Uncertainty-Aware Contrastive Learning With Hard Negative Sampling for Code Search Tasks

Han Liu[1 2], **Jiaqing Zhan**[1], **Qin Zhang**[1*]

[1]College of Computer Science and Software Engineering, Shenzhen University
[2]Guangdong Provincial Key Laboratory of Intelligent Information Processing, Shenzhen University
han.liu@szu.edu.cn, zhanjiaqing2023@email.szu.edu.cn, qinzhang@szu.edu.cn

Abstract

Code search is a highly required technique for software development. In recent years, the rapid development of transformer-based language models has made it increasingly more popular to adapt a pre-trained language model to a code search task, where contrastive learning is typically adopted to semantically align user queries and codes in an embedding space. Considering that the same semantic meaning can be presented using diverse language styles in user queries and codes, the representation of queries and codes in an embedding space may thus be non-deterministic. To address the above-specified point, this paper proposes an uncertainty-aware contrastive learning approach for code search. Specifically, for both queries and codes, we design an uncertainty learning strategy to produce diverse embeddings by learning to transform the original inputs into Gaussian distributions and then taking a reparameterization trick. We also design a hard negative sampling strategy to construct query-code pairs for improving the effectiveness of uncertainty-aware contrastive learning. The experimental results indicate that our approach outperforms 10 baseline methods on a large code search dataset with six programming languages. The results also show that our strategies of uncertainty learning and hard negative sampling can really help enhance the representation of queries and codes leading to an improvement of the code search performance.

Introduction

In software development, a code search technique is highly required for programmers to retrieve relevant programming libraries (Singer et al. 2010; Nie et al. 2016; McMillan et al. 2011). In recent years, transform-based language models have become increasingly more powerful, so it has been a popular strategy to adapt a pre-trained language model to a code search task (Feng et al. 2020; Guo et al. 2021, 2022; Romera-Paredes et al. 2024; Hu et al. 2024; Phan and Jannesari 2024). In this context, code search can be achieved by measuring the matching degree of each query-code pair and then ranking the retrieved codes according to their degrees of matching with the given natural language query.

In recent years, related studies have paid more attention to improving the effectiveness of measuring the similarity between queries and codes through enhancing their representation in an embedding space. Specifically, a pre-trained language model that is adapted to a code search task is expected to produce meaningful embeddings of queries and codes, such that the similarity of each query-code pair can be measured accurately. In other words, it is generally expected to align queries and codes in an embedding space, where contrastive learning has been adopted typically to achieve the alignment by pulling each positive query-code pair closer and pushing each negative pair further away (Wang et al. 2021b,a; Jain et al. 2021; Guo et al. 2022). Some recent code search methods driven by contrastive learning include CodeRetriever (Li et al. 2022) and CoCoSoDa (Shi et al. 2023), which have demonstrated their effectiveness in representation alignment of each positive query-code pair.

Despite the above-mentioned effectiveness of existing code search methods, it is worth noting that diverse language styles can be used in user queries and codes for presenting the same semantic meaning, so it may be inappropriate to consider the representation of queries and codes in an embedding space to be deterministic. In other words, it seems more reasonable to produce diverse embeddings for queries and codes for expecting a representation enhancement.

We address above concerns by proposing an uncertainty-aware contrastive learning with hard negative sampling approach. Specifically, we propose a strategy of uncertainty learning to produce diverse embeddings for queries and codes, and design a way of constructing hard negative query-code pairs, in order to improve the effectiveness of inter-modality alignment (for positive query-code pairs) and intra-modality alignment (for positive pairs of queries and codes). The contributions of this paper are summarised as follows:

- We have proposed an uncertainty-aware embedding strategy to obtain diverse representation of queries and codes for better aligning positive pairs of queries and codes.
- We have designed a strategy of hard negative sampling to augment the negative query-code pairs for better aligning positive query-code pairs.
- Experimental results show that our approach outperforms baselines of code search on a dataset with six programming languages and indicate the effectiveness of incorporating uncertainty learning and hard negative sampling into a contrastive learning-driven method of code search.

[*]Corresponding author

Related Work

Code Search Topics about learning code representation become increasingly more popular, proving benefits for tasks like code summarization, search, completion, and commit message generation in software engineering. Code search is especially important, aiding development and maintenance by swiftly locating relevant code snippets that meet product needs and enhance productivity.

Early code search engines use information retrieval to find code fragments. These techniques rely on keyword matching between a query and a code, focusing on textual similarity. Recently, deep learning has received great attention, leading to more studies on deep code search that leverages neural networks to extract high level semantic information of codes and queries. Code2Vec (Alon et al. 2019) is a foundational work in code understanding, which represents code snippets as fixed-length vectors. Gu, Zhang, and Kim proposed the first deep learning-based model for code search tasks, which is referred to as 'CODEnn' and maps codes and queries into a unified semantic space using a recurrent neural network. The code represented as a graph can retain more information than a sequence representation. Some methods (Ling et al. 2021; Liu et al. 2023; Zeng et al. 2023) construct codes as graph-structured data, where graph neural networks are used to encode the data. In recent years, the utilization of large pre-trained models has markedly improved the understanding of code semantics. CodeBERT (Feng et al. 2020), which is pre-trained on query-code pairs, learns the semantic relationship through masked language modeling (MLM) and replaced token detection (RTD) tasks. Guo et al. proposed GraphCodeBERT, which undertakes MLM, edge prediction, and node alignment tasks to guide a model to learn code structure and data dependencies. UniXcoder (Guo et al. 2022) is pre-trained with MLM, unidirectional language modeling (ULM), and denoising tasks. It enhances semantic code representations using multi-modal contrastive learning with abstract syntax tree (AST) and cross-modal generation with code comments.

Code Search With Contrastive Learning Recently, many researchers (Gao, Yao, and Chen 2021; Yan et al. 2021; Zhang et al. 2022) incorporated contrastive learning into Transformers for aligning positive text pairs while distancing negative ones, and the results have indicated the effectiveness of contrastive learning in enhancing the text representation. Thus, there have been several approaches that were designed to integrate contrastive learning into code search (Li et al. 2023; Hu et al. 2023; Li, Zhou, and Shen 2024; Phan and Jannesari 2024) for improving performance by constructing positive and negative code pairs. For example, CodeRetriever (Li et al. 2022) implemented unimodal and bimodal contrastive learning. Code pairs are collected through unsupervised learning based on documentation and function names. Shi et al. proposed CoCoSoDa that involves a soft data augmentation method to construct positive code pairs and utilize a momentum encoder for consistent negative samples generation to enhance code search performance using contrastive learning. Unlike the above-mentioned sampling techniques, we design a strategy that can achieve positive pairs sampling based on the data distribution obtained by uncertainty learning. Also, we explore combining BM25 (Robertson et al. 1995) and cosine similarity, considering pair relation from multiple perspectives.

Uncertainty Learning In recent years, there have been increasingly more studies on uncertainty modelling in deep learning (Shen et al. 2023; Zhang and Wang 2023; Zhang et al. 2024). There are two main types of uncertainty, i.e., model uncertainty (Gal and Ghahramani 2016; Kendall and Cipolla 2016), which is related to parameter noise and indicates the prediction confidence of a model, and data uncertainty, which is associated with noise in training data.

Gal and Ghahramani proposed a theoretical framework which leverages dropout training as an approximate Bayesian inference in deep Gaussian processes and effectively reduces model uncertainty while maintaining computational efficiency. Advances in data uncertainty learning have been applied to computer vision tasks such as face recognition (Khan et al. 2019), semantic segmentation (Isobe and Arai 2017; Salvi et al. 2024; Gupta et al. 2024), and person Re-ID (Yu et al. 2019; Dou et al. 2022), which enhances the model robustness and interpretability by addressing uncertainty. Chang et al. proposed to identify inherent data uncertainty in the continuous mapping spaces and on specific face datasets, and incorporate data uncertainty learning into the face recognition model. To the best of our knowledge, uncertainty learning in code search is under-explored, so we design in the next section a strategy of learning representation uncertainty to address the diversity in language styles used for presentations of queries and codes.

Proposed Approach

In this section, we present our proposed approach of code search, which consists of contrastive learning, hard negative sampling and uncertainty-aware embedding of queries and codes. Specifically, we set two subtasks of contrastive learning for inter-modality alignment and intra-modality alignment, respectively. The former subtask is to align positive query-code pairs and the latter one aims at aligning positive pairs of queries and codes, where a hard negative sampling strategy is designed to create hard negative query-code pairs and an uncertainty-aware embedding method is designed to construct query and code pairs. The entire process of the proposed approach is illustrated in Figure. 1.

For aligning the positive query-code pairs given in a data set, we need to construct more negative pairs. Specifically, in each batch, we set to have k positive pairs and construct $2k^2-k$ negative pairs in total, where $k(k-1)$ negative pairs are obtained by combining query q_i ($i=1,2,\ldots,k$) in the i-th positive pair with each of the codes $\{c_{i'}\}_{i'\neq i}$ in the other $k-1$ positive pairs and the remaining k^2 negative pairs are produced by taking our strategy of hard negative sampling.

In the setting of hard negative sampling, our strategy is to find k queries for each code to construct k hard negative query-code pairs, where k is the number of positive pairs in each batch for a contrastive learning task. Specifically, while each query q_i in the set Q is used in turn as an anchor, each of the other $|Q|-1$ queries is combined with q_i to

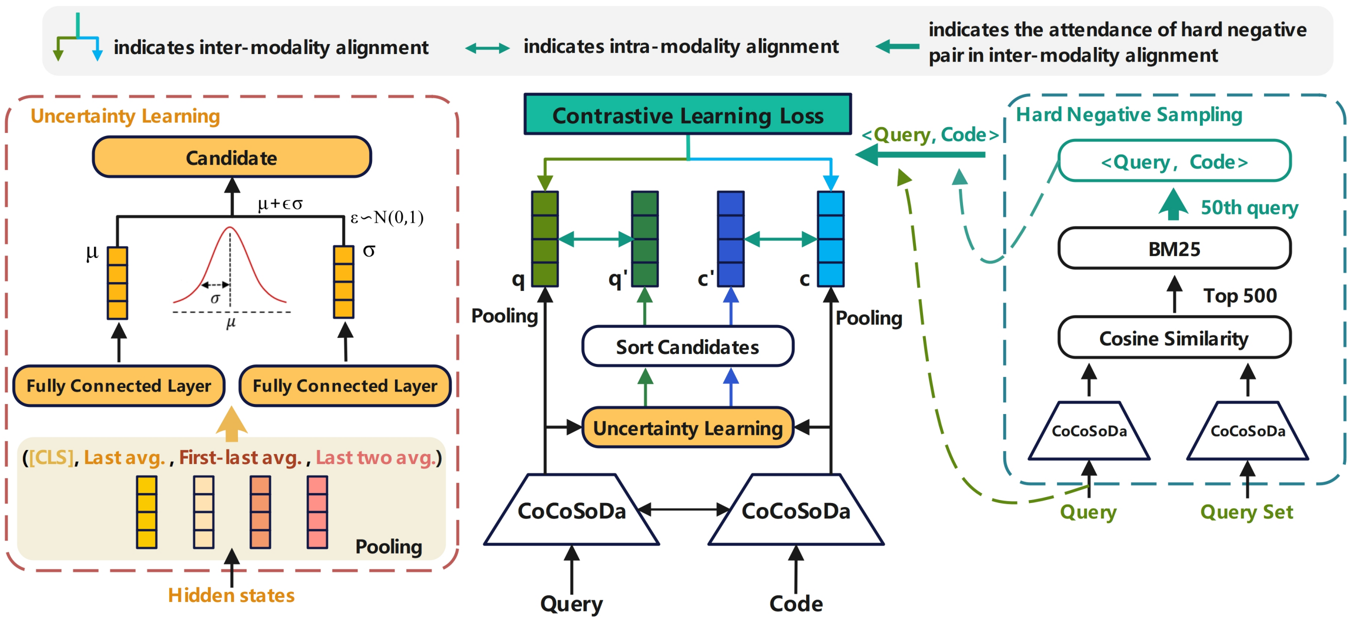

Figure 1: Overview of our proposed approach.

make up a query pair that is supposed to be a negative one, and the similarity of each pair is calculated to select Top-k candidates. Furthermore, the BM25 strategy (Robertson et al. 1995) is taken to finally select the $\frac{k}{10}$-th query pair from the candidates in order to avoid sampling a false negative. Finally, the anchor q_i in the selected query pair and the code c_i^- corresponding to the other query q_i^- in the selected pair make up a hard negative query-code pair in the batch. In addition, we also set to combine c_i^- with each of the other $k-1$ queries in the same batch to make up a hard negative query-code pair. Therefore, given each query q_i as an anchor, k hard negative query-code pairs are constructed, and there are totally k^2 hard negative query-code pairs created in each batch that has k queries used as anchors.

For aligning positive query pairs, as shown in Figure. 1, a query q_i is initially transformed into a vector z_{q_i} by using CoCoSoDa with a pooling operation that takes the average of the token embeddings from the last layer. In the setting of uncertainty-aware embedding, q_i is initially transformed into vector z'_{q_i} by using CoCoSoDa with our designed pooling operation that concatenates the average of the token embeddings from the last layer, the average of that from the first and last layers, the average of that from the last two layers and the [CLS] embedding from the last layer. Furthermore, z'_{q_i} is mapped into a Gaussian distribution, where the mean μ_{q_i} and the variance σ_{q_i} of the distribution are learned simultaneously by setting two fully connected layers in parallel and are considered as the expected representation of a query semantically similar to q_i and the corresponding uncertainty, respectively. Through a reparameterization trick $(\mu_{q_i} + \epsilon\sigma_{q_i}, \epsilon \sim \mathcal{N}(0, I))$, a vector of query q_i is obtained, which can be combined with z_{q_i} to make up a positive pair in a batch for contrastive learning. Since ϵ is obtained by sampling from $\mathcal{N}(0, I)$, we set to sample multiple values of ϵ for reparameterization to produce multiple vectors for each query and then select the vector $z_{q_i}^+$ that is the most similar to z_{q_i} for constructing the positive pair. In addition, the selected vector $z_{q_i}^+$ is also combined with each other vector $z_{q_{i'}}$ to make up negative pairs, where $z_{q_{i'}}$ is obtained using CoCoSoDa for representing each other query $q_{i'}$ in the same batch. In this context, while each batch consists of k queries, there are k positive query pairs and $k(k-1)$ negative pairs. The same procedure specified above is also applied to producing diverse embeddings for each code c_i and aligning positive code pairs.

The loss function set for our proposed approach is shown in Eq. (1), which consists of three terms, namely, contrastive loss $\mathcal{L}_{cl_1}$ illustrated in Eq. (2) for aligning positive query-code pairs, contrastive loss $\mathcal{L}_{cl_2}$ illustrated in Eq. (3) for aligning positive pairs of queries and codes and KL divergence regulariser $\mathcal{L}_{kl}$ illustrated in Eq. (4).

In Eq. (1), λ is a regularisation coefficient. In Eq. (2), z_{q_i} and z_{c_i} make up a positive query-code pair, z_{q_i} and z_{c_j} make up an in-batch negative query-code pair, z_{q_i} and $z_{c_j}^-$ constitute a hard negative query-code pair and τ represents the temperature parameter of a softmax function.

$$\mathcal{L}_{total} = \mathcal{L}_{cl_1} + \mathcal{L}_{cl_2} + \lambda\mathcal{L}_{kl} \tag{1}$$

$$\mathcal{L}_{cl_1} = -\log \frac{e^{sim(z_{q_i}, z_{c_i})/\tau}}{\sum_{j=1}^{k}(e^{sim(z_{q_i}, z_{c_j})/\tau} + e^{sim(z_{q_i}, z_{c_j}^-)/\tau})} \tag{2}$$

In Eq. (3), z_{q_i} and $z_{q_j}^+$ make up a negative query pair, $z_{q_i}^+$ is a positive sample with respect to the query anchor z_{q_i}, z_{c_i} and $z_{c_j}^+$ constitute a negative pair of codes and $z_{c_i}^+$ is a positive

Language	Training	Dev	Test	Candidates size
Python	251,820	13,914	14,918	43,827
PHP	241,241	12,982	14,014	52,660
Go	167,288	7,325	8,122	28,120
Java	164,923	5,183	10,955	40,347
JavaScript	58,025	3,885	3,291	13,981
Ruby	24,927	1,400	1,261	4,360

Table 1: Dataset statistics.

sample with respect to the code anchor z_{c_i}.

$$\mathcal{L}_{cl_2} = -(\log \frac{e^{sim(z_{q_i}, z_{q_i}^+)/\tau}}{\sum_{j=1}^{k} e^{sim(z_{q_i}, z_{q_j}^+))/\tau}} + \log \frac{e^{sim(z_{c_i}, z_{c_i}^+)/\tau}}{\sum_{j=1}^{k} e^{sim(z_{c_i}, z_{c_j}^+)/\tau}}) \tag{3}$$

$$\mathcal{L}_{kl} = KL[\mathcal{N}(z_{q_i}^+ | \mu_{q_i}, \sigma_{q_i}^2) \parallel \mathcal{N}(\epsilon | 0, \mathrm{I})] + KL[\mathcal{N}(z_{c_i}^+ | \mu_{c_i}, \sigma_{c_i}^2) \parallel \mathcal{N}(\epsilon | 0, \mathrm{I})] \tag{4}$$

In Eq. (4), the two terms are provided to drive the uncertainty learning for queries and codes, respectively. As inspired from Chang et al., it is encouraged to have large σ for some queries and codes and small σ for the others. In our setting, it is expected intuitively to obtain large σ for those queries and codes that can be presented in more diverse language styles, and to have small σ for the others that have limited language styles for presentations.

Experiments

Datasets

We conducted experiments on the CodeSearchNet code corpus (Husain et al. 2019), to be consistent with Guo et al.. CodeSearchNet contains six languages, namely, Ruby, JavaScript, Go, Python, Java, and PHP, and has been widely used in previous studies. To make the experimental setup closer to the real scenarios, Guo et al. expanded the candidate dataset and filtered out low-quality queries on each code corpus through rules, where the data statistics are shown in Table 1. Following previous studies, in the inference phase, a model retrieves the semantically closest code snippets from the candidate set based on the given query.

Implementation Details

In previous studies on code search (Feng et al. 2020; Guo et al. 2021; Wang et al. 2021b), it is common practice to use other code snippets in other query-code pairs within the same batch as negative examples for a given query-code pair. This paper evaluates this conventional approach, which is referred to as “In-Batch Negative” in Table 2, and compares the approach with the “Hard Negative” method, which is integrated into our proposed negative sampling technique. The analysis of model performance under the “Hard Negative” setting will be further detailed in this section.

We implement a dual encoder model using a Transformer architecture that has 12 layers, 768-dimensional hidden states, and 12 attention heads. The parameters of the pre-trained CoCoSoDa (Shi et al. 2023) are utilized to initialize this dual encoder. To reduce the total number of parameters, we follow previous studies by sharing parameters between the query encoder and the code one. For training, we set the batch size to 128, the temperature hyperparameter to 0.03, the number of epochs to 10, and the random seed to 123456. The maximum sequence lengths are set to 256 for code snippets and 128 for queries. We use the AdamW optimizer with a learning rate of 8e-6. All experiments were conducted on a machine equipped with four NVIDIA GeForce RTX 4090 GPUs which each has 24GB of memory.

Baselines

We compare the proposed approach with 10 baseline methods specified as follows:

- **RoBERTa (code)** (Guo et al. 2021) is pre-trained with MLM objective on large code corpus. To improve model performance, key hyperparameters are modified and the next sentence prediction task is omitted.
- **CodeBERT** (Feng et al. 2020) extends the RoBERTa framework by pre-training on both bimodal and unimodal data. To leverage both modalities, a hybrid objective that integrates MLM and RTD tasks is constructed.
- **GraphCodeBERT** (Guo et al. 2021) is pre-trained on large code corpus by considering semantic structures of code, where edge prediction and variable alignment tasks are set for learning to capture structural information.
- **SynCoBERT** (Wang et al. 2021a) integrates source code, AST, and comment for syntax-driven multimodal contrastive pre-training, with objectives for identifier prediction, AST edge prediction and contrastive learning.
- **UniXcoder** (Guo et al. 2022) is a unified cross-modal pre-trained model that takes AST and code comments as inputs, involving MLM, ULM, denoising objectives, and multi-modal contrastive learning.
- **CodeRetriever** (Li et al. 2022) employs both unimodal and bimodal contrastive learning, and integrates SimCSE to train a matcher, while retrieving through function names and documentation to collect positive pairs.
- **Soft-InfoNCE** (Li et al. 2023) is an extension of vanilla InfoNCE with a weight term to explicitly model relationships of negative pairs. The weight terms are estimated using BM25, SimCSE, and Trained Models.
- **TOSS** (Hu et al. 2023) involves a two-stage fusion framework that employs text matching or bi-encoder methods in the first stage and uses cross-encoder reranking to enhance performance in the second stage.
- **CoCoSoDa** (Shi et al. 2023) integrates momentum encoders to produce consistent negative code pairs and involves a soft data augmentation method to create positive pairs to set unimodal and bimodal contrastive learning.
- **SEA** (Hu et al. 2024) splits long code via AST, encodes blocks into embeddings, and aggregates these based on attention to form comprehensive code representations.

Model	Ruby	Javascript	GO	Python	Java	PHP	AVG.
RoBERTa (code) (Guo et al. 2021)	0.628	0.562	0.859	0.610	0.620	0.579	0.643
CodeBERT (Feng et al. 2020)	0.693	0.706	0.840	0.869	0.748	0.706	0.760
GraphCodeBERT (Guo et al. 2021)	0.703	0.644	0.897	0.692	0.691	0.649	0.713
SYNCOBERT (Wang et al. 2021a)	0.722	0.677	0.913	0.724	0.723	0.678	0.740
UniXcoder (Guo et al. 2022)	0.740	0.684	0.915	0.720	0.726	0.676	0.744
CodeRetriever (Li et al. 2022)	0.771	0.719	0.924	0.758	0.765	0.708	0.774
Soft-InfoNCE (Li et al. 2023)	0.753	0.693	0.916	0.728	0.733	0.684	0.751
TOSS (Hu et al. 2023)	0.765	0.696	0.918	0.760	0.750	0.692	0.763
CoCoSoDa (Shi et al. 2023)	0.818	0.764	0.921	0.757	0.763	0.703	0.788
SEA (Hu et al. 2024)	0.776	0.742	0.921	0.754	0.768	**0.748**	0.785
Our (In-Batch Negative)	0.825	0.775	0.923	0.768	0.769	0.717	0.796
Our (Hard Negative)	**0.829**	**0.778**	**0.926**	**0.773**	**0.772**	0.719	**0.800**

Table 2: Performance of different approaches on MRR. Results on compared models are from the previous papers

In our experiments, the effectiveness of our proposed method is evaluated by comparing with the state-of-the-art pre-trained models. The results on baseline models are taken from their original papers or previous studies.

Evaluation Metrics

We use four evaluation metrics to measure the performance, namely, mean reciprocal rank (MRR) and top-k recall ($R@k$, $k = 1, 5, 10$), which are widely used in information retrieval. MRR is defined as the mean of the reciprocal rankings of the target codes (corresponding to the given queries in set Q) in the candidate sets. $R@k$ metric quantifies the proportion of queries which each identifies the correct code snippet within the returned list of k ranked candidates. They can be calculated as follows:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{Rank_i}, \quad R@k = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\delta(Rank_i \leq k) \tag{5}$$

where $Rank_i$ is the serial number of the target code fragment in the candidate queue corresponding to the i-th natural language query. $\delta(\cdot)$ is an indicator function that returns 1 if $Rank_i \leq k$ and returns 0 otherwise.

Results on Comparisons with Baselines

We compare our proposed method with six pre-training-based methods and four fine-tuned models on CodeSearchNet to evaluate the effectiveness. To ensure fairness in the comparison, the results reported in the previous papers for the baseline methods are used. Experimental results on MRR are presented in Table 2, while the results on $R@k(k = 1, 2, 3)$ are shown in Figure 2. Due to space limitations, we only present $R@k$ results for some of the models. Conclusions on MRR are consistent with ones on $R@k$.

The results shown in Table 2 indicate that our approach performs better than all those baseline methods. The results also demonstrate that incorporating hard negative sampling to construct more negative query-code pairs in addition to in-batch negative pairs leads to performance improvements,

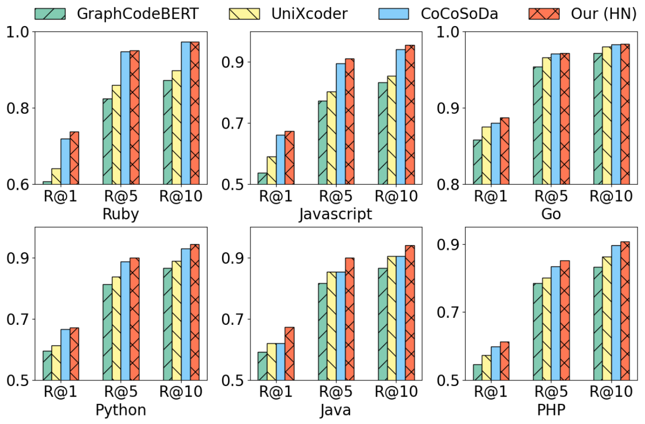

Figure 2: Performance of the various approaches evaluated on $R@k(k = 1, 5, 10)$, with identical models indicated by the same colour. HN is short for Hard Negative.

suggesting that negative query-code pairs with higher similarity have some impacts on model performance. Figure 2 shows that the improvement in $R@k$ metrics becomes more apparent as the size of the candidate set of code snippets is decreased for a given query. The phenomenon indicates that our approach can achieve accurately matching and selecting relevant code snippets even when the candidate set is small.

Moreover, compared with models trained on both codes and natural language queries, Roberta (Guo et al. 2021), which is trained only on a code corpus, demonstrates limited performance in downstream tasks. Among the rest of the models, GraphCodeBERT introduces dataflow information considering the semantic structure of the code and thus outperforms CodeBERT. The contrastive learning based models outperform others in code search tasks, demonstrating the superior effectiveness of contrastive learning in representation enhancement. Unixcoder and SynCoBERT perform better, while syntax information is utilized and AST of the code is incorporated into multi-modal contrastive learning

Method	Ruby	Javascript	Go	Python	Java	PHP	AVG.
Bi+Uni+sort+HN	**0.829**	**0.778**	**0.926**	**0.773**	**0.772**	**0.719**	**0.800**
Bi+Uni+sort+RS	0.823	0.777	0.923	0.768	0.770	0.716	0.796
Bi+sort+HN	0.822	0.771	0.924	0.764	0.767	0.710	0.793
Bi+Uni+sort	0.825	0.775	0.923	0.768	0.769	0.717	0.796
Bi+Uni	0.822	0.774	0.919	0.769	0.768	0.714	0.794

Table 3: Ablation study.

task during pre-training period. Considering the potential relationships between negative codes, UniXcoder, which is fine-tuned using Soft-InfoNCE loss, may learn better representation that helps improve the code search performance. TOSS combines different methods to increase recall diversity and achieve higher accuracy. CodeRetriever, which is fine-tuned on GraphCodeBERT, achieves better performance than UniXcoder and SynCoBERT, due possibly to an effective positive pair construction method that enlarges the corpus. We can also observe that the SEA method using GraphCodeBERT as a code encoder improves the code search performance over other Transformer-based models, thanks to the AST-based splitting and attention-based aggregation. CoCoSoDa, which involves soft data augmentation and utilises a momentum encoder to enrich samples used in a batch, shows its superiority over the other baselines.

Ablation Study

To evaluate the impacts of those components of our approach, some ablation experiments are conducted. Specifically, we analyze the contributions of hard negative sampling, uncertainty-aware embedding, and contrastive learning methods. Herein, **Bi** and **Uni** denote bimodal and unimodal contrastive learning, respectively, whereas **HN** and **RS** represent hard negative sampling and random sampling methods. **Sort** indicates the candidate sorting module.

Table 3 shows that the removal of any component results in a degradation of the model performance, demonstrating that each component can impact the code search performance. The performance can be improved by incorporating the candidate sorting module to select candidates closer to the query in an embedding space. Specifically, the absence of uncertainty-aware embedding and intra-modal contrastive learning leads to a considerable decline in performance. This decline may be attributed to the case that uncertainty-aware embedding enhances representation diversity and increases the difficulty of aligning positive pairs of queries and codes, while intra-modal contrastive learning ensures a uniform distribution of embeddings for queries and codes. Inter-modal contrastive learning, which aims at aligning positive query-code pairs while distancing the negative pairs, is important for developing a well aligned semantic space between code snippets and queries. The removal of hard negative samples or the substitution of semantics-based negative sampling with random sampling can further impair model performance. Those negative query-code pairs which each shows high similarity assist the model in capturing subtle differences between positive and negative code snippets, thereby facilitating the representation enhancement.

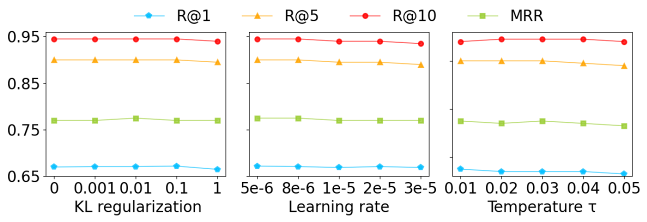

Figure 3: Impacts of the hyperparameters on Python corpus. Identical metrics are marked with the same color and marker.

Impacts of Hyperparameters

This section analyses the sensitivity of various hyperparameters on the Python corpus, providing insights for optimized configurations. Hyperparameters include the KL divergence regularization coefficient, learning rate and temperature in our loss function. Referring to previous studies, we explore the effects of various hyperparameters within typical ranges. The experimental results are shown in Figure 3.

The results obtained using varied temperatures show that the performance is generally stable for small temperature variations (between 0.01 and 0.05). However, a temperature above 0.04 can have an obvious effect on the model performance. The above finding suggests that setting an appropriate temperature is important. The learning rate, which is considered as a key hyperparameter, has a stable range between 5e-6 and 2e-5, where a high learning rate leads to performance degradation. Setting the learning rate value to 8e-6 results in the overall peak performance of the model. In terms of the impact of the KL divergence coefficient in our loss function, the results show that small adjustments in the regularisation coefficient cause only small fluctuations in performance, while setting a large coefficient can significantly degrade the model performance, suggesting that it is more encouraging to set the parameter to a small value.

Analysis

Quantitive analysis Recently, Wang and Isola identified two key properties related to contrastive learning alignment and uniformity. Alignment measures the closeness of features for positive pairs and reflects precise pair matching. Uniformity evaluates the distribution of features on the hypersphere and promotes a uniform distribution of embeddings for better separation of negative pairs. Effective models promote both of these properties, thereby improving the overall performance and the quality of representation. Mathematically, they can be computed as follows (assuming the representations to be already normalized):

$$\begin{aligned}\ell_{\text{align}} &\triangleq \underset{(x,x^+)\sim p_{\text{pos}}}{\mathbb{E}} \|f(x) - f\left(x^+\right)\|^2 \\ \ell_{\text{uniform}} &\triangleq \log \underset{\substack{i.i.d.\\ x,y\sim p_{\text{data}}}}{\mathbb{E}} e^{-2\|f(x)-f(y)\|^2}\end{aligned} \tag{6}$$

where $(X, Y) \sim p_{\text{pos}}$ means that X and Y are positive pairs , while $\overset{i.i.d.}{x,y\sim p_{\text{data}}}$ denotes independently and identically distribution of X and Y. To investigate the impact of our

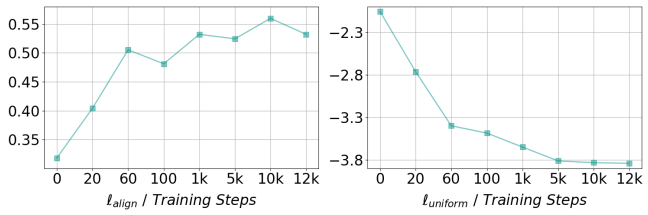

Figure 4: Alignment and uniformity curve during training.

method on the embedding space, we use the metrics mentioned above to observe changes in the distribution of embeddings during training.

As shown in Figure 4, we can observe a gradual decrease in the uniformity loss of the model, indicating alleviation in anisotropy. Combining with the findings drawn from Table 2, which shows that our method effectively improves the model performance, it is suggested that our approach helps to align code and query representations. However, the loss of alignment is increased during training. These two findings consistently indicate that the model sacrifices some degree of alignment in order to optimise the overall uniformity of the feature space. This trade-off helps to prevent overfitting and leads to a better semantic space, thus improving semantic understanding. The aforementioned results indicate that our approach enhances the uniformity of hypersphere despite compromising on alignment. In this situation, the model can be enhanced to analyse the semantic relationships between codes and natural language queries, leading to an improvement of the performance in code search tasks.

Qualitative analysis t-SNE is a dimensionality reduction technique that visualises high-dimensional data while preserving pairwise similarities. For our analysis, we used t-SNE to visualise embeddings obtained from the CoCoSoDa model and our proposed method. We randomly sampled 300 query-code pairs from the Go corpus, reduced their dimensionality to 2D, and compared the embeddings to analyse the performance. This visualisation helps us to understand the effectiveness of a model in preserving meaningful structure in code representation.

As illustrated in Figure 5, code snippets and queries are highlighted in purple and pink, respectively, with green lines indicating the pairwise distances that query-code pairs have. The figure demonstrates that most sample pairs exhibit relatively short distances, indicating that both the pre-trained CoCoSoDa model and the model enhanced using our method effectively map positive samples to closely related embeddings in the feature space. Although the right image shows more long lines compared with the left image, these distances are not as extreme as those observed in the left image. Additionally, the right image displays points that are more uniformly and widely distributed, in contrast to the left image where points are more concentrated in the center.

This observation aligns with our analysis shown in the curves for ℓ_{align} and $\ell_{uniform}$ during model training (see Figure 4). Throughout the training process, the model adjusts the feature space structure to achieve a more uniform

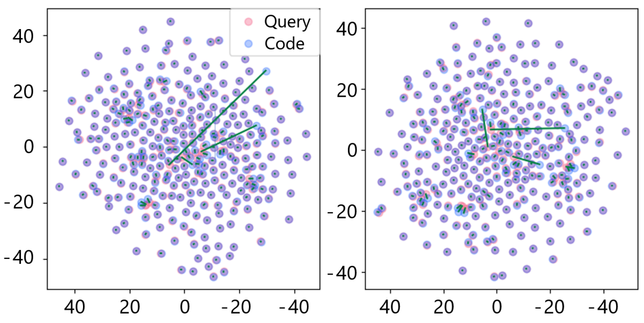

Figure 5: T-SNE visualizations of representations produced by CoCoSoda (left image) and our method (right image) are shown, while blue denotes code(Go) and the ones on queries are in pink, with green lines indicating distances between pairs. Purple signifies overlapping query and code, indicating that the paired code was successfully retrieved.

distribution of feature embeddings, while involving a compromise in alignment. Accordingly, in comparison with the left image, the right image shows a larger number of long lines and thus indicates that the model fine-tuned using our method exhibits slightly lower alignment than the pre-trained model. However, as shown in Table 2, this alignment compromise does not adversely affect the performance of the model. Although there are more long lines in the right image, their lengths are smaller than the ones of those long lines in the left image. Meanwhile, the points in the right image are more uniformly distributed, reflecting a more coordinated feature space. The phenomenon suggests that our method leads to a more balanced and effective embedding space. Overall, the visualization results indicate that the model enhanced using our method produces superior embeddings compared with the pre-trained CoCoSoDa model.

Conclusion

In this paper, we have proposed an approach of uncertainty-aware contrastive learning with hard negative sampling for code search tasks. Specifically, for improving the effectiveness of inter-modal alignment (for positive query-code pairs) and intra-modality alignment (for positive pairs of queries and codes) in an embedding space, we have proposed a strategy of uncertainty learning to produce diverse embeddings for queries and codes and have designed a way of constructing hard negative query-code pairs. The experimental results on a large benchmark dataset with six programming languages indicate that our approach outperforms 10 baseline methods. The results of ablation experiments show that the incorporation of uncertainty learning and hard negative sampling into contrastive learning can really help enhance the representation of queries and codes leading to an improvement of the code search performance. In the future, we will explore diffusion-driven strategies of generating diverse user queries that can be used for searching same code snippets.

Acknowledgments

This paper is supported by National Natural Science Foundation of China (Grants 62106147 and 62206179), Guangdong Provincial Key Laboratory (Grant 2023B1212060076) and Shenzhen Science and Technology Program (Grant ZDSYS20220527171400002).

References

Alon, U.; Zilberstein, M.; Levy, O.; and Yahav, E. 2019. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–29.

Chang, J.; Lan, Z.; Cheng, C.; and Wei, Y. 2020. Data uncertainty learning in face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5710–5719.

Dou, Z.; Wang, Z.; Chen, W.; Li, Y.; and Wang, S. 2022. Reliability-aware prediction via uncertainty learning for person image retrieval. In *European Conference on Computer Vision*, 588–605. Springer.

Feng, Z.; Guo, D.; Tang, D.; Duan, N.; Feng, X.; Gong, M.; Shou, L.; Qin, B.; Liu, T.; Jiang, D.; et al. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1536–1547.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6894–6910.

Gu, X.; Zhang, H.; and Kim, S. 2018. Deep code search. In *Proceedings of the 40th International Conference on Software Engineering*, 933–944.

Guo, D.; Lu, S.; Duan, N.; Wang, Y.; Zhou, M.; and Yin, J. 2022. UniXcoder: Unified Cross-Modal Pre-training for Code Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 7212–7225.

Guo, D.; Ren, S.; Lu, S.; Feng, Z.; Tang, D.; Shujie, L.; Zhou, L.; Duan, N.; Svyatkovskiy, A.; Fu, S.; et al. 2021. GraphCodeBERT: Pre-training Code Representations with Data Flow. In *International Conference on Learning Representations*.

Gupta, S.; Zhang, Y.; Hu, X.; Prasanna, P.; and Chen, C. 2024. Topology-aware uncertainty for image segmentation. *Advances in Neural Information Processing Systems*, 36.

Hu, F.; Wang, Y.; Du, L.; Li, X.; Zhang, H.; Han, S.; and Zhang, D. 2023. Revisiting code search in a two-stage paradigm. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 994–1002.

Hu, F.; Wang, Y.; Du, L.; Zhang, H.; Zhang, D.; and Li, X. 2024. Tackling Long Code Search with Splitting, Encoding, and Aggregating. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 15500–15510.

Husain, H.; Wu, H.-H.; Gazit, T.; Allamanis, M.; and Brockschmidt, M. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*.

Isobe, S.; and Arai, S. 2017. Deep convolutional encoder-decoder network with model uncertainty for semantic segmentation. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 365–370. IEEE.

Jain, P.; Jain, A.; Zhang, T.; Abbeel, P.; Gonzalez, J.; and Stoica, I. 2021. Contrastive Code Representation Learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 5954–5971.

Kendall, A.; and Cipolla, R. 2016. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, 4762–4769. IEEE.

Khan, S.; Hayat, M.; Zamir, S. W.; Shen, J.; and Shao, L. 2019. Striking the right balance with uncertainty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 103–112.

Li, H.; Zhou, X.; Luu, A.; and Miao, C. 2023. Rethinking Negative Pairs in Code Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 12760–12774.

Li, H.; Zhou, X.; and Shen, Z. 2024. Rewriting the Code: A Simple Method for Large Language Model Augmented Code Search. In *The 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*.

Li, X.; Gong, Y.; Shen, Y.; Qiu, X.; Zhang, H.; Yao, B.; Qi, W.; Jiang, D.; Chen, W.; and Duan, N. 2022. Coderetriever: A large scale contrastive pre-training method for code search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2898–2910.

Ling, X.; Wu, L.; Wang, S.; Pan, G.; Ma, T.; Xu, F.; Liu, A. X.; Wu, C.; and Ji, S. 2021. Deep graph matching and searching for semantic code retrieval. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5): 1–21.

Liu, S.; Xie, X.; Siow, J.; Ma, L.; Meng, G.; and Liu, Y. 2023. Graphsearchnet: Enhancing gnns via capturing global dependencies for semantic code search. *IEEE Transactions on Software Engineering*, 49(4): 2839–2855.

McMillan, C.; Grechanik, M.; Poshyvanyk, D.; Xie, Q.; and Fu, C. 2011. Portfolio: finding relevant functions and their usage. In *Proceedings of the 33rd International Conference on Software Engineering*, 111–120.

Nie, L.; Jiang, H.; Ren, Z.; Sun, Z.; and Li, X. 2016. Query expansion based on crowd knowledge for code search. *IEEE Transactions on Services Computing*, 9(5): 771–783.

Phan, H.; and Jannesari, A. 2024. Leveraging Statistical Machine Translation for Code Search. In *Proceedings of the*

28th International Conference on Evaluation and Assessment in Software Engineering, 191–200.

Robertson, S. E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M. M.; Gatford, M.; et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp*, 109: 109.

Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Balog, M.; Kumar, M. P.; Dupont, E.; Ruiz, F. J.; Ellenberg, J. S.; Wang, P.; Fawzi, O.; et al. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995): 468–475.

Salvi, M.; Mogetta, A.; Raghavendra, U.; Gudigar, A.; Acharya, U. R.; and Molinari, F. 2024. A Dynamic Uncertainty-Aware Ensemble Model: Application to Lung Cancer Segmentation in Digital Pathology. *Applied Soft Computing*, 112081.

Shen, H.; Chen, S.; Wang, R.; and Wang, X. 2023. Adversarial Learning with Cost-Sensitive Classes. *IEEE Transactions on Cybernetics*, 53(8): 4855–4866.

Shi, E.; Wang, Y.; Gu, W.; Du, L.; Zhang, H.; Han, S.; Zhang, D.; and Sun, H. 2023. Cocosoda: Effective contrastive learning for code search. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2198–2210. IEEE.

Singer, J.; Lethbridge, T.; Vinson, N.; and Anquetil, N. 2010. An examination of software engineering work practices. In *CASCON First Decade High Impact Papers*, 174–188.

Wang, T.; and Isola, P. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, 9929–9939. PMLR.

Wang, X.; Wang, Y.; Mi, F.; Zhou, P.; Wan, Y.; Liu, X.; Li, L.; Wu, H.; Liu, J.; and Jiang, X. 2021a. Syncobert: Syntax-guided multi-modal contrastive pre-training for code representation. *arXiv preprint arXiv:2108.04556*.

Wang, Y.; Wang, W.; Joty, S.; and Hoi, S. C. 2021b. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 8696–8708.

Yan, Y.; Li, R.; Wang, S.; Zhang, F.; Wu, W.; and Xu, W. 2021. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. 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)*, 5065–5075.

Yu, T.; Li, D.; Yang, Y.; Hospedales, T. M.; and Xiang, T. 2019. Robust person re-identification by modelling feature uncertainty. In *Proceedings of the IEEE/CVF international conference on computer vision*, 552–561.

Zeng, C.; Yu, Y.; Li, S.; Xia, X.; Wang, Z.; Geng, M.; Bai, L.; Dong, W.; and Liao, X. 2023. degraphcs: Embedding variable-based flow graph for neural code search. *ACM Transactions on Software Engineering and Methodology*, 32(2): 1–27.

Zhang, T.; and Wang, X. 2023. Anchor-Wise Fuzziness Modeling in Convolution-Transformer Neural Networks for Left Atrium Image Segmentation. *IEEE Transactions on Fuzzy Systems*, 32(2): 398–408.

Zhang, W.; Lv, Z.; Zhou, H.; Liu, J.-W.; Li, J.; Li, M.; Li, Y.; Zhang, D.; Zhuang, Y.; and Tang, S. 2024. Revisiting the Domain Shift and Sample Uncertainty in Multi-source Active Domain Transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16751–16761.

Zhang, Y.; Zhang, R.; Mensah, S.; Liu, X.; and Mao, Y. 2022. Unsupervised sentence representation via contrastive learning with mixing negatives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 11730–11738.