Repo4QA: Answering Coding Questions via Dense Retrieval on GitHub Repositories

Minyu Chen[1], **Guoqiang Li**[1]*, **Chen Ma**[2], **Jingyang Li**[1], **Hongfei Fu**[1]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]City University of Hong Kong, Hong Kong S.A.R.
{minkow,li.g,lijjjjj}@sjtu.edu.cn
chenma@cityu.edu.hk, fuhf@cs.sjtu.edu.cn

Abstract

Open-source platforms such as GitHub and Stack Overflow both play significant roles in current software ecosystems. It is crucial but time-consuming for developers to raise programming questions in coding forums such as Stack Overflow and be navigated to actual solutions on GitHub repositories. In this paper, we dedicate to accelerating this activity. We find that traditional information retrieval-based methods fail to handle the long and complex questions in coding forums, and thus cannot find suitable coding repositories. To effectively and efficiently bridge the semantic gap between repositories and real-world coding questions, we introduce a specialized dataset named Repo4QA, which includes over 12,000 question-repository pairs constructed from Stack Overflow and GitHub. Furthermore, we propose QuRep, a CodeBERT-based model that jointly learns the representation of both questions and repositories. Experimental results demonstrate that our model simultaneously captures the semantic features in both questions and repositories through supervised contrastive loss and hard negative sampling. We report that our approach outperforms existing state-of-art methods by 3%-8% on MRR and 5%-8% on P@1. [1]

1 Introduction

With the increasing popularity of software development, Stack Overflow and GitHub (two large-scale communities widely recognized in open source ecosystems) have attracted growing research interests. As the idiom goes, "Don't reinvent the wheel". Tackling programming problems with existing codes and documents is more effective and economical, since various resources in repositories can provide more helpful information than text-formed answers. Specifically, developers can get help and advice to solve the technical challenges they face, and be provided a variety of solutions and tools in repositories on GitHub. As a vast number of challenges have already been considered and settled by the community, sophisticated schemes posted on GitHub can help satisfy requirements or solve problems discussed in Stack Overflow. Naturally, many answers in Stack Overflow provide links to GitHub repositories, and a large amount of these answers are acknowledged to be high-quality and helpful by the community. This phenomenon is worth investigating to determine its contribution to the efficiency improvement and the code reuse of the open source ecosystem.

Motivated by the interplay between Stack Overflow and GitHub, we introduce a novel question-repository matching task. Given a natural-language-formed question in the programming domain, we aim to search for the most relevant GitHub repository from all candidate repositories as the answer. Figure 1 illustrates the interaction between a question and a repository, where the key information for problem-solving is framed.

To this end, we introduce Repo4QA, a dataset consisting of 12,995 question-repository pairs[2] for complex coding question solving. The questions are collected from Stack Overflow, and the repositories are crawled from GitHub through the hyperlinks provided in corresponding Stack Overflow answers. Each repository is instrumental for trouble-shooting confirmed by forum users with upvotes. Such that proper repositories can be used as answers to coding questions.

Different from previous information retrieval (IR) tasks, the challenge of the proposed task lies in the long-form questions. Typical IR methods and tricks, such as query expansion, are designed for short and keyword-based user inputs, while our task focuses on questions written in natural lan-

*Corresponding author.

[1]Our dataset and code are available at https://github.com/Minkow/Repo4QA

[2]We also call them query-document pairs from the perspective of IR.

Proceedings of the 29th International Conference on Computational Linguistics, pages 1580–1592
October 12–17, 2022.

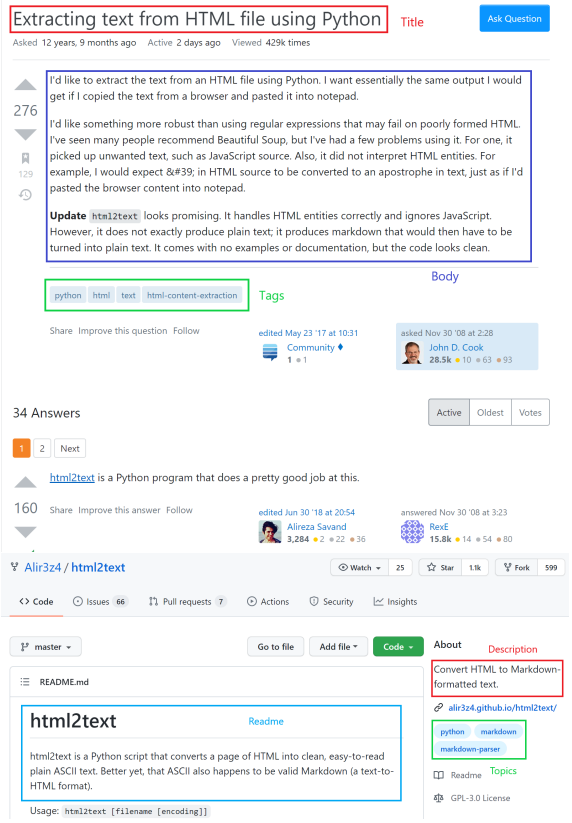

Figure 1: An example of interaction between question at Stack Overflow and repository at GitHub.

guages containing semantic information, which is harder to resolve.

Different from code searching task (Husain et al., 2019; Cambronero et al., 2019), our task is supposed to find a reasonable semantic alignment between two long-form sentences. Questions and repositories have more complex structures and richer information than short-form web queries and code snippets. Compared with community-based QA task (Qiu and Huang, 2015a; Zhao et al., 2017), our task is a cross-platform task resulting in semantic gaps between questions and answers, which is more challenging for traditional IR-based methods such as BM-25 (Robertson et al., 1995). Besides, unlike common questions, questions about programming are more challenging and specialized. The terminology of programming is widely used to form questions; there are even some questions described with codes. Moreover, complex models considering more interactions between QA pairs are more computationally expensive in our task. Finally, the interactions inside queries or documents corpus should not be neglected. The relationship of the elements in the same semantic category is informative while discriminating positive samples from negatives.

To address the aforementioned issues, we propose a novel jointly learning scheme, **QuRep**, for **Qu**estion and **Re**pository **p**airs. QuRep employs CodeBERT (Feng et al., 2020) as the initial weight of its transformer-based encoder. Our work leverages the supervised contrastive loss for better separating elements from each other in the representation space. An unsupervised hard negative sampling strategy enables better discrimination performance. The shared weights between encoders reduce the computation cost, while the relevance between questions and answers can be ranked by the similarity score of embedding vectors. With experimental evaluation and comprehensive analyses, we show that our method strongly outperforms baselines.

To summarize, the main contributions of this paper are concluded into three points respectively.

- *Dataset*: A novel cross-platform Question-Answering task is presented, aiming at answering real-world programming questions with existing GitHub repositories. We especially collect a dataset, Repo4QA, for this task.
- *Methodology*: A practical joint embedding model, QuRep, is proposed to optimize the classification and contrastive loss, which can represent both questions in language and repositories.
- *Experiment*: Experimental results demonstrate the effectiveness and efficiency of the proposed QuRep model compared with baselines.

2 Related Work

Datasets. Existing datasets in the programming domain focus on text-code interaction. CodeSearchNet (Husain et al., 2019), Deep Code Search (Gu et al., 2018) and CoDesc (Hasan et al., 2021) collect large-scale corpus of code snippet with corresponding descriptions. CoSQA (Huang et al., 2021) collects pairs of web query and function code for code question answering. Stack Overflow resources are mined (Yin et al., 2018) as long-form natural language queries to retrieval code snippets (Nie et al., 2016; Yao et al., 2018; Heyman and Van Cutsem, 2020). CodeXGLUE (Lu et al., 2021) includes text-to-code generation task and code memorization task. The only text-to-text task

is documentation translation in CodeXGLUE. Besides, LinkSO (Liu et al., 2018) discovered the similarity between repositories.

Neural Matching Networks. Ranking methods are widely used in text matching and semantic search tasks, such as community-based question answering (Qiu and Huang, 2015b; Zhang et al., 2021b), retrieval-based question answering (Qu et al., 2020; Cohen et al., 2018), and visual question answering (Lee et al., 2020). Specifically, in the programming domain, traditional IR-based ranking models regard code as text and match keywords in queries (Bajracharya et al., 2006). Recently, deep learning based methods represent coding questions and answers with vectors and leverage similarities to rank answers (Gu et al., 2018; Cambronero et al., 2019; Wan et al., 2019).

Considering the computational cost for matching, representation-based learning approaches (Huang et al., 2013) encode a query paired with a document separately into a vector and judge the relevancy by the similarity of vectors. Towards a better representation of repositories, paper2repo (Shao et al., 2020) maps the embeddings of academic papers and repositories into the same space for ranking and recommendation. Very recently, pre-trained models, including CodeBERT (Feng et al., 2020) that trained from data in the programming domain, have been applied to improve representation learning.

LM Based Retrieval and Ranking. Pretrained language models (Kenton and Toutanova, 2019; Liu et al., 2019b) dramatically advance the state of the art on various NLP tasks. However, limitations on text length and the trade-off of effectiveness and efficiency are important issues, as the cross-attention operations are too expensive in pair-wise cross-encoders. Recent work (Karpukhin et al., 2020; Xiong et al., 2020; Khattab and Zaharia, 2020; Nie et al., 2020; Gao et al., 2021a) try to reduce the computational interaction between query and document and move it to the online rerank procedure. By storing the representation of documents offline, these methods facilitate cheap runtime costs while achieving promising results on retrieval tasks.

Moreover, contrastive learning on pre-training models has recently been broadly applied to several sentence-level tasks. SBERT (Reimers and Gurevych, 2019) uses siamese and triplet network to derive embedding of two sentences, then fine-tune the model to yield useful sentence embeddings. Some work aims to improve BERT sentence embeddings in an unsupervised way (Gao et al., 2021b; Kim et al., 2021) by data augmentation.

Dataset	Type	Samples	Avg. length
Large	Question	12,995	7.97 + 104.50
	Repository	9,954	9.79 + 572.58
Small	Question	3,766	8.02 + 105.73
	Repository	2,862	9.77 + 688.03

Table 1: The statistics of Repo4QA dataset. Avg. length means title length + body length for question, and description length + Readme length for repository. Readme file has the maximum character length of 8192.

3 Dataset Description

3.1 Repo4QA Dataset

Questions. We collect complex programming questions from the well-known coding forum Stack Overflow. Stack Overflow provides data dump [3] from 2014 to 2021. To avoid ambiguous answers such as "We can deploy A after B to tackle this", answers less than 200 characters that contain only a hyperlink to a GitHub repository are selected.

Responders are required not only to get through the requirements raised by questioners, but also to be familiar with repositories stored on GitHub. The goal of our research is to fill in the gap between the questioner and various open source tools.

To filter out answers without specific repository-for-solution and repository-structured intent such as bug-reporting discussion, we discard answers linking to particular resources in repositories such as issues, commits, and releases (e.g., the question is "Assets serving paths in Rails 3.1 - how to customize it?" and the answer is "GitHub.com/chriseppstein/compass/issues/337 - there is a big discussion about it"). We control the quality and correctness of answers by only mining posts with at least one upvote. After removing answers with inaccessible GitHub repository links, these answers and corresponding questions compose 12,995 QA pairs with 12,713 unique questions. Codes are marked with *[code]* token to help our model learn the combination of natural language and code in questions.

[3] https://archive.org/details/stackexchange

Repositories. We crawl repositories through the GitHub API with given hyperlinks in answers. For our task, we collect basic information such as the name, description, topics (also called tags), and stars of a repository. The description is short textual documentation to describe a repository briefly. Topics are keywords to classify a repository. For instance, the tag "python" indicates the programming language that the repository uses, and the tag "deep-learning" shows that the repository is used in the deep learning domain. In addition, the Readme file is also obtained for detailed documentation. Most repositories introduce the main contribution and usage in the head of the Readme file. We investigate 50 repositories randomly and find that the most informative part of a Readme file is the starting 2 or 3 paragraphs, which is far less than 512 tokens. Hence, we cut the first 8192 characters of a Readme file after text cleaning to represent the repository in natural language. Likewise, 9,663 unique repositories are investigated in this step, in which 2,862 repositories have at least one topic.

Construction. Questions and repositories are aligned according to QA pairs mined in questions to constitute a QA pair sample. For each pair, the original answer is replaced by the repository mentioned. All samples in the small dataset have at least one topic, while most repositories in the large dataset lack topics. The statistics of both datasets are listed in Table 1.

Comparison. To the best of our knowledge, this is the first dataset applied to solving complex realistic programming problems with existing web resources. In this part, we conduct a comparison between Repo4QA and datasets from two aspects: 1) Code intelligence and 2) Community-based QA. Datasets in the programming domain (Yao et al., 2018; Nie et al., 2016; Yin et al., 2018; Heyman and Van Cutsem, 2020) tend to adopt only the title of the question in Stack Overflow or a short description as the query. However, for some complex questions raised in Stack Overflow, the title is too short to fully reveal the logic or semantics of those complex questions. On the other hand, the details of those questions are often clearly illustrated in the question body, and the neglect of the question body will lead to an incomplete understanding of the question. We also find that in Community-based QA datasets (Lyu et al., 2019; Zhang et al., 2021b), their QA pairs are on the same webpage, while our Repo4QA aims to bridge the semantic gap between natural languages and GitHub repositories, which is challenging to represent due to its length (averagely over 500 words) and heterogeneous text structures. In comparison, the average length of question/answer on the CoSQA dataset (Huang et al., 2021) is 6.60/71.51, and 7.86/81.6 on the CQA-SO (Lyu et al., 2019) dataset. A detailed comparison can be found in appendix A.

4 Methodology

4.1 Task Description

Before diving into the details of our model, we first describe some notations used in the answer selection problem. Each question consists of a title, content, and tags, while each repository served as an answer candidate has descriptions and documentation. Note that tags may not be contained in some repositories. Given a question in a natural language question set $q \in Q$, and a set of repositories $R = \{r_1, r_2, \cdots, r_n\}$ from GitHub, our main task is to find the most helpful repository $r \in R$ to solve the question q.

Due to the limitation of computation resources in real-world applications, joint embedding is an effective and efficient way to find repositories related to the raised question. Ideally, we train a model that jointly learns the embedding of Q and R with a transformer-based encoder network. Specifically, given any question and its dense representation q_i and repositories' embedding r_i^+, r_i^-, where r_i^+ is one of the answer to q_i, and r_i^- is not related to q_i. We expect the model to distinguish r_i^+ from others via similarity metrics, that is, to make $s(q_i, r_i^+)$ and $s(q_i, r_i^-)$ satisfying the inequality: $s(q_i, r_i^+) > s(q, r_i^-)$, where s denotes a similarity function, e.g., cosine similarity or Euclidean distance-based similarity. Then we rank all the answers for a given question according to the similarity between them.

4.2 Model Architecture

We present our QuRep in Figure 2. Different from the natural language in common domains, the language used in the programming domain contains various out-of-vocabulary words, e.g., "flask" is a tool used for web in Python programming. To solve this problem, we leverage CodeBERT as our text encoder. CodeBERT is a bi-modal pre-trained RoBERTa-based (Liu et al., 2019b) model for natural language (NL) and programming language

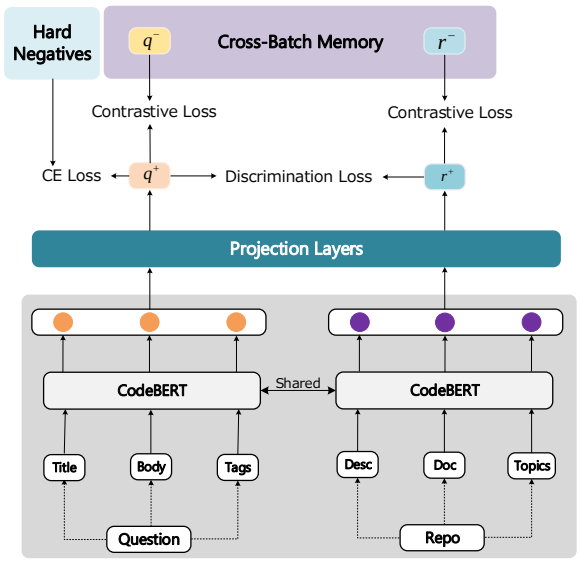

Figure 2: The QuRep applies a weight-sharing CodeBERT for encoding questions and repositories. Similarity is computed between model outputs and embeddings stored in Cross-Batch Memory for model training.

(PL) tasks. It is a bidirectional Transformer with 12 layers, 768 dimensional hidden states and 12 attention heads pre-trained on the large-scale CodeSearchNet (Husain et al., 2019) corpus. CodeBERT achieves the state-of-the-art in most NL-PL, tasks such as the natural language code search and code documentation generation. By expanding the vocabulary of RoBERTa, CodeBERT can properly represent programming terms that occur in the training corpus, especially for word-combining terminologies commonly used in programming. For example, "pyflask" is an OOV word in most models' vocabulary, but the WordPiece encoding will split "pyflask" into "py" and "flask".

For each question with title, body, and tags, we put a [CLS] token at the beginning of these three parts and separate them by [SEP] after tokenization. Then we feed the tokenized sequences into CodeBERT to acquire pooled contextualized representations of them, respectively. A [Q] is placed in the start to identify the question. In practice, we adopt the mean pooling value of contextual representations as the output of CodeBERT. Similarly, for each repository, we conduct the same manner as its description, Readme documentation, and topics. The weights are shared between the model for both question encoding and answer encoding.

As few Readme files exceed the token length limit of 512 in CodeBERT, we only take the first 509 tokens (3 tokens are left for [Q]/[A], [CLS] and [SEP]) of the Readme file as the documentation. In practice, the head content of Readme file is descriptive and summative enough, which is more informative than the usage description and changelogs in the later part of Readme file.

4.3 Loss Functions

Our approach considers three kinds of losses, namely discrimination loss, supervised contrastive loss, and classification loss. These loss functions are designed for specific tasks to better represent questions and repositories.

Discrimination Loss. Given a randomly sampled minibatch of positive pairs $\mathcal{D} = (q_i, r_i)^N$, the goal of Discrimination loss is to separate the answer r_i apart from all other $2N-2$ repositories in the same batch $\mathcal{D}$. It matches the target to satisfy the inequality : $s(q^+, r^+) > s(q^+, r^-)$. where $s(\cdot)$ represents the similarity. We choose cosine similarity as the metric. Following previous work (Karpukhin et al., 2020), the discrimination loss is defined as the negative log likelihood of positive pairs of instances:

$$\mathcal{L}_D = -\log \frac{\mathrm{e}^{s(q_i,r_i)/\tau}}{\mathrm{e}^{s(q_i,r_i)/\tau} + D}, \tag{1}$$

where τ refers to the temperature parameter, $D = \sum_{j \neq i} \mathrm{e}^{s(q_j,r_j)/\tau}$ denotes the normalization part.

Supervised Contrastive Loss. The discrimination loss discussed above focuses on modeling the distance between questions and answers. However, compared with other QA or NLI tasks (Ren et al., 2021; Zhang et al., 2021a), our questions are more complex and long-formed, which means different questions implicate diverse semantic information. We insist that a good dense representation model should not only control the distance of questions and answers, but can also represent the semantic difference between questions and repositories. Hence, we further leverage the supervised contrastive loss, to enforce the model learn to distinguish similar questions and answers, that is, $sim(q^+, r^+) > sim(q^+, q^-)$, and $sim(q^+, r^+) > sim(r^+, r^-)$. We also employ the negative log likelihood to achieve the supervised contrastive loss at the instance level:

$$\begin{aligned} \mathcal{L}_{Cq} &= -\log \frac{\mathrm{e}^{s(q_i,r_i)/\tau}}{\mathrm{e}^{s(q_i,r_i)/\tau} + Cq}, \\ \mathcal{L}_{Cr} &= -\log \frac{\mathrm{e}^{s(q_i,r_i)/\tau}}{\mathrm{e}^{s(q_i,r_i)/\tau} + Cr}, \end{aligned} \tag{2}$$

where $Cq = \sum_{j \neq i} e^{s(q_i,q_j)/\tau}$, and $Cr = \sum_{j \neq i} e^{s(r_i,r_j)/\tau}$. They refer to the similarity of questions and repositories themselves, respectively.

We can observe that the difference between L_{Cq}, L_{Cr}, and L_D is the normalization part. Instead of simply summing them up, we can uniformly conclude these different types of loss functions into one negative log likelihood as the following Discrimination-Contrastive loss:

$$\mathcal{L}_{DC} = -\log \frac{e^{s(q_i,r_i)/\tau}}{e^{s(q_i,r_i)/\tau} + D + \alpha \cdot Cq + \beta \cdot Cr}, \tag{3}$$

where the weight hyper-parameter α and β control the importance of distinguishing similar questions and repositories. The combined Discrimination-Contrastive loss consists of four interactions:

(1) $q^+ \rightarrow\leftarrow r^+$: The main element that gathers paired question and repository together.

(2) $q^+ \longleftrightarrow q^-$: The factor that separates the embedding of questions.

(3) $r^+ \longleftrightarrow r^-$: The component that makes repositories to be distant from each other.

(4) $q^+ \longleftrightarrow r^-$: An important role that cause unmatched question-repository pairs segregated.

Hard Negative for Classification Loss. Previous studies (Luan et al., 2021; Karpukhin et al., 2020; Xiong et al., 2020) provide effective methods of hard negative sampling, and further demonstrate the importance of hard negatives. Ideally, we aim to pay more attention to these hard negatives, which are not semantically similar but are mapped close to the anchor in the vector space.

We employ BM25 as the base sentence matching algorithm. For each question in the training set, we query it in the repository corpus, and then select the top 10 results as the hard negatives except the correct answer. Then we construct triplets (q^+, r^+, r^-) as training instances, where the negative r^- is randomly sampled from these hard negatives at a ratio of p to avoid the model collapse. We minimize the cross entropy loss of enhanced representations of q and r as follows:

$$\mathcal{L}_{CE} = CrossEntropy\left(f(q, r, |q - r|), y\right), \tag{4}$$

where f denotes a feed-forward classification network, and y represents the classification label.

Overall Loss. The overall loss is the weighted sum of the DC loss and the CE loss, which is:

$$\mathcal{L} = \mathcal{L}_{DC} + \gamma \mathcal{L}_{CE}, \tag{5}$$

where γ is a hyper-parameter to balance the loss.

4.4 Cross-Batch Memory Augmentation

Cross-batch memory (XBM) (Wang et al., 2020) can considerably boost the performance of contrastive learning tasks. The XBM module stores embeddings and labels for data points. It is maintained as a first-input-first-output (FIFO) queue. Enqueue and dequeue procedures are conducted when a mini-batch arrives. As mentioned above, for an anchor question q_i, we pair it with rest $M-1$ repositories $\{r_j | j \neq i\}$ in the M-size mini-batch as negative pairs. In practice, those heavyweight BERT-based model has acute GPU memory cost issues. The size of mini-batch is often limited in NLP tasks using BERT-based models. By pairing anchors with samples stored in XBM, the information provided by negative pairs is significantly enriched.

5 Experiments

5.1 Experimental Setup

Dataset. We conduct experiments on the Repo4QA-small dataset, by randomly splitting Repo4QA-small dataset into 2,966/400/400 for training/testing/validation. For repositories retrieval, we evaluate the performance from 3 different corpora: the test split, the whole Repo4QA-small repositories, and the whole Repo4QA-large repositories, which is a more realistic setting since the repositories do not occur during the training period. We always only raise all 400 questions in the testing set. However, we retrieve answers from our dataset with different scales. As for the test split, we search for the best answer in the 400 repositories. For the Repo4QA-small split, the range of ranking is the 400 repositories in the testset and the rest 3,366 repositories in the Repo4QA-small set. Note that though some repositories have occurred in the training step, we never learn the relationship between questions and these repositories. For the Repo4QA-large split, we seek suitable repositories among the union of the 400 repositories in the testset and all the repositories in the Repo4QA-large set. If given more repositories, such as whole GitHub repositories, a practical solution is to filter

Models	Test			Small			Large		
	MRR	P@1	P@5	MRR	P@1	P@5	MRR	P@1	P@5
BERT-base	7.98	4.25	9.50	3.08	1.50	3.75	1.23	0.50	1.75
CodeBERT-base	2.22	0.25	2.25	0.34	0	0.25	0.05	0	0
Glove	19.82	13.00	26.00	10.33	7.00	14.50	5.74	3.75	6.50
BM25	43.22	35.25	51.75	28.23	22.00	35.50	22.13	17.50	27.00
Universal Sentence Encoder	62.72	49.75	78.25	41.24	31.25	51.75	27.09	20.00	33.00
S-BERT	80.23	72.00	91.00	59.22	47.00	73.50	43.89	34.50	55.75
Siamese-CodeBERT	79.37	70.75	88.50	56.62	44.50	68.25	41.67	32.25	53.00
Triplet-CodeBERT	82.96	76.00	89.75	61.86	50.25	76.50	46.24	36.75	57.25
QuRep-BERT-base	77.23	70.25	87.00	59.61	48.50	73.00	44.93	32.75	60.75
QuRep (ours)	**86.26**	**81.00**	**93.00**	**69.04**	**58.50**	**82.25**	**54.26**	**41.75**	**70.00**

Table 2: Experimental results on the Repo4QA-small test set. The best figure of MRR and P@1 metric is in **bold**. Our QuRep model outperforms both unsupervised and supervised baselines.

several repositories with traditional IR approaches such as BM-25, then re-rank these repositories via our model.

Metrics. We adopt two common metrics widely used in answer ranking to measure the effectiveness of our proposed model, namely, Mean Reciprocal Rank (MRR) and Precision@K, which means the rate that correct answer appears in the top-K ranked candidates. In practice, we evaluate the performance of $K = 1$ and $K = 5$.

Baselines. As Repo4QA is a new challenge, no model is specifically designed for it. Existing methods such as ColBert (Khattab and Zaharia, 2020) focus on passage ranking tasks such as MS MARCO (Nguyen et al., 2016), with short queries and related passages. While query expansion methods (Nogueira et al., 2019b,a) are not so helpful because of the complexity of our questions. Pairwise cross-encoders are more suitable for the re-rank task after we retrieve the dense representation for the consideration of the effective-efficient trade-off. For a fair comparison, diversified methods for similar tasks, such as sentence embedding and metric learning, are introduced as baselines. We use the exact same processed data for our model as the input of these baselines. To be specific, BM25(okapi) (Robertson et al., 1995) and GloVe (Pennington et al., 2014) is widely used in IR tasks, while Universal Sentence Encoder (Cer et al., 2018), BERT (Kenton and Toutanova, 2019), CodeBERT (Feng et al., 2020), S-BERT (Reimers and Gurevych, 2019) are transformer-based models with different pre-training and fine-tuning strategies. To compare with our designed loss functions, we train the CodeBERT model with vanilla siamese loss and triplet loss. Appendix B provides implementation details of baselines.

Implementation Details. We implement our model based on the Hugging Face transformers library. The initial weight is adopt from `microsoft/codebert-base`[4]. We use RAdam (Liu et al., 2019a) as the optimizer, with a cosine annealing learning rate from 1e-5 to 1e-7. The temperature τ in the loss function is 0.01. Factors of weights are set as $\alpha = \beta = 1.5$ and $\gamma = 0.4$. The negative sampling ratio is 0.2. For cross-batch memory, we set the size of the memory bank to 64, and this mechanism starts working from epoch 6.

5.2 Model Comparisons

The performance of different approaches on the Repo4QA task is in Table 2. From these experimental results, we have the following observations:

First, our proposed model outperforms all competitive baselines on MRR, P@1, and P@5 metrics. Especially, for the retrieval task on Repo4QA-large dataset, traditional IR-based and word embedding approaches cannot differentiate target from similar repositories, while contrastive-learning methods strongly outperform others. This result demonstrates the difficulty of understanding long-form questions compared with short queries, and the necessity of precise retrieval on large corpora via semantic search instead of lexical matching.

Second, poor performance is shown by BERT

[4] https://github.com/microsoft/CodeBERT

Loss	MRR	P@1	P@5
Ours	**86.26**	**81.00**	**93.00**
Vanilla triplet loss	82.96	76.00	89.75
Hinge loss	83.72	77.25	91.75
Circle loss	82.44	75.00	91.25
InfoNCE	84.44	78.25	92.25

Table 3: Results with different comparable loss function.

Models	MRR	P@1	P@5
QuRep	**86.26**	**81.00**	**93.00**
w/o XBM	-3.28	-4.00	-2.50
w/o CodeBERT	-9.03	-10.75	-6.00
w/o Constrative Loss	-21.24	-26.75	-13.25
w/o CE Loss	-0.83	-1.50	-1.00
w/o Hard Negatives	-1.43	-2.75	-1.25

Table 4: Results of ablation study conducted on Test split about model structure.

and CodeBERT if directly employing mean pooling output to similarity computation. The results are even worse than using average GloVe embeddings. Universal Sentence Encoder shows effectiveness among unsupervised learning methods, and SBERT achieves great performance since they are trained on large QA corpora.

Third, nearly all supervised methods achieve higher performance than unsupervised models, though no early interaction, such as cross-attention between questions and repositories, is considered. This phenomenon demonstrates the sound effect of contrastive learning.

Fourth, although some repositories in the Repo4QA-small dataset have been modeled in the training period, the performance still drops a lot compared with the result in the Repo4QA-test. The root cause is the diversity of questions. As the questions are always from the testset, they keep unseen for the model in the inference step. There is also no issue of data leakage in our Repo4QA-small dataset because the training data consists of QA pairs, but the questions in testset never pair with answers in the Repo4QA-small dataset. When questions have a longer form instead of simple keywords query, the semantic information contained is richer and more diversified, which increases the difficulty of encoding.

5.3 Loss Comparisons

We agree that an empirical study for comparable loss functions contributes to the soundness of our method. As reported in Table 3, experimental results demonstrate the hybrid loss we design significantly improves the performance compared with the classical loss functions of contrastive study(Sun et al., 2020; Gao et al., 2021b).

The strongest baseline, InfoNCE, has a similar form to our Discrimination-Contrastive loss. The difference is that we distinguish the effect from questions and repositories as stated in Sect. 4.3.

5.4 Ablations Study

As the experimental results show the same trend in all three splits, in this part, we only conduct experiments in the test split. We consider the model components and loss function components as two main aspects of our ablation studies.

Effects of model components. Removing or replacing components of QuRep decreases the performance, as Table 4 shows. To be more specific, we replace CodeBERT with BERT-base, just as the result of QuRep-BERT-base shows. The performance drop proves the effectiveness of pre-training in the programming domain. Cross-batch memory is proved to be a necessary mechanism for constructing more QA pairs to be trained.

Effects of loss components. We also investigate the ablation of constituents in our proposed hybrid loss function. Each part of our loss contributes to the improvement of performance. As illustrated in Table 4, contrastive loss is the backbone of our learning target, and the hard negative sampling strategy also plays an important role.

5.5 Case Study

To better understand the difficulty of the task and the effectiveness of our solution, we present a case study in this part. As a typical method in the IR field, BM25 is selected as the baseline method to compare. All these cases can be viewed on StackOverflow. We only display key parts of questions/repositories considering the full length.

The first and second cases in Table 5 are examples that QuRep outperforms BM25. Question 1 means to find a dictionary implementation in JavaScript. BM25 returns results containing the keyword "dictionary," as the repository "stig/json-framework" (now redirected to SBJson/SBJson) mentions NSMutableDictionary type in Objective-

Question Title	BM25 Rank	QuRep Rank	BM25 Top1	QuRep Top1
Is there a dictionary implementation in JavaScript	13	1	stig/json-framework This framework implements a strict parser and generator in Objective-C	jieter/django-tables2 A complete, fully tested and documented data structure library written in pure JavaScript.
What is the simplest way to graph rrd files in Grafana	30	1	bookkojot/mp4fixer Recover damaged/unfinished mp4 files with h264 video	doublemarket/grafana-rrd-server A simple HTTP server that reads RRD files and responds to requests from Grafana.
Validate URL with standard package in GO	32	1	GitHub/fetch Promise-based mechanism for programmatically making web requests in the browser	asaskevich/govalidator Go Package of validators and sanitizers for strings, numerics, slices and structs.
Angular: How to force run unit tests when running Git push	245	6	mozilla-mobile/fenix all-new Firefox for Android browser, based on GeckoView and Mozilla Android Components.	lerna/lerna A tool for managing JavaScript projects with multiple packages

Table 5: Case study on BM25 and QuRep.

C, which is not so close to the question. In comparison, our proposed model recommends the collection of data structures written in JavaScript. Though the word "dictionary" is not discussed in the repository so frequently, "dictionary" is contained by the concept "data structures." This is what language models bring us. Case 2 and 3 also show that BM25 tend to return similar repositories at the word level, which results in repositories from different disciplines or programming languages. While QuRep can identify the purpose of questions and recommend better answers.

Case 4 is a bad case in which both models fail. The best answer posted on StackOverflow is "typicode/husk." QuRep ranks "lerna/lerna" as the best because it discusses both Angular and Git. However, the phrase "unit test" is ignored.

From the case study section, we can learn that understanding the intent of a question is a key factor for our task, especially when the question contains rich semantic information. The relations of concepts are also an important issue to study.

6 Conclusion

In this paper, we introduce an automatically collected novel dataset, Repo4QA, for the proposed task. Furthermore, we propose QuRep, a contrastive learning method to fine-tune CodeBERT for our task. Experimental results demonstrate that our method outperforms baseline models in effectiveness and efficiency. Detailed analyses are conducted to investigate the impact on performance brought by components of our model. We look forward to other applications and more research interest in our task. Moving forward, we plan to deploy our dataset and method to solve programming questions in software engineering practice and consider code stored in repositories for better modeling of multi-modal GitHub repositories.

Acknowledgements

This work is supported by the National Science Foundation of China Grant No. 62161146001.

References

Sushil Bajracharya, Trung Ngo, Erik Linstead, Yimeng Dou, Paul Rigor, Pierre Baldi, and Cristina Lopes. 2006. Sourcerer: a search engine for open source code supporting structure-based search. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 681–682.

J. Cambronero, Hongyu Li, Seohyun Kim, Koushik Sen, and S. Chandra. 2019. When deep learning met code search. *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Daniel Cohen, Liu Yang, and W. Croft. 2018. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. Codebert: A

pre-trained model for programming and natural languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1536–1547.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. 2018. Deep code search. *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 933–944.

Masum Hasan, Tanveer Muttaqueen, Abdullah Al Ishtiaq, Kazi Sajeed Mehrab, Md. Mahim Anjum Haque, Tahmid Hasan, Wasi Ahmad, Anindya Iqbal, and Rifat Shahriyar. 2021. CoDesc: A large code–description parallel dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 210–218, Online. Association for Computational Linguistics.

Geert Heyman and Tom Van Cutsem. 2020. Neural code search revisited: Enhancing code snippet retrieval through natural language intent. *arXiv preprint arXiv:2008.12193*.

J. Huang, D. Tang, L. Shou, M. Gong, and N. Duan. 2021. Cosqa: 20,000+ web queries for code search and question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.

H. Husain, Hongqi Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *ArXiv*, abs/1909.09436.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021. Self-guided contrastive learning for BERT sentence representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2528–2540, Online. Association for Computational Linguistics.

Kyungjae Lee, Nan Duan, Lei Ji, Jason Li, and Seungwon Hwang. 2020. Segment-then-rank: non-factoid question answering on instructional videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8147–8154.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019a. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. 2018. Linkso: a dataset for learning to retrieve similar question answer pairs on software development forums. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, pages 2–5.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *ArXiv*, abs/2102.04664.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Shanshan Lyu, Wentao Ouyang, Yongqing Wang, Huawei Shen, and Xueqi Cheng. 2019. What we vote for? answer selection from user expertise view

in community question answering. In *The World Wide Web Conference*, pages 1198–1209.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Liming Nie, He Jiang, Zhilei Ren, Zeyi Sun, and Xiaochen Li. 2016. Query expansion based on crowd knowledge for code search. *IEEE Transactions on Services Computing*, 9:771–783.

Ping Nie, Yuyu Zhang, Xiubo Geng, Arun Ramamurthy, Le Song, and Daxin Jiang. 2020. Dc-bert: Decoupling question and document for efficient contextual encoding. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1829–1832.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019a. From doc2query to doctttttquery. *Online preprint*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015a. Convolutional neural tensor network architecture for community-based question answering. In *Twenty-Fourth international joint conference on artificial intelligence*.

Xipeng Qiu and Xuanjing Huang. 2015b. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*.

Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 539–548.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Pair: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Huajie Shao, Dachun Sun, Jiahao Wu, Zecheng Zhang, A. Zhang, Shuochao Yao, Shengzhong Liu, Tianshi Wang, C. Zhang, and T. Abdelzaher. 2020. paper2repo: Github repository recommendation for academic papers. *Proceedings of The Web Conference 2020*.

Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. 2020. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6398–6407.

Yao Wan, Jingdong Shu, Yulei Sui, Guandong Xu, Zhou Zhao, Jian Wu, and Philip S. Yu. 2019. Multi-modal attention network learning for semantic source code retrieval. *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 13–25.

Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. 2020. Cross-batch memory for embedding learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6388–6397.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Ziyu Yao, Daniel S. Weld, Wei-Peng Chen, and Huan Sun. 2018. Staqc: A systematically mined question-code dataset from stack overflow. *Proceedings of the 2018 World Wide Web Conference*.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 476–486.

Dejiao Zhang, Shang-Wen Li, Wei Xiao, Henghui Zhu, Ramesh Nallapati, Andrew O Arnold, and Bing Xiang. 2021a. Pairwise supervised contrastive learning of sentence representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5786–5798.

Wei Zhang, Zeyuan Chen, Chao Dong, Wen Wang, Hongyuan Zha, and Jianyong Wang. 2021b. Graph-based tri-attention network for answer ranking in cqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14463–14471.

Zhou Zhao, Hanqing Lu, Vincent W Zheng, Deng Cai, Xiaofei He, and Yueting Zhuang. 2017. Community-based question answering via asymmetric multi-faceted ranking network learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

A Dataset comparison

As shown in Table 6, we compare our dataset with several datasets on Code Intelligence and Community-based QA. The scale of our dataset is similar to SO-DS (Heyman and Van Cutsem, 2020), CQA-Quora (Lyu et al., 2019) and CQA-SO (Zhang et al., 2021b). Our QA pairs are more complex then short-form query/code pairs, which resulting in less available resources on the web. Moreover, our dataset is the only cross-platform one.

B Baseline Implementation Details

We select GloVe average embedding, BERT *[CLS]* token output and CodeBERT *[CLS]* token output as baselines without supervision. For the supervised learning method, Siamese-formed CodeBERT and Triplet-formed CodeBERT are compared.

- BM25(okapi) (Robertson et al., 1995) BM25 is a well-known lexical retrieval model. We employ the implementation from the *rank_bm25* library.[5]
- GloVe (Pennington et al., 2014) The mean embedding of the whole sentence is regarded as the representation of the sentence. Sentence representation similarity is computed for ranking.
- Universal Sentence Encoder (Cer et al., 2018) It is a transformer-based network which augments unsupervised learning with training on SNLI dataset. [6]
- BERT (Kenton and Toutanova, 2019) We use the *[CLS]* token output for sentence embedding.
- CodeBERT Similar to the strategy applied on BERT, the *[CLS]* token output is adopted. Besides, we train a Siamese-formed CodeBERT and a Triplet-formed CodeBERT for comparison in supervised learning. The implementation of CosQA (Huang et al., 2021) is based on the Siamese-formed CodeBERT.
- S-BERT (Reimers and Gurevych, 2019) is a Siamese BERT-Networks. We employ the *all-roberta-large-v1* model hosted on huggingface, which is pretrained over 1B+ QA pairs for better sentence embedding. [7]

C Hyper-parameter Configuration

To figure out the different role that our proposed loss functions played in the optimization phrase, we conduct a discussion on weight hyper-parameters, α, β, and γ. α and β controls the importance of supervised contrastive loss over discrimination loss. Since Cq, Cr and D occurs together at the normalization part of equation 4.3, the larger their weight are, the more importance the have. We set $\alpha = \beta$ during our experiment, and report result as Table 7. We can find that the model performs better while $\alpha > 1$, which means the model focus more on separating elements from the same semantic catogory apart. The output demonstrates the importance and effectiveness of surpervised contrastive learning. Besides, the weight γ and the hard negative sampling rate p are set to 0.2 to achieve best performance.

α	MRR	P@1	P@5
2	86.01	80.50	**93.50**
1.5	**86.26**	**81.00**	93.00
1	85.30	79.00	92.75
0.5	85.12	78.00	92.75
0	84.48	77.75	92.50

Table 7: Results of different hyperparameter $\alpha = \beta$ settings.

[5]https://GitHub.com/dorianbrown/rank_bm25
[6]https://tfhub.dev/google/universal-sentence-encoder/4
[7]https://huggingface.co/sentence-transformers/all-roberta-large-v1

Dataset	Domain	Size	Query Type	Answer Type	Annotation
CSN (Husain et al., 2019)	Coding	2.3M	Short description	Function code	No
Deep Code Search (Gu et al., 2018)	Coding	18.2M	Short description	Function code	No
CoSQA (Huang et al., 2021)	Coding	20.6K	Web Query	Function code	Yes
QECK (Nie et al., 2016)	Coding	312.9K	SO question title	Code block	No
StaQC (Yao et al., 2018)	Coding	268K	SO question title	Code block	Partly
CoNaLa (Yin et al., 2018)	Coding	598.2K	SO question title	Code block	Partly
SO-DS (Heyman and Van Cutsem, 2020)	Coding	12.1k	SO question title	Code block	No
CQA-Quora (Lyu et al., 2019)	Open	76.2k	Quora question	Quora Answer	No
CQA-SO (Zhang et al., 2021b)	Coding	13.9k	SO question	SO Answer	No
Repo4QA (ours)	Coding	13.0k	SO question	GitHub repository	No

Table 6: Overview of existing datasets on Code Intelligence and Community-based QA