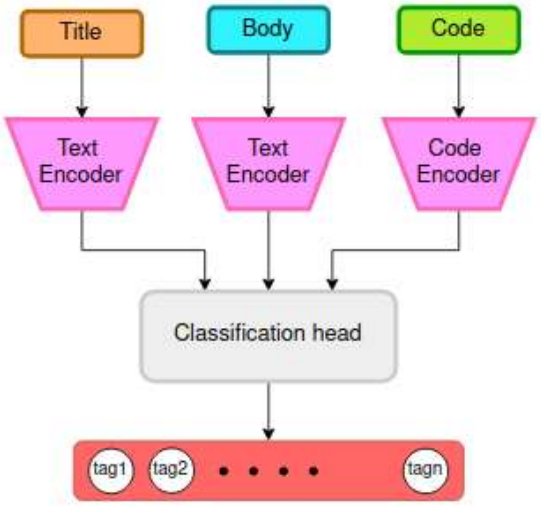

Figure 3: Architecture of the Tag Classifier

The individual feature representations are passed to the classification header where they are concatenated to give a final feature representation. This final representation is passed through a sequential network of dense layers to perform multi-label classification. The final output vector from this sequential network is the probabilities of a tag belonging to a question. These probabilities are arranged in decreasing order of their probabilities and the tags which are having probability greater than 0.5 are given as the final output from the tag classifier.

To identify and use the best combination of Text Encoder and Code Encoder, we executed four different combinations of the latest and advanced transformer models for text and code. The text transformers selected are all-distilroberta-v1 and T5 [12]. all-distilroberta-v1 is a sentence transformer which is an Encoder type model. T5 is a Encoder-Decoder type model. The code transformers used are CodeT5 [13] and CodeBERT. The CodeBERT and CodeT5 are based on BERT [14] and T5 architectures respectively and are trained on coding data.

3.4 Fusion Layer

The keywords obtained from Keyword extraction and the tags obtained from the Tag Classifier are then combined to generate a final set of tags that will be recommended to the user.

4. IMPLEMENTATION DETAILS

The data required for training our models for our system was obtained from Kaggle. This dataset was extracted from Stack Exchange Data Explorer. The data is collected from 2009 to 2020. A tag is considered rare when its frequency of presence falls below a given cutoff θ. The rationale is that a tag that appears seldom throughout a large dataset, like the one used in this study, is either poorly designed or little used, making it unfamiliar to programmers. Rare tags are eliminated to increase the dataset's usability. According to [3] [15] [16], the cutoff θ was fixed at value 50 for detecting and eliminating rare tags. As a result, we remove from the dataset any posts that contain rare tags and posts that only contain rare tags. The dataset, which started with 23,020,127 postings, now has 29,284 common tags and 34,369 unusual tags after this filtering. There is an 80:20 split between the training and test sets in the final data set.

The Hugging Face Transformers module in Python was used to create the suggested system. After that, it was run in a Kaggle environment on a GPU P100. The batch size and learning rate are set to 8 and 0.001, respectively. Adam is utilized as the optimizer, and the maximum length of input text is

256. Over 200 epochs, the fine-tuning process has been finished.

5. RESULT ANALYSIS

The effectiveness of the approach was evaluated using Precision@k, Recall@k, and F1-Score@k which were considered as the evaluation metrics for the baseline models.

- **Precision@k:** The average ratio of anticipated ground truth tags among the top-k suggested tags is measured by precision@k.
- **Recall@k:** The ratio of accurately predicted ground truth tags in the list of ground truth tags is reported by Recall@k.
- **F1-score@k:** Typically used as a summary metric, F1-score@k is the harmonic mean of Precision@k and Recall@k.

The four variants of the tag classifier are trained with the trained set and are then evaluated based on their performance on the test set. Their performance was evaluated using Precision@1, Recall@1, and F1-Score@1 metrics.

Table 1: Evaluation Metrics for Text and Code Encoder Models

Text Encoder	Code Encoder	Precision@1	Recall@1	F1-score@1
Sentence Transformer (all-distilroberta-v1)	CodeT5	0.464	0.464	0.464
Sentence Transformer (all-distilroberta-v1)	CodeBERT	0.117	0.117	0.117
T5	CodeBERT	0.590	0.590	0.590
T5	**CodeT5**	**0.843**	**0.843**	**0.843**

Their performance can be seen in Table 1. From the table it can be seen that the combination of all-distilroberta-v1 and CodeBERT performed worst whereas the combination of T5 and CodeT5 performed best. Therefore, T5 and CodeT5 combination is selected as the optimal combination of the models for tag classifier.

The Figure 4, Figure 5, Figure 6 are the Epochs vs Precision, Epochs vs Recall and Epochs vs F1-score respectively of different model combinations after training for 200 epochs.

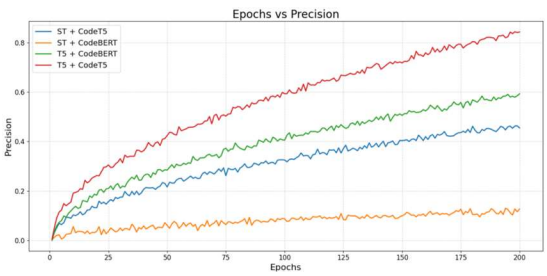

Figure 4: Epochs vs Precision graph

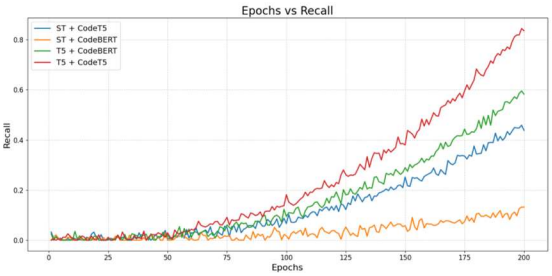

Figure 5: Epochs vs Recall graph

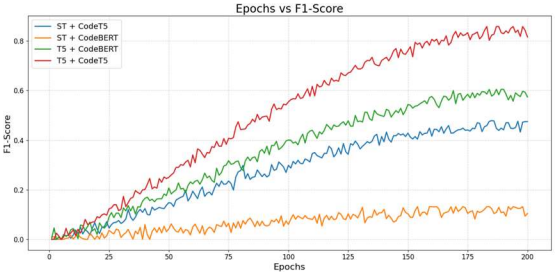

Figure 6: Epochs vs F1-Score graph

The performance of our TriTag Recommender made up of T5 and CodeT5 encoders with KeyBERT is compared with the competitive models or state-of-the-art approaches as shown in Table 2 and is also represented as a bar plot in Figure 7.

Their performance is evaluated with metrics Precision@5, Recall@5, and F1-Score@5. The Recall@5 of our model is 0.885 which shows an increase of 17% when compared with 0.757 which is the highest of the competitive models. Similarly, The Precision@5 of our model is 0.516 which shows an increase of 5% when compared with 0.493 which is the highest of the competitive models. Similarly, The F1-Score@5 of our model is 0.652 which shows an increase of 9% when compared with 0.597 which is the highest of the competitive models. The metrics clearly show that our model outperforms the existing approaches

Table 2: Comparative Analysis on StackOverflow Dataset

Approach	Recall@5	Precision@5	F1-Score@5
Roostaee [2]	0.720	0.392	0.493
PTM4Tag [16]	0.757	0.493	0.513
RACM [17]	0.653	0.452	0.345
T5+CodeT5+KeyBERT	**0.885**	**0.516**	**0.652**

To further analyze the effectiveness of the TriTag Recommender beyond numerical metrics, we performed a qualitative evaluation by observing the model's tag predictions on a variety of real-world questions. This allowed us to assess the semantic correctness, practical relevance, and generalization capability of the models. Table 3 shows a few sample inputs with their corresponding actual tags (as labeled in the dataset) and the tags predicted by the TriTag Recommender.

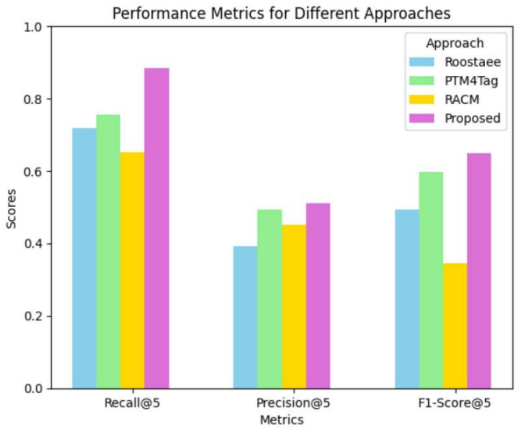

Figure 7: Performance metrics result

Table 3: Tag Prediction Comparison

Question Title	Actual Tags	Predicted Tags
application routes with spring mvc	["java"]	["java"]
regex python with unicode japanese character issue	["regex", "python"]	["regex", "python", "string"]
challenge optimize unlisting easy	["optimization", "r"]	["algorithm", "c++", "java", "optimization", "performance", "math"]
why false output false in	["python-3.x"]	["python", "python-3.x"]

6. CONCLUSION

Tags play an important role in community platforms. They are readily used to direct questions to the feed of the users with expertise in answering those questions. While these tags are helpful if they are relevant or correct, this is not always true since users have limited understanding. This affects the platform's performance. To solve this, we presented a novel deep learning-based Recommender for automated tag suggestions for CQA platforms using domain-specific pre-trained transformer models, T5 and CodeT5 and KeyBERT. Our approach captures the context of natural language and code and recommends relevant tags. The scientific contribution of our work lies in integrating domain-specific transformer models (CodeT5 and T5) with keyword extraction techniques (KeyBERT) to create a hybrid tag recommender that outperforms existing state-of-the-art solutions. This approach advances the field by effectively combining code and natural language understanding to achieve superior performance in Recall@5, Precision@5, and F1-Score@5 compared to prior works.

Even though our approach performed well, there are a few limitations. There was semantic overlap in tags, where the model sometimes struggled to differentiate between tags with similar meanings like Python and python-3.x. Our approach can only suggest tags that are used during training and which are present in the text. It cannot effectively suggest tags for new or evolving topics or tags that are not explicitly mentioned in the text.

In the future, we plan to explore tag generation approaches to generate tags for even new or evolving tags. It is to be noted that personalized recommendations can be made by utilizing user information like user profiles, user tags, user history, etc.

REFERENCES:

[1] Chehreh, Isun, Ebrahim Ansari, and Bahram Sadeghi Bigham. "Advanced Automated Tagging for Stack Overflow: A Multi-Stage Approach Using Deep Learning and NLP Techniques." In 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), pp. 1-6. IEEE, 2024.

[2] Roostaee, M. "Language-independent Profile-based Tag Recommendation for Community Question Answering Systems."

[3] Li, Can, Ling Xu, Meng Yan, and Yan Lei. "TagDC: A tag recommendation method for software information sites with a combination of deep learning and collaborative filtering." Journal of Systems and Software 170 (2020): 110783.

[4] Li, Lin, PeiPei Wang, Xinhao Zheng, and Qing Xie. "Code-enhanced fine-grained semantic matching for tag recommendation in software information sites." In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1-5. IEEE, 2023.

[5] He, Junda, Bowen Xu, Zhou Yang, DongGyun Han, Chengran Yang, Jiakun Liu, Zhipeng Zhao, and David Lo. "PTM4Tag+: Tag recommendation of stack overflow posts with pre-trained models." Empirical Software Engineering 30, no. 1 (2025): 1-41.

[6] Amsa–nguan, Thitiworada, Nawakarn Leerattanachote, Ponlawat Suparat, Piyanit Wepulanon, and Santitham Prom–on. "Development of the Topic Tagging System for Thai and English-Translated Web Contents." In 2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE), pp. 239-245. IEEE, 2024.

[7] Raja Ram A, Sanjay kumar V, Ratheesh B. "Automated Question Tagging Using Machine Learning"

[8] Li, Lin, Peipei Wang, Xinhao Zheng, Qing Xie, Xiaohui Tao, and Juan D. Vel´asquez. "Dual-interactive fusion for code-mixed deep representation learning in tag recommendation." Information Fusion 99 (2023): 101862.

[9] Devine, Peter, and Kelly Blincoe. "Unsupervised extreme multi label classification of stack overflow posts." In Proceedings of the 1st International Workshop on Natural Language-based Software Engineering, pp. 1-8. 2022.

[10] Subramani, Srinivas, Sangeetha Rajesh, Kirti Wankhede, and Bharati Wukkadada. "Predicting tags of stack overflow questions: A deep learning approach." In 2023 Somaiya International Conference on Technology and Information Management (SICTIM), pp. 64-68. IEEE, 2023.

[11] Wiratchawa, Kannika, Tanutcha Khunthong, and Thanapong Intharah. "LegalBERT-th: development of legal Q&A dataset and automatic question tagging." In 2021 18th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON), Chiang Mai. 2021.

[12] Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer." Journal of machine learning research 21, no. 140 (2020): 1-67.

[13] Wang, Yue, Weishi Wang, Shafiq Joty, and Steven CH Hoi. "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation." arXiv preprint arXiv:2109.00859 (2021).

[14] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pp. 4171-4186. 2019.

[15] Xu, Bowen, Thong Hoang, Abhishek Sharma, Chengran Yang, Xin Xia, and David Lo. "Post2vec: Learning distributed representations of Stack Overflow posts." IEEE Transactions on Software Engineering 48, no. 9 (2021): 3423-3441.

[16] He, Junda, Bowen Xu, Zhou Yang, DongGyun Han, Chengran Yang, and David Lo. "PTM4Tag: sharpening tag recommendation of stack overflow posts with pre-trained models." In Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, pp. 1-11. 2022.

[17] Lu, Sijin, Pengyu Xu, Bing Liu, Hongjian Sun, Liping Jing, and Jian Yu. "Retrieval Augmented Cross-Modal Tag Recommendation in Software Q&A Sites." arXiv preprint arXiv:2402.03635 (2024).