CHICOT: A Developer-Assistance Toolkit for Code Search with High-Level Contextual Information

Terufumi Morishita, Yuta Koreeda*, Atsuki Yamaguchi*, Gaku Morio, Osamu Imaichi, Yasuhiro Sogawa

Hitachi, Ltd. Research and Development Group
terufumi.mowirhita.wp@hitachi.com.

Abstract

We propose a source code search system named CHICOT (**C**ode search with **HI**gh level **CO**n**T**ext) to assist developers in reusing existing code. While previous studies have examined code search on the basis of code-level, fine-grained specifications such as functionality, logic, or implementation, CHICOT incorporates *high-level contextual information*, such as the purpose or domain of a developer's project. It achieves this feature by using context information from additionally extracted from codebases. It provides a VSCode plugin for daily coding assistance, and the built-in crawler ensures up-to-date code suggestions. The case study attests to the utility of CHICOT in real-world scenarios.

1 Introduction

Over the past two decades, large-scale codebases such as GitHub and BitBucket have emerged on the web. Reusing codes from such codebases significantly enhances the efficiency and quality of software development. When looking for codes, *semantic code search* is a key technique for finding the code snippets that best match a given natural language query.

Many studies have focused on searching for codes on the basis of code-level information, such as functionality, implementation, and logic (Li, Hu, and Peng 2020; Feng et al. 2020; Guo et al. 2020). Accordingly, queries have been framed to request such code-level specifics, e.g., "Estimate CO2 emissions." For example, a leading benchmark CodeSearchNet (Husain et al. 2019) aims to find a function from the GitHub that aligns with the functional specification.

However, in a real-world scenario of software development, not only code-level information but also the broader context is crucial. Since developers craft codes that reflect the specific purpose/domain of their projects, their queries should include this contextual information, e.g., "How can we estimate the CO2 emissions *from our company-operated flights?*" Accordingly, the returned codes must incorporate context-specific logic, such as the estimation of CO2 emissions *on the basis of flight attributes* like aircraft type/-model, flight distance/duration/altitude, and passenger and cargo load, rather than generic but useless logic based on "the type and amount of consumed fossil fuels."

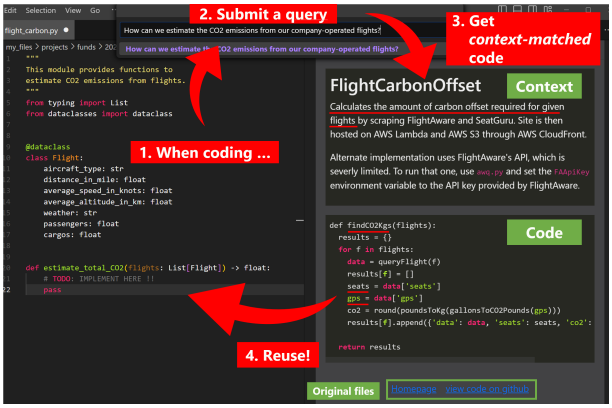

Figure 1: A CHICOT VSCode use case.

	Context	UI	Data
CodeSearchNet	–	–	up to 2018
CHICOT	✓	**VSCode / Browser**	**up to date**

Table 1: A comparison of CHICOT with the CodeSearchNet.

In light of this background, we propose CHICOT (*semantic **C**ode search with **HI**gh-level **CO**n**T**extual information*), a toolkit to find code snippets that best match the developer's unique context. CHICOT achieves this feature through two key mechanisms. (1) As preparation, it crawls functions from GitHub, each paired with its additionally extracted contextual information. (2) Upon receiving a query, it searches for a best matching *pair*, i.e., a function and its context. The architecture of CHICOT is depicted in Figure 2.

Further, to assist with everyday coding, we integrated CHICOT into VSCode as a plugin (Figure 1). CHICOT supports major languages, including Java, JavaScript, PHP, and Python. Its built-in crawler ensures up-to-date code suggestions. The contributions of CHICOT are outlined in Table 1.

*These authors contributed equally.

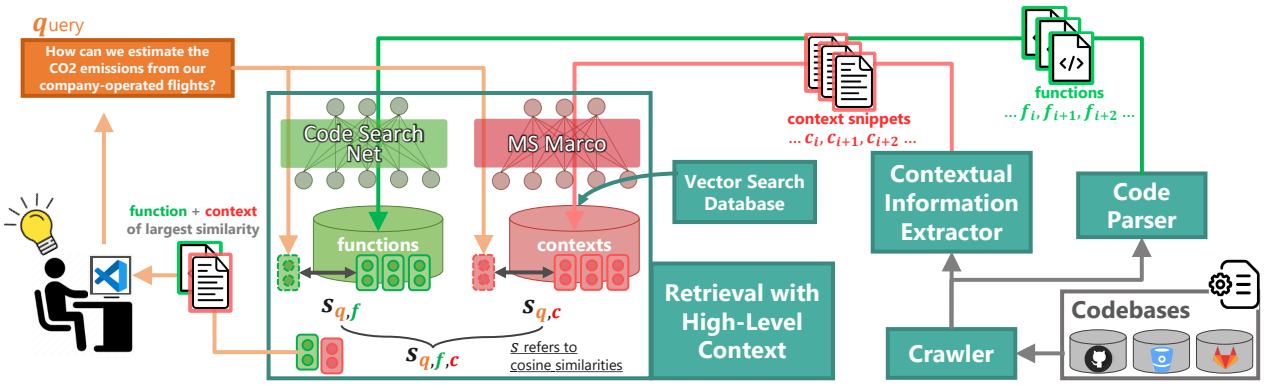

Figure 2: Overview of CHICOT. Starting from "Codebases", data flows from the right to the left. See Section 2 for details.

2 CHICOT Overview

Crawler collected 20,000 GitHub repositories with over 100 stars to ensure quality.

Code Parser extracts functions from each repository. Each function f_i is a parsed string, e.g., in Python, the string between "def my_function(...)" and "return my_value". Six million functions were collected in total.

Contextual Information Extractor extracts a piece of text that describes the context of each function f_i (*context snippet*, c_i). We utilize the repository's README for the context, as it typically outlines the project's purpose or domain. Note that a single README is shared by multiple functions. In processing a README, we first remove noise such as HTML and markdown tags using regular expressions, and then we apply a simple *LEAD-k* summarization ($k = 5$), extracting the first k lines to omit irrelevant information, such as installation instructions and release notes.

Retrieval with High-Level Context We now possess a collection of function and context snippet pairs $(f_1, c_1), \ldots, (f_N, c_N)$. In contrast to the previous systems that seek the function f_i aligning the best with a given query q, CHICOT aims to identify the optimal *pair* $p_j = (f_j, c_j)$.

We first divide this task into two subtasks: (I) determining a relevance score $s_{q,f}$ between q and f, and (II) $s_{q,c}$ between q and c. Then, the final score is derived as:

$$s_{q,f,c} = (1 - w) \times s_{q,f} + w \times s_{q,c}. \quad (1)$$

The pair $p_j = (f_j, c_j)$ of the highest score s_{q,f_j,c_j} is returned to the user. The advantage of this subtask-based formalization is that users can configure w depending on how much they value contextual information.

To calculate $s_{q,f}$ and $s_{q,c}$, we utilize cosine similarity by DNNs tailored for each subtask. Subtask (I) is the same as the previous code search *without* context, so we utilize a DNN trained on CodeSearchNet. Specifically, we use the lightweight model of (Wu and Yan 2022) to ensure compatibility with resource-constrained devices. Subtask (II) resembles the retrieval phase in question answering (QA) where a model calculates the relevance score between a question and a document written in natural language. Here, we adopt a compact language model[1] trained on MS-Marco (Nguyen et al. 2016), the largest QA dataset.

[1] "msmarco-distilbert-base-v4" (Reimers and Gurevych 2021).

A PHP 5.5+ tax management library.

```php
public function calculatePrices()
    {
        if ($this->getId()) {
            return;
        }
        $taxFactor = $this->getTaxRate() / 100;
        $total = $pretax = $tax = 0;
        $sums = [];
        foreach ($this->getProducts() as $produ
            $pPreTax = $product->getPrice();
            $pTax    = $pPreTax * $taxFactor;
```

(a) "Calculate prices of equipment in tax management systems"

Calculates the amount of carbon offset required for given flights by scraping FlightAware and SeatGuru. Site is then

```python
def findCO2Kgs(flights):
  results = {}
  for f in flights:
    data = queryFlight(f)
    results[f] = []
    seats = data['seats']
    gps = data['gps']
    co2 = round(poundsToKg(gallonsToCO2Pounds(gps)))
    results[f].append({'data': data, 'seats': seats, 'co2'
  return results
```

(b) "Estimate the amount of carbon offset for a flight"

Figure 3: For each query, we show a retrieved pair; an excerpt of the context snippet (upper) and the function (lower).

Vector Search Databases store vectorized items. We used Annoy (Bernhardsson 2018) for memory efficiency.

User Interface The VSCode plugin presents the top 20 retrieved pairs and links to the original repository for supplemental details. Users can configure parameters like w (1) in the settings tab. A web browser interface is also available.

3 Assessment of CHICOT

Precision at Ten To evaluate context-dependent search, we need a query that specifies *both* code- and context-level requirements. We manually crafted 25 such queries. We then assessed the top ten pairs retrieved for each query, checking whether *both* the function and context snippet align with the code- and context-level specifications, respectively. CHICOT achieved a precision@10 of 6.4%. Context-dependent code search is challenging due to the two alignment criteria, yet with over one accurate pair on the first page (default 20 pairs), CHICOT can be beneficial for developers.

Case Study Figure 3 shows that the retrieved pairs satisfy both the code- and context-level requirements of the query. Developers can easily reuse these items through copy-pasting, making modifications as needed.

Acknowledgements

We thank the three anonymous reviewers and the meta-reviewer, who gave us insightful comments and suggestions. We thank Dr. Masaaki Shimizu and Yasunori Kaneda at Hitachi for the convenience of additional computational resources. We thank Osamu Imaichi and Masashi Egi at Hitachi for insightful comments. We thank Dr. Naoaki Okazaki, professor at Tokyo Institute of Technology, for the keen comments. Computational resources of AI Bridging Cloud Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST) were used.

References

Bernhardsson, E. 2018. *Annoy: Approximate Nearest Neighbors in C++/Python*. Python package version 1.13.0.

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

Guo, D.; Ren, S.; Lu, S.; Feng, Z.; Tang, D.; Shujie, L.; Zhou, L.; Duan, N.; Svyatkovskiy, A.; Fu, S.; et al. 2020. GraphCodeBERT: Pre-training Code Representations with Data Flow. In *International Conference on Learning Representations*.

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*.

Li, R.; Hu, G.; and Peng, M. 2020. Hierarchical Embedding for Code Search in Software Q&A Sites. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–10.

Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; and Deng, L. 2016. MS MARCO: A human generated machine reading comprehension dataset. *choice*, 2640: 660.

Reimers, N.; and Gurevych, I. 2021. The Curse of Dense Low-Dimensional Information Retrieval for Large Index Sizes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 605–611. Online: Association for Computational Linguistics.

Wu, C.; and Yan, M. 2022. Learning Deep Semantic Model for Code Search using CodeSearchNet Corpus. *CoRR*, abs/2201.11313.