Enhancing Code Search Intent with Programming Context Exploration

Yanmin Dong[1], Zhenya Huang[1,2]*, Zheng Zhang[1], Guanhao Zhao[1] , Likang Wu[3], Hongke Zhao[3], Binbin Jin[1] , Qi Liu[1]

[1]State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China

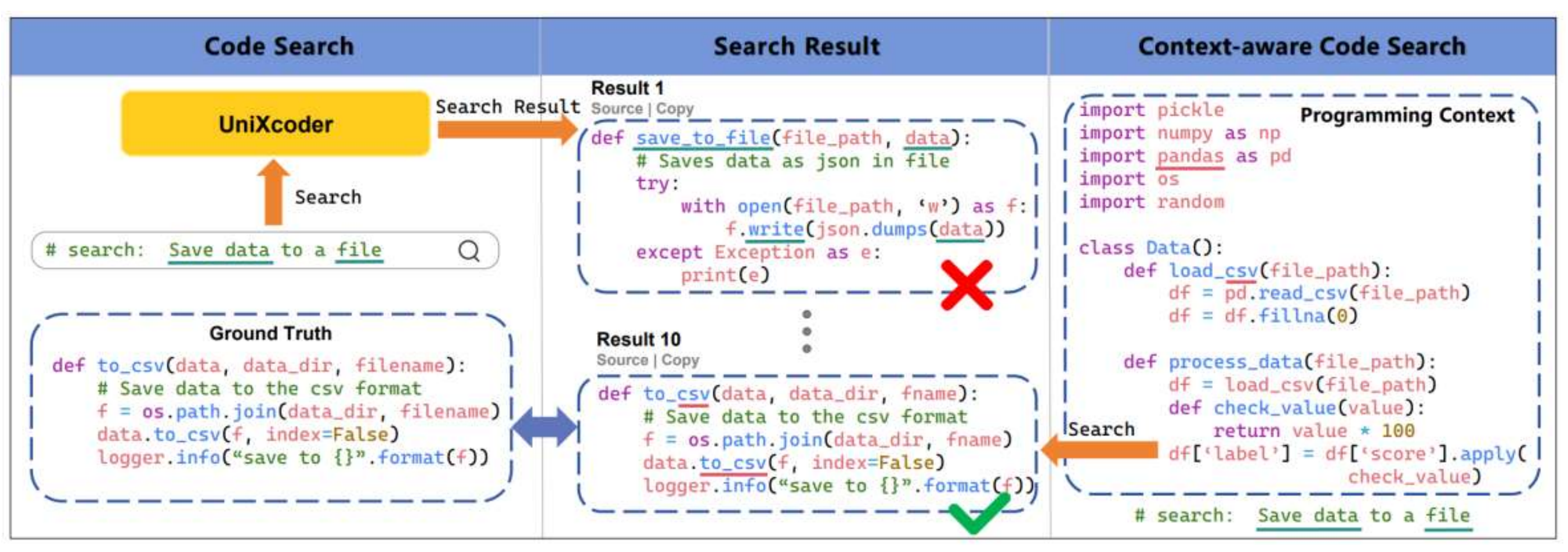
[2]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center

[3]College of Management and Economics, Tianjin University;

Introduction

Motivation

- Code search is a crucial task for developers. However, short and ambiguous queries often lead to suboptimal search results. For example, a developer searches for "save data to a file," the top-ranked result retrieved by UniXcoder is incorrect, while the ground truth appears much lower in the ranking. A deeper analysis reveals that the retrieved top result does implement saving data to a file, but it saves in a different format than what the developer intended (e.g., a generic file format instead of CSV). This discrepancy highlights a fundamental challenge: developers often struggle to express their precise intent in short queries, leading to ineffective search results. Addressing this issue is essential to improving the accuracy and usability of code search systems.



```
# search:  Save data to a file
```

Ground Truth

```
def to_csv(data, data_dir, filename):
    # Save data to the csv format
    f = os.path.join(data_dir, filename)
    data.to_csv(f, index=False)
    logger.info("save to {}".format(f))
```

Result 1

```
def save_to_file(file_path, data):
    # Saves data as json in file
    try:
        with open(file_path, 'w') as f:
            f.write(json.dumps(data))
    except Exception as e:
        print(e)
```

Result 10

```
def to_csv(data, data_dir, fname):
    # Save data to the csv format
    f = os.path.join(data_dir, fname)
    data.to_csv(f, index=False)
    logger.info("save to {}".format(f))
```

Programming Context

```
import pickle
import numpy as np
import pandas as pd
import os
import random

class Data():
    def load_csv(file_path):
        df = pd.read_csv(file_path)
        df = df.fillna(0)

    def process_data(file_path):
        df = load_csv(file_path)
        def check_value(value):
            return value * 100
        df['label'] = df['score'].apply(
                            check_value)
# search:  Save data to a file
```

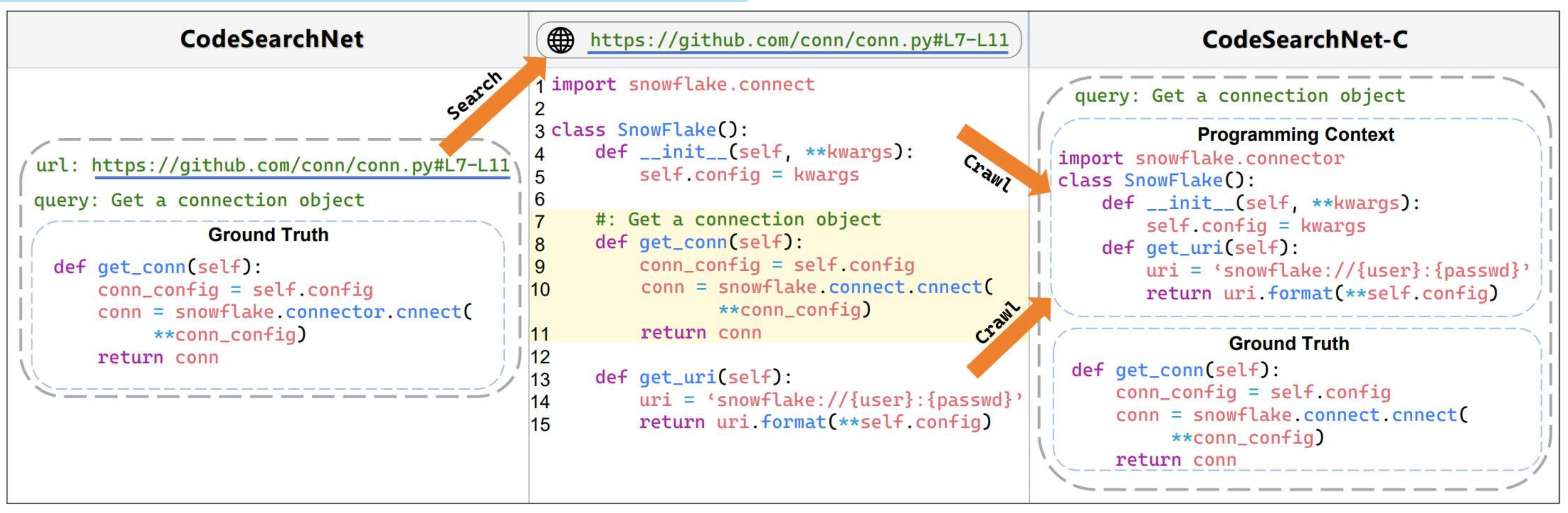
Fig1: The left represents the code search process in UniXcoder. The middle is the search result. The right represents the scenario of writing code in an IDE.

Challenge

- **Existing benchmarks do not provide programming context, making it difficult to study context-aware code search effectively.**
- **Developers write code in different orders and query at different stages, requiring a realistic simulation of their habits.**
- **The programming context has a complex hierarchical structure, and extracting relevant information while filtering out noise remains a challenge.**

Model

CodeSearchNet-C Dataset Construction



CodeSearchNet

```
url: https://github.com/conn/conn.py#L7-L11
query: Get a connection object
```

Ground Truth

```
def get_conn(self):
    conn_config = self.config
    conn = snowflake.connector.cnnect(
           **conn_config)
    return conn
```

https://github.com/conn/conn.py#L7-L11

```
1  import snowflake.connect
2
3  class SnowFlake():
4      def __init__(self, **kwargs):
5          self.config = kwargs
6
7      #: Get a connection object
8      def get_conn(self):
9          conn_config = self.config
10         conn = snowflake.connect.cnnect(
                   **conn_config)
11         return conn
12
13     def get_uri(self):
14         uri = 'snowflake://{user}:{passwd}'
15         return uri.format(**self.config)
```

CodeSearchNet-C

```
query: Get a connection object
```

Programming Context

```
import snowflake.connector
class SnowFlake():
    def __init__(self, **kwargs):
        self.config = kwargs
    def get_uri(self):
        uri = 'snowflake://{user}:{passwd}'
        return uri.format(**self.config)
```

Ground Truth

```
def get_conn(self):
    conn_config = self.config
    conn = snowflake.connect.cnnect(
           **conn_config)
    return conn
```

Fig2: The process of constructing our CodeSearchNet-C dataset.

Table1: The statistics of the CodeSearchNet-C dataset..

Language	Java	Python
Num. queries	181061	280652
Avg. tokens of query	13.2	11.4
Num. codes in codebase	40347	43827
Num. context	178622	275347
Avg. tokens of context	7877.8	8312.3

Framework

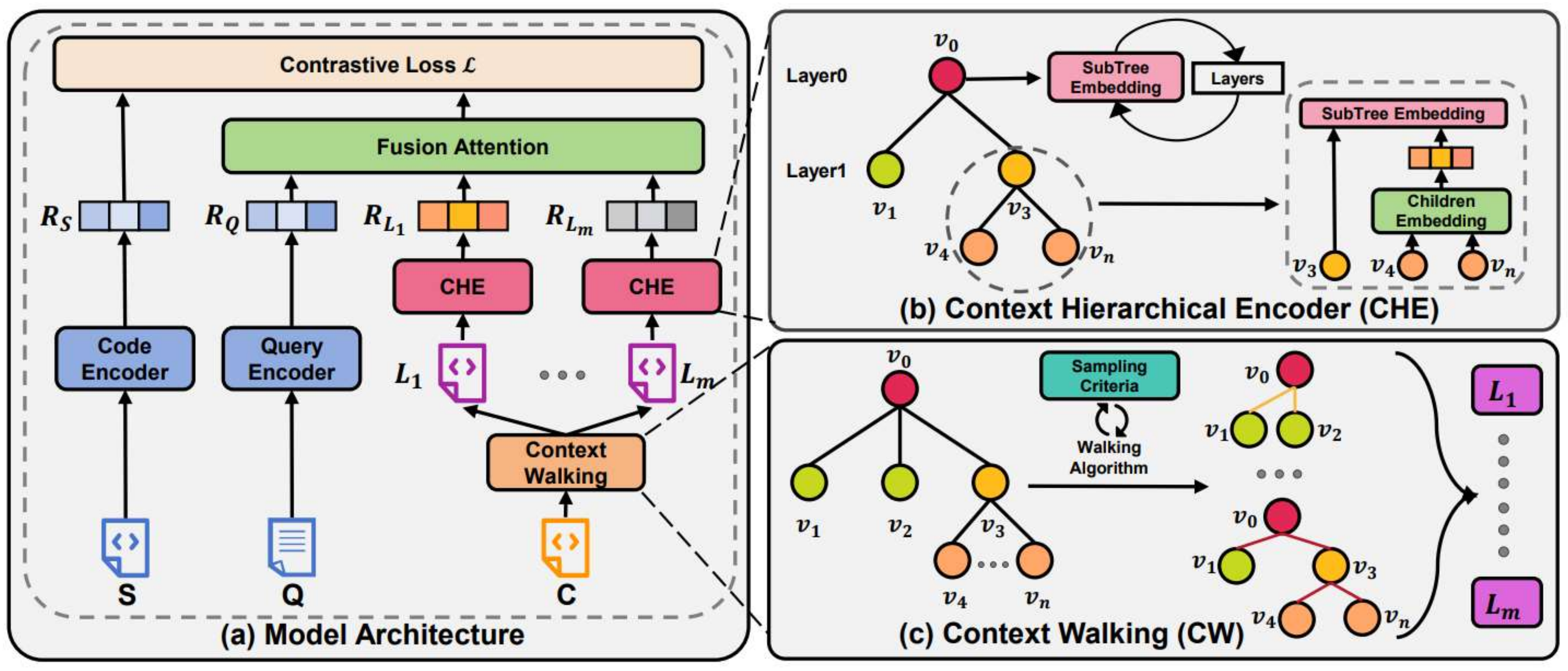


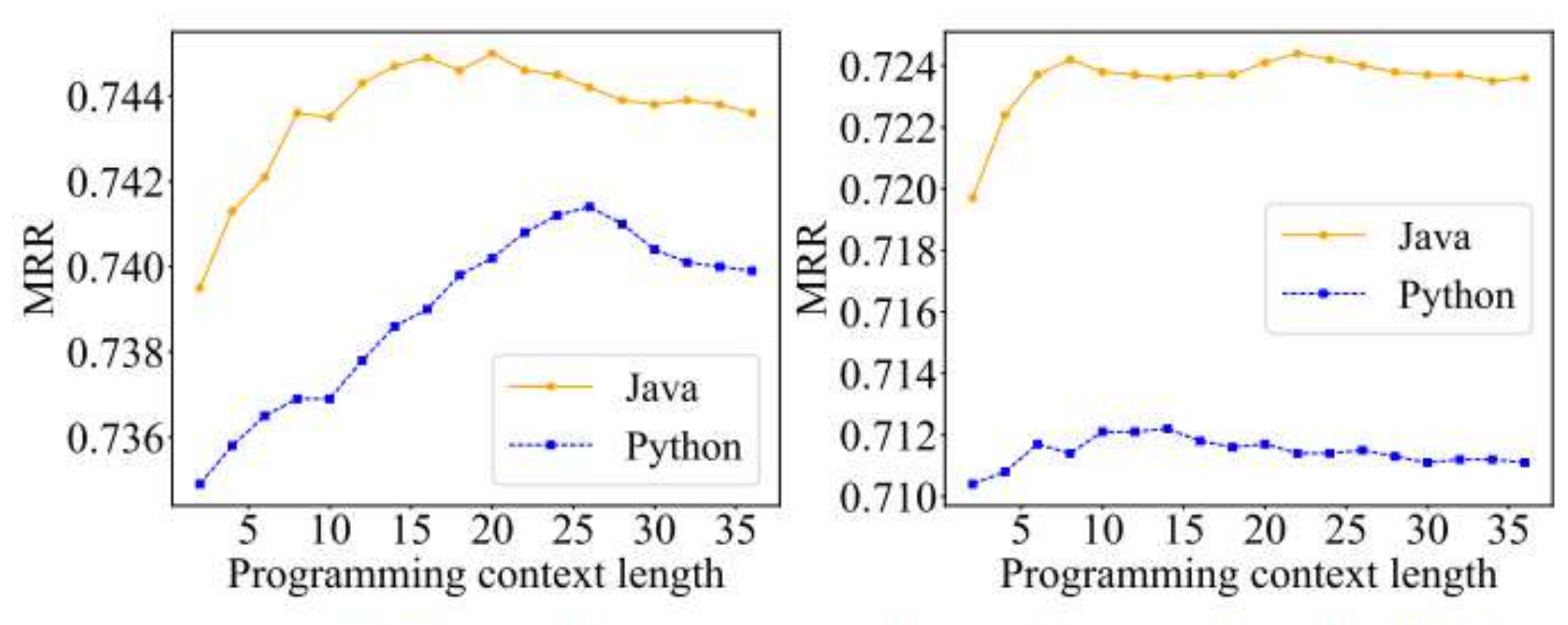
Fig3: The proposed ConCR framework. The left part (a) shows the architecture of ConCR, which consists of two stages: Context Walking and Context Hierarchical Encoder. The right part (b)(c) shows the details of these two stages, respectively.

- **Overview:** We propose a novel two-stage ConCR framework. Figure 3(a) shows the overall framework of ConCR. In the first stage, the Context Walking (CW) (in Figure 3(c)) samples the CHT, generating variable-length and unordered programming contexts to simulate different programming habits. In the second stage, the Context Hierarchical Encoder (CHE) (in Figure 3(b)) hierarchically model the sampled programming context from local to global to extract useful information for searching.
- **Context Walking:** Programming habits vary among developers, such as differences in the order of writing code and where they perform queries. For example, one developer might write "load_csv" before "process_data," while another does it in the opposite order. These variations result in different orders and lengths of programming context. To address this, we design a context walking algorithm that generates variable-length and unordered contexts to simulate different developers' habits, using two sampling criteria to ensure diversity and correctness.
- **Context Hierarchical Encoder:** The programming context has a hierarchical structure, with elements like classes and functions at different levels, and often includes irrelevant information. To address this, we introduce CHE, which models the context from local to global levels, mimicking how developers understand code. CHE uses two types of embeddings—node and subtree embeddings—to capture relevant information for search.

Experiments

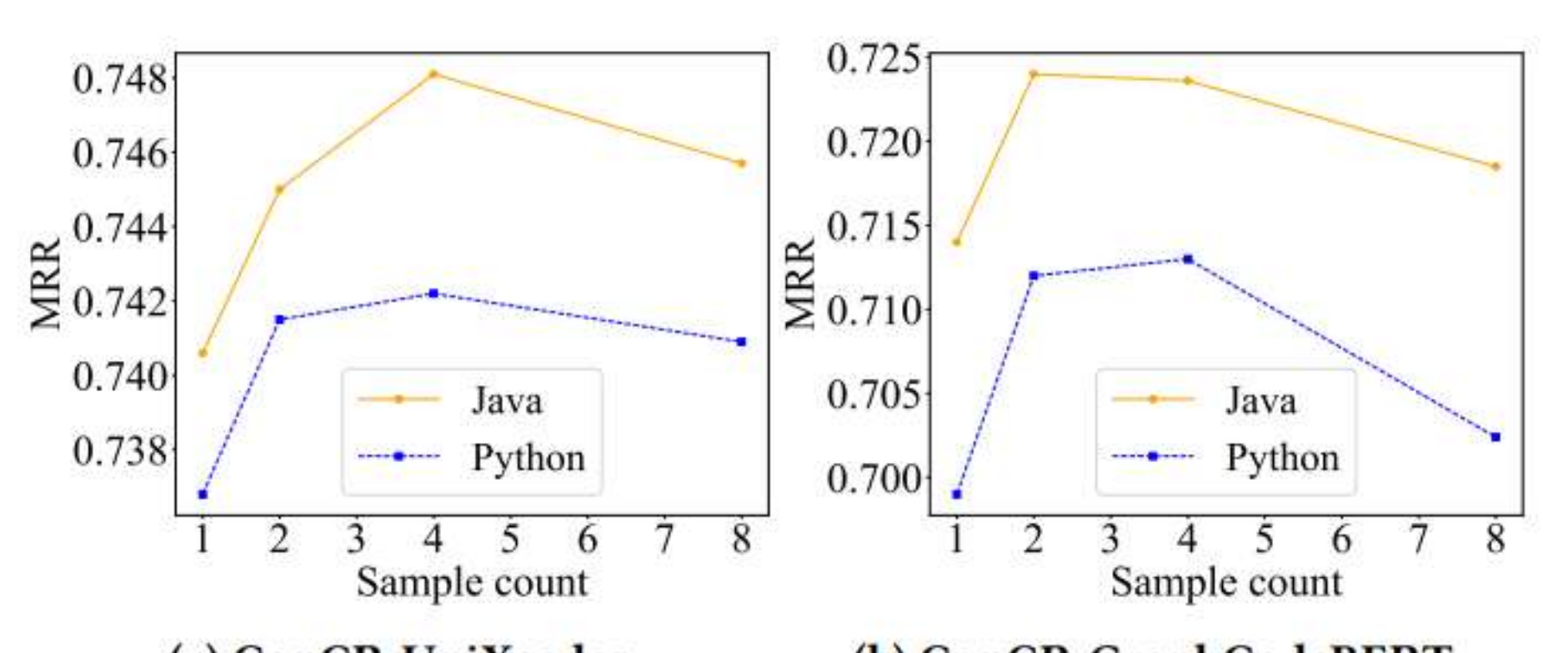
Table 2: Results on baseline and context-aware code search performance(* : $p < 0.05$ in the t-test).

Dataset	Java					Python				
Metrics	MRR	R@1	R@5	R@10	R@100	MRR	R@1	R@5	R@10	R@100
BM25	0.181	0.131	0.241	0.298	0.566	0.206	0.140	0.269	0.331	0.607
BiRNN	0.304	0.203	0.421	0.531	0.672	0.29	0.223	0.408	0.499	0.661
CodeBERT-bi	0.674	0.572	0.801	0.852	0.944	0.673	0.571	0.799	0.856	0.960
CodeBERT-bi-Context	0.689*	0.581*	0.819*	0.876*	0.961*	0.682*	0.578*	0.812*	0.868*	0.966*
ConCR-CodeBERT-bi	**0.701***	**0.600***	**0.832***	**0.882***	**0.962***	**0.690***	**0.585***	**0.819***	**0.875***	**0.968***
w/o CW	0.696	0.590	0.827	0.877	0.962	0.686	0.581	0.817	0.871	0.966
w/o CHE	0.689	0.581	0.819	0.876	0.961	0.682	0.578	0.812	0.868	0.966
GraphCodeBERT	0.687	0.589	0.808	0.858	0.946	0.694	0.594	0.821	0.874	0.966
GraphCodeBERT-Context	0.714*	0.614*	0.839*	0.891*	0.960*	0.699*	0.591*	0.832*	0.883*	0.972*
ConCR-GraphCodeBERT	**0.724***	**0.622***	**0.854***	**0.899***	**0.966***	**0.712***	**0.610***	**0.839***	**0.887***	**0.972***
w/o CW	0.721	0.618	0.850	0.894	0.965	0.706	0.605	0.831	0.883	0.972
w/o CHE	0.714	0.614	0.839	0.891	0.960	0.699	0.591	0.832	0.883	0.972
UniXcoder	0.726	0.634	0.842	0.885	0.956	0.722	0.624	0.843	0.893	0.974
UniXcoder-Context	0.732*	0.635*	0.854*	0.899*	0.967*	0.731*	0.627*	0.858*	0.905*	0.979*
ConCR-UniXcoder	**0.749***	**0.648***	**0.874***	**0.913***	**0.973***	**0.742***	**0.641***	**0.866***	**0.911***	**0.981***
w/o CW	0.741	0.639	0.871	0.912	0.973	0.737	0.639	0.856	0.905	0.980
w/o CHE	0.732	0.635	0.854	0.899	0.967	0.731	0.627	0.858	0.905	0.979



(a) ConCR-UniXcoder (b) ConCR-GraphCodeBERT

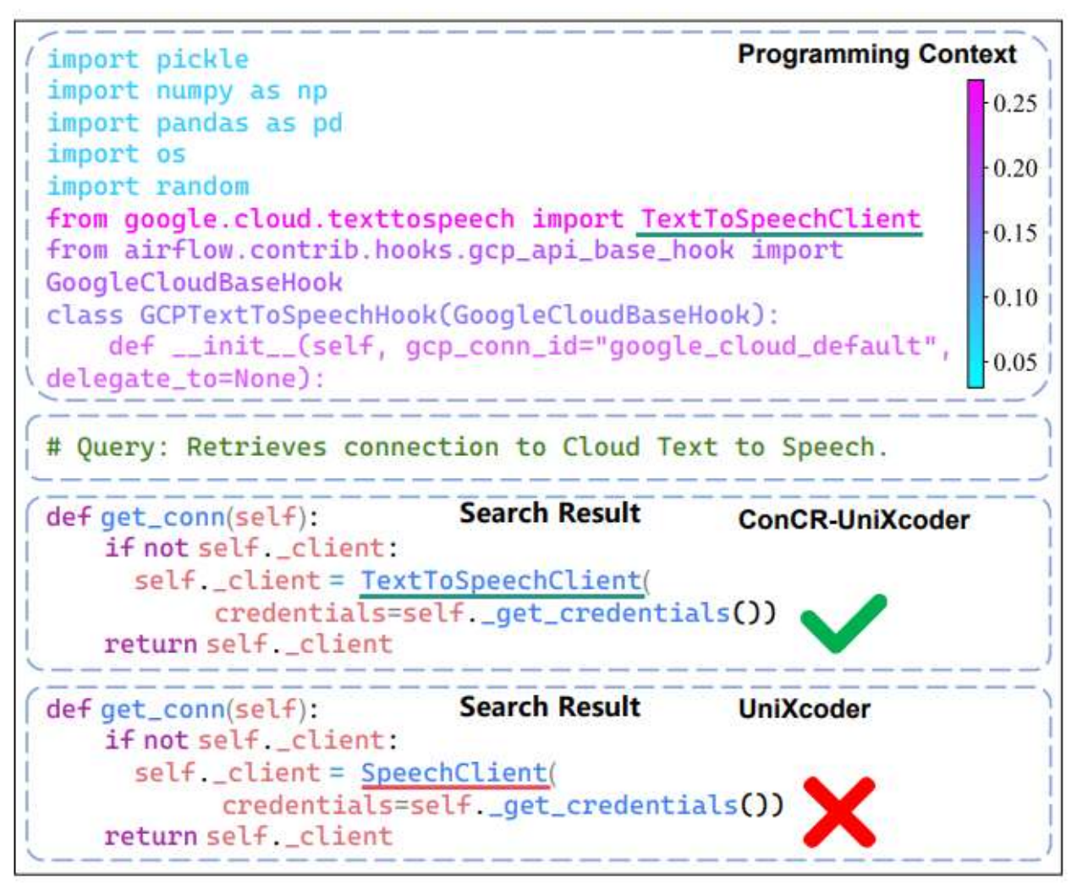
Figure 5: Performance over different context length.



(a) ConCR-UniXcoder (b) ConCR-GraphCodeBERT

Figure 6: Performance over different hyperparameter m.

- The results show that methods incorporating programming context (e.g., UniXcoder-Context, ConCR-UniXcoder) outperform traditional code search models, demonstrating the importance of capturing developers' search intent through context. Notably, the ConCR framework, which models context more effectively, yields better performance than methods that treat context as a single vector.
- Increasing the context length initially improves performance, but excessive length introduces redundancy that hampers effectiveness. Additionally, the model performs better on the Java dataset, Java's strongly typed nature provides clearer developer intent compared to Python's dynamic typing.
- Ablation experiments reveal that both CHE and CW components are essential for model performance, with the CHE component being particularly crucial for enhancing context modeling. Furthermore, the number of context samples m positively impacts performance up to a threshold, beyond which redundant information starts to degrade results.
- In the case study, ConCR effectively captures the developer's intent by leveraging context, shown through the attention mechanism, while UniXcoder, which does not consider context, retrieves results that do not align with the developer's true intent.



Programming Context

```
import pickle
import numpy as np
import pandas as pd
import os
import random
from google.cloud.texttospeech import TextToSpeechClient
from airflow.contrib.hooks.gcp_api_base_hook import
GoogleCloudBaseHook
class GCPTextToSpeechHook(GoogleCloudBaseHook):
    def __init__(self, gcp_conn_id="google_cloud_default",
delegate_to=None):
```

```
# Query: Retrieves connection to Cloud Text to Speech.
```

Search Result — ConCR-UniXcoder

```
def get_conn(self):
    if not self._client:
      self._client = TextToSpeechClient(
           credentials=self._get_credentials())
    return self._client
```

Search Result — UniXcoder

```
def get_conn(self):
    if not self._client:
      self._client = SpeechClient(
           credentials=self._get_credentials())
    return self._client
```

Figure 7: A case of ConCR and UniXcoder search results.