# Snip Finder
# Team Rocket

Fatih Ay
e273915@metu.edu.tr

Hasan Ege Şengül
e217202@metu.edu.tr

## Abstract

We propose a code snippet similarity search engine built on top of an information retrieval framework. The system allows users to input a small code fragment and retrieve the most similar code snippets from a large corpus. This functionality serves several use cases, such as plagiarism detection and code debugging assistance. For instance, users can paste a buggy piece of code and find related working examples to resolve issues. Our backend will be based on PyTerrier for fast retrieval and re-ranking.

## 1 Introduction

The goal of our project is to build a retrieval system capable of identifying code snippets that are semantically or syntactically similar to a given query snippet. Our system targets two main application scenarios: (1) academic plagiarism detection and (2) support for developers in locating related implementations to debug or extend their code. We will use the CodeSearchNet dataset, which provides code snippets in six major programming languages including Python, Java, and JavaScript. The dataset consists of over 6 million functions, making it suitable for large-scale retrieval experiments. For evaluation, we will use the accompanying challenge topics (queries) and qrels provided with ir-datasets. This dataset has 2.070.536 qrels and query except the challenge sub dataset.

| | Number of Functions | |
|---|---|---|
| | w/ documentation | All |
| Go | 347 789 | 726 768 |
| Java | 542 991 | 1 569 889 |
| JavaScript | 157 988 | 1 857 835 |
| PHP | 717 313 | 977 821 |
| Python | 503 502 | 1 156 085 |
| Ruby | 57 393 | 164 048 |
| All | 2 326 976 | 6 452 446 |

**Figure 1.** Dataset size statistics from the dataset paper.

## 2 Overview of the Proposed System

- Data Parsing and Indexing: We will use the CodeSearchNet dataset, which consists of over 6 million code snippets in six programming languages (Python, Java, JavaScript, Ruby, Go, and PHP). Before indexing, we will perform preprocessing steps including:
  Tokenization of code (coding languages exclusive)
  Removal of comments and non-informative tokens
  Stop word removal and stemming for natural language tokens (e.g., function names or docstrings)
  The indexing will be handled by PyTerrier, an extensible IR platform built on top of Lucene and Terrier. We will use: Inverted index (standard term-based) A field index for separating code tokens and docstrings (if applicable) and positional index
- Query processing: The system will allow users to input a code snippet as a query. Supported query format is raw code snippet queries (e.g., partial or buggy code) Initially, we will not support boolean or wildcard queries, but we may explore phrase or proximity queries depending on feasibility. Queries will be parsed using simple heuristics, and optionally passed through a syntax parser or embedding model if we use neural methods.
- Ranking and retrieval: Our baseline retrieval model will be BM25 implemented via PyTerrier. To enhance performance, we plan to implement: Dense vector re-ranking using code embeddings (e.g., CodeBERT, GraphCodeBERT)
  Optional use of pseudo-relevance feedback or cross-encoders for re-ranking top-k results
  Query-document similarity will be computed using cosine similarity in the embedding space
  This hybrid architecture will allow us to combine lexical and semantic similarities.
- User Interface: We will implement a simple and user-friendly web interface using Python Streamlit. The front-end will allow users to input a code snippet or a description and view the top-k most similar code.
- View source language, similarity score. If local resources are insufficient, we plan to deploy the model with Ray Framework as a REST API on a cloud platform (e.g., Hugging Face Spaces, Google Colab, or a lightweight cloud instance), with the Flask app consuming the results via API calls.