Repo4QA: Answering Complex Coding Questions via Dense Retrieval on GitHub Repositories

Anonymous ACL submission

Abstract

Open-source platforms such as Github and Stack Overflow both play important roles in our software ecosystem. It is crucial but time-consuming for programmers to raise their specific programming questions on coding forums such as Stack Overflow, which guides them to actual solutions on Github repositories. We show our interest in accelerating such a process and find that traditional Information Retrieval based methods fail to handle the long and complex questions in coding forums and thus cannot find the suitable coding repositories. In order to bridge the semantic gap between repositories and real-world coding questions effectively and efficiently, we introduce a specialized dataset named Repo4QA, which includes over 12,000 question-repository pairs constructed from Stack Overflow and Github. Furthermore, we propose QuReCL, a contrastive learning model based on CodeBERT, to jointly learn the representation of both questions and repositories. Experimental results demonstrate that our model can simultaneously capture the semantic features in both questions and repositories through jointly embedding, and outperforms existing state-of-art methods.

1 Introduction

With the increasing popularity of software developers, Stack Overflow and Github, the two large-scale communities widely used in open source ecosystems have attracted growing research interests. As the idiom goes, "Don't reinvent the wheel", tacking programming problems with existed codes and documents is a more effective and economical way, while various resources in repositories can provide more useful 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 for their software development. Note that a vast number of challenges have already been considered and settled by the community, sophisticated schemes posted on Github can help to satisfy requirements or to 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 researching to determine its contribution to the efficiency improvement and code reuse of the whole open source ecosystem.

Motivated by the trend of interplay between Stack Overflow and Github, we introduce a novel question-repository matching task. Given a natural-language-formed question in the programming domain, the task is defined as searching for the most relevant and helpful Github repository in repositories corpus as answer. Figure 1 illustrates the interaction between a question and a repository. Key information for problem solving is framed in the figure.

To this end, we introduce Repo4QA, a dataset consisting of 12,995 question-repository pairs 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 answers. Each repository is instrumental for trouble-shooting confirmed by forum users with upvotes.

The proposed task has its own characteristics. Different from code searching task (Husain et al., 2019; Cambronero et al., 2019), our task needs to find a reasonable semantic alignment between two long-form sentences. Questions and repositories have more complex structure 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 gap between questions and answers, which is more challenging for traditional IR-based meth-

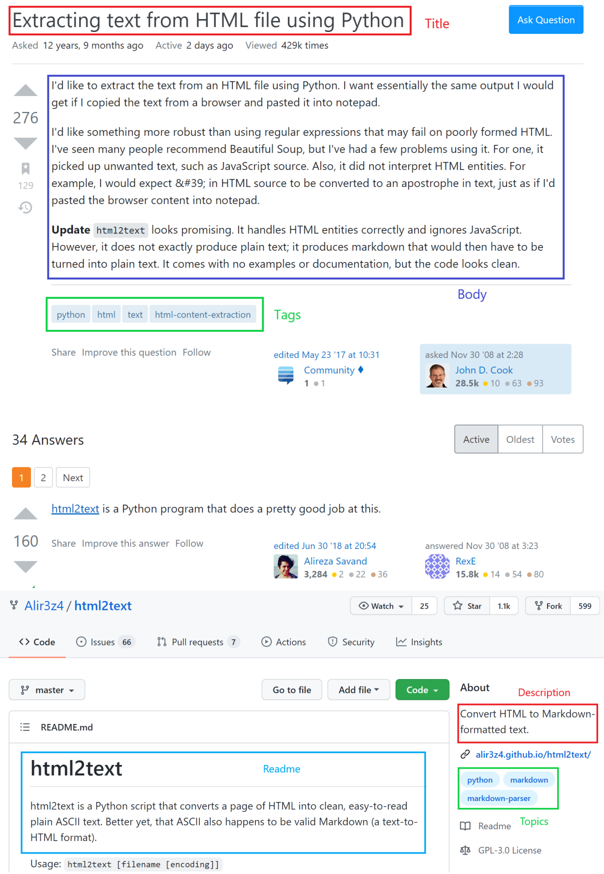

Figure 1: An example of interaction between question at Stack Overflow and repository at Github.

ods such as BM-25(Robertson et al., 1995). Besides, unlike common questions, questions about programming are more difficult and specialized. 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 computational expensive in our task.

To address the aforementioned issues, we propose a contrastive learning based model, QuReCL, to jointly learn the representation of **Qu**estions and **Re**positories. QuReCL computes a single vector for each question and repository, and similarity score of the vectors is calculated to measure the relatedness for ranking. With experimental evaluation and comprehensive analyses, we show that our method strongly outperforms baselines. We also demonstrate that QuReCL is more computationally efficient than baselines in inference step.

In conclusion, 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 also collect a dataset, Repo4QA for this task.
- *Methodology*: A practical contrastive learning model, QuReCL, is proposed to jointly learn embedding of both questions and repositories, which can be applied to related products with great flexibility.
- *Experiment*: Experimental results are given to demonstrate the effectiveness and efficiency of the proposed QuReCL 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; **?**). 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.

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., 2021), open domain 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 query and document each into a vector and judge the relevancy by the similarity of vectors. Towards a better representation of repository, 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 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 re-rank procedure. By storing representation of document offline, these methods facilitate cheap runtime cost while achieving promising results on retrieval tasks.

Moreover, contrastive learning on pre-training models is broadly applicable to several sentence-level tasks recently. SBERT (Reimers and Gurevych, 2019) use siamese and triplet network to derive embedding of two sentences, and 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.

3 Preliminaries

3.1 Repo4QA Dataset

Questions We collect complex programming questions from the well-known coding forum Stack Overflow. Stack Overflow provides data dump [1] from 2014 to 2021. Answers within 200 characters which contain a hyperlink to a Github repository are selected. Questions with such kind of answers are often complicated.Responders are required not only to get through the requirements raised by questioners, but also to be familiar to repositories stored in Github. The goal of our research is to fill in the gap between the questioner and various of open source tools.

To filter out topics without specific repository-for-solution intent such as bug-reporting discussion, we discard answers discussing particular resources in repository including issues, commits and releases. We control the quality and correctness of answers by only mining posts with one or more upvotes. After removing answers with inaccessible Github repository links, these answers and corresponding questions compose 12,995 QA pairs consisting of 12,713 unique questions. Codes are marked with *[code]* token to help our model learn the combination of natural language and code in questions.

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

Repositories We crawl repositories through the Github API with given hyperlinks in answers. For our task, we mine basic information such as a name, a description, topics (also called tags) and stars of a repository. A description is a short textual documentation to describe a repository briefly. Topics are keywords to classify a repository. For instance, "python" tag indicates the programming language that the repository uses, and "deep-learning" tag shows that the repository is used in the deep learning domain. The Readme file is obtained to provide documentation in detail. Most repositories introduce the main contribution and usage in the head of Readme file. We investigate 30 repositories randomly and find that the most informative part of a Readme file is about 2-3 paragraphs at the start, which is far less then 512 tokens. Hence, we cut the first 8192 characters of a Readme file after text cleaning to represent the repository in natural language. 9,663 unique repositories are mined 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. We then select a small dataset from the entire QA pairs with high quality. All samples in small dataset have at least one topic. The statistics of both datasets is listed in Table 2.

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. (2) Community-based QA. As presented in Table 1, datasets in the programming domain tend to use only the title of the question in Stack Overflow or a short textual query including web query and short description as query. However, we find out that in complex questions

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., 2021)	Coding	13.9k	SO question	SO Answer	No
Repo4QA (ours)	Coding	13.0k	SO question	Github repository	No

Table 1: Overview of existing datasets on Code Intelligence and Community-based QA

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 2: 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.

raised in Stack Overflow, problems are almost described in detail in the body of questions. The importance of the question body should not be neglected. We find out that in Community-based QA datasets, their QA pairs are on the same page, while our Repo4QA aims to bridge the semantic gap between natural languages and Github repositories, which is a challenge to represent for its length (averagely over 500 words) and heterogeneous text structure. In comparison, the average length of question/answer in CoSQA dataset (Huang et al., 2021) is 6.60/71.51, and the average length of answer in CQA-SO (Zhang et al., 2021) dataset is 85.6.

4 Methodology

4.1 Task Description

Before diving into detail of our model, we first describe symbols used in the answer selection problem. 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. Each question has the title, the content and the tags, while each repository has descriptions and the documentation. Tags are not contained in some repositories. Our main task is to find the most possibly helpful repository $r \in R$ to solve the question q.

Due to the limitation of calculating resources in real-life application, joint embedding is an effective and efficient way to find repositories related to question raised. Ideally, we would train a model that jointly learns the embedding of Q and R with a triplet network. To be specifically, given any question q_i and repository 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 aim to learn a representation function $e_x = f(x)$ to make $s(e_{q_i}, e_{r_i^+})$ and $s(e_{q_i}, e_{r_i^-})$ satisfying the inequality: $s(e_{q_i}, e_{r_i^+}) > s(e_q, e_{r_i^-})$, where s denotes similarity, e.g cosine similarity or Euclidean distance-based similarity. Then we rank all the answers for a given question according to similarity.

4.2 Model Architecture

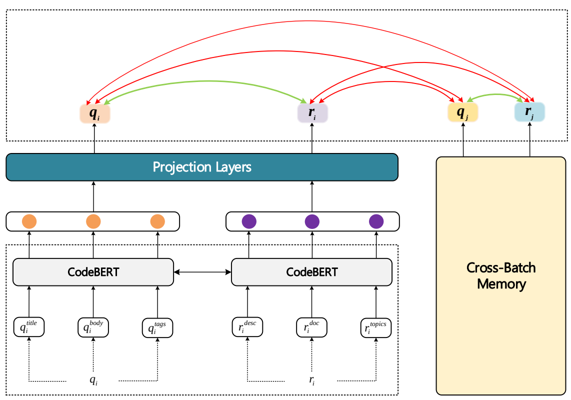

Figure 2: The QuReCL applies a weight-sharing CodeBERT for encoding questions and repositories. Similarity is computed between model outputs and embeddings stored in Cross-Batch Memory for model training.

We present our QuReCL as Figure 2 illustrated in this part. Different from natural language in common domain, language used in programming domain contains various of out-of-vocabulary words, e.g "flask" is a tool used for web in python programming. In order 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 (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 natural language code search and code documentation generation. By expanding its vocabulary from RoBERTa, CodeBERT can represent programming terms that occur in the training corpus properly, especially for word-combining terminologies common used in programming domain. For example, "pyflask" is an OOV word in most model's vocabulary, but the WordPiece encoding will cut "pyflask" to "py" and "flask".

In detail, for each question $q_i = (q_i^{title}, q_i^{body}, q_i^{tag})$, where $q_i^{title}, q_i^{body}, q_i^{tag}$ denotes the title, the body and the tags of q_i, we put a [CLS] token in the front of the 3 sentences 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 query. In practice, we adopt the mean pooling value of contextual representation as the output of CodeBERT:

$$\mathbf{q}_i = \text{C-BERT}([Q]q_i^{tag}[S]q_i^{title}[S]q_i^{body}[S]) \quad (1)$$

Where [S] denotes [SEP] Similarly, for each repository, $r_j = (r_j^{desc}, r_j^{doc}, r_j^{topic})$, where $r_j^{desc}, r_j^{doc}, r_j^{topic}$ denotes the description, readme documentation and topics of r_j, we have formalation as follows:

$$\mathbf{r}_i = \text{C-BERT}([A]r_i^{topic}[S]r_i^{desc}[S]r_i^{doc}[S]) \quad (2)$$

Note that few readme files exceed the token length limit of 512 in CodeBERT, we only take the first 510 tokens (2 tokens are left for [Q]/[A] and [SEP]) of the Readme file as documentation. In practice, the head content of readme file is descriptive and summative enough for our task, which is more informative than usage and example part.

4.3 Contrastive Learning for Joint Embedding

We obtain an encoding model of questions and repositories by a CodeBERT encoder and projection layer. To make this model learn the joint embedding of both questions and repositories, we incorporate contrastive learning methods to CodeBERT. Contrastive learning aims to learn representations by contrasting positive and negative examples as the name implies. It matches the target to satisfy the inequality : $s(e_{q_i}, e_{r_i^+}) > s(e_q, e_{r_i^-})$.

We select N QA pairs into a batch, resulting $2N$ data points. It is clear that the QA pair is a positive pair. Instead of choosing negative pairs explicitly, every non-matched Q-A sample pair is regarded as negative pairs.

Loss Function Denote $X = \{\mathbf{x}|\mathbf{x} \in \{\mathbf{q_i}\} \cup \{\mathbf{r_i}\}\}$ is the set of computed embedding vectors of questions and repositories in batch.

Typical metric-learning based loss function focuses on modeling the distance between questions and answers. However, compared with other QA tasks, 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 queries and documents, but can also represent the semantic difference between questions and repositories.

To achieve this, we leverage the NT-Xent loss applied in SimCLR (Chen et al., 2020) is our training target at first. The loss function for a positive pair (i, j) is defined as follow :

$$L_{i,j}^{base} = -log\frac{e^{(sim(\mathbf{x_i},\mathbf{x_j})/\tau)}}{\sum_{k=1,k\neq i}^{2N} e^{(sim(\mathbf{x_i},\mathbf{x_k})/\tau)}} \quad (3)$$

where τ is a temperature hyperparameter. Cosine similarity is implemented as a similarity function. The loss is calculated in all positive pairs bidirectionally including (i, j) and (j, i). The overall base is the average loss of all positive pairs [2]:

$$L^{base} = \frac{1}{2N}\sum_{k=1}^{N}(L_{2k-1,2k}^{base} + L_{2k,2k-1}^{base}) \quad (4)$$

[2]Pairs are placed orderly in mini-batch

Loss Function Revisited Inspired by the discussion of NT-Xent loss optimization in the recent works (Chen and He, 2021; Kim et al., 2021) , we revisit our task and NT-Xent loss. The NT-Xent loss consists of four interactions as follows:

(1) $\mathbf{q_i} \rightarrow\leftarrow \mathbf{r_i}$: The main element that gathers paired question and repository together in the vector space.

(2) $\mathbf{q_i} \leftarrow\rightarrow \mathbf{q_j}$: The factor that separates embedding of questions.

(3) $\mathbf{r_i} \leftarrow\rightarrow \mathbf{r_j}$: The component that make repositories to be distant from each other.

(4) $\mathbf{q_i} \leftarrow\rightarrow \mathbf{r_j}$: An important role that cause unmatched question-repository pairs segregated.

Unlike unsupervised methods (Gao et al., 2021b; Kim et al., 2021) neglecting the impact of similarity computing between data points and their augmentations. The semantic gap exists not only between different questions but also different repositories in our task. It is necessary to consider how important the factor (2) and (3) are in the learning procedure. To this end, we reformulate NT-Xent to Weighted-NT-Xent :

$$L^w = \frac{1}{2N}\sum_{k=1}^{N}(L^w_{2k-1,2k} + L^{base}_{2k,2k-1}) \tag{5}$$

$$L^w_{i,j} = -log\frac{e^{(sim(\mathbf{x_i},\mathbf{x_j})/\tau)}}{\sum_{k=1,k\neq i}^{2N} w(i,k)e^{(sim(\mathbf{x_i},\mathbf{x_k})/\tau)}} \tag{6}$$

where

$$w(i,k) = \begin{cases} \alpha & if\{\mathbf{x_i},\mathbf{x_k}\} \subseteq \cup\mathbf{q_j} \\ \beta & if\{\mathbf{x_i},\mathbf{x_k}\} \subseteq \cup\mathbf{r_j} \\ 1 & otherwise \end{cases} \tag{7}$$

The Weighted-NT-Xent gives the weight for the self-model similarity result. We can control the contribution of case (2) and case (3) as mentioned above, by changing the value of α and β. We expect that this refinement can reveal the impact of self-modal contrastive learning during the cross-modal contrastive learning task. We will report and discuss the fact that the setting of α and β greater than 1 results better performance in the experiments section.

Cross-Batch Memory Augmentation and Negatives Sampling 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 happen when a mini-batch arrives. As mentioned above, for an anchor question $\mathbf{q_i}$, we pair it with rest $N-1$ repositories $\{r_j|j \neq i\}$ in the N-size mini-batch as negative pairs. In practice, heavy-weight BERT-based model has acute GPU memory cost issue. The size of mini-batch is often limited in NLP tasks using BERT-based model[3]. By pairing anchors with samples stored in XBM, information provided by negative pairs is significantly enriched. Moreover, we select hard negatives with tags/topics labeling, from the intuition that tags/topics overlap leads to similar discussion.

5 Experiments

We carry out our experiments to evaluate the performance of methods on our Repo4QA dataset. Further, we discuss whether our proposed QuReCL model outperforms state-of-the-art methods in similar tasks. In addition, an ablation study is conducted to explore how components of our proposed model impact performance.

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 corpus: the test split, the whole Repo4QA-small repositories and the whole Repo4QA-large repositories, which is a more realistic setting since the documents do not occur in training period. If given more repositories, for example, the whole Github repositories, a piratical solution is to filter several repositories with traditional IR approaches such as BM-25, then rerank these repositories via our model.

Metrics We adopt two common metrics to measure the effectiveness of our proposed model, namely, Mean Reciprocal Rank (MRR) and Precision@K. In practice, we evaluate the performance of $K = 1, 5$. The two metrics are widely used in information retrieval and answer ranking.

[3]Maximum mini-batch size is 8 in this work

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 query and passages related. While query expansion methods (Nogueira et al., 2019b,a) are not so helpful because of the complexity of our queries. Pair-wise cross-encoders are more suitable for the rerank task after we retrieval the dense representation for the consideration of effective-efficient trade-off. For a fair comparison, diversified methods for similar tasks such as sentence embedding and metric learning models are introduced as baselines. We use the exact same processed data for our model as the input of these baselines.

- BM25(okapi) (Robertson et al., 1995) BM25 is a well-known lexical retrieval model. We employ the implementation from the *rank_bm25* library.[4]

- 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. [5]

- 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.

- 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. [6]

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

5.2 Model Comparisons

The performance of different approaches on Repo4QA task is reported in Table 3. From these experimental results, we can obtain the following summaries :

- Our proposed model outperforms all competitive baselines on MRR, P@1 and P@5 metrics. Especially, for retrieval task on Repo4QA-large dataset, traditional IR-based and word embedding approaches cannot differ 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 corpus via semantic search instead of lexical matching.

- Poor performance is shown by BERT 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 corpus.

- 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.

5.3 Ablation Study

Effects of model components To discover the impact of different aspects of QuReCL, an empirical ablation study is performed in this part. Removing or replacing components of QuReCL decreases the performance as Table 4 shows. We replace the NT-Xent loss with Circle loss (Sun et al., 2020) for comparison.

Discussion of Weighted-NT-Xent As mentioned above, we design the Weighted-NT-Xent, to diversify the impact of the four types of interactions between QA pairs. The hyperparameter α and β controls the weight of similarity computation. In our practice, we set $\alpha = \beta$ for better hyperparameter serach. The original NT-Xent is the case

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
QuReCl (ours)	**86.11**	**80.50**	**93.00**	**69.20**	**59.00**	**82.25**	**53.95**	**41.00**	**68.50**

Table 3: Experimental results on the Repo4QA-small test set. The best figure of MRR and P@1 metric is in **bold**. Our QuReCL model outperforms both unsupervised and supervised baselines.

Models	**MRR**	**P@1**
QuReCL	**86.11**	**80.50**
w/o XBM	-3.07	-3.75
w/o NT-Xent	-5.61	-6.25

Table 4: Results of ablation study on model structure.

α	**MRR**	**P@1**	**P@5**
2.5	84.70	78.00	94.00
2	**86.11**	**80.50**	**95.00**
1.5	85.56	79.75	93.25
1	85.30	79.00	94.75
0.5	85.12	78.00	94.75
0	84.29	77.00	93.50

Table 5: Results of different hyperparameter α settings.

of $\alpha = 1$, which means interactions are treated equally during learning. Some previews works (Chen and He, 2021; Kim et al., 2021) discard the self-augmentation interplay, which equals the situation that $\alpha = 0$. In our task, we acknowledge the opinion that the most important issue in training is to make model gather (q_i, r_i) and separate (q_i, r_j). However, samples in (q_i, q_j) and (r_i, r_j) pairs are semantically diverse. Learning to dissociate them is beneficial at the top view of our task. So we search for the suitable hyperparameter α from 0 to 2.5 and expect this refinement can precisely lead to better performance for our task, as we can see in Table 5.

We report that $\alpha = 2$ is the best among our settings, ascending the original NT-Xent loss by 0.81%, which demonstrates the importance of learning to differ queries and documents. Ignoring the interaction $(\alpha = 0)$ between repositories themselves will reduce the performance badly.

6 Conclusion

In this paper, we introduce an automatically collected novel dataset Repo4QA for the proposed task. Furthermore, we propose QuReCL, 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 analysis are conducted to investigate the impact on performance brought by components of our model. We look forward to other applications and more research interest on our task. Moving forward, we are planning to deploy our dataset and method to solve programming questions in software engineering practice, and consider code stored in repositories for better informatively modeling bi-modal Github repositories.

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.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

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. *arXiv preprint arXiv:2104.08821*.

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 Seung-won 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*.

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.

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.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

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.

Wei Zhang, Zeyuan Chen, Chao Dong, Wen Wang, Hongyuan Zha, and Jianyong Wang. 2021. 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 Implementation Details

Our implementation is based on the HuggingFace's Transformers (Wolf et al., 2019), some baseline models are implemented with Sentence-Transformers (Reimers and Gurevych, 2019). We leverage *microsoft/codebert-base* to initialize parameters and weights in CodeBERT model. Mini-batch size is set to 8. Temperature hyperparameter τ is 0.2. Rectified Adam (Liu et al., 2019a) with betas = (0.9,0.999) is used as our optimizer. Learning rate is set to 1e-5 at first, then decrease to 2e-7 at epoch 20 with a *CosineAnnealingLR*. Cross-batch memory with size 64 begins to enqueue at epoch 5. All experiments are conducted on an NVIDIA GTX 3090 with 24GB GPU memory.

B Effects of Question/Repository Components

To explore the importance of different components of questions and repositories, we conduct an ablation study by removing a component from the question of repository. As listed in Table 6, the performance drops heavily when we delete constituents. Documentation is the most important element, as the most huge performance decrease is caused by deleting Readme documentation from repositories. The loss decreases more slowly , because the difference between samples is harder to distinguish without such an informative constituent.

Models	**MRR**	**P@1**
Complete Data	**86.11**	**80.50**
w/o question title	-3.77	-4.75
w/o question body	-35.43	-36.25
w/o question tags	-3.85	-4.75
w/o repository desc	-2.77	-3.50
w/o repository doc	-47.83	-38.50
w/o repository topics	-3.96	-5.00

Table 6: Results of ablation study on data structure by removing a component.