# **THÔNG TIN CHUNG CỦA NHÓM**

* Link YouTube video của báo cáo (tối đa 5 phút):   
  [*https://youtu.be/4gRbNdDxUm8*](https://youtu.be/4gRbNdDxUm8)
* Link slides (dạng .pdf đặt trên Github của nhóm):   
  [*https://github.com/nttrung309/CS519/blob/main/FinalReport.Slide.pdf*](https://github.com/nttrung309/CS519/blob/main/FinalReport.Slide.pdf)

| * Họ và Tên: Nguyễn Thành Trung * MSSV: 20520831 | * Lớp: CS519.O11 * Tự đánh giá (điểm tổng kết môn): 8.5/10 * Số buổi vắng: 1 * Số câu hỏi QT cá nhân: 14 * Số câu hỏi QT của cả nhóm: 6 * Link Github: https://github.com/nttrung309/CS519 * Mô tả công việc và đóng góp của cá nhân cho kết quả của nhóm:   + Lên ý tưởng đề tài   + Viết nội dung đề cương   + Làm video YouTube |
| --- | --- |

# **ĐỀ CƯƠNG NGHIÊN CỨU**

| **TÊN ĐỀ TÀI**  MÔ HÌNH TRANSFORMER CHO BÀI TOÁN TỰ SINH TESTCASE TRONG KIỂM THỬ PHẦN MỀM |
| --- |
| **TÊN ĐỀ TÀI TIẾNG ANH**  TRANSFORMER MODEL FOR AUTO-GENERATING TESTCASES IN SOFTWARE TESTING |
| **TÓM TẮT**  Trong đề tài này, chúng tôi sẽ thực hiện nghiên cứu chi tiết về việc tích hợp mô hình Transformer trong bài toán tự sinh test case trong kiểm thử phần mềm. Xuất phát từ thực tế ngày càng phức tạp của quá trình kiểm thử, chúng tôi nhận thức được tầm quan trọng của tự động hóa để giảm thiểu công sức và tăng hiệu quả. Kỹ thuật cụ thể mà chúng tôi áp dụng trong nghiên cứu là cơ chế “Attention” trong mô hình Transformer giúp mô hình tập trung vào các đoạn mã nguồn và thông tin từ tài liệu quan trọng, từ đó tăng cường khả năng tổng hợp thông tin và cải thiện chất lượng của các trường hợp kiểm thử được tạo ra. Đề tài của chúng tôi tập trung vào ba mục tiêu chính. Mục tiêu thứ nhất của đề tài là phát triển một hệ thống tự động hóa có khả năng tạo ra các trường hợp kiểm thử từ source code và tài liệu liên quan. Chúng tôi đặt ra bài toán làm thế nào có thể sử dụng mô hình Transformer và các kỹ thuật học máy để hiệu quả và tự động hóa quá trình này. Thứ hai là tối ưu hóa mô hình và quy trình tự động hóa để đảm bảo rằng trường hợp kiểm thử không chỉ đáp ứng yêu cầu chức năng mà còn tối ưu hóa thời gian và công sức kiểm thử. Cuối cùng là tích hợp liên tục và tự động hóa quy trình kiểm thử, tăng cường khả năng triển khai và kiểm thử liên tục trong quá trình phát triển phần mềm. |
| **GIỚI THIỆU**  Theo thống kê từ trang Global App Testing công bố vào năm 2023, có đến 2/3 công ty sử dụng tỷ lệ kiểm thử 75:25 (kiểm thử thủ công: tự động) hoặc 50:50. Điều đáng chú ý là chỉ có 9% hiện kiểm thử hoàn toàn thủ công. Trong quá trình phát triển phần mềm hiện đại, việc kiểm thử trở nên phức tạp, đòi hỏi sự áp dụng các phương pháp mới để đáp ứng sự phát triển nhanh chóng của ứng dụng và hệ thống phần mềm. Phương pháp truyền thống, mặc dù đóng góp lớn trong quá khứ, ngày càng gặp khó khăn trong việc đáp ứng tính đa dạng của quá trình phát triển. Từ dữ liệu thống kê chỉ ra rằng, trong những dự án phần mềm lớn, tự động hóa kiểm thử mang lại lợi ích đáng kể về thời gian và nguồn lực. Nó giúp giảm thiểu sai sót con người và tăng cường tính hiệu quả của quá trình kiểm thử. Từ đó, thấy được tầm quan trọng của bài toán tự động hóa kiểm thử trong phát triển phần mềm ngày nay.  Kiểm thử tự động là một phương pháp quan trọng trong quá trình phát triển phần mềm, nhằm tự động hóa việc thực hiện các bước kiểm thử mà trước đây thường phải thực hiện thủ công. Sự ra đời của kiểm thử tự động xuất phát từ nhu cầu giảm thiểu công sức con người, tăng tốc quá trình kiểm thử, và đảm bảo tính chính xác của quá trình kiểm thử. Tuy nhiên, mặc dù mang lại nhiều lợi ích nhưng vẫn còn những thách thức chưa được giải quyết. Phương pháp truyền thống thường gặp khó khăn khi đối mặt với các dự án lớn hay sự thay đổi nhanh chóng trong mã nguồn, dẫn đến sự không linh hoạt và chi phí cao trong việc duy trì hệ thống kiểm thử tự động.  Chính vì những thách thức này, đề tài nghiên cứu của chúng tôi hướng đến việc cải thiện khả năng tự động hóa trong quá trình tạo trường hợp kiểm thử. Bằng cách sử dụng trí tuệ nhân tạo, cụ thể là mô hình Transformer nổi bật với khả năng xử lý thông tin song song thông qua cơ chế self-attention, giúp cải thiện hiệu suất trong các nhiệm vụ xử lý ngôn ngữ tự nhiên mà cụ thể ở đây là mã nguồn và các tài liệu liên quan. Chúng tôi đặt ra mục tiêu không chỉ tự động hóa quá trình kiểm thử, mà còn làm cho nó thông minh, đáng tin cậy, và linh hoạt hơn đáp ứng với sự phát triển đa dạng và nhanh chóng của phần mềm hiện đại.  ***Input***: Mã nguồn dự án cần kiểm thử, tài liệu mô tả, tài liệu đặc tả yêu cầu.  ***Output***: Tài liệu hướng dẫn kiểm thử, báo cáo kiểm thử, danh sách lỗi. |
| **MỤC TIÊU**   * Mục tiêu chính của đề tài là phát triển một hệ thống tự động hóa có khả năng tạo ra các trường hợp kiểm thử từ mã nguồn và tài liệu liên quan. Chúng tôi đặt ra bài toán làm thế nào có thể sử dụng mô hình Transformer và các kỹ thuật học máy để hiệu quả và tự động hóa quá trình này. [3] * Tối ưu hóa mô hình và quy trình tự động hóa để đảm bảo rằng các testcase không chỉ đáp ứng yêu cầu chức năng mà còn tối ưu hóa thời gian và công sức kiểm thử. * Tích hợp liên tục và tự động hóa quy trình kiểm thử, tăng cường khả năng triển khai và kiểm thử liên tục trong quá trình phát triển phần mềm. |
| **NỘI DUNG VÀ PHƯƠNG PHÁP**  **a. NỘI DUNG**   * Tìm hiểu về các biến thể và cải tiến của mô hình Transformer trong ngữ cảnh kiểm thử phần mềm [1]. Phân tích chi tiết cơ chế Attention, bao gồm self-attention và multi-head attention. * Tìm hiểu về ứng dụng của xử lý ngôn ngữ tự nhiên trong việc hiểu và tổng hợp thông tin từ tài liệu yêu cầu và mã nguồn. Áp dụng các mô hình NLP như BERT để cải thiện khả năng hiểu biểu đồ ngôn ngữ và tăng cường khả năng sinh test case. * Nghiên cứu sâu rộng về các phương pháp tự sinh test case từ mã nguồn trong ngữ cảnh kiểm thử tự động. Xem xét cách tích hợp thông tin từ tài liệu đặc tả yêu cầu vào quá trình tạo testcase. [2] * Nghiên cứu về ứng dụng của kiểm thử dựa trên rủi ro để ưu tiên các trường hợp kiểm thử dựa trên mức độ rủi ro của chúng. Xây dựng một mô hình rủi ro tích hợp với hệ thống tự động hóa để tối ưu hóa quá trình kiểm thử. * Áp dụng kỹ thuật Fine-tuning để điều chỉnh mô hình Transformer cho ngữ cảnh kiểm thử phần mềm. Nghiên cứu về quy trình tự động hóa linh hoạt có thể thích nghi với sự thay đổi nhanh chóng trong mã nguồn. * Nghiên cứu về các kỹ thuật tích hợp liên tục tiên tiến, bao gồm "Containerization" và "Microservices architecture". Xem xét cách tự động hóa quy trình kiểm thử có thể tương tác với các công cụ CI/CD (Continuous Integration/Continuous Deployment). * Khảo sát các kỹ thuật Transfer Learning để chia sẻ kiến thức từ một dự án sang dự án khác. Nghiên cứu về các phương pháp Ensemble Learning để cải thiện độ chính xác của mô hình.   **b. PHƯƠNG PHÁP**   * Tiến hành đánh giá và so sánh giữa các biến thể của mô hình Transformer. Xem xét các công trình nghiên cứu về transfer learning và ensemble learning trong ngữ cảnh kiểm thử tự động. * Nghiên cứu về kỹ thuật Active Learning để mô hình có thể tận dụng sự tham gia của người kiểm thử để cải thiện độ chính xác. Khảo sát cách sử dụng học tăng cường để mô hình có thể tương tác và học từ môi trường kiểm thử thực tế. * Thực hiện thử nghiệm trên nhiều dự án phần mềm để đánh giá hiệu suất và sự chuyên sâu của mô hình. Áp dụng kỹ thuật fine-tuning và ensemble learning để cải thiện kết quả thử nghiệm. * Sử dụng kết quả thử nghiệm để tối ưu hóa mô hình và điều chỉnh quy trình tự động hóa. Áp dụng kỹ thuật auto-configuration để mô hình có thể thích ứng với môi trường và yêu cầu cụ thể. [5] * Áp dụng các kỹ thuật tích hợp liên tục tiên tiến và xây dựng quy trình tự động hóa linh hoạt và dễ mở rộng. Sử dụng công cụ CI/CD để kết nối mô hình và quy trình tự động hóa vào quy trình phát triển tự động. [4] |
| **KẾT QUẢ MONG ĐỢI**   * Tạo ra một hệ thống tự động hóa có khả năng sinh test case từ mã nguồn và tài liệu liên quan, giảm đáng kể công sức và thời gian kiểm thử. * Đạt được khả năng hiệu quả và linh hoạt thông qua sử dụng mô hình Transformer và kỹ thuật học máy. * Tối ưu hóa mô hình để đảm bảo rằng các test case đáp ứng yêu cầu chức năng, tiết kiệm thời gian và công sức kiểm thử. * Xây dựng quy trình tự động hóa linh hoạt, có khả năng thích nghi với sự biến động trong mã nguồn và yêu cầu chức năng. * Tích hợp liên tục quy trình kiểm thử vào quá trình phát triển phần mềm, giảm thiểu sai sót con người và tăng cường khả năng triển khai. Tạo ra một môi trường kiểm thử liên tục, đảm bảo tính nhất quán và chất lượng của phần mềm trong môi trường phát triển nhanh chóng. * Nâng cao khả năng hiểu và tổng hợp thông tin từ tài liệu yêu cầu và mã nguồn bằng cách sử dụng xử lý ngôn ngữ tự nhiên (NLP). * Cải thiện chất lượng test case sinh ra thông qua áp dụng mô hình Transformer trong xử lý ngôn ngữ tự nhiên. |
| **TÀI LIỆU THAM KHẢO** *(Định dạng DBLP)*  [1]. Yan Xiao, Xinyue Zuo, Lei Xue, Kailong Wang, Jin Song Dong, Ivan Beschastnikh:  Empirical Study on Transformer-based Techniques for Software Engineering. 2023: 1-10  [2]. Shaukat Ali; Lionel C. Briand; Hadi Hemmati; Rajwinder Kaur Panesar-Walawege:  A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. IEEE 2010: 1-50  [3]. Chengcheng Wan, Shicheng Liu, Sophie Xie, Yifan Liu, Henry Hoffmann, Michael Maire, Shan Lu:  Automated testing of software that uses machine learning APIs. ICSE '22: 1–20  [4]. Mitesh Soni:  End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery. IEEE 2015: 1-50  [5]. Julia Ling, Erin Antono, Saurabh Bajaj, Sean Paradiso, Maxwell Hutchinson, Bryce Meredig, Brenna M. Gibbons:  Machine Learning for Alloy Composition and Process Optimization. ASME 2018: 1-10 |