RETRIEVAL AUGMENTED CODE GENERATION AND SUMMARIZATION

LE MINH THANH TU[1]

[1] University of Information Technology - UIT

What ?

We propose a framework for automatic source code generation and summarization, in which we:

- Constructed a comprehensive legal dataset that consolidates various Vietnamese legal documents, including laws, decrees, and circulars
- Developed REDCODER, a hybrid system that combines Retrieval-Augmented Generation (RAG) with pre-trained models like PLBART and GraphCodeBERT to improve the accuracy and contextual relevance of generated code and summaries

Why ?

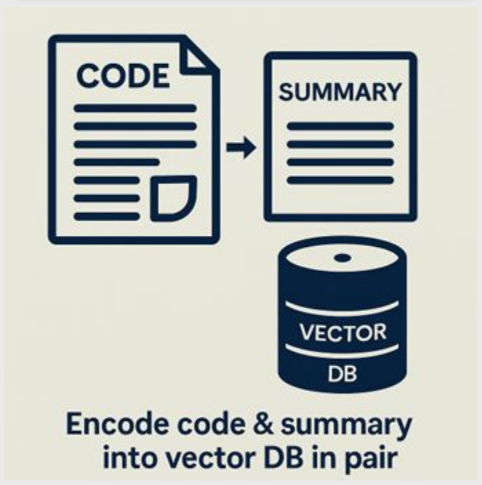
- Source code generation and summarization are essential in software development, aiding productivity and code comprehension. While developers frequently seek related examples during coding, most existing models depend solely on pre-trained architectures without leveraging external knowledge, limiting performance on complex tasks. By integrating retrieval with generation, our approach improves contextual understanding and delivers more accurate, developer-like outputs

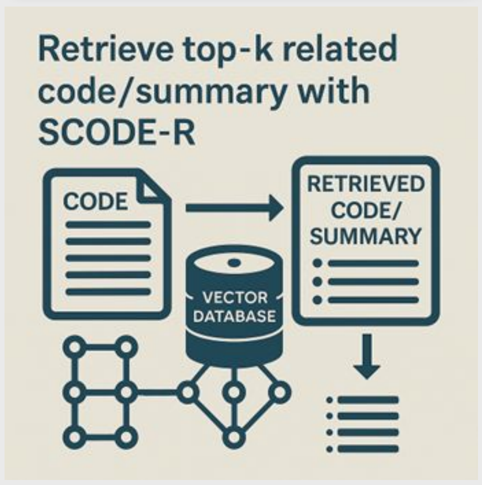
Overview

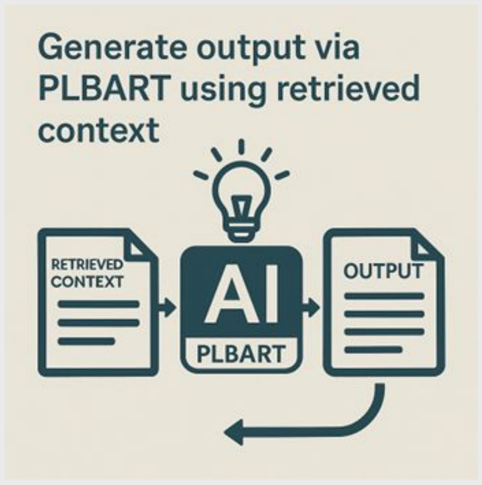
Indexing

Retrieval

Generation







Description

1. Indexing

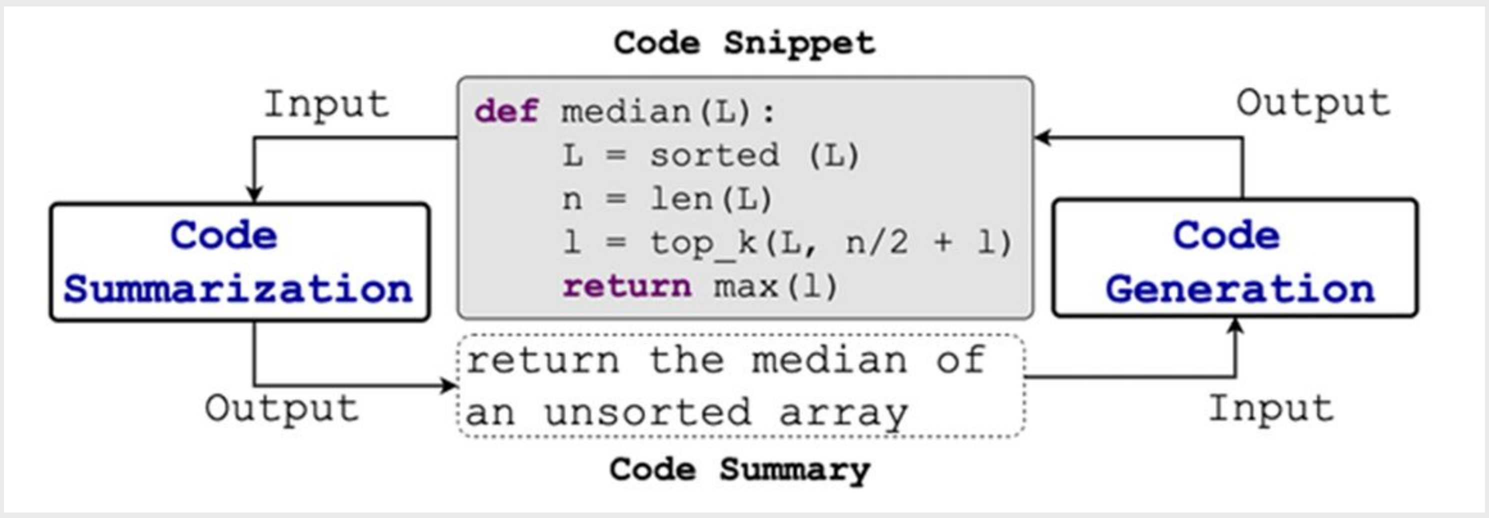
- Code and summary pairs, along with standalone entries, are encoded using pretrained models like CodeBERT and GraphCodeBERT, then indexed in a vector database. This setup enables REDCODER to efficiently retrieve semantically relevant context for both generation and summarization, reflecting real-world developer practices.

2. Retrieval

- Given a code snippet or summary as input, REDCODER uses SCODE-R—built on Dense Passage Retriever—to retrieve top-k relevant code or summaries from the vector database. It leverages pretrained encoders like CodeBERT and GraphCodeBERT to measure semantic similarity, enhancing input with meaningful context before generation.

3. Face-Track Matching

- The retrieved context is combined with the original input and passed to SCODE-G, a generation module based on PLBART. This enhanced input allows the model to generate accurate code or summaries, with support for both single and paired retrieval formats, improving fluency, correctness, and contextual relevance



```
def median(L):
    L = sorted (L)
    n = len(L)
    l = top_k(L, n/2 + 1)
    return max(l)
```

```
return the median of
an unsorted array
```

Figure 1. Example input/output for code generation and summarization tasks.

NII

Lê Minh Thanh Tú – Trường Đại học Công Nghệ Thông Tin (UIT)
TEL : 0911675681 Email : tulmt.18@grad.uit.edu.vn