


Cross-Community Question Relevance Prediction for Stack Overflow and GitHub


Song Yu

(Central South University, Hunan, China)

 <https://orcid.org/0009-0000-1286-2343>, ys@csu.edu.cn


Bugao Jiang

(Central South University, Hunan, China)

 <https://orcid.org/0009-0002-7923-9710>, 1169182681@qq.com


Danni Zhang

(Central South University, Hunan, China)

 <https://orcid.org/0009-0002-6908-4487>, 1074271129@qq.com

Zhifang Liao

(Central South University, Hunan, China)

 <https://orcid.org/0000-0002-5525-904X>, zfliao@csu.edu.cn

Abstract: As the open-source community has evolved, Stack Overflow (SO) has gained extensive usage. The question-and-answer community's mechanism for recommending related questions helps users discover more content relevant to their current problems, expediting issue resolution. However, the recommendation of relevant questions in a single community context limits the amount of available content and the diversity of content, and the recommendation results rely heavily on the existing knowledge of the community. Stack Overflow still harbors a substantial number of unresolved questions. To address this situation, this paper proposes a cross-community question relevance prediction model, CCQRP, to predict the relevance of Stack Overflow questions and GitHub(GH) issues, and recommend relevant GitHub issues. CCQRP aims to assist developers in effectively resolving problems and enhancing development efficiency. We design an embedding layer incorporating BERTOverflow and Bi-LSTM and devise a weighted attention matrix based on named entity types of tokens. This matrix assigns different weights to tokens of varying named entity types during the prediction process, capturing critical information to predict the relevance of SO questions and GH issues. Due to the lack of existing datasets, we construct a dataset named Question-Issue dataset (QI), consisting of Stack Overflow questions, GitHub issues, and the corresponding question-issue relevance, containing 240,000 related SO question-GH issue pairs and 470,000 unrelated pairs. We evaluate the effectiveness of CCQRP on QI. Compared to the latest models (MQDD, CodeBERT, ASIM), CCQRP demonstrates an improvement in F1-score ranging from 0.60% to 10.86% and exhibits robust generalization capabilities.

Keywords: Relevant Question, Relevance Prediction, Stack Overflow, GitHub, Deep Learning

Categories: I.2, D.12

DOI: 10.3897/jucs.119772

1 Introduction

As one of the most representative question-and-answer communities in the field of software engineering, Stack Overflow (SO) has surpassed a monthly visitor count of over 100 million, accumulating more than 70 million questions and answers. However, despite such high user activity, there remains a significant number of unresolved questions within SO. As of August 2019, 47% of SO questions were still unresolved [Yazdaninia et al. 2021]. According to Stack Exchange¹, as of October 2022, there were over 500,000 questions posted on Stack Overflow from January 2022 to October 2022 that still had no answer.

Existing work has aimed to mitigate this situation by detecting related questions and generating answers for target questions. Xu et al. [Xu et al. 2016] and Liao et al. [Liao et al. 2021] analyzed the relevance between target questions and other questions on SO, and recommended related questions or answers. Additionally, Cai [Cai et al. 2019], Wu [Wu et al. 2021], Silva [Silva et al. 2019], defined solutions as seq2seq tasks and had respectively introduced methods like AnswerBot, CAGKG, and CROKAGE to generate answers directly. Existing work has concentrated the emphasis on solutions within the SO community, leveraging the similar knowledge organization structure within the same community, which is easier to handle. However, the effectiveness of recommendations relies significantly on the development of SO.

As the largest hosting platform for both open-source and private software projects, GitHub (GH) has over 83 million developers and more than 200 million repositories. GitHub hosts a large number of issues presented in question-and-answer format. There's a significant overlap in topic and question-and-answer formats between SO questions and GH issues. Additionally, there's a substantial amount of spontaneously established cross-community URL links between developers on these two platforms. In our analysis of the Stack Overflow dataset², we find a total of 320,000 GH issue links. As illustrated in Figure 1, SO users often append GH issue URL links in their comments (C), directing an SO question (Q) to a related GH issue (I), and indicating within the comment (C) that the solution to Q exists within the Q&A content of I.

We further analyze the Stack Overflow dataset based on the "score" metric to explore whether the introduction of GH issues has been beneficial. The "score" metric on Stack Overflow allows users to evaluate answers by voting: choosing "useful" adds one point to an answer, while "not useful" deducts one point. The statistical findings reveal that the average score for all answers is 3.11, whereas answers containing GH issue links have an average score of 4.68. Additionally, the average score for all comments is 0.32, while comments containing GH issue links have an average score of 0.43. Both averages show varying degrees of improvement, indicating that integrating related GH issues into SO discussions can enhance the completeness of answers and comments by providing additional context, or can provide solutions directly. Answers and comments containing related GH issues receive greater user approval. Cross-community sharing of issues significantly augments the quantity and diversity of retrievable answers and content, serving as a reference to improve the quality of answers in SO. However, there is currently limited research on the relevance of cross-community questions, and cross-community question sharing primarily relies on users' spontaneous actions. There is a lack of automated detection tools, making it challenging for users to find GH issues

¹ <https://data.stackexchange.com/stackoverflow/queries>

² based on data retrieved from the Stack Exchange Data Explorer in October 2022, including questions, answers, and comments

related to current SO questions from a huge amount of data.

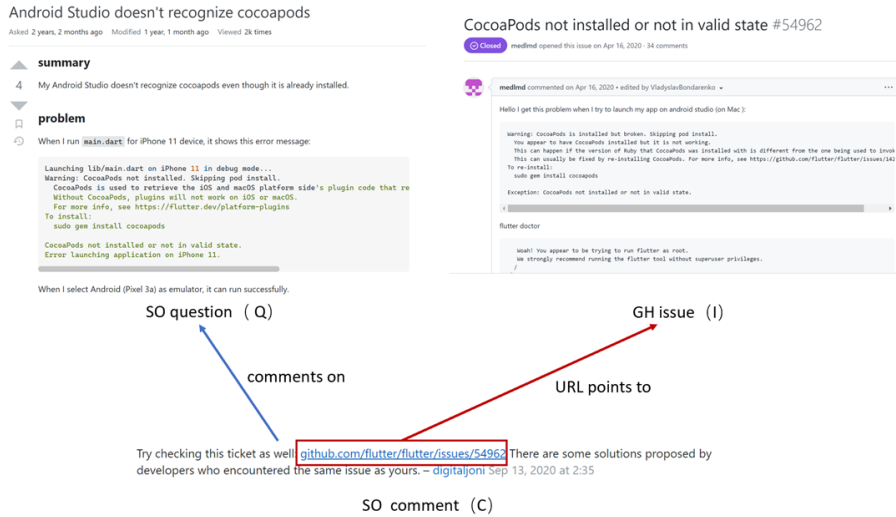


Figure 1: An instance of linking SO questions to GH issues through SO comments

In this paper, we propose an innovative approach from a cross-community perspective to assist developers in problem-solving. Specifically, it associates SO questions with relevant GH issues and recommends relevant GH issues for SO questions. To the best of our knowledge, no previous methods have been implemented from this perspective.

We construct a dataset named Question-Issue dataset (QI) based on the GH issue links within SO. This dataset comprises SO questions, GH issues, and the relevance. Additionally, we propose a cross-community question relevance prediction model (CCQRP) aiming at predicting the relevance between SO questions and GH issues. CCQRP design an embedding layer incorporating BERTOverflow and Bi-LSTM, and devise a weighted attention matrix based on named entity types of tokens, adjusting the weight of each token in the attention matrix according to its named entity types. This adjustment can make the model focus more on tokens associated with crucial named entity types during the relevance prediction process. Ultimately, leveraging the predicted results, related GH issues can be recommended for SO questions.

The contributions of this paper are as follows:

1. A diverse dataset named QI. Based on the GH issue URL links within SO, we construct a dataset QI, which comprises 240,000 pairs of related SO questions - GH issues and 470,000 pairs of unrelated samples. QI includes title and body of each SO question and GH issue.
2. A novel cross-community question relevance prediction model(CCQRP) based on the named entity types of tokens. The model adjusts the weights of each token according to its named entity types and computes the weighted attention matrix for a SO question and a GH issue to realize the relevance prediction.

3. A Comparison of three state-of-the-art question relevance prediction model(MQDD, ASIM and CodeBERT) and ChatGPT-3.5-Turbo from multiple points of view. We conduct several experiments on QI and compared with existing models, the experiment results show that our model outperforms the existing models in Precision, Recall, and F1 score to varying degrees.

2 Related Work

In this section, we will briefly introduce related work. Firstly, we introduce related question detection in SO. Secondly, we introduce cross-community research on SO and GH.

2.1 Related Question Detection

Question relevance prediction refers to the analysis and comparison of a pair of questions by machine learning or natural language processing to determine their similarity or relevance. It is commonly utilized in question-and-answer community to ascertain the similarity between a given question and those existing within a database or knowledge repository, thereby identifying the most relevant question. Benefiting from memory structure, RNN and its variants are widely used in natural language processing [Godbole et al. 2018] [Ye et al. 2017]. Wang [Wang et al. 2020] built three deep learning approaches WV-CNN, WV-RNN, and WV-LSTM based on Word2Vec, CNN, RNN, and LSTM to fully capture the word-level and document-level semantic information of each question pair in Stack Overflow. Three types of word embeddings involving Google news vector embedding, FastText crawl embedding with 300 dimensions, and FastText crawl sub-words embedding with 300 dimensions were implemented individually to vectorize all the questions and train the model. Similarly, Z. Imtiaz [Imtiaz et al. 2020] combined same three word embedding features and applied Siamese MaLSTM (“Ma” for Manhattan distance) Neural Network model for prediction of duplicate questions in Quora Question Pairs dataset. Xu [Xu et al. 2016] regarded an SO question along with its answer as a knowledge unit, adopting a neural language model (word embeddings) and a convolutional neural network (CNN) to capture word- and document-level semantics of knowledge units. Liao [Liao et al. 2021] employed Siamese Bi-LSTM to encode question pairs and capture the semantic interaction information of title and body through soft align attention and inference composition. Different from recent language representation models, BERT [Kenton et al. 2019] was designed to pre-train deep bidirectional representations from unlabeled text. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question similarity prediction. Pašek J [Pašek et al. 2022] leveraged data collected on the Stack Overflow website to pre-training a multimodal model for searching duplicates on question-answering websites. They also designed two new learning objectives to improve duplicate detection capabilities.

2.2 Cross-Community Research Based on SO and GH

As two popular communities, Stack Overflow and GitHub overlap to a large extent both in terms of users and technologies, providing the possibility of knowledge sharing between the two communities. On one hand, Anastasia Reinhardt [Reinhardt et al. 2018] used

API usage patterns mined from 380,000 GitHub projects to detect API usage violations in SO posts. They quantified the number of GitHub examples that follow API usage patterns and showed how to fix the detected violations in a given SO code fragment. On the other hand, Yang [Yang et al. 2017] and Sebastian Baltes [Baltes et al. 2019] found that there was a large amount of duplicate code in Stack Overflow and GitHub and that developers often looked for and reused code in Stack Overflow when writing open source projects, enabling code flow from Stack Overflow to GitHub. In a study of the change history of reused code, Saraj Singh Manes [Manes et al. 2020] found that 76% of code snippets changed on Stack Overflow, while only 22% of reused code snippets changed on GitHub. The Stack Overflow code snippets experienced fewer changes compared to the changes on GitHub. To enable timely propagation of changes to reused code in SO to GH, Chaiyong Ragkhitwetsagul [Ragkhitwetsagul et al. 2022] developed Matcha, a code recommendation tool that leverages Stack Overflow snippets with version history and code clone search techniques to identify sub-optimal code in a GitHub project and suggested their optimized versions. Shipra Sharma [Sharma et al. 2019] used the data provided by Stack Overflow to identify potential defects in a given source code and tested it on source code samples from over 300 GitHub open source repositories. Roy Ka-Wei Lee [Lee et al. 2017] proposed similarity scores to measure the similarity of developers' interests within and across social collaboration platforms and applied the proposed similarity scores to empirical studies on GitHub and Stack Overflow. Overall, the depth and breadth of knowledge mining are improved by crossing Stack Overflow and GitHub communities, and the results are more valuable and informative compared to studies on individual communities.

However, there has been limited attention from researchers on the correlation between Stack Overflow (SO) questions and GitHub (GH) issues. When recommending related questions, the candidate set of related questions has primarily been confined to SO. Despite the similarity in content between SO questions and GH issues, the answers are not effectively reused because they are distributed in different communities.

3 Method

In this section, we mainly introduce the workflow of CCQRP and the overall structure of the model. Figure 2 presents the overall framework of CCQRP, which contains three phases, data preparation, data preprocessing, and model building. We first scrape GH issue URL links from SO comments and answers, match SO question-GH issue pairs, clean the data, remove duplicate pairs and construct the Question-Issue dataset (QI). The relevance prediction model consists of two parts: named entity recognition and relevance prediction. It computes the attention matrix between SO questions and GH issues, incorporating named entity type features of SO questions and GH issues to predict the relevance.

3.1 Dataset Construction

As shown in Figure 1, SO users spontaneously share links to GH issues relevant to the respective SO question within SO answers or comments. Therefore, we take the GH question URL link in the answer or comment as a starting point, take the GH issue pointed to by the issue link contained in the comment or answer (C/A) as I, and take the SO question to which the C/A belongs as Q, and take the $\langle Q, I \rangle$ pair as a positive sample (correlation pair). All the related question-issue pairs are mined from the SO dataset

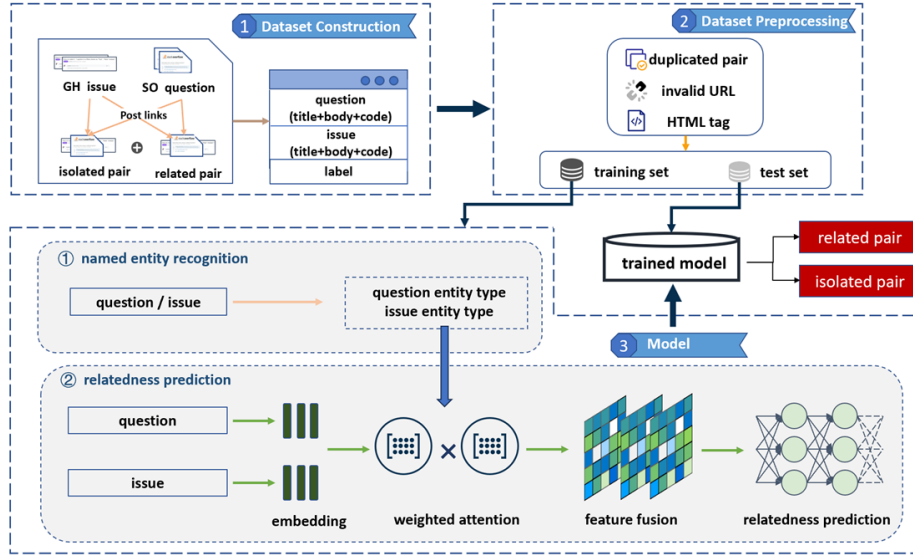


Figure 2: Overall framework of CCQRP

released by Stack Exchange in October 2022. To be specific, we use a regular expression³ to mine comments or answers that contain GH issue URL links. After filtering out a target answer or comment, we match it to its respective Stack Overflow questions (Q) by the parent ID property. Then, by leveraging PyGithub, we scrape the content of the GH issue(I) linked by the issue URL mentioned in the answer or comment, including its title and body. After completing these steps, we construct a dataset of 240,000 pairs of related question-issue pairs (QI-positive). Following Liao et al.[Liao et al. 2021], we categorize QI-positive based on programming languages.

A SO question and a GH issue are randomly selected to form a $\langle Q, I \rangle$, and if the $\langle Q, I \rangle$ does not appear in QI-positive, it will be classified as unrelated. For each programming language, we construct two negative datasets in two ways. For the first negative dataset, we randomly select n unrelated pairs (n is the number of positive samples for the programming language). To gain a deeper understanding of the actual performance of CCQRP, we try to increase the prediction difficulty by increasing the textual similarity of unrelated question-issue pairs. Therefore, for the second negative dataset, we rank the textual similarity of all possible unrelated question-issue pairs and select top n pairs to construct another negative dataset. The similarity is defined as follows.

$$\text{sim}_{(q,i)} = \frac{\text{TF-IDF}(q) \cdot \text{TF-IDF}(i)}{|\text{TF-IDF}(q)| \times |\text{TF-IDF}(i)|} \quad (1)$$

Since the data provided by Stack Exchange is not preprocessed, we use bs4 library to remove HTML tags in the SO dataset, including `<pre>` and so on. In the process of crawling GH issues, many issue links are published early, and the issue may have been deleted or the links are not working. PyGithub will return an exception. So we

³ `.*https://[/]github.com[/] [a-zA-Z0-9]+[/] [a-zA-Z0-9]+[/] issues[/]\d+.*`

need to remove it from the dataset. Finally, the same issue URL link may be posted in different answers or comments to the same question, resulting in duplicate question-issue pairs to remove.

3.2 Model Description

CCQRP consists of two main modules, the first one is the named entity recognition module, which fuses three features including word embedding of each token of the question or the issue, whether the token is a code or not, and whether the token is an entity or not to predict the entity type of the token. The second module takes a SO question and a GH issue as inputs, calculating the initial attention matrix E from the word embedding representations of the SO question and GH issue, transforming the named entity types of tokens predicted by the first module into named entity type weight matrices (w_q, w_i) through convolution operation, constructing new representations of questions and issues by dot-multiplying w_q and w_i with E respectively, and fusing the new and old representations to predict relevance.

3.2.1 Named Entity Recognition Module

Named Entity Recognition (NER) refers to the identification of proper nouns from natural language text and classifying them into named entity types, such as person, location, and organization [Ahmad et al. 2020]. However, named entity types defined based on the general corpus are not applicable to the field of software engineering. Inspired by Jeniya Tabassum [Tabassum et al. 2020], we define the named entity types into 24 categories, as shown in Table 1. “O” means the token is not a named entity. In traditional named entity recognition, as a named entity may consist of multiple tokens — for instance, “San Francisco” is a “Location” entity — BI annotation is often utilized to indicate whether a token is at the beginning, middle, or end of a named entity. However, in the subsequent prediction process of this paper, we treat each token of the same entity uniformly. Consequently, the BI annotation is omitted.

id	entity type	id	entity type	id	entity type
1	value	9	file type	17	output block
2	application	10	function	18	UI element
3	class	11	HTML/XML tag	19	algorithm
4	code block	12	keyboard input	20	variable
5	data structure	13	library	21	device
6	data type	14	license	22	language
7	error name	15	operating system	23	version
8	filename	16	organization	24	O

Table 1: 24 named entity types

The specific implementation details of named entity recognition are illustrated in Figure 3. During the recognition process, the named entity types of tokens of SO questions and issues are predicted separately. We will illustrate with an example of the named entity recognition for a SO question. Firstly, we extract the BERT embedding features $V1$

for the question $\{v_{11}, v_{21}, v_{31}, \dots, v_{m1}\}$, where v_{k1} represents the BERT embedding for the k^{th} token in the question, and m denotes the total number of tokens for the question. While composing SO questions, users may not strictly adhere to the community's code formatting guidelines, causing some code to be unrecognizable by HTML tags. Therefore, we introduce a code recognition module to predict whether the token belongs to a code snippet. The prediction results form feature $V2\{v_{12}, v_{22}, v_{32}, \dots, v_{m2}\}$, where v_{k2} is a vector indicating whether the k^{th} token is code or not. $V3$ is generated by the entity existence recognition module. It's worth noting that this module doesn't predict the named entity type of the token, but rather predicts whether the token is part of a named entity. $V3$ is presented as $\{v_{13}, v_{23}, v_{33}, \dots, v_{m3}\}$, where v_{k3} is a vector indicating whether the k^{th} token is part of a named entity or not.

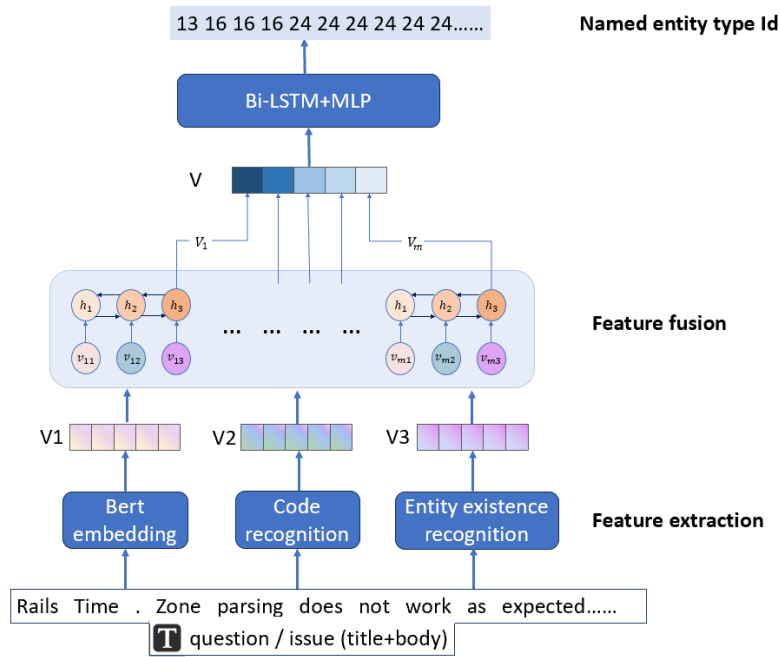


Figure 3: Named Entity Recognition Module

Each token's three features will be sequentially input into the Bi-LSTM to obtain a fused feature v_k :

$$v_k = \text{BiLSTM}(v_{k1}, v_{k2}, v_{k3}) \quad (2)$$

The representation V of the question is defined as:

$$V = \{v_1, v_2, v_3, \dots, v_m\} \quad (3)$$

V is then sent into another Bi-LSTM which is introduced to capture semantic interactions between tokens. A multi-layer perceptron is used to predict the named entity

type of each token $\{t_1, t_2, t_3, \dots, t_m\}$. When the number of tokens of a SO question or GH issue is less than the max sequence length of BERTOverflow, we default the named entity type of the padded portions to “O”.

During the training process, the named entity recognition module and the relevance prediction module are trained independently. We will train the named entity recognition module first and evaluate its recognition effect. In the subsequent training and prediction process, the parameters of this module will be fixed.

3.2.2 Relevance Prediction Module

In this section, we will detail the process of integrating the named entity types of tokens predicted in Section 3.2.1 into the question-issue attention matrix and implement question-issue relevance prediction. The specifics implementation are outlined in Figure 4.

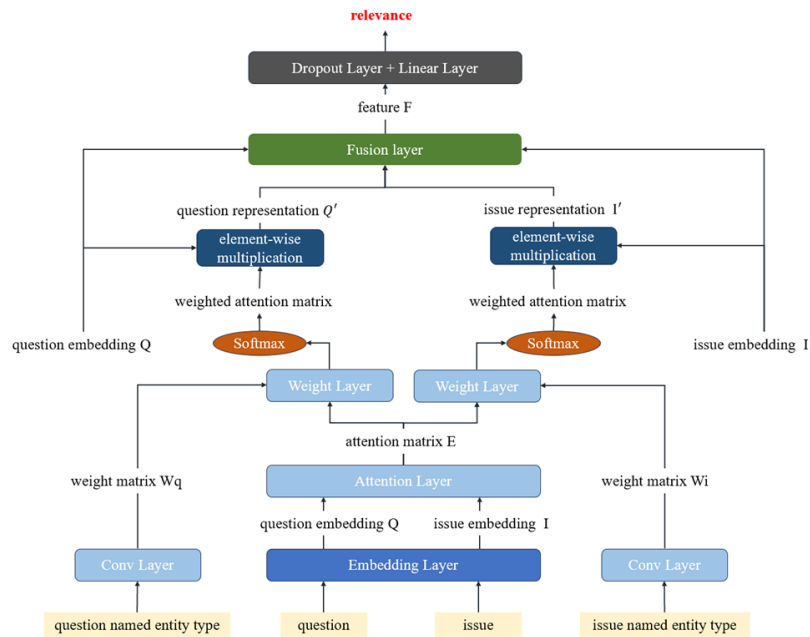


Figure 4: Relevance Prediction Module

3.2.2.1 Word Embedding Module

To better capture the semantic information of questions and issues, we’ve devised a dual-layered word embedding module, comprising BERTOverflow [Tabassum et al. 2020] and Bi-LSTM. Since BERT [Kenton et al. 2019] was proposed, a large number of variants have appeared (e.g. RoBERTa [Liu et al. 2019], ALBERT [Lan et al. 2019]), and have been widely applied to solve a variety of specialized domain tasks based on differences in pre-training datasets. Considering the characteristics of QI, we design the

first layer of the word embedding module as BERTOverflow, which is based on BERT and is pre-trained on 152 million sentences from Stack Overflow posts. The second layer of the word embedding module is Bi-LSTM used to capture semantic interactions between tokens. The representations of the question and issue are depicted as follows:

$$Q = (eq_1, eq_2, \dots, eq_m) \quad (4)$$

$$I = (ei_1, ei_2, \dots, ei_m) \quad (5)$$

eq_k and ei_k represent the word embeddings of the k^{th} token of question and issue, respectively.

3.2.2.2 Weighted Attention Module

In this section, we design a weighted attention mechanism based on token named entity types to capture semantic interaction between question and issue. The initial attention matrix $E \in R^{n \times m}$ is defined as follows:

$$E_{ul} = eq_u \cdot ei_l, u \in [1, 2, \dots, m], l \in [1, 2, \dots, n] \quad (6)$$

E_{ul} represents the inner product of the word embeddings of the u^{th} token of the question and the l^{th} token of the issue.

In Section 3.2.1, we obtained the named entity type for each token of question and issue $Qt = (qt_1, qt_2, \dots, qt_m)$, $It = (it_1, it_2, \dots, it_m)$ through the named entity recognition module. To convert Qt and It into model-usable data, we use a 24-bit one-hot encoding to represent the 24 named entity types, and map Qt and It to one-hot encoding matrices. To obtain the weights of each token based on its named entity type, we devised a 1D convolutional neural network whose $in_channels = 24$, $out_channels = 1$, $kernel_size = 1$, and $stride = 1$. The 24 parameters of the convolutional kernel represent the weights of the 24 named entity types. The specific calculation process is shown in Figure 5. Through convolution operations on the one-hot encoding matrix, we can convert the one-hot encoding named entity types into corresponding weights for those named entity types, resulting in the named entity type weight matrix $Nq \in R^{m \times 1}$ for the question (for issue, it is $Ni \in R^{n \times 1}$). Nq_k and Ni_k represent the weight of the k^{th} token of the question, issue based on the named entity types. Before training, the 24 parameters of the convolutional kernel are uniformly initialized as 1. It means that we assign the same initial weight to each named entity type, allowing the model to continuously adjust them during the subsequent training process.

Based on the attention matrix E and named entity type weight matrix Nq , we can get the weighted attention matrix E' of the question, E' is defined as:

$$E'_{ul} = Nq_u \times E_{ul}, u \in [1, 2, \dots, m], l \in [1, 2, \dots, n] \quad (7)$$

E'_{ul} is computed from the initial attention value E_{ul} and the weight Nq_u . Different named entity types correspond to distinct weights Nq_u . The greater the weight, the higher the attention value for token u . Eventually, we obtain a new representation Q' based on the weighted attention matrix E' . E' indicates the correlation between Q' and I .

$$\hat{E}_{ul} = \text{softmax}\left(\frac{E'_{ul}}{\sum_{l=1}^n E'_{ul}}\right), u \in [1, 2, \dots, m], l \in [1, 2, \dots, n] \quad (8)$$

$$Q' = \hat{E} \cdot I \quad (9)$$

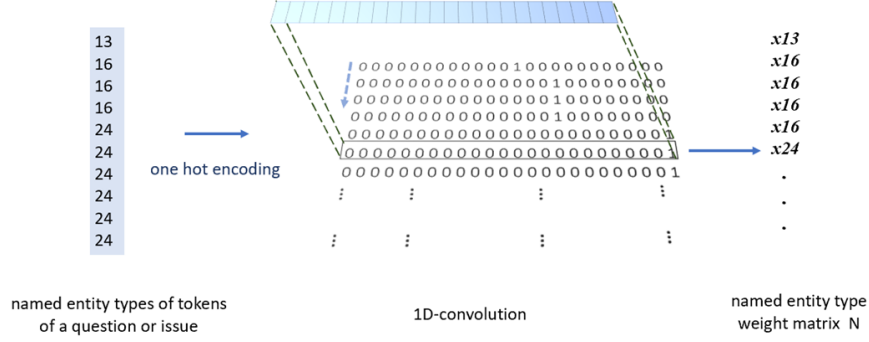


Figure 5: Token weights based on named entity types

Similarly, we can also obtain the new representation I' for the issue.

3.2.2.3 Prediction Module

The prediction module consists of a fusion layer and a prediction layer. Q and Q' , I and I' are fused before predicting. We refer to the fusion method in the paper [Yang et al. 2019] and use three ways to fuse the features:

$$\hat{Q} = \text{maxpool}(Q; Q') \quad (10)$$

$$\hat{I} = \text{maxpool}(I; I') \quad (11)$$

$$F = ((\hat{Q}; \hat{I}); (\hat{Q} - \hat{I}); (\hat{Q} \circ \hat{I})) \quad (12)$$

“;” denotes concatenation operation, “-” denotes subtraction operation, and “ \circ ” denotes element-wise multiplication. MaxPool layers are added to reduce the size of the features.

The prediction layer consists of a dropout layer preventing model overfitting and linear layers. Each of these linear layers is preceded by a LeakyReLU activation function.

We use the cross-entropy loss function to minimize prediction error:

$$L = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y})) \quad (13)$$

y is the real label of the relevance of the SO question and GH issue, with 1 being relevant and 0 being irrelevant, and \hat{y} is the result of the model, i.e., the probability that the GH issue is relevant to the SO question.

4 Experiment Setup

To assess the effectiveness of CCQRP, we conduct multiple experiments on QI, aiming to address the following four research questions:

- RQ1: Is it necessary to predict question relevance across communities?

- RQ2: How does the performance of our model compare with other existing methods?
- RQ3: How much of a role do the entity types of tokens in the title and the entity types of tokens in the body play respectively?
- RQ4: How well does the model generalize in predicting relevance across other programming languages?

4.1 Experiment Details

QI dataset: In Section 3.1, We’ve created a total of 240,000 relevant pairs, categorized them into sub-datasets based on programming languages, and build two corresponding subsets of irrelevant pairs. Each pair is stored in the format <question, issue, 0/1> (where 1 represents relevant and 0 represents irrelevant). We conduct experiments on the four programming languages with the largest sample sizes, including Python, JavaScript, C#, and Java. The specific details are illustrated in Table 2.

Language	Positive Dataset	Negative Dataset (random)	Negative Dataset (textual similarity)
Python	29529	29517	28763
JavaScript	22996	22920	23019
C#	11162	12025	10303
Java	9877	9963	9899

Table 2: Composition of dataset

Experiment setup: The ratio of the training set, validation set, and test set is 14:3:3. We set the max sequence length of BERTOverflow to 512, dropout probability to 0.6, learning rate to 0.00001, and each model is trained for 15 epochs using the Adam optimizer for backpropagation.

Evaluation metrics: The evaluation of CCQRP is based on three metrics: Precision, Recall and F1-score, defined as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (14)$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (15)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (16)$$

4.2 RQ1: Is it necessary to predict question relevance across communities?

To explore the necessity of predicting cross-community question relevance, we conduct a comparative analysis of TF-IDF cosine similarity between questions and issues. Firstly, we employ a method similar to Section 3.1 to mine relevant SO question q' for a given

SO question q (referred to as dataset S , formatted as $\langle q, q' \rangle$), using regular expression⁴ for matching. Subsequently, dataset S underwent filtration, preserving $\langle q, q' \rangle$ pairs where q had previously appeared in QI-positive, forming dataset S' . Finally, merging S' and QI-positive generated dataset SD , with data formatted as $\langle q, q', i \rangle$, where q' represents a SO question related to SO question q , and i denotes a GH issue related to SO question q .

We analyze and compare the average text similarity of $\langle q, q' \rangle$, $\langle q, i \rangle$, and $\langle q, q', i \rangle$ in the SD dataset. The specific calculation process is represented as follows:

$$\text{avg}_{\text{sim}(q,q')} = \frac{\sum_{k=1}^L \text{sim}(q_k, q'_k)}{L} \quad (17)$$

$$\text{avg}_{\text{sim}(q,i)} = \frac{\sum_{k=1}^L \text{sim}(q_k, i_k)}{L} \quad (18)$$

$$\text{avg}_{\text{sim}(q,q',i)} = \frac{\sum_{k=1}^L \max(\text{sim}(q_k, q'_k), \text{sim}(q_k, i_k))}{L} \quad (19)$$

L denotes the sample count in the dataset SD , $\text{sim}(q_k, q'_k)$ denotes the TF-IDF cosine similarity between q_k and q'_k in the k^{th} data entry $\langle q_k, q'_k, i_k \rangle$ in SD , $\text{sim}(q_k, i_k)$ denotes the TF-IDF cosine similarity between q_k and i_k in the k^{th} data entry $\langle q_k, q'_k, i_k \rangle$ in SD .

Dataset	Variable Name	Average TF-IDF Similarity Score
SO-SO	$\text{avg}_{\text{sim}(q,q')}$	0.2924
SO-GH	$\text{avg}_{\text{sim}(q,i)}$	0.2949
SO-SO-GH	$\text{avg}_{\text{sim}(q,q',i)}$	0.3818

Table 3: Average TF-IDF cosine similarity across different communities

The average TF-IDF cosine similarity of various data types is presented in Table 3. From the table data, it's evident that for an SO question q , the average similarity between the recommended SO question q' and q is 0.2924. When recommending a GH issue from a different community, the similarity can still reach 0.294. For a SO question, it's feasible to uncover pertinent content within GH issues without being constrained by community boundaries. By comparing $\text{avg}_{\text{sim}(q,q')}$ and $\text{avg}_{\text{sim}(q,i)}$, it's evident that after incorporating the GH issue dataset, the average similarity between question q and its most similar SO question or GH issue increased from 0.2924 to 0.3818. This demonstrates that in certain scenarios, the recommended GH issue might exhibit higher relevance to question q than the suggested SO question q' . In summary, the introduction of the GH issue dataset significantly enhances the similarity of recommended content. Implementing cross-community question relevance prediction is crucial to further improve content recommendations.

⁴ [http\[s\]://stackoverflow.com/questions/\d+](http[s]://stackoverflow.com/questions/\d+)

4.3 RQ2: How does the performance of our model compare with other existing methods?

To assess the effectiveness of CCQRP, we compare it against three state-of-the-art SO question relevance prediction models. Among these, MQDD [Pašek et al. 2022] was introduced by Pašek J et al. in 2022. It is trained on question descriptions and source codes in multiple programming languages. Pašek J et al. designed two new learning objectives to enhance duplicate detection capabilities. ASIM [Pei et al. 2021] was proposed by Jiayan Pei in 2021. Considering that questions/issues encompass programming language in addition to natural language, we include CodeBERT [Feng et al. 2020] as the third baseline model. The predictive performance of the various models on our two Python datasets is demonstrated in Tables 4 and 5.

Model	Precision	Recall	F1
MQDD	81.27	87.67	84.34
CodeBERT	92.46	95.31	93.87
ASIM	93.47	95.82	94.63
CCQRP	94.12	96.30	95.20

Table 4: Results of different models in the Python dataset with random negative dataset

Model	Precision	Recall	F1
MQDD	79.34	81.41	80.36
CodeBERT	88.75	78.26	83.17
ASIM	86.95	77.56	81.99
CCQRP	87.61	80.25	83.77

Table 5: Results of different models in the Python dataset with random similar dataset

As shown in Tables 4 and 5, on the Python dataset with random negative data, our model perform well compared to three baseline models in precision, recall, and F1 metrics, with improvements of 0.67%, 0.48%, and 0.57%, respectively, over the second-best performer in each metric. By comparing the results in Table 4 and Table 5, it's evident that as the similarity of the negative data increases, the precision, recall, and F1 scores of each model show varying degrees of decline. Despite MQDD and CodeBERT excelling in recall and precision, respectively, the scores in another metric are not as favorable. MQDD exhibits the lowest precision, and the recall of CodeBERT is 3.15% lower than the highest. Meanwhile, CCQRP manages to balance both metrics, maintaining a lead of 0.6% over the second-best performer in F1 score.

The latest advancements in Artificial Intelligence (AI) have brought various new opportunities to the field of software engineering. ChatGPT⁵ is a large language model based on Transformer. Through extensive unsupervised learning, ChatGPT extracts language patterns and knowledge from vast amounts of data, demonstrating strong

⁵ <https://openai.com/blog/chatgpt>

versatility in understanding languages across multiple tasks and domains. Its flexibility and performance in language generation make it an essential tool for researchers and developers exploring natural language processing and dialogue systems. Therefore, we choose ChatGPT-3.5-Turbo as an additional comparative baseline to analyze and compare its performance with CCQRP in cross-community question relevance prediction.

We use the ChatGPT API with the “gpt-3.5-turbo”. After testing several variations, we decide to test one by one to ChatGPT using the following prompt: “Here are two questions, please label them as related or unrelated. Question 1: ‘ $\{SO\ question\}$ ’. Question 2: ‘ $\{GH\ issue\}$ ’.” Where $\{SO\ question\}$ and $\{GH\ issue\}$ will be filled respectively with the titles and bodies of SO questions and GH issues. We conduct experiments on 2000 $\langle Q, I \rangle$ pairs randomly selected from the Python dataset. The experiment results are shown in Table 6.

Model	Precision	Recall	F1
ChatGPT-3.5	74.55	80.10	77.22

Table 6: Performance of ChatGPT

From the experiment results, we can see that the precision of ChatGPT-3.5-Turbo falls far behind CCQRP. To further explore the underlying reason, we attempted to have ChatGPT-3.5-Turbo explain why it deemed certain $\langle Q, I \rangle$ pairs related or unrelated, particularly those it predicted incorrectly. Figure 6 illustrates ChatGPT-3.5-Turbo’s explanation for a specific $\langle Q, I \rangle$ prediction. Through analysis, we’ve discovered that although Q and I discuss the same topic, their focus on the “exception” differs, which makes them unrelated. However, ChatGPT-3.5-Turbo simply considers them “related” because they discuss the same topic. This could be the reason behind ChatGPT-3.5-Turbo having high recall but low precision.

The two questions are related as both involve issues related to Gson, but they address different problems within Gson usage:

- The first question deals with a specific exception (`java.lang.reflect.InaccessibleObjectException`) occurring due to the logger (`logger.info()`) when combined with Gson’s serialization in a JUnit test.
- The second question discusses an issue with null fields serialization during a request call process and proposes a fix by directly using the Gson instance instead of the `TypeAdapter` within `GsonConverterFactory`.

Given their connection through the commonality of using Gson and dealing with different aspects of Gson’s behavior, I would output “1”.

Figure 6: ChatGPT-3.5-Turbo’s analysis of a $\langle Q, I \rangle$ pair

4.4 RQ3: How much of a role do the entity types of tokens in the title and the entity types of tokens in the body play respectively?

In this section, we divide the question/issue into two parts: title and body. To investigate the effect of the named entity types of different parts on the model’s predictive performance, we set all named entity types of tokens in the target part to “O” while retaining the named entity types in another part. Similar to Section 4.3, we conduct ablation experiments on the Python datasets. The experiment results are presented in Tables 7 and 8, where “Part” indicates the part where the named entity types of tokens are retained.

Part	Precision	Recall	F1
title + body	94.12	96.30	95.20
title	94.74	95.35	95.04
body	94.24	95.68	94.96

Table 7: Results in the Python dataset with random negative dataset

Part	Precision	Recall	F1
title + body	87.61	80.25	83.77
title	88.67	78.52	83.29
body	87.91	79.00	83.22

Table 8: Results in the Python dataset with similar negative dataset

The table illustrates that incorporating both the named entity types from the title and body improves the model’s performance. Across both datasets, there are enhancements in the F1 metric by 0.16%, 0.24%, 0.48%, and 0.55% when compared to using only the named entity type from the title or body. Notably, when comparing experiments using solely the named entity type from the title with those using only the body, the model demonstrates superior performance when considering the title’s named entity type. Despite titles being generally shorter than bodies, they often encapsulate the essence of the entire content. Titles tend to highlight keywords or critical statements, making them more valuable compared to the body.

4.5 RQ4: How is the generalization ability of CCQRP?

To explore the generalization ability of CCQRP, we apply our model on JavaScript, C#, and Java datasets shown in Section 3.1. The experimental results, as depicted in Tables 9 and 10, demonstrate the model’s consistently excellent prediction capability on dataset with random negative data, achieving F1 scores of 95.23%, 94.57%, and 94.49%. Moreover, the model maintains its F1 scores to a certain extent with an increase in the textual similarity of negative data, reaching 86.68%, 81.49%, and 86.56% (as shown in Table 10), respectively. It’s verified that our model has a certain generalization ability.

Language	Precision	Recall	F1
JavaScript	0.94319	0.96167	0.95234
C#	0.94497	0.94646	0.94572
Java	0.95701	0.93313	0.94492

Table 9: Results of CCQRP in different language datasets with random negative dataset

Language	Precision	Recall	F1
JavaScript	0.89711	0.83856	0.86685
C#	0.86042	0.77399	0.81492
Java	0.90858	0.82657	0.86564

Table 10: Results of CCQRP in different language datasets with similar negative dataset

5 Threats to Validity

In this section, we analyze several aspects that may pose threats to our method and the effectiveness of our experiments.

Threats to Internal Validity. Internal validity concerns potential errors within our experiments. To ensure the correct implementation of CCQRP, we meticulously reviewed the code to eliminate logical errors. For the three comparative methods mentioned in this paper, we replicated the models using the source code provided by the authors. Additionally, we conducted thorough debugging and validation to ensure the integrity of our implementation.

Threats to External Validity. External validity refers to the generalizability of our results. To ensure a more reliable evaluation of the model, we employed ten-fold cross-validation during both training and evaluation to test various performance metrics. Additionally, we tested CCQRP on four different programming languages and built two negative datasets for each language to validate its robustness.

Threats to Construct Validity. Threats to construct validity relate to the suitability of our evaluation metrics. We used precision, recall, and F1-score, which are widely accepted for evaluating model performance in classification tasks in software engineering. We evaluated other models on the same dataset and metrics to ensure consistency.

6 Conclusion

In this paper, we constructed a dataset concerning the relevance of Stack Overflow questions to GitHub issues named QI, and proposed a model named CCQRP to predict the relevance. CCQRP incorporates named entity type features, and continuously adjusts the weights of each named entity type through a convolutional neural network. It calculates the weighted attention matrix N based on named entity types of tokens of questions and issues. Through N , it fine-tunes the influence of tokens with different named entity types in the attention matrices, facilitating the prediction of relevance. Different from existing approaches, CCQRP surpasses community constraints, enabling cross-community prediction of question relevance. Experiments conducted on datasets in four programming languages—Python, JavaScript, C# and Java, indicate an improvement in F1-score ranging from 0.60% to 10.86% and demonstrate strong generalization capability.

Our work contributes to the software engineering community and the broader society by enhancing the efficiency of knowledge transfer between different platforms like Stack Overflow and GitHub. By predicting the relevance of questions across these communities, CCQRP facilitates quicker resolution of issues, thereby accelerating the software development process. This improvement not only benefits developers by saving time and effort but also contributes to the overall quality and reliability of software products, which are integral to various aspects of modern life. Additionally, by enabling cross-community knowledge sharing, our approach promotes a more collaborative and informed developer ecosystem, fostering innovation and continuous learning.

Possible future work will include the following aspects.

Integration of Project-Related Information. We plan to extract project-related information from the GitHub issue repositories and integrate this project data into the relevance prediction. In GitHub, issues are often raised in the context of specific project content. By appending project information, we aim to comprehensively capture various features of the issues, thereby further enhancing our analysis and prediction processes. This step will enable us to more accurately understand the background and context of the issues, providing a more comprehensive and in-depth information foundation for our work.

Exploring Semantic Similarity Measures. We aim to explore the possibility of using semantic similarity measures when combining the datasets. This could potentially make the proposed approach more efficient and generalizable, enhancing its performance and applicability.

Generalizability to Other Domains and Varying Datasets. We aim to test the generalizability of the proposed approach to other domains and varying datasets to understand the robustness and applicability of our model across different contexts and data sources. By extending the application of CCQRP to other question-and-answer communities within the software engineering domain, we seek to broaden the model's applicability. This expansion will allow our model to better serve diverse software development ecosystems, making it more versatile and capable of adapting to different development environments and workflows. Such an approach will provide users with more comprehensive and nuanced support, enhancing the overall utility and impact of our system.

Addressing Class Imbalance. We will investigate whether the variations in accuracy across different programming languages are due to class imbalance. And we plan to employ appropriate sampling methods to reduce this imbalance, potentially improving the accuracy and fairness of our model.

Acknowledgements

This work was funded in part by National Key Research and Development Program of China(2022YFC3601604).

References

[Ahmad et al. 2020] Ahmad, Muhammad Tayyab and Malik, Muhammad Kamran and Shahzad, Khurram and Aslam, Faisal and Iqbal, Asif and Nawaz, Zubair and Bukhari, Faisal : “ Named entity recognition and classification for Punjabi Shahmukhi ”; ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP),19,4(2020),1-13

- [Baltes et al. 2019] Baltes, Sebastian and Diehl, Stephan: “ Usage and attribution of Stack Overflow code snippets in GitHub projects ”; *Empirical Software Engineering*, 24, 3(2019), 1259-1295
- [Cai et al. 2019] Cai, Liang and Wang, Haoye and Xu, Bowen and Huang, Qiao and Xia, Xin and Lo, David and Xing, Zhenchang : “ AnswerBot: an answer summary generation tool based on stack overflow”; *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, 1134-1138
- [Feng et al. 2020] Feng, Zhangyin and Guo, Daya and Tang, Duyu and Duan, Nan and Feng, Xiaocheng and Gong, Ming and Shou, Linjun and Qin, Bing and Liu, Ting and Jiang, Daxin and others : “ CodeBERT: A pre-trained model for programming and natural languages ”; *Findings of the Association for Computational Linguistics: EMNLP,2020*,1536-1547
- [Godbole et al. 2018] Godbole, Ameya and Dalmia, Aman and Sahu, Sunil Kumar: “ Siamese neural networks with random forest for detecting duplicate question pairs”; *arXiv preprint arXiv:1801.07288*, 2018
- [Imtiaz et al. 2020] Imtiaz, Zainab and Umer, Muhammad and Ahmad, Muhammad and Ullah, Saleem and Choi, Gyu Sang and Mehmood, Arif: “ Duplicate questions pair detection using siamese malstm”; *IEEE Access* 8(2020), 21932-21942
- [Kenton et al. 2019] Kenton, Jacob Devlin Ming-Wei Chang and Toutanova, Lee Kristina: “ BERT: Pre-training of deep bidirectional transformers for language understanding”; *Proceedings of NAACL-HLT*, 2019,4171-4186
- [Lan et al. 2019] Lan, Zhenzhong and Chen, Mingda and Goodman, Sebastian and Gimpel, Kevin and Sharma, Piyush and Soricut, Radu: “ ALBERT: A lite BERT for self-supervised learning of language representations ”; *International Conference on Learning Representations*,2019
- [Lee et al. 2017] Lee, Roy Ka-Wei and Lo, David : “ Github and stack overflow: Analyzing developer interests across multiple social collaborative platforms ”; *Social Informatics: 9th International Conference, SocInfo 2017, Oxford, UK, September 13-15, 2017, Proceedings, Part II*,2017,245-256
- [Liao et al. 2021] Liao, Zhifang and Li, Wenlong and Zhang, Yan and Yu, Song : “ Detecting duplicate questions in stack overflow via semantic and relevance approaches ”; *28th Asia-Pacific Software Engineering Conference (APSEC)*, 2021, 111-119
- [Liu et al. 2019] Liu, Yinhan and Ott, Myle and Goyal, Naman and Du, Jingfei and Joshi, Mandar and Chen, Danqi and Levy, Omer and Lewis, Mike and Zettlemoyer, Luke and Stoyanov, Veselin : “ RoBERTa: A robustly optimized BERT pretraining approach ”; *arXiv preprint arXiv:1907.11692*,2019
- [Manes et al. 2020] Manes, Saraj Singh: “Studying the Change History of Code Snippets on Stack Overflow and GitHub”; *Carleton University*,2020
- [Pašek et al. 2022] Pašek, Jan and Sido, Jakub and Konopík, Miloslav and Pražák, Ondřej: “ MQDD: Pre-training of Multimodal Question Duplicity Detection for Software Engineering Domain”; *arXiv preprint arXiv:2203.14093*, 2022
- [Pei et al. 2021] Pei, Jiayan and Wu, Yimin and Qin, Zishan and Cong, Yao and Guan, Jingtao : “ Attention-based model for predicting question relatedness on Stack Overflow ”; *IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*,2021,97-107
- [Ragkhitwetsagul et al. 2022] Ragkhitwetsagul, Chaiyong and Paixao, Matheus: “ Recommending Code Improvements Based on Stack Overflow Answer Edits”; *arXiv preprint arXiv:2204.06773*, 2022
- [Reinhardt et al. 2018] Reinhardt, Anastasia and Zhang, Tianyi and Mathur, Mihir and Kim, Miryung: “ Augmenting stack overflow with API usage patterns mined from GitHub”; *Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*,2018,880-883
- [Sharma et al. 2019] Sharma, Shipra and Sodhi, Balwinder: “ Using Stack Overflow content to

assist in code review ”; *Software: Practice and Experience*, 49, 8(2019), 1255-1277

[Silva et al. 2019] Silva, Rodrigo FG and Roy, Chanchal K and Rahman, Mohammad Masudur and Schneider, Kevin A and Paixao, Klerisson and de Almeida Maia, Marcelo: “ Recommending comprehensive solutions for programming tasks by mining crowd knowledge ”; *IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*,2019,358-368

[Tabassum et al. 2020] Tabassum, Jeniya and Maddela, Mounica and Xu, Wei and Ritter, Alan : “ Code and Named Entity Recognition in StackOverflow ”; *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,2020,4913-4926

[Wang et al. 2020] Wang, Liting and Zhang, Li and Jiang, Jing: “ Duplicate question detection with deep learning in stack overflow ”; *IEEE Access* 8(2020), 25964-25975

[Wu et al. 2021] Wu, Yongliang and Zhao, Shuliang: “ Community answer generation based on knowledge graph ”; *Information Sciences* 545(2021), 132-152

[Xu et al. 2016] Xu, Bowen and Ye, Deheng and Xing, Zhenchang and Xia, Xin and Chen, Guibin and Li, Shanping : “ Predicting semantically linkable knowledge in developer online forums via convolutional neural network ”; *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, 51-62

[Yang et al. 2017] Yang, Di and Martins, Pedro and Saini, Vaibhav and Lopes, Cristina: “ Stack overflow in github: any snippets there? ”; *IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*,2017,280-290

[Yang et al. 2019] Yang, Runqi and Zhang, Jianhai and Gao, Xing and Ji, Feng and Chen, Haiqing : “ Simple and effective text matching with richer alignment features ”; *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*,2019,4699-4709

[Yazdaninia et al. 2021] Yazdaninia, Mohamad and Lo, David and Sami, Ashkan : “ Characterization and prediction of questions without accepted answers on stack overflow ”; *IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*,2021,59-70

[Ye et al. 2017] Ye, Borui and Feng, Guangyu and Cui, Anqi and Li, Ming : “ Learning question similarity with recurrent neural networks ”; *ieec international conference on big knowledge (icbk)*, 2017 , 111-118