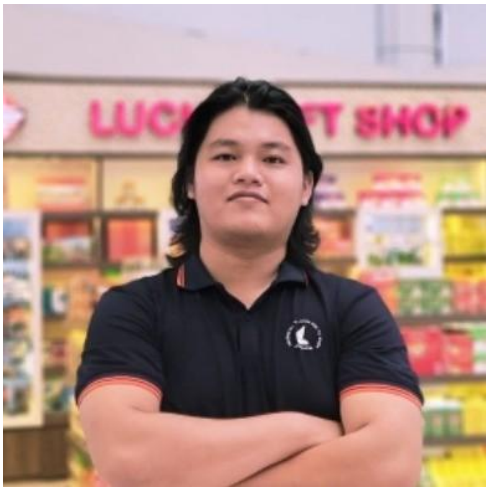
THÔNG TIN CHUNG CỦA NHÓM

- Link YouTube video của báo cáo (tối đa 5 phút):
  Phương pháp nghiên cứu khoa học - CS2205.CH190 - Lê Minh Thanh Tú
- Link slides (dạng .pdf đặt trên Github của nhóm):
  CS2205.CH190/CS2205_CH190_Slide.pdf at main · tulmt-work/CS2205.CH190
- *Mỗi thành viên của nhóm điền thông tin vào một dòng theo mẫu bên dưới*
- *Sau đó điền vào Đề cương nghiên cứu (tối đa 5 trang), rồi chọn Turn in*
- *Lớp Cao học, mỗi nhóm một thành viên*

- Họ và Tên: Lê Minh Thanh Tú
- MSSV: 230101032



- Lớp: CS2205.CH190
- Tự đánh giá (điểm tổng kết môn): 8.5/10
- Số buổi vắng: 2
- Số câu hỏi QT cá nhân: 3
- Số câu hỏi QT của cả nhóm: 3
- Link Github:
  https://github.com/tulmt-work/CS2205.CH190

ĐỀ CƯƠNG NGHIÊN CỨU

TÊN ĐỀ TÀI (IN HOA)

HỆ THỐNG TẠO SINH VÀ TÓM TẮT MÃ NGUỒN DỰA TRÊN TRUY XUẤT TĂNG CƯỜNG

TÊN ĐỀ TÀI TIẾNG ANH (IN HOA)

RETRIEVAL AUGMENTED CODE GENERATION AND SUMMARIZATION

TÓM TẮT

Đề tài "Retrieval Augmented Code Generation and Summarization" (Tạo sinh và tóm tắt mã nguồn dựa trên truy xuất tăng cường) tập trung giải quyết hai bài toán quan trọng trong xử lý ngôn ngữ tự nhiên: (1) tạo mã nguồn từ mô tả ngôn ngữ tự nhiên và (2) tóm tắt mã nguồn thành văn bản.

Phương pháp được đề xuất là xây dựng REDCODER – một framework kết hợp truy xuất thông tin và mô hình sinh ngôn ngữ nhằm nâng cao chất lượng đầu ra cho cả hai tác vụ. REDCODER gồm hai thành phần chính: bộ truy xuất (retriever) giúp tìm kiếm các đoạn mã hoặc tóm tắt liên quan và bộ tạo sinh (generator) sinh mã hoặc văn bản dựa trên kết quả truy xuất được.

Bộ dữ liệu huấn luyện và đánh giá đến từ các nguồn chất lượng cao như CodeSearchNet và CodeXGLUE, tập trung vào hai ngôn ngữ lập trình phổ biến là Java và Python. Ngoài ra, đề tài còn sử dụng bộ dữ liệu Concode cho bài toán sinh mã. Cơ sở dữ liệu truy xuất gồm hơn 1.4 triệu hàm và 1.1 triệu tóm tắt không trùng lặp, giúp tăng cường hiệu quả truy xuất và sinh kết quả.

Các mô hình mã hóa tiên tiến như CodeBERT, GraphCodeBERT và PLBART được áp dụng để đảm bảo độ chính xác và hiệu quả huấn luyện. Đề tài hướng đến hỗ trợ lập trình viên – đặc biệt là những người thường xuyên viết hoặc đọc hiểu mã – bằng cách tự động hóa việc viết mã và tài liệu kỹ thuật, từ đó giảm tải công việc thủ công và

tăng năng suất phát triển phần mềm.

Việc thiết kế hệ thống theo hướng tách biệt giữa truy xuất và tạo sinh cũng tạo điều kiện linh hoạt trong việc tích hợp với các mô hình khác trong tương lai.

Tóm lại, REDCODER là một hướng tiếp cận mới mẻ, hiệu quả và có tính ứng dụng cao trong việc kết hợp truy xuất tăng cường và sinh ngôn ngữ cho xử lý mã nguồn, hứa hẹn mang lại nhiều giá trị thực tiễn trong lĩnh vực phát triển phần mềm hiện đại.

GIỚI THIỆU

Trong những năm gần đây, các mô hình học sâu trong lĩnh vực Xử lý ngôn ngữ tự nhiên (NLP) đã thúc đẩy mạnh mẽ quá trình tự động hóa trong phát triển phần mềm, đặc biệt là hai tác vụ: tạo sinh mã nguồn từ mô tả tự nhiên và tóm tắt mã nguồn thành văn bản dễ hiểu. Tuy nhiên, các mô hình hiện tại thường thiếu khả năng khai thác bối cảnh bên ngoài, dẫn đến chất lượng đầu ra còn hạn chế, nhất là với các đoạn mã dài hoặc phức tạp.

Để khắc phục hạn chế này, đề tài đề xuất mô hình REDCODER – một framework kết hợp giữa truy xuất thông tin và mô hình ngôn ngữ sinh, nhằm cải thiện độ chính xác và tính liên kết ngữ cảnh của kết quả đầu ra. REDCODER gồm hai thành phần chính: SCODE-R (truy xuất) được xây dựng từ Dense Passage Retriever, dùng để tìm các đoạn mã hoặc tóm tắt liên quan; và SCODE-G (tạo sinh) dựa trên PLBART, sinh ra kết quả dựa trên đầu vào được tăng cường thông tin.

Bài toán gồm hai hướng:

- Input: mô tả ngôn ngữ tự nhiên → Output: đoạn mã nguồn tương ứng (code generation).
- Input: đoạn mã nguồn → Output: mô tả ngắn gọn, dễ hiểu bằng tiếng tự nhiên (code summarization).

Bằng cách kết hợp thông tin truy xuất vào quá trình tạo sinh, REDCODER không chỉ giúp cải thiện chất lượng kết quả mà còn phản ánh sát hơn hành vi thực tế của lập trình viên trong quá trình phát triển phần mềm.

MỤC TIÊU

Tạo ra một bộ dữ liệu liên quan đến mã nguồn/chú thích có chất lượng cao.

Xây dựng được một mô hình kết hợp giữa truy xuất thông tin và tạo sinh ngôn ngữ nhằm nâng cao hiệu quả cho hai tác vụ quan trọng trong phát triển phần mềm: tạo sinh mã nguồn từ mô tả tự nhiên và tóm tắt mã nguồn thành văn bản dễ hiểu.

Tạo ra một công cụ hỗ trợ đắc lực cho lập trình viên trong việc viết mã và đọc hiểu mã nguồn, góp phần giảm tải khối lượng công việc, tăng năng suất lập trình, và cải thiện khả năng duy trì tài liệu kỹ thuật.

NỘI DUNG VÀ PHƯƠNG PHÁP

1. Nội dung:

Dựa vào các mục tiêu đã đặt ra, đề tài sẽ được tiến hành thực hiện theo các nội dung sau:

- Xây dựng cơ sở dữ liệu mã nguồn và chú thích chất lượng cao:
    - Khai thác các bộ dữ liệu mã nguồn uy tín như CodeSearchNet, CodeXGLUE và Concode.
    - Tiến hành lọc nhiễu, loại bỏ trùng lặp và chuẩn hóa dữ liệu đầu vào bao gồm mã nguồn, chú thích và tóm tắt.
    - Xây dựng hai cơ sở dữ liệu riêng biệt: một cho mã nguồn và một cho tóm tắt, bao gồm cả dạng đơn (singleton) và cặp song song (pair).
- Phát triển hệ thống REDCODER:
    - Thành phần truy xuất SCODE-R:
        - Sử dụng mô hình Dense Passage Retriever (DPR) được tinh chỉnh dựa trên dữ liệu mã nguồn và chú thích.
        - Huấn luyện với phương pháp in-batch negatives, loại bỏ các mẫu tiêu cực “cứng.

        - Sử dụng encoder dựa trên các mô hình mạnh như CodeBERT và GraphCodeBERT.
    - Thành phần tạo sinh SCODE-G:
        - Áp dụng mô hình PLBART – mô hình seq2seq được tiền huấn luyện trên dữ liệu mã nguồn.
        - Kết hợp dữ liệu đầu vào với kết quả truy xuất để tạo thành đầu vào tăng cường cho quá trình sinh mã hoặc tóm tắt.
        - Hỗ trợ cả hai chế độ tăng cường: singleton và pair (REDCODER-EXT).
- Đánh giá mô hình:
    - Thực hiện đánh giá REDCODER trên hai tác vụ: tạo sinh và tóm tắt mã nguồn.
    - So sánh kết quả với các mô hình hiện đại hiện có như CodeGPT, GraphCodeBERT, GPT-2, RoBERTa,...

2. Phương pháp:

Ứng với từng nội dung trong đề tài, đề tài đã lựa chọn các phương pháp thực hiện phù hợp để đảm bảo độ chính xác, hiệu quả và tính linh hoạt của hệ thống:

- Phương pháp xây dựng mô hình REDCODER: REDCODER là một framework hai thành phần gồm truy xuất và tạo sinh. Phương pháp trung tâm là Retrieval-Augmented Generation (RAG), trong đó dữ liệu đầu vào được tăng cường bởi các thông tin truy xuất liên quan, nhằm cải thiện đầu ra cho cả hai tác vụ: tạo sinh và tóm tắt mã nguồn. Phương pháp này mô phỏng hành vi tự nhiên của lập trình viên – vốn thường tìm kiếm mã mẫu hoặc tài liệu trước khi viết mã hoặc hiểu mã nguồn.
- Truy xuất thông tin với SCODE-R: Thành phần SCODE-R được xây dựng dựa trên Dense Passage Retriever (DPR), sử dụng hai bộ mã hóa độc lập để mã hóa truy vấn và các đoạn mã/tóm tắt trong cơ sở dữ liệu. Mô hình được huấn luyện

lại với dữ liệu song song gồm mã nguồn và chú thích để đảm bảo độ chính xác cao trong việc truy xuất các tài liệu có liên quan.

- Sinh mã và tóm tắt với SCODE-G: Sử dụng PLBART – một mô hình Transformer dạng seq2seq đã được tiền huấn luyện trên tập lớn dữ liệu mã nguồn.
- Các kỹ thuật huấn luyện:
    - Huấn luyện mô hình SCODE-R bằng chiến lược in-batch negatives.
    - Tinh chỉnh mô hình PLBART bằng cách sử dụng cặp dữ liệu song song (code-summary).
    - Áp dụng kỹ thuật giới hạn độ dài đầu vào (512 tokens) để phù hợp với kiến trúc PLBART.
- Đánh giá mô hình: Các độ đo hiệu quả được áp dụng:
    - BLEU / BLEU-4: đánh giá chất lượng ngôn ngữ của đầu ra.
    - CodeBLEU: đánh giá độ chính xác cú pháp và ngữ nghĩa của mã nguồn.
    - Exact Match (EM): đo phần trăm kết quả đầu ra trùng khớp hoàn toàn với nhãn mục tiêu.
    - Mô hình cũng được so sánh với nhiều mô hình hiện đại khác như CodeGPT, GraphCodeBERT, RoBERTa để kiểm chứng hiệu quả vượt trội.

KẾT QUẢ MONG ĐỢI

- Xây dựng được một cơ sở dữ liệu mã nguồn và chú thích chất lượng cao, được xử lý và chuẩn hóa từ các nguồn uy tín như CodeSearchNet, CodeXGLUE và Concode, hỗ trợ tốt cho các tác vụ sinh và tóm tắt mã nguồn.
- Xây dựng được một hệ thống hỗ trợ sinh mã và tóm tắt mã nguồn thông minh, có khả năng kết hợp hiệu quả giữa truy xuất thông tin và mô hình ngôn ngữ lớn. Hệ thống này cho phép sinh mã hoặc tóm tắt kèm theo thông tin tham chiếu từ các ví dụ liên quan, giúp kết quả đầu ra rõ ràng, hợp ngữ cảnh và sát

với thực tiễn sử dụng của lập trình viên.

TÀI LIỆU THAM KHẢO *(Định dạng DBLP)*

[1]. Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

[2]. Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016. Deep api learning. In Proceed ings of the 2016 24th ACM SIGSOFT International Symposiumon Foundations of Software Engineering, pages 631–642.

[3]. Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A transformer-based ap proach for source code summarization. In Proceed ings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4998–5007, Online. Association for Computational Linguistics.

[4]. Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code generation as a dual task of code summariza tion. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Ad vances in Neural Information Processing Systems 32, pages 6563–6573. Curran Associates, Inc.