Sử dụng các mô hình được đào tạo trước để tăng cường tự động hóa đánh giá mã

Rosalia Tufano
SEART @ Software Institute
Università della Svizzera italiana
Thụy Sĩ

Simone Masiero
SEART @ Software Institute
Università della Svizzera italiana
Thụy Sĩ

Antonio Mastropaolo
SEART @ Software Institute
Università della Svizzera italiana
Thụy Sĩ

Luca Pascarella
SEART @ Software Institute
Università della Svizzera italiana
Thụy Sĩ

Denys Poshyvanyk
SEMERU @ Khoa Khoa học Máy tính
William và Mary
Hoa Kỳ

Gabriele Bavota
SEART @ Software Institute
Università della Svizzera italiana
Thụy Sĩ

TÓM TẮT

Rà soát mã là một thực tiễn được áp dụng rộng rãi trong các dự án mã nguồn mở và trong công nghiệp. Với chi phí không đáng kể của một quy trình như vậy, các nhà nghiên cứu bắt đầu điều tra khả năng tự động hóa các nhiệm vụ xem xét mã cụ thể. Gần đây, chúng tôi đã đề xuất các mô hình Deep Learning (DL) nhắm mục tiêu tự động hóa hai nhiệm vụ: mô hình đầu tiên lấy làm đầu vào cho một mã được gửi để xem xét và thực hiện trong đó các thay đổi có thể được khuyến nghị bởi người xem xét; mô hình thứ hai lấy làm đầu vào cho mã được gửi và nhận xét của người xem xét được đăng trong ngôn ngữ tự nhiên và tự động thực hiện thay đổi theo yêu cầu của người xem xét.Mặc dù kết quả sơ bộ mà chúng tôi đạt được là đáng khích lệ, cả hai mô hình đã được thử nghiệm trong các kịch bản xem xét mã khá đơn giản, về cơ bản đơn giản hóa vấn đề được nhắm mục tiêu.Điều này cũng là do những lựa chọn của chúng tôi khi thiết kế cả hai thí nghiệm công nghệ và thí nghiệm. Trong bài báo này, chúng tôi xây dựng trên cơ sở công việc đó bằng cách chứng minh rằng một mô hình Máy biến áp chuyển tiếp văn bản (T5) được đào tạo trước có thể vượt trội hơn các mô hình DL trước đó để tự động hóa các nhiệm vụ xem xét mã. Ngoài ra, chúng tôi đã tiến hành các thử nghiệm của mình trên một bộ dữ liệu lớn hơn và thực tế hơn (và đầy thách thức) về các hoạt động đánh giá mã.

Các từ khoá

Đánh giá mã, Nghiên cứu thực nghiệm, Học máy về mã

1. GIỚI THIỆU

Lợi ích của việc xem xét mã đã được công nhận rộng rãi, với một số nghiên cứu cung cấp bằng chứng về chất lượng cao hơn của mã được xem xét [15, 29, 31]. Ngoài ra, đánh giá mã giúp ngăn chặn lỗi và thúc đẩy chuyển giao kiến thức giữa các nhà phát triển [10, 40]. Tuy nhiên, các nghiên cứu về đánh giá mã cũng nhấn mạnh một chi phí bổ sung mà một quy trình như vậy đòi hỏi: Bằng chứng thực nghiệm cho thấy rằng các dự án phần mềm lớn có thể trải qua hàng trăm đánh giá mã mỗi tháng. Điều này áp dụng cho cả nguồn mở (ví dụ: 500 đánh giá mỗi tháng trong Linux [39]) và công nghiệp (ví dụ:3k đánh giá mỗi tháng trong các dự án Microsoft Bing [38]). Do đó, các nhà phát triển có thể dành nhiều giờ mỗi tuần để xem xét mã [16].

Với chi phí xem xét mã không đáng kể, gần đây chúng tôi đã đề xuất tự động hóa các nhiệm vụ xem xét mã cụ thể: Mục tiêu không phải là thay thế các nhà phát triển, mà là giúp họ tiết kiệm thời gian trong hai tình huống. Đầu tiên là người đóng góp (tức là nhà phát triển gửi mã để xem xét) muốn nhận phản hồi nhanh về mã họ đã viết trước khi gửi mã để xem xét. Phản hồi được cung cấp bởi một mô hình Deep Learning (DL) được đào tạo để lấy làm đầu vào cho mã để gửi để xem xét và cung cấp làm đầu ra cho một phiên bản sửa đổi của (tức là) thực hiện các thay đổi mã có khả năng được khuyến nghị bởi một người xem xét.

Kịch bản thứ hai liên quan đến (các) người đánh giá tham gia vào quy trình: một mô hình DL được đào tạo để lấy làm đầu vào (i) mã trình nộp để xem xét, và (ii) một nhận xét được viết bởi người xem xét bằng ngôn ngữ tự nhiên để yêu cầu một thay đổi cụ thể về . Kết quả đầu ra của mô hình là một phiên bản sửa đổi của (tức là,) thực hiện thay đổi được đề nghị trong . Ý tưởng ở đây là người đánh giá có thể sử dụng mô hình để cung cấp cho người đóng góp một ví dụ cụ thể của các thay đổi mã mà họ muốn thấy được thực hiện.

Trong công việc trước đây [46] chúng tôi đã đào tạo và thử nghiệm các mô hình DL trên một tập dữ liệu bao gồm bộ ba 17k, , được trích xuất từ các đánh giá mã được thực hiện trong GitHub [2] và Gerrit [1]. Cụ thể, mô hình khuyến nghị thay đổi mã cho bộ đóng góp là mô hình mã hóa-decoder với một bộ mã hóa lấy làm đầu vào và một bộ giải mã tạo ra . Đánh giá của chúng tôi cho thấy mô hình này có thể đề xuất thay đổi như một người đánh giá sẽ làm trong 3% (dự đoán đơn lẻ) đến 16% các trường hợp (10 dự đoán khác nhau).Mô hình được sử dụng trong kịch bản thứ hai (tức là, việc thực hiện tự động nhận xét được khuyến nghị bởi người đánh giá), thay vào đó có hai bộ mã hóa lấy đầu vào và , tương ứng, và một bộ giải mã tạo ra . Mô hình này có thể thực hiện thành công một thay đổi được khuyến nghị bởi một người đánh giá trong 12% (dự đoán đơn lẻ) đến 31% (10 dự đoán khác nhau) của các trường hợp.

Mặc dù những kết quả này đại diện cho bước đầu tiên của chúng tôi đối với các nhiệm vụ xem xét mã tự động, cách tiếp cận của chúng tôi [46] cũng như nghiên cứu thực nghiệm được tiến hành phải chịu một số hạn chế mà chúng tôi cố gắng khắc phục trong bài báo này. Đầu tiên, chúng tôi áp dụng một quy trình trừu tượng hóa mã để giảm kích thước từ vựng và đơn giản hóa việc học mô hình DL. Điều này có nghĩa là mô hình không hoạt động trên mã nguồn thô, nhưng trên một phiên bản trừu tượng của nó, trong đó, ví dụ, các định danh biến được thay thế bằng mã thông báo VAR_ID đặc biệt, trong đó ID là một số lũy tiến (ví dụ: biến được khai báo thứ hai được đại diện bởi VAR_2).Khả năng quay lại mã nguồn thô được đảm bảo bằng cách giữ một bản đồ liên kết được trừu tượng hóa với các mã thông báo thô trong (ví dụ: var_1 → i).

Mặc dù một quy trình như vậy đơn giản hóa việc học mô hình, nó đặt ra một giới hạn mạnh mẽ về nhiều nhiệm vụ xem xét mã có thể được hỗ trợ bởi mô hình như vậy. Thật vậy, quá trình trừu tượng hóa
buộc phải loại trừ khỏi bộ dữ liệu bộ ba , , tất cả những người giới thiệu định danh hoặc chữ không có mặt
in. Điều này là cần thiết vì bản đồ trừu tượng được xây dựng trên và, nếu một biến mới VAR_2 được giới thiệu trong quá trình xem xét, một biến như vậy không thể được ánh xạ trở lại mã nguồn thô, làm cho cách tiếp cận như vậy không thể sử dụng trong thực tế. Điều đó có nghĩa là
bộ ba , , mà chúng tôi đánh giá cách tiếp cận của chúng tôi [46] là những thay đổi tương đối đơn giản được thực hiện trong quá trình xem xét mã,
không yêu cầu giới thiệu bộ nhận dạng hoặc chữ viết mới. Thứ hai, để đơn giản hóa việc học, chúng tôi chỉ xem xét bộ ba , , trong đó cả mã được gửi để xem xét ( ) và mã sửa đổi ( ) có không quá 100 token [46]. Một lần nữa, điều này
giảm độ phức tạp của vấn đề được giải quyết.

Về cơ bản, hai lựa chọn trên dẫn đến việc đào tạo và trải nghiệm các mô hình được đề xuất về các trường hợp xem xét mã khá đơn giản chỉ đại diện cho một số ít các chuyển đổi mã được thực hiện trong quá trình xem xét mã.

Trong bài báo này, chúng tôi xây dựng dựa trên công việc trước đây của chúng tôi [46] với các mô hình DL để tự động hóa đánh giá mã trong các tình huống thực tế và thách thức hơn. Chúng tôi bắt đầu bằng cách đào tạo mô hình Máy biến áp chuyển tiếp văn bản (T5) được đề xuất gần đây [35] trên một tập dữ liệu tương tự như được sử dụng trong [46]. Tuy nhiên, chúng tôi áp dụng một trình giải nén (tức là, SentencePiece [26]) cho phép chúng tôi làm việc với mã nguồn thô, mà không cần trừu tượng hóa mã. Ngoài ra, chúng tôi cung cấp độ dài tối đa của các thành phần mã được xem xét từ 100 token “trừu tượng” đến 512 token “SentencePiece” (tức là, 390 token “trừu tượng”). Sự vắng mặt của một trừu tượng mecha-nism và tăng giới hạn trên cho đầu vào/đầu ra chiều dài al-lowed chúng tôi để xây dựng một bộ dữ liệu lớn hơn đáng kể so với bộ dữ liệu được sử dụng trong [46] (168k trường hợp so với 17k) và, quan trọng hơn, để nổi bật trong một tập dữ liệu như vậy, một loạt các chuyển đổi mã được thực hiện trong quá trình xem xét mã, bao gồm các trường hợp khá thách thức như những người yêu cầu giới thiệu bộ nhận dạng và chữ mới (chiếm 63% tập dữ liệu mới mà chúng tôi xây dựng). Ngoài ra, chúng tôi đã thử nghiệm với việc tự động hóa nhiệm vụ thứ ba liên quan đến quy trình xem xét mã: Với mã được gửi để xem xét ( ), tạo nhận xét bằng ngôn ngữ tự nhiên yêu cầu thay đổi mã người đóng góp như một người xem xét sẽ làm (ví dụ: mô phỏng một người xem xét nhận xét về mã đã gửi).

Chúng tôi cũng so sánh mô hình T5 với mô hình mã hóa - mã hóa được trình bày trong công việc trước đây của chúng tôi về bộ dữ liệu ban đầu được sử dụng trong [46]. Kết quả của chúng tôi cho thấy hiệu suất vượt trội của T5, thể hiện một bước tiến quan trọng trong việc tự động hóa các nhiệm vụ xem xét mã. Tóm lại, những đóng góp của công trình này là:

(i) Một cách tiếp cận mới để tự động hóa xem xét mã khắc phục một số hạn chế của kỹ thuật hiện đại [46];

(ii) Một đánh giá thực nghiệm toàn diện về cách tiếp cận như vậy, bao gồm so sánh với kỹ thuật trước đây của chúng tôi [46];

(iii) Tự động hóa nhiệm vụ thứ ba: Với mã được gửi để xem xét, tự động tạo nhận xét bằng ngôn ngữ tự nhiên yêu cầu thay đổi như người xem xét sẽ làm;

(iv) Một bộ dữ liệu đánh giá mã để đào tạo và kiểm tra các mô hình DL trong các tình huống thực tế hơn so với các mô hình được sử dụng trong [46];

(v) Một gói sao chép toàn diện [8].

2 T5 ĐỂ TỰ ĐỘNG XEM XÉT MÃ

Chúng tôi mô tả mô hình DL mà chúng tôi áp dụng, quá trình xây dựng các bộ dữ liệu cần thiết cho đào tạo của nó và quy trình được sử dụng để tìm kiếm siêu thông số, đào tạo mô hình và tạo dự đoán.

2.1 Máy biến áp chuyển văn bản sang văn bản tiếp theo (T5)

Máy biến áp chuyển văn bản sang văn bản tiếp theo, hoặc đơn giản là T5, không chỉ đơn thuần là một mô hình. Raffel và cộng sự [35] so sánh "mục tiêu đào tạo trước, cấu trúc, bộ dữ liệu không được gắn nhãn, phương pháp truyền tải và các yếu tố khác trên hàng chục nhiệm vụ hiểu ngôn ngữ".

Kết quả của việc thăm dò này là sự kết hợp tốt nhất giữa kiến thức và kỹ thuật đào tạo, cụ thể là T5. T5 dựa trên kiến trúc Transformer [48]. Việc triển khai được đề xuất chỉ khác nhau ở một số chi tiết (liên quan đến lớp chuẩn hóa và sơ đồ nhúng) so với biểu mẫu ban đầu của nó. Raffel et al. đã đề xuất một số phiên bản của T5, khác nhau về kích thước (ví dụ, số lớp) và kết quả là, độ phức tạp đào tạo. Trong công việc này, chúng tôi áp dụng phiên bản nhỏ của T5 bao gồm: chú ý 8 đầu, 6 lớp trong cả bộ mã hóa và bộ giải mã, mỗi lớp có kích thước 512 và kích thước đầu ra 2.048 (thông số 60M).

Mô hình phải trải qua khóa đào tạo đầu tiên (đào tạo trước) với mục đích cung cấp cho nó kiến thức chung hữu ích để giải quyết một tập hợp các nhiệm vụ liên quan. Ví dụ, giả sử chúng ta muốn đào tạo một mô hình có thể (i) dịch tiếng Anh sang tiếng Đức và (ii) tóm tắt văn bản tiếng Anh. Thay vì bắt đầu bằng cách đào tạo mô hình cho hai nhiệm vụ này, T5 có thể được đào tạo trước theo cách không có giám sát bằng cách sử dụng mục tiêu khử giám sát (hoặc mô hình ngôn ngữ che đậy): Mô hình được cung cấp với các câu có 15% mã thông báo của họ (ví dụ: từ trong câu tiếng Anh hoặc mã thông báo trong các tuyên bố Java) được che đậy một cách ngẫu nhiên và được yêu cầu dự đoán chúng. Bằng cách học cách dự đoán các token bị che đậy, mô hình có thể có được kiến thức chung về ngôn ngữ ưa thích. Trong ví dụ của chúng tôi, chúng tôi có thể đào tạo trước mô hình về câu tiếng Anh và tiếng Đức.

Sau khi được đào tạo trước, T5 được tinh chỉnh về các nhiệm vụ hạ nguồn theo cách có giám sát. Mỗi nhiệm vụ được xây dựng trong một "văn bản đến văn bản" for-mat (tức là, cả đầu vào và đầu ra của mô hình được biểu diễn dưới dạng văn bản). Ví dụ, đối với nhiệm vụ dịch một tập dữ liệu bao gồm các cặp câu tiếng Anh và tiếng Đức cho phép tinh chỉnh mô hình. Tương tự, nhiệm vụ tóm tắt yêu cầu văn bản tiếng Anh đầu vào và một bản tóm tắt tương ứng. Trong các phần tiếp theo, chúng tôi giải thích cách chúng tôi đào tạo trước và tinh chỉnh T5 để hỗ trợ các nhiệm vụ xem xét mã.

2.2 Dữ liệu đào tạo

Chúng tôi mô tả quy trình được sử dụng để xây dựng các bộ dữ liệu cần thiết cho việc đào tạo trước (Phần 2.2.1) và tinh chỉnh (Phần 2.2.2) của T5. Một phần của bộ dữ liệu tinh chỉnh đã được sử dụng để tìm kiếm siêu thông số (Phần 2.3) và để kiểm tra hiệu suất của T5 (Phần 3).

2.2.1 Tập dữ liệu đào tạo trước.

Với mục tiêu của giai đoạn đào tạo trước (tức là cung cấp cho mô hình kiến thức chung về các hướng dẫn chi tiết của các nhiệm vụ xuôi dòng), chúng tôi đã xây dựng một bộ dữ liệu cho phép đào tạo T5 về Java và tiếng Anh kỹ thuật.

Thật vậy, bên cạnh mã nguồn, tiếng Anh kỹ thuật là công cụ trong quá trình xem xét mã trong đó người xem xét đăng nhận xét bằng ngôn ngữ tự nhiên về mã.

Chúng tôi bắt đầu từ hai bộ dữ liệu có các phiên bản bao gồm cả mã nguồn và tiếng Anh kỹ thuật: kết xuất Stack Overflow chính thức (SOD) [7] và CodeSearchNet (CSN) [25]. Stack Overflow là một trang web Hỏi & Đáp dành cho các lập trình viên. Dữ liệu kết xuất mà chúng tôi sử dụng thu thập tất cả các câu hỏi và câu trả lời tương đối từ năm 2006 đến 2020 cho tổng số khoảng 51 TRIỆU bài đăng (trong đó một bài đăng là một câu hỏi hoặc câu trả lời duy nhất). Một bài đăng bao gồm văn bản tiếng Anh (theo hướng dẫn của SO) và/hoặc đoạn mã. Các bài đăng thường đi kèm với các thẻ đặc trưng cho chủ đề của họ (ví dụ: Java, Android) và có thể được xếp hạng với phiếu bầu lên/xuống và, đối với những gì liên quan đến câu trả lời, chúng có thể được đánh dấu là "câu trả lời được chấp nhận" từ câu hỏi của tác giả.

Chúng tôi trích xuất từ SOD tất cả các câu trả lời (i) có thẻ Java; (ii) chứa ít nhất một thẻ <pre><code> HTML để đảm bảo sự hiện diện của ít nhất một đoạn mã trong câu trả lời; và (iii) có ít nhất 5 phiếu tăng giá và/hoặc là câu trả lời được chấp nhận. Những bộ lọc này được chứng minh bởi mục tiêu đào tạo trước của chúng tôi. Thật vậy, chúng tôi muốn mô hình có được kiến thức về tiếng Anh kỹ thuật và Java: tập trung vào câu trả lời có chứa ít nhất một đoạn mã làm tăng cơ hội văn bản ngôn ngữ tự nhiên của họ đề cập đến một nhiệm vụ triển khai, tương tự như những gì xảy ra trong xem xét mã. Ngoài ra, bộ lọc phiếu tăng/trả lời được chấp nhận nhằm mục đích loại bỏ các phiên bản chất lượng thấp chứa, ví dụ, các giải pháp mã sai. Đây cũng là lý do tại sao chúng tôi tập trung vào các câu trả lời chất lượng cao có khả năng chứa các giải pháp làm việc hơn là các câu hỏi, ngay cả khi được bầu lên (ví dụ: vì chúng có liên quan đến nhiều người dùng) có thể kết luận các triển khai sai. Từ bước này, chúng tôi thu được 1.018.163 trường hợp ứng viên từ SOD.

Trên mỗi câu trả lời đã chọn, chúng tôi thực hiện các bước làm sạch sau: Chúng tôi xóa biểu tượng cảm xúc, ký tự không phải Latinh, ký tự điều khiển, khoảng trắng và nhiều khoảng trắng. Một số ký hiệu đặc biệt được thay thế bằng cách sử dụng các ký tự Latinh có cùng ý nghĩa, ví dụ: "≥" được thay thế bằng ">=". Hơn nữa, chúng tôi thay thế bất kỳ liên kết nhúng với một thẻ đặc biệt<LINK_i> "", với là một số nguyên nằm trong khoảng từ 0 đến − 1, trong đó là số lượng các liên kết trong . Cuối cùng, chúng tôi đã loại bỏ tất cả các trường hợp có ít hơn mười mã thông báo hoặc nhiều hơn 512 (40.491). Điều này để lại cho chúng tôi 977.379 trường hợp hợp hợp lệ.

Bộ dữ liệu Java CSN [25] có 1.5M phương pháp Java độc đáo, một số trong đó có Javadoc của họ. Chúng tôi đã lọc ra tất cả những trường hợp không có sẵn Javadoc hoặc không có bất kỳ chữ cái nào, loại bỏ 1.034.755 chữ cái trong số đó. Không giống như SOD, CSN có thể chứa các trường hợp trong đó "phần văn bản" (tức là, nhận xét phương pháp) không phải bằng tiếng Anh. Để giải quyết một phần vấn đề này, chúng tôi loại trừ các cặp không tìm thấy ký tự Latinh nào. Mặc dù điều này không loại trừ tất cả các nhận xét không phải bằng tiếng Anh, ít nhất xác định và loại bỏ những nhận xét được viết bằng các ngôn ngữ cụ thể (ví dụ: tiếng Nga, tiếng Trung) (15,229). Chúng tôi đã quyết định chấp nhận một số mức độ nhiễu trong tập dữ liệu trước khi đào tạo (ví dụ: nhận xét được viết bằng tiếng Pháp) vì (i) do kích thước của tập dữ liệu này, lượng nhiễu nhỏ này không ảnh hưởng đáng kể đến hiệu suất của mô hình và (ii) tập dữ liệu trước khi đào tạo không được sử dụng làm bộ kiểm tra để đánh giá hiệu suất của phương pháp tiếp cận. Như chúng tôi sẽ giải thích ở phần sau, việc làm sạch hạt mịn hơn đã được thực hiện cho bộ dữ liệu tinh chỉnh, thay vào đó, được sử dụng để đánh giá hiệu suất. Trong 519.905 trường hợp còn lại, chúng tôi đã thực hiện các bước làm sạch tương tự được mô tả cho SOD (ví dụ: loại bỏ biểu tượng cảm xúc). Cuối cùng, từ mỗi cặp, chúng tôi thu được một chuỗi duy nhất kết hợp nhận xét Javadoc và mã, giữ lại những người có nhiều hơn mười và ít hơn 512 token (còn lại 507.947 trường hợp).

Bằng cách kết hợp các phiên bản được thu thập từ SOD và CSN, chúng tôi đã thu được bộ dữ liệu trước khi đào tạo bao gồm 1.485.326 phiên bản. Để thực hiện đào tạo trước, chúng tôi ngẫu nhiên che mặt trong mỗi lượt 15% mã thông báo của nó. Các mã thông báo được che đậy được thay thế bằng mã thông báo canh<extra_id_i>, trong đó có số lượng ngày càng tăng từ 0 đến − 1, trong đó là số lượng mã thông báo được che đậy trong một trường hợp nhất định. Nếu một số token liền kề được che đậy, chúng sẽ được thay thế bằng một mã thông báo duy nhất. Các "trường hợp được che đậy" này đại diện cho đầu vào của mô hình trong quá trình đào tạo trước. Mục tiêu (tức là chuỗi mà mô hình dự kiến sẽ tạo ra) được xây dựng để tổng hợp các mã thông báo trọng tài và (các) mã thông báo mà chúng đang che đậy. Một mã thông báo canh bổ sung được thêm vào để chỉ ra sự kết thúc của chuỗi.

Bộ dữ liệu trước đào tạo của chúng tôi được cung cấp công khai [8].

2.2.2 Tinh chỉnh các Tập dữ liệu. Để tạo tập dữ liệu tinh chỉnh, chúng tôi đã khai thác các dự án mã nguồn mở Java từ GitHub bằng cách sử dụng bản sao ap web của Dabic et al. 19. Sử dụng giao diện truy vấn [5], chúng tôi đã chọn tất cả các dự án Java có ít nhất 50 yêu cầu kéo (PR), mười người đóng góp, mười ngôi sao và không phải là dĩa. Các bộ lọc nhằm mục đích (i) đảm bảo có đủ tài liệu "đánh giá mã" trong các dự án (tức là ít nhất 50 PR); (ii) loại bỏ các dự án cá nhân/đồ chơi (ít nhất mười người đóng góp và ngôi sao); và (iii) giảm cơ hội khai thác mã trùng lặp. Điều này dẫn đến danh sách 4.901 dự án. Chúng tôi cũng khai thác sáu cài đặt Gerrit [1] được sử dụng trong [46] có chứa dữ liệu xem xét mã về 6.388 dự án.

Từ cả bộ dữ liệu GitHub và Gerrit, chúng tôi trích xuất bộ ba < , , >, trong đó là một phương pháp được gửi để xem xét; là nhận xét của người xem xét duy nhất đề xuất thay đổi mã cho ; và là phiên bản sửa đổi của việc thực hiện Đề xuất Lưu ý rằng (i) chúng tôi chỉ tìm kiếm các PR được chấp nhận khi kết thúc đánh giá mã, vì chúng tôi muốn tìm hiểu cách để đề xuất những thay đổi, cuối cùng, có thể dẫn đến mã được coi là tốt từ quan điểm của người đánh giá; và (ii) một PR duy nhất trong GitHub và Gerrit có thể dẫn đến một số bộ ba cho tập dữ liệu của chúng tôi. Thật vậy, chúng tôi khai thác các vòng đánh giá khác nhau trong mỗi lần PR. Ví dụ: một phương pháp có thể được gửi để xem xét, nhận được nhận xét yêu cầu thay đổi (vòng đầu tiên). Phiên bản sửa đổi của địa chỉ sau đó được gửi lại ( ), dẫn đến vòng xem xét thứ hai (có thể dẫn đến các ý kiến bổ sung và sửa đổi phương pháp). Chúng tôi dừng lại khi mã được chính thức chấp nhận.

Nhìn chung, chúng tôi đã khai thác 382.955 bộ ba hợp lệ từ GitHub và Gerrit bằng cách sử dụng đường ống từ [46] mà chúng tôi tóm tắt trong phần sau (xem [46] để biết thêm chi tiết). Chúng tôi nhắm mục tiêu bộ ba trong đó một nhận xét

đã được người đánh giá đăng trên một phương thức . Chúng tôi có thể xác định các trường hợp này vì cả GitHub và Gerrit (i) đều cung cấp thông tin

về các nhà phát triển gửi mã và đăng nhận xét trong quá trình xem xét; và (ii) cho phép truy xuất mã cụ thể

(các) dòng đề cập đến (tức là mã trong đã được người đánh giá đánh dấu khi đăng nhận xét).

Chúng tôi loại trừ tất cả các nhận xét được đăng bởi các tác giả của mã (ví dụ: để trả lời người đánh giá), vì họ không đại diện cho việc đánh giá mã. Do đó, bộ ba trong bộ dữ liệu của chúng tôi có một nhận xét duy nhất được đăng bởi một người đánh giá. Ngoài ra, chúng tôi loại trừ liên kết với nhận xét nội tuyến (thay vì dòng mã) trong , vì chúng tôi nhắm mục tiêu sửa chữa

của các vấn đề liên quan đến mã. Để coi bộ ba là hợp lệ, phải là nhận xét duy nhất được người đánh giá đăng trong bài đánh giá cụ thể đó

vòng đàm phán

Bằng cách này, chúng ta có thể tự tin rằng phiên bản sửa đổi được gửi phụ sau này bởi tác giả ( ) thực sự nhằm thực hiện . Ngoài ra, phải khác với (tức là, một thay đổi phải được thực hiện trong mã để giải quyết). Từ quan điểm kỹ thuật, phân tích cú pháp của các phương pháp từ các bản vá lỗi được gửi cho đánh giá đã được thực hiện bằng thư viện thằn lằn [4]. Lưu ý rằng, việc loại bỏ bộ ba trong đó bao gồm nhiều hơn một nhận xét đã được thực hiện sau đó trong đường ống xử lý (chúng tôi sẽ quay lại điểm này). Thật vậy, trước khi chúng tôi phải làm sạch bình luận có thể chỉ đại diện cho tiếng ồn.

Như đã thực hiện đối với bộ dữ liệu trước khi đào tạo, chúng tôi đã thực hiện một số bước làm sạch. Chúng tôi đã thay thế bất kỳ liên kết nào bằng mã thông báo được đánh số<LINK_i>, với là một số nguyên nằm trong khoảng từ 0 đến − 1, trong đó là tổng số liên kết trong và . Nếu cùng một liên kết xuất hiện trong các phần khác nhau (ví dụ: trong và), nó được thay thế bằng cùng một mã thông báo. Chúng tôi cũng đã xóa bất kỳ ký tự biểu tượng cảm xúc và ký tự không cảm xúc nào khỏi các tập hợp, khoảng trống bổ sung và ký tự điều khiển khỏi cả nhận xét và phương pháp, và nhận xét nội tuyến khỏi các phương pháp (chúng tôi không quan tâm đến việc giải quyết các vấn đề liên quan đến nhận xét nội bộ). Sau quá trình làm sạch, chúng tôi đã thu được một số bộ ba trong đó trở thành một chuỗi trống hoặc ở đâu và trở nên bằng nhau (ví dụ: chúng chỉ khác nhau đối với một số khoảng trống trước khi làm sạch). Chúng ta loại bỏ các trường hợp này (-33,005) cũng như các trường hợp có + hoặc dài hơn 512 token (-61,233). Chúng tôi đã xem xét tổng của và về độ dài bởi vì, đối với một trong các nhiệm vụ (tức là, việc thực hiện tự động một nhận xét được đăng bởi người đánh giá), chúng sẽ được kết hợp để tạo thành đầu vào cho mô hình.

Sau đó, chúng tôi đã xóa khỏi bộ ba nhận xét không liên quan của mình (-28.581), tức là nhận xét không đề xuất đề xuất thay đổi mã (ví dụ: "có vẻ tốt với tôi"). Trong [46], chúng tôi chế tạo thủ công một tập hợp các mẫu ngôn ngữ tự nhiên để phát hiện các nhận xét không liên quan (ví dụ: nhận xét một từ có chứa các từ như “cảm ơn”, “tử tế”, v.v.). Chúng tôi đã mở rộng bộ này kể từ khi chúng tôi nhận thấy rằng trong tập dữ liệu phong phú hơn của chúng tôi một số nhận xét không liên quan đã được để lại bởi các mô hình này. Phân tích như vậy đã được thực hiện bởi một trong các tác giả bằng cách kiểm tra thủ công tất cả các bộ ba bao gồm ít hơn sáu từ. Các heuristics cập nhật có sẵn trong gói sao chép của chúng tôi [8].

Chúng tôi cũng loại trừ các bộ ba bao gồm các ý kiến không phải bằng tiếng Anh (-4,815) thông qua một đường ống dẫn bao gồm ba công cụ dò ngôn ngữ. Một phân loại sơ bộ đã được thực hiện bằng cách sử dụng các thư viện Python langdetect [3] và pycld3 [6]. Nếu cả hai công cụ này phân loại nhận xét là không phải tiếng Anh, chúng tôi đã dựa vào API phát hiện lan-guage của Google để đưa ra quyết định cuối cùng. Quá trình như vậy là cần thiết vì chúng tôi nhận thấy rằng API của Google là chính xác nhất trong việc loại bỏ ngôn ngữ, đặc biệt là khi các nhận xét cũng có các cấu trúc mã trong đó. Trong kịch bản này, các thư viện Python thường tạo ra âm sai (tức là, phân loại một câu tiếng Anh là không phải tiếng Anh). Tuy nhiên, chúng tôi có một số lượng hạn chế các yêu cầu có sẵn cho API Google. Do đó, chúng tôi đã thực hiện lọc trước bằng cách sử dụng thư viện Python và, khi cả hai đều báo cáo nhận xét không phải bằng tiếng Anh, chúng tôi đã kiểm tra lại bằng cách sử dụng Google API.

Sau quy trình vệ sinh này, chúng tôi đã loại trừ tất cả bộ ba có nhiều hơn một bình luận trong (-86.604). Cuối cùng, chúng tôi đã loại bỏ tất cả các bản sao từ tập dữ liệu tinh chỉnh (-918). Là người bảo thủ, chúng tôi xác định là trùng lặp hai bộ ba có cùng (do đó, thậm chí bộ ba có cùng/nhưng khác nhau đã được loại bỏ).

Bộ dữ liệu kết quả có 167.799 bộ ba đã được sử dụng để xây dựng ba bộ dữ liệu tinh chỉnh cần thiết cho ba nhiệm vụ mà chúng tôi hướng đến việc tự động hóa. Trong nhiệm vụ đầu tiên (từ mã đến mã), mô hình lấy làm đầu vào với mục tiêu tự động tạo phiên bản sửa đổi của nó, thực hiện các thay đổi mã có thể được yêu cầu trong quá trình xem xét mã. Do đó, tập dữ liệu tinh chỉnh được thể hiện bằng các cặp→.

Trong tác vụ thứ hai (code&comment-to-code) mô hình có dạng đầu vào và một bình luận được đăng bởi người xem xét và nhắm mục tiêu tạo ra , phiên bản sửa đổi của việc thực hiện thay đổi mã được đề nghị trong .

Mã chứa hai thẻ đặc biệt<START>, <END> đánh dấu phần của mã đề cập đến. Tập dữ liệu tinh chỉnh của tác vụ thứ hai này được thể hiện bằng các cặp < , >→.

Cuối cùng, trong nhiệm vụ thứ ba (code-to-comment) mô hình lấy làm đầu vào và nhằm mục đích tạo ra một nhận xét ngôn ngữ tự nhiên ( ) đề xuất thay đổi mã như một người đánh giá sẽ làm. Tập dữ liệu tinh chỉnh được thể hiện bằng các cặp→.

Bảng 1: Tập dữ liệu trước khi đào tạo và tinh chỉnh (# instance)

Tập dữ liệu	xe kéo	ước lượng	kiểm tra
Trước tập huấn			
Trần chông CodeSearchNet	977.379	-	-
	507.947	-	-
sự điều chỉnh tinh tế	134.239	16.780	16.780

Cả ba bộ dữ liệu tinh chỉnh đã được chia thành 80% đào tạo, 10% đánh giá và 10% kiểm tra. Bảng 1 tóm tắt số lần trong các bộ dữ liệu: Việc đào tạo trước chỉ được sử dụng để đào tạo, trong khi các bộ dữ liệu tinh chỉnh được khai thác cũng để điều chỉnh siêu máy đo (đánh giá) và để đánh giá hiệu suất của mô hình (kiểm tra). Trong Bảng 1, chúng tôi chỉ báo cáo thông tin cho một tập dữ liệu tinh chỉnh duy nhất (thay vì cho ba tập dữ liệu được mô tả trước đó), vì cả ba tập dữ liệu tinh chỉnh đều chứa cùng một số trường hợp. Thật vậy, tất cả chúng đều bắt nguồn từ cùng một bộ ba.

2.3 Đào tạo và Tìm kiếm siêu thông số

Raffel et al. [35] đã chỉ ra vai trò chính của việc đào tạo trước đối với hiệu suất của các mô hình T5. Tầm quan trọng của việc đào tạo trước cũng đã được xác nhận (đối với các mô hình dựa trên Transformer khác) trong bối cảnh các nhiệm vụ liên quan đến mã như tạo trường hợp thử nghiệm [44]. Để nghiên cứu thêm về khía cạnh này, chúng tôi đã quyết định thử nghiệm với cả mô hình được đào tạo trước và mô hình không được đào tạo trước, cả hai đều phải tuân theo quy trình điều chỉnh siêu thông số.

Vì chúng tôi đã áp dụng phiên bản nhỏ của T5 được trình bày bởi Raffel et al. [35], chúng tôi đã không thử nghiệm với các biến thể liên quan đến kiến trúc của nó (ví dụ: thay đổi số lớp hoặc số đơn vị ẩn). Mặc dù, như được thực hiện bởi Mastropaolo và cộng sự [28], chúng tôi đã thử nghiệm với các cấu hình tỷ lệ học tập khác nhau: (i) Tỷ lệ học tập tốn kém (C-LR), trong đó giá trị tỷ lệ học tập được cố định trong quá trình đào tạo; (ii) Tỷ lệ học tập căn bậc hai nghịch đảo (ISR-LR), trong đó giá trị tỷ lệ học tập phân rã thành căn bậc hai nghịch đảo của bước đào tạo; (iii) Tỷ lệ học tập tam giác nghiêng (ST-LR), trong đó tỷ lệ học tập đầu tiên tăng tuyến tính và sau đó nó phân rã tuyến tính trở lại giá trị bắt đầu; (iv) Tỷ lệ học tập phân rã đa thức (PD-LR), trong đó tỷ lệ học tập phân rã đa thức thành một giá trị cố định trong một số bước nhất định.

Việc điều chỉnh siêu thông số chỉ được thực hiện cho giai đoạn tinh chỉnh. Thật vậy, mặc dù chúng ta chỉ tập trung vào một siêu máy đo, một quá trình như vậy vẫn còn khá tốn kém, đòi hỏi phải đào tạo tám mô hình T5 khác nhau (tức là đào tạo trước và không đào tạo trước mỗi mô hình với bốn tỷ lệ học tập khác nhau).

Đối với đào tạo trước, chúng tôi sử dụng cùng một cấu hình được đề xuất bởi Raffel et al. trong [35]. Chúng tôi đã đào tạo trước mô hình trên bộ dữ liệu đào tạo trước (Bảng 1) cho 200k bước (34 kỷ nguyên). Bắt đầu từ mô hình được đào tạo trước, chúng tôi đã tinh chỉnh cho 75k bước bốn mô hình khác nhau, mỗi mô hình sử dụng một trong những tỷ lệ học tập thử nghiệm.

Vì mục tiêu của quy trình này là tìm ra tỷ lệ học tập tốt nhất cho ba nhiệm vụ đánh giá mã, chúng tôi đã tinh chỉnh từng mô hình này bằng cách sử dụng sự kết hợp của ba nhiệm vụ: Một mô hình duy nhất được đào tạo để hỗ trợ cả ba nhiệm vụ bằng cách sử dụng sự kết hợp của các bộ đào tạo của họ. Đây là một trong những đặc điểm của T5, khả năng đào tạo một mô hình duy nhất cho nhiều nhiệm vụ. Cách tiếp cận tương tự đã được sử dụng cho mô hình không được đào tạo trước: Trong trường hợp này, bốn mô hình T5 (một cho mỗi tỷ lệ học tập) đã được tinh chỉnh trực tiếp.

Chúng tôi đánh giá hiệu suất của tám mô hình trên bộ đánh giá của mỗi nhiệm vụ về "dự đoán hoàn hảo", cụ thể là các trường hợp trong đó đầu ra được tạo ra giống hệt với chuỗi mục tiêu (dự kiến). Bảng 2 báo cáo kết quả đạt được. Như vậy có thể thấy, không có tỷ lệ học tập nào đạt được kết quả tốt nhất trong tất cả các nhiệm vụ. Tuy nhiên, ST-LR cho thấy hiệu suất tổng thể tốt hơn và, vì lý do này, là phương pháp chúng tôi áp dụng trong các thử nghiệm của mình.

Bảng 2: Kết quả điều chỉnh siêu thông số

Nhiệm vụ	Chiến lược tỷ lệ học tập			
	C-LR	ISR-LR	ST-LR	PD-LR
	Được đào tạo trước			
mã hóa thành mã & Ma˜ sô´ đê´ n ma˜ sô´	|||UNTRANSLATED_CONTENT_START|||2.68% |||UNTRANSLATED_CONTENT_END||| 10,39%	3.68 9,23%	4,64 8,46%	2,53 9,89%
mã để ghi chú	0.15%	0.32	0.60	0.15%
	Không được đào tạo trước			
mã hóa thành mã ma˜ & tơ´ i ma˜ sô´	1.23 5,05	3.71 6,41	4.16 −6,24	1.22 5.18
mã để ghi chú	0.09	0.44	0.49%	0.03

Với cấu hình tốt nhất cho cả các mô hình được đào tạo trước và các mô hình không được đào tạo trước, chúng tôi đã tinh chỉnh chúng tối đa 300k bước bằng cách sử dụng chiến lược dừng sớm. Điều này có nghĩa là chúng tôi đã lưu một điểm kiểm tra của mô hình cứ sau 10k bước tính toán hiệu suất của nó về "dự đoán hoàn hảo" trên bộ đánh giá và dừng đào tạo nếu hiệu suất của mô hình không tăng cho ba điểm kiểm tra liên tiếp (để tránh quá tải).

2.4 Tạo dự đoán

Một khi các mô hình được đào tạo, chúng có thể được sử dụng để tạo ra các mối quan tâm. Như đã thực hiện trong công việc trước đây, chúng tôi áp dụng chiến lược tìm kiếm chùm tia [36] để tạo ra nhiều dự đoán cho một đầu vào duy nhất. Đối với ví dụ cũ, trong trường hợp nhiệm vụ mã hóa, đối với một phương pháp được cung cấp dưới dạng đầu vào nhiều ứng viên có thể được tạo ra. Khi chúng tôi yêu cầu mô hình tạo ra các dự đoán, nó tạo ra các chuỗi token có khả năng nhất cho chuỗi đầu vào; được gọi là kích thước chùm tia và chúng tôi thử nghiệm với = 1, 3, 5, 10.

Đối với mỗi dự đoán được tạo ra bởi T5, chúng tôi cũng khai thác chức năng điểm số của nó để đánh giá sự tự tin của mô hình về đầu vào được cung cấp.

Giá trị được trả về bởi hàm này dao động từ trừ vô cực đến 0 và nó là log-likalytics ( ) của dự đoán. Do đó, nếu nó là 0, nó có nghĩa là khả năng dự đoán là 1 (tức là độ tin cậy tối đa, kể từ (1) = 0), trong khi khi khi nó đi đến trừ vô cùng, độ tin cậy có xu hướng là 0. Trong nghiên cứu thực nghiệm của chúng tôi (Phần 3), chúng tôi đánh giá độ tin cậy của mức độ tin cậy như một đại diện cho chất lượng của các dự đoán.

3 Thiết kế nghiên cứu

Mục tiêu của đánh giá của chúng tôi là đánh giá thực nghiệm hiệu suất của mô hình T5 trong các nhiệm vụ tự động hóa xem xét mã. Các danh mục ngữ cảnh của (i) các bộ dữ liệu mà chúng tôi đã trình bày trong Phần 2; và (ii) bộ dữ liệu từ công việc trước đây của chúng tôi [46]. Từ bây giờ, chúng tôi đề cập đến cách tiếp cận được trình bày trước đây của chúng tôi như là đường cơ sở. Nghiên cư´ u na` y nhă` m gia˰ i quyê´ t năm câu ho˰ i nghiên cư´ u (RQ).

RQ1: T5 có thể tự động đề xuất thay đổi mã cho các nhà phát triển như những người đánh giá sẽ làm ở mức độ nào? Chúng tôi cung cấp đầu vào cho T5 một phương pháp Java được gửi để xem xét và đánh giá mức độ mà mô hình có thể cung cấp đầu ra như một phiên bản sửa đổi của ( ) thực hiện các thay đổi mã có thể sẽ được yêu cầu trong quá trình xem xét mã. Ý tưởng ở đây là một mô hình như vậy có thể được sử dụng trước khi mã được gửi để xem xét dưới dạng kiểm tra tự động cho người đóng góp.

RQ2: T5 có thể tự động thực hiện các thay đổi mã theo khuyến nghị của người đánh giá ở mức độ nào? Được cung cấp một phương pháp Java
đã gửi để xem xét ( ) và nhận xét ngôn ngữ tự nhiên ( ) trong đó người xem xét yêu cầu thực hiện các thay đổi mã cụ thể trong , chúng tôi đánh giá khả năng của T5 để tự động sửa đổi để giải quyết (do đó có được một phương pháp sửa đổi).

RQ thứ ba tập trung vào nhiệm vụ liên quan đến đánh giá mã mới mà chúng tôi giới thiệu trong bài báo này:

RQ3: T5 có thể tự động đề xuất những thay đổi trong ngôn ngữ tự nhiên ở mức độ nào như những người đánh giá sẽ làm? Trong RQ T5 này được cung cấp dưới dạng đầu vào với một phương pháp Java được gửi để xem xét
( ) và bắt buộc phải tạo nhận xét về ngôn ngữ tự nhiên ( ) yêu cầu thay đổi mã như người đánh giá sẽ làm.

Đối với RQ1-RQ3, chúng tôi thử nghiệm với các biến thể khác nhau của mô hình T5. Đặc biệt, chúng tôi đánh giá chất lượng dự đoán T5 cho cả ba nhiệm vụ khi (i) mô hình được đào tạo trước hoặc không; và (ii) các dự đoán có mức độ tin cậy khác nhau. Nhờ những phân tích này, chúng ta có thể trả lời RQ thứ tư của mình:

RQ4: Vai trò của mô hình trước khi đào tạo về hiệu suất của T5 là gì? Sự tự tin của các dự đoán trước ảnh hưởng đến chất lượng của họ như thế nào? Như đã giải thích trong Phần 2.3, chúng tôi thực hiện một nghiên cứu xóa trong đó T5 được tinh chỉnh mà không cần bất kỳ đào tạo trước nào (tức là, bằng cách bắt đầu từ các trọng số ngẫu nhiên trong mạng thần kinh). Điều này cho phép đánh giá sự đóng góp của đào tạo trước vào hiệu suất của mô hình. Đối với độ tin cậy của các dự đoán trước, chúng tôi đánh giá liệu nó có thể được sử dụng như một ủy quyền đáng tin cậy cho chất lượng của các dự đoán hay không (tức là, độ tin cậy càng cao, khả năng dự đoán càng cao là chính xác). Nếu trường hợp này xảy ra, một phát hiện như vậy sẽ có ý nghĩa đối với việc sử dụng mô hình T5 trong thực tế: Một nhà phát triển sử dụng mô hình có thể quyết định nhận được các khuyến nghị có độ tin cậy cao hơn , làm giảm cơ hội nhận được các dự đoán vô nghĩa.

Cuối cùng, RQ cuối cùng so sánh hiệu suất của mô hình T5 với phương pháp tiếp cận mà chúng tôi đã trình bày trong [46]:

RQ5: Hiệu suất của T5 so với kỹ thuật hiện đại là gì? Chúng tôi sử dụng việc triển khai và các bộ dữ liệu từ công việc trước đây của chúng tôi để so sánh hiệu suất của mô hình T5 với đường cơ sở [46].

3.1 Thu thập và phân tích dữ liệu

Để trả lời bốn câu hỏi nghiên cứu đầu tiên, chúng tôi thử nghiệm với cấu hình tốt nhất của cả mô hình T5 được đào tạo trước và không được đào tạo trước trên bộ thử nghiệm của bộ dữ liệu tinh chỉnh được báo cáo trong Bảng 1.

Hãy nhớ rằng đối với mỗi nhiệm vụ trong số ba nhiệm vụ mà chúng tôi hỗ trợ (ví dụ: các nhiệm vụ ánh xạ đến RQ1, RQ2 và RQ3) các trường hợp bộ kiểm tra 16.779
là các bộ ba giống nhau < , , >. Sự khác biệt duy nhất là: trong RQ1 mô hình đã được đào tạo (và được kiểm tra) để lấy đầu vào
và sản xuất ; trong RQ2 nó lấy làm đầu vào và và sản xuất ;
trong RQ3 nó lấy làm đầu vào và sản xuất .

Bằng cách chạy các mô hình trên các bộ thử nghiệm, chúng tôi báo cáo cho mỗi trong ba nhiệm vụ tỷ lệ phần trăm của "dự đoán hoàn hảo", cụ thể là các trường hợp trong đó đầu ra của mô hình là một trong những dự kiến. Ví dụ: trong trường hợp RQ3, điều này có nghĩa là mô hình có thể, được đưa ra dưới dạng đầu vào, tạo ra một nhận xét giống hệt với nhận xét được viết thủ công bởi người đánh giá đã kiểm tra .

Bên cạnh việc tính toán các dự đoán hoàn hảo, trong RQ3 (tức là, nhiệm vụ mà mô hình được yêu cầu để tạo ra văn bản ngôn ngữ tự nhiên), chúng tôi cũng tính toán điểm BLEU (Nghiên cứu đánh giá song ngữ) của các dự đoán [32]. BLEU đánh giá chất lượng của văn bản được tạo tự động. Điểm BLEU nằm trong khoảng từ 0 đến 1, với 1 cho thấy, trong trường hợp của chúng tôi, nhận xét ngôn ngữ tự nhiên được mô hình đưa ra giống hệt với nhận xét được viết bằng tay bởi người đánh giá. Chúng tôi sử dụng biến thể BLEU-4, tính toán sự chồng chéo về mặt 4 chương trình giữa văn bản được tạo và văn bản tham chiếu.

Trong RQ1 và RQ2 (tức là, trong các nhiệm vụ mà mô hình được yêu cầu để tạo mã), chúng tôi áp dụng CodeBLEU [37], một số liệu tương tự được đề xuất gần đây lấy cảm hứng từ điểm BLEU nhưng được điều chỉnh để đánh giá chất lượng của mã được tạo tự động.

Khác với BLEU, CodeBLEU không chỉ tính toán độ tương đồng dựa trên n-gram mà còn xem xét mức độ tương tự của cây cú pháp trừu tượng và luồng dữ liệu được tạo ra và mã tham chiếu. Ren et al. [37], người đề xuất CodeBLEU, cho thấy số liệu của họ tương quan tốt hơn với nhận thức của các nhà phát triển về sự tương đồng của mã so với số liệu BLEU.

Liên quan đến RQ4, chúng tôi so sánh các kết quả (ví dụ: dự đoán hoàn hảo, BLEU, CodeBLEU) đạt được bởi mô hình T5 với và không có đào tạo trước. Chúng tôi cũng thống kê so sánh hai mô hình (tức là có/không có đào tạo trước) bằng cách sử dụng bài kiểm tra McNemar [30] và Tỷ lệ cược (ORS) trên các dự đoán hoàn hảo mà chúng có thể tạo ra. Đối với sự tự tin của các dự đoán, chúng tôi lấy mô hình hoạt động tốt nhất (tức là mô hình có đào tạo trước) và chia các dự đoán của nó thành mười nhóm dựa trên sự tự tin của họ đi từ 0,0 đến 1,0 ở các bước 0,1 (tức là, khoảng thời gian đầu tiên bao gồm tất cả các dự đoán có sự tự tin với 0 < ≤ 0,1, khoảng thời gian cuối cùng có 0,9 < ≤ 1). Sau đó, chúng tôi báo cáo cho mỗi khoảng tỷ lệ phần trăm dự đoán hoàn hảo.

Cuối cùng, trong RQ5, chúng tôi so sánh T5 với đường cơ sở [46] về hai nhiệm vụ tự động hóa trong công việc trước đây của chúng tôi (tức là những nhiệm vụ liên quan đến RQ1 và RQ2).

Là số liệu cho các so sánh, chúng tôi đã sử dụng tỷ lệ phần trăm dự đoán hoàn hảo và CodeBLEU của các dự đoán. Chúng tôi đã so sánh hai kỹ thuật trong một số tình huống. Đầu tiên, chúng tôi đã sử dụng tập dữ liệu

từ [46] có 17.194 bộ ba < , , >. Bằng cách thực hiện một số kiểm tra trên bộ dữ liệu này, chúng tôi nhận thấy rằng một vài trường hợp (97)

đã có nhận xét ( ) không được viết bằng tiếng Anh hoặc chứa các ký tự unicode không hợp lệ không cho phép tokenizer của chúng tôi hoạt động. Do đó, chúng tôi loại trừ những trường hợp đó khỏi đào tạo và các bộ kiểm tra được chia sẻ bởi các tác giả. Sau đó, bộ đào tạo đã được sử dụng để (i) đào tạo đường cơ sở [46]; và (ii) tinh chỉnh mô hình T5 mà không cần đào tạo trước. Bằng cách này, chúng ta có thể so sánh hiệu suất của hai mô hình trên bộ kiểm tra khi được đào tạo trên chính xác cùng một dữ liệu. Điều quan trọng cần lưu ý là đường cơ sở đã được đào tạo và thử nghiệm về mã trừu tượng (như được thực hiện trong [46]), trong khi T5 làm việc trực tiếp với mã nguồn thô.

Trên hết, chúng tôi cũng báo cáo hiệu suất của mô hình T5 được đào tạo trước khi chạy trên bộ thử nghiệm từ [46]. Mô hình đào tạo trước này đã được tinh chỉnh bằng cách sử dụng bộ dữ liệu đào tạo trong [46]. Rõ ràng, phân tích này ủng hộ T5 vì nó đã được đào tạo về nhiều dữ liệu hơn (tức là tập dữ liệu trước khi đào tạo). Tuy nhiên, nó cung cấp các gợi ý bổ sung về vai trò của việc đào tạo trước và về hiệu quả của mô hình T5 nói chung. Bên cạnh việc báo cáo số liệu thống kê mô tả, chúng tôi thống kê so sánh hai mô hình bằng cách sử dụng bài kiểm tra McNemar [30] và Tỷ lệ cược (ORS) trên các dự đoán hoàn hảo mà chúng có thể tạo ra. Vì có liên quan đến nhiều so sánh (ví dụ: so sánh mô hình được đào tạo trước và mô hình không được đào tạo trước với đường cơ sở), chúng tôi điều chỉnh các giá trị bằng cách sử dụng hiệu chỉnh Holm [24].

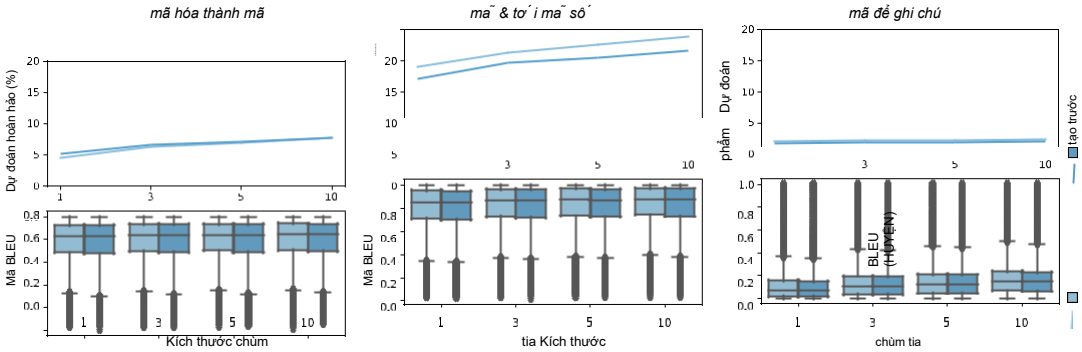
4 THẢO LUẬN KẾT QUẢ

Chúng ta bắt đầu bằng cách trả lời RQ1 -RQ 3 (Phần 4.1), trình bày hiệu suất mỗi phần của T5 trong ba nhiệm vụ mà chúng ta hướng đến là tự động hóa. Sau đó, chúng ta thảo luận về tác động đến hiệu suất của đào tạo trước và độ tin cậy của mức độ tin cậy như là một đại diện cho chất lượng của các dự đoán (Phần 4.2). Cuối cùng, chúng tôi so sánh T5 với đường cơ sở [46] (Phần 4.3).

4.1 RQ 1 -RQ 3: Hiệu suất của T5

Hình 1 báo cáo hai biểu đồ cho mỗi nhiệm vụ. Biểu đồ đường ở trên cùng cho thấy tỷ lệ phần trăm dự đoán hoàn hảo (trục) đạt được bởi T5 cho các kích thước chùm tia khác nhau (trục); đường liên tục đại diện cho phiên bản được đào tạo trước của mô hình, trong khi đường đứt nét là đường không được đào tạo trước. Các ô ở dưới cùng báo cáo CodeBLEU cho hai tác vụ tạo mã (tức là, code-to-code và code&comment-to-code) và điểm BLEU cho tác vụ tạo mã trong đó văn bản được tạo. Màu xanh nhạt hơn đại diện cho mô hình được đào tạo trước.

Chúng ta bắt đầu bằng cách bình luận về các dự đoán hoàn hảo (biểu đồ đường). Thoạt nhìn, hiệu suất của mô hình có vẻ khá thấp. Ví dụ, trong trường hợp mã hóa = 1 (tức là, một dự đoán duy nhất được đề xuất bởi T5), cả các mô hình được đào tạo trước và các mô hình không được đào tạo trước đều đạt được5% dự đoán hoàn hảo (751 và 863 trường hợp được dự đoán chính xác tương ứng với và không có đào tạo trước). Tuy nhiên, kết quả như vậy nên được xem xét trong bối cảnh của những gì đã được báo cáo bởi kỹ thuật hiện đại [46] rằng, trên một tập dữ liệu thử nghiệm đơn giản hơn nhiều, đạt được cho cùng một nhiệm vụ và cùng kích thước chùm tia 2,91% dự đoán hoàn hảo.



Hình 1: Kết quả tập dữ liệu T5 lớn

Các quan sát tương tự có thể được thực hiện cho nhiệm vụ mã hóa và mã hóa, trong đó tại = 1 T5 có thể tạo ra 14,08% (2.363 trường hợp) và 12,06% (2.024) dự đoán hoàn hảo khi được đào tạo trước và không, tương ứng. Đối với nhiệm vụ này, trong công việc trước đây của chúng tôi [46], chúng tôi đã đạt được trên một tập dữ liệu đơn giản hơn 12,16% dự đoán hoàn hảo. Chúng tôi trực tiếp so sánh hai cách tiếp cận trong RQ5.

Điều thú vị là, việc tăng kích thước chùm tia từ 1 lên 10 chỉ dẫn đến cải thiện biên cho tất cả các nhiệm vụ. Cải tiến lớn nhất là thu được cho mã & tham khảo mã, nơi chúng tôi di chuyển từ 14,08% (= 1) đến 18,88% (= 10) dự đoán hoàn hảo cho mô hình được đào tạo trước. Với mục tiêu tiếp cận của mình, chúng tôi tin rằng hiệu suất phù hợp nhất là những hiệu suất đạt được ở mức = 1.

Thật vậy, cung cấp một số khuyến nghị để kiểm tra cho một trình khử phát triển có thể phản tác dụng, đặc biệt là xem xét rằng các khuyến nghị là toàn bộ các phương pháp trong trường hợp hai nhiệm vụ tạo mã.

Chuyển sang nhiệm vụ code-to-comment, T5 gặp khó khăn trong việc xây dựng các nhận xét ngôn ngữ tự nhiên giống với những nhận xét được viết bởi người đánh giá. Mô hình được đào tạo trước, tại = 1, tạo ra 356 nhận xét đúng (2,12%) so với 324 (1,93%) của mô hình không được đào tạo trước. Những con số này chỉ tăng nhẹ ở mức = 10, với tối đa 2,44% dự đoán hoàn hảo đạt được khi đào tạo trước.

Phần trên cùng của Hình 2 cho thấy hai ví dụ về các dự đoán hoàn hảo được tạo ra bởi mô hình cho mỗi nhiệm vụ. Một đường gạch nối ngăn cách hai ví dụ trong mỗi nhiệm vụ. Đối với nhiệm vụ mã hóa, mã đầu tiên trong mỗi ví dụ đại diện cho đầu vào của mô hình, trong khi mã thứ hai đại diện cho đầu ra của nó. Chúng tôi đã tô đậm các phần của mã được thay đổi bởi mô hình và thay thế các phần không liên quan của các phương pháp bằng [...] để tiết kiệm không gian. Trong ví dụ mã thành mã đầu tiên, T5 loại bỏ kiểm tra phiên bản không cần thiết, vì FileSystemDataset là một phân lớp của Tập dữ liệu. Thay vào đó, ví dụ thứ hai đơn giản hóa việc kiểm tra sự tồn tại của một cụm, cung cấp một thông báo lỗi có ý nghĩa. Trường hợp thứ hai này không thể được hỗ trợ bởi đường cơ sở [46], vì nó yêu cầu giới thiệu mã token mới không có trong mã đầu vào. Hãy nhớ rằng, đây là những dự đoán hoàn hảo, những thay đổi được triển khai giống hệt với những thay đổi được thực hiện bởi các nhà phát triển trong quá trình xem xét mã.

Đối với tác vụ từ mã đến mã, đầu vào được cung cấp bởi mô hình bao gồm nhận xét được viết bởi người đánh giá và yêu cầu thay đổi cụ thể đối với phần mã được đánh dấu bằng màu cam. Trong ví dụ đầu tiên, người đánh giá đề nghị sử dụng một đối tượng cụ thể để thực hiện kiểm tra null và T5 thực hiện chính xác thay đổi.

Điều thứ hai rất thú vị bởi vì, mặc dù người đánh giá đánh giá cao trả về null là mã có liên quan cho nhận xét của họ ("khác là dư thừa"), mô hình hiểu chính xác rằng hành động cần thực hiện là loại bỏ tuyên bố khác không cần thiết.

Cuối cùng, đối với nhiệm vụ mã hóa nhận xét, chúng tôi báo cáo mã được cung cấp dưới dạng đầu vào cho mô hình (dòng đầu tiên) với nhận xét nó tạo ra dưới dạng đầu ra (dòng thứ hai). Trong ví dụ đầu tiên, T5 đề nghị (như được thực hiện bởi người đánh giá thực sự) thêm một kiểm tra null, cũng hiển thị mã cần thiết cho việc thực hiện nó. Mã này không chỉ là một mẫu, nhưng nó phù hợp với mã đầu vào được cung cấp (nó đề cập đến đối tượng nhà cung cấp). Trong ví dụ thứ hai, T5 đề xuất đổi tên định danh, cung cấp các khuyến nghị hợp lệ cho việc đổi tên.

Nhìn vào cuối Hình 1, các kết quả về CodeBLEU cho thấy một trung vị cao hơn 0,80 cho tất cả các kích thước chùm tia và cho cả hai nhiệm vụ tạo mã. Tuy nhiên, mặc dù chúng tôi báo cáo các giá trị này cho tính đầy đủ và phù hợp với những gì được thực hiện trong các công việc tương tự [45, 46, 50], họ nói ít về chất lượng của các dự đoán và chúng chủ yếu hữu ích cho công việc trong tương lai mà muốn so sánh với cách tiếp cận của chúng tôi (phân phối hoàn chỉnh có sẵn trong gói sao chép của chúng tôi [8]). Thật vậy, rất khó để giải thích đúng các giá trị này vì hai lý do. Đầu tiên, không có ngưỡng được chấp nhận mà hiệu suất tốt có thể được yêu cầu. Thứ hai, như cũng đã làm trong các công trình trước đây đề xuất các mô hình lấy làm đầu vào cho một đoạn mã và cung cấp đầu ra cho cùng một mã "sửa đổi" theo một cách nào đó (ví dụ: với một lỗi cố định [45], với một tuyên bố duy nhất được thêm vào [50], hoặc với các thay đổi liên quan đến xem xét được thực hiện [46]), chúng tôi đã tính toán Code-BLEU giữa mã dự đoán và mã mục tiêu (hai phương pháp trong trường hợp của chúng tôi). Tuy nhiên, đầu vào được cung cấp cho mô hình đã khá giống với đầu ra mục tiêu, có nghĩa là một mô hình lấy đầu vào làm phương pháp và không thực hiện bất kỳ thay đổi nào trên nó, có khả năng thu được giá trị cao của CodeBLEU. Vì lý do này, chúng tôi chủ yếu tập trung thảo luận về các dự đoán hoàn hảo. Liên quan đến điểm BLEU đạt được trong bài tập code-to-comment, khoảng trung bình khoảng 0,10 (xem Hình 1). Kết quả như vậy được mong đợi dựa trên tỷ lệ phần trăm thấp các dự đoán hoàn hảo đạt được cho nhiệm vụ này.

Quay trở lại dự đoán hoàn hảo, các kết quả được báo cáo trong biểu đồ đường trong Hình 1 đại diện cho một ràng buộc thấp hơn cho hiệu suất của phương pháp tiếp cận của chúng tôi. Thật vậy, chúng tôi chỉ xem xét một dự đoán là “hoàn hảo” nếu nó giống với dự đoán tham chiếu. Ví dụ: trong trường hợp nhiệm vụ mã hóa nhận xét, nhận xét ngôn ngữ tự nhiên được tạo bởi T5 chỉ được phân loại là chính xác nếu nó bằng với tham chiếu, bao gồm cả dấu câu.

Hình 2: Ví dụ về dự đoán hoàn hảo và dự đoán thay thế

Tuy nhiên, có thể một nhận xét ngôn ngữ tự nhiên được trình bày bởi T5 là khác nhau nhưng về mặt ngữ nghĩa tương đương với nhận xét được viết bởi nhà phát triển (ví dụ: "biến nên là riêng tư" so với "thay đổi chế độ hiển thị thành riêng tư"). Các quan sát tương tự giữ cho hai nhiệm vụ tạo mã (ví dụ: nhận xét của người đánh giá có thể được giải quyết theo những cách khác nhau nhưng tương đương về mặt ngữ nghĩa).

Để có ý tưởng về số lượng các dự đoán có giá trị có mặt trong số những dự đoán được phân loại là "sai" (tức là các dự đoán không hoàn hảo), ba tác giả đã phân tích thủ công một mẫu gồm 100 dự đoán "sai" cho mỗi nhiệm vụ (tổng cộng 300). Phân tích được thực hiện trong hai cuộc họp trong đó mỗi trường hợp được thảo luận bởi cả ba tác giả. Mục tiêu là phân loại mỗi trường hợp thành một trong ba loại: (i) “tương đương về mặt ngữ nghĩa” (nghĩa là mã/nhận xét được tạo ra khác nhau nhưng tương đương về mặt ngữ nghĩa với mã tham chiếu); (ii) “giải pháp thay thế” (nghĩa là mã/nhận xét được tạo ra không tương đương về mặt ngữ nghĩa, nhưng có giá trị); hoặc (iii) “sai” (nghĩa là mã/nhận xét được tạo ra không có ý nghĩa đối với đầu vào được cung cấp). Vì chúng tôi cũng tính toán độ tin cậy cho mỗi dự đoán được tạo ra bởi T5, thay vì chọn ngẫu nhiên 300 trường hợp để kiểm tra, chúng tôi quyết định nhắm mục tiêu cho mỗi nhiệm vụ 100 dự đoán sai hàng đầu do mô hình tạo ra về độ tin cậy. Thật vậy, những trường hợp này đặc biệt thú vị, vì chúng đại diện cho những dự đoán sai, tuy nhiên, mô hình khá tự tin.

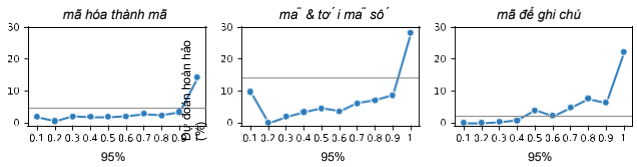
Bảng 3: Phân tích thủ công 100 dự đoán "sai" cho mỗi nhiệm vụ

Nhiệm vụ	Tương đương về mặt ngữ nghĩa	giải pháp có thể lựa chọn	Sai
mã hóa thành mã	1	10	89
mã & tới mã số	6	56	38
mã để ghi chú	36	10	54

Bảng 3 trình bày kết quả phân tích thủ công của chúng tôi. Đối với mã thành mã, chúng tôi quan sát thấy rằng, trong hầu hết các trường hợp (89%) mô hình thực sự tạo ra các dự đoán sai không phù hợp với những thay đổi được thực hiện bởi nhà phát triển. Có một vài ngoại lệ cho các trường hợp này, chủ yếu liên quan đến các thay đổi nhỏ trong đó mô hình đưa ra quyết định khác với mô hình của nhà phát triển nhưng vẫn có hiệu lực (ví dụ: trích xuất một chuỗi vào một biến và sử dụng tên khác cho biến được trích xuất). Thú vị hơn là kết quả cho hai nhiệm vụ còn lại.

Trong trường hợp code&comment-to-code, chúng tôi nhận thấy rằng 62 trong số 100 dự đoán "sai" mà chúng tôi kiểm tra thực sự là triển khai hợp lệ của thay đổi được khuyến nghị bởi người đánh giá. Một ví dụ được trình bày ở cuối Hình 2 (nền đen), trong đó chúng tôi hiển thị đầu vào được cung cấp cho mô hình (tức là mã trong dòng đầu tiên và nhận xét của người đánh giá "Inline biến này") và đầu ra của mô hình ngay bên dưới. T5 đã gửi thành công nhận xét của người đánh giá.

Sử dụng các mô hình được đào tạo trước để tăng cường tự động hóa đánh giá mã



Hình 3: Dự đoán hoàn hảo bằng sự tự tin của mô hình

Tuy nhiên, dự đoán này khác với việc thực hiện mục tiêu, vì dự đoán này cũng bao gồm một thay đổi khác không được yêu cầu rõ ràng trong việc xem xét mã. Trường hợp này là đại diện cho tất cả 56 trường hợp mà chúng tôi phân loại là "các giải pháp thay thế" cho nhiệm vụ này và, với mục tiêu mã hóa thành mã, chúng tôi tin rằng chúng đại diện cho các dự đoán tốt.

Cuối cùng, cũng đối với nhiệm vụ mã hóa nhận xét, chúng tôi tìm thấy một số lượng lớn các dự đoán "sai" thực sự có giá trị, với 36 trong số chúng thậm chí tương đương về mặt ngữ nghĩa (tức là, T5 xây dựng một nhận xét yêu cầu những thay đổi tương tự theo yêu cầu của người đánh giá, nhưng sử dụng một từ khác). Một ví dụ được báo cáo ở dưới cùng của Hình 2. Mặc dù mô hình chỉ nhận được mã làm đầu vào, chúng tôi cũng hiển thị nhận xét của người đánh giá ban đầu (ví dụ: "Vui lòng biến nhận xét này thành một biến") để giúp dễ dàng đánh giá mức độ liên quan của nhận xét được tạo bởi T5 (ví dụ: "Trích xuất tòa nhà ...").

Nhìn chung, phân tích của chúng tôi cho thấy rằng các dự đoán hoàn hảo thực sự đại diện cho sự ràng buộc thấp hơn đối với hiệu suất của T5, đặc biệt là đối với hai nhiệm vụ có liên quan đến nhận xét ngôn ngữ tự nhiên.

4.2 RQ4: Tiền tập luyện và sự tự tin

Trong Hình 1, chúng tôi quan sát thấy hiệu suất tốt hơn cho mô hình được đào tạo trước trong tác vụ code&comment-to-code và trong tác vụ code-to-comment, trong khi mô hình không được đào tạo trước hoạt động tốt hơn trong tác vụ code-to-code. Kết quả của bài kiểm tra McNemar về các dự đoán tại =1, xác nhận những phát hiện như vậy: bên cạnh sự khác biệt đáng kể được xác nhận cho tất cả các nhiệm vụ (- value < 0.01), ORS cho thấy tỷ lệ cao hơn 85% và 59% để có được một dự đoán hoàn hảo bằng cách sử dụng mô hình được đào tạo trước trong nhiệm vụ code-to-code (OR=1.85) và trong nhiệm vụ code-tocomment (OR=1.59), trong khi tỷ lệ thấp hơn 34% trong nhiệm vụ code-to-code (OR=0.66). Hai quan sát đáng được thực hiện. Thứ nhất, nhìn chung, mô hình được đào tạo trước dường như đại diện cho một giải pháp có giá trị hơn. Thứ hai, sự thiếu cải thiện trong nhiệm vụ mã hóa có thể được giải thích bằng việc đào tạo trước và tinh chỉnh mà chúng tôi đã thực hiện. Trong khi thực hiện, tác vụ từ mã đến mã chỉ tập trung vào mã nguồn, không có ngôn ngữ tự nhiên trong đầu vào cũng như đầu ra. Giai đoạn tinh chỉnh, tập trung vào mã nguồn, có lẽ đã đủ để mô hình tìm hiểu về cú pháp mã và các biến đổi có thể thực hiện. Các khóa đào tạo bổ sung, bao gồm cả tiếng Anh kỹ thuật, đã không mang lại lợi ích cho mô hình cho nhiệm vụ từ mã đến mã. Thay vào đó, hai nhiệm vụ còn lại bao gồm ngôn ngữ tự nhiên làm đầu vào (code&comment-to-code) hoặc yêu cầu tạo ra ngôn ngữ tự nhiên làm đầu ra (code-to-comment), thu được hiệu suất tăng lên từ quá trình đào tạo trước.

Hình 3 mô tả tỷ lệ phần trăm dự đoán hoàn hảo (trục) trong mỗi khoảng tin cậy (từ 0,0-0,1 lên đến 0,9-1,0, trục) khi sử dụng mô hình được đào tạo trước và =1. Để diễn giải tốt hơn các kết quả được nhập lại, đường màu xám thể hiện hiệu suất tổng thể của mô hình khi xem xét tất cả các dự đoán (ví dụ: 4,48% dự đoán hoàn hảo cho nhiệm vụ mã hóa).

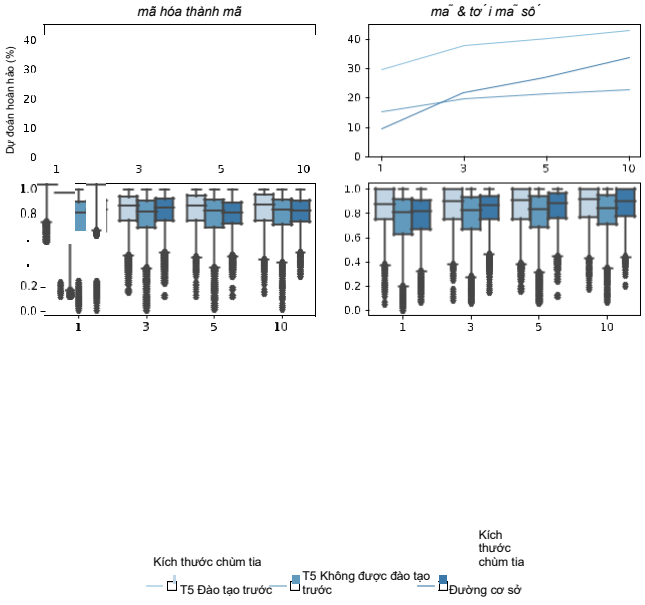
Trong cả ba nhiệm vụ, chúng tôi quan sát một xu hướng rõ ràng, với các dự đoán trong nhóm độ tin cậy cao nhất (0.9-1.0) đảm bảo hiệu suất về cơ bản tốt hơn xu hướng chung. Khi chỉ xem xét các dự đoán trong nhóm này, tỷ lệ phần trăm các dự đoán hoàn hảo tăng lên: 14,24% đối với mã thành mã (từ tổng thể 4,48%), 28,23% đối với mã&comment-to-code (tổng thể= 14,08%) và 22,23% đối với mã thành mã (tổng thể=2,12%). Xem xét sự phức tạp của các nhiệm vụ được giải quyết, bước nhảy trong hiệu suất là đáng kể và cho thấy khả năng sử dụng của mức độ tin cậy như một đại diện cho chất lượng dự đoán. Ngoài ra, mặc dù tỷ lệ dự đoán hoàn hảo là khá hạn chế, với bảy trong số mười dự đoán là sai trong trường hợp tốt nhất (28,23% cho code&comment-to-code), cần xem xét những gì đã được quan sát trước đó trong phân tích thủ công của chúng tôi, với các dự đoán "có giá trị" được phân loại là "sai" trong phân tích định lượng của chúng tôi.

4.3 RQ_5: So sánh với đường cơ sở [46]

Hình 4 so sánh hiệu suất đạt được bởi mô hình T5 với hiệu suất thu được bởi đường cơ sở [46].

Trong các biểu đồ đường, các đường liên tục đại diện cho T5 được đào tạo trước, các đường đứt nét không được đào tạo trước T5 và các đường chấm chấm của đường cơ sở. Hai điểm quan trọng cần ghi nhớ: Đầu tiên, các kết quả trong Hình 4 đã được tính toán trên bộ kiểm tra được sử dụng trong [46]. Thật vậy, hiệu suất về các dự đoán hoàn hảo cao hơn một chút so với các dự đoán trong Hình 1 (xem các giá trị trên trục), do các trường hợp đơn giản hơn được nêu bật trong bộ dữ liệu này. Thứ hai, đường cơ sở đã được đào tạo và thử nghiệm về mã trừu tượng (như trong bài báo gốc), trong khi T5 làm việc về mã nguồn thô.

Khi =1, T5 đạt được hiệu suất tốt hơn đáng kể. Kết quả của bài kiểm tra thống kê trong Bảng 4 luôn cho thấy sự khác biệt đáng kể có lợi cho T5 (giá trị điều chỉnh < 0,01), với ORS dao động từ 1,69 (T5 không được đào tạo trước so với [46] trong nhiệm vụ mã hóa) đến 11,48 (T5 được đào tạo trước so với [46] trong nhiệm vụ mã hóa). T5 được đào tạo trước trong trường hợp này hoạt động tốt hơn so với T5 không được đào tạo trước cho cả hai nhiệm vụ. Điều này có thể là do kích thước hạn chế của bộ dữ liệu tinh chỉnh được sử dụng trong so sánh này. Thật vậy, để có một so sánh công bằng với [46], chúng tôi đã tinh chỉnh T5 trên tập huấn luyện chúng tôi sử dụng trong [46] và soạn thảo bởi 13.5ktrường hợp (so với 134k chúng tôi có trong bộ dữ liệu tinh chỉnh của chúng tôi khi trả lời RQ_1 -RQ 4). Điều này có lẽ không đủ để đào tạo hiệu quả một mô hình lớn như T5, và làm cho các trường hợp được sử dụng trong đào tạo trước là cơ bản để tìm hiểu thêm về ngôn ngữ. Tuy nhiên, ngay cả khi không được đào tạo trước, T5 vẫn vượt trội hơn mức cơ bản khi =1. Ví dụ, trong nhiệm vụ mã hóa và mã hóa, đường cơ sở đạt được 9,48% dự đoán hoàn hảo, so với 15,46% của T5 không được đào tạo trước và 29,74% của T5 được đào tạo trước. Đường cơ sở quan sát sự cải thiện mạnh mẽ hơn với việc tăng trong (tức là, kích thước chùm tia) so với T5 (xem Hình 4). Chúng tôi tin rằng điều này là do cách sử dụng trừu tượng. Thật vậy, khi làm việc với mã trừu tượng, "không gian tìm kiếm" (tức là, số lượng các giải pháp có thể được tạo ra với từ vựng nhất định) bị hạn chế hơn nhiều vì mô hình không giải quyết được các định danh và chữ viết. Thử mười dự đoán trong một không gian tìm kiếm nhỏ hơn có nhiều khả năng dẫn đến dự đoán chính xác. Kết quả của CodeBLEU xác nhận xu hướng được quan sát với các dự đoán hoàn hảo, với T5 được đào tạo trước là mô hình tốt nhất.



Hình 4: T5 so với đường cơ sở [46]

Chúng tôi cũng xem xét sự kết hợp của các dự đoán hoàn hảo được tạo ra bởi hai phương pháp tiếp cận trên bộ kiểm tra để xác minh tính bổ sung của các kỹ thuật. Về nhiệm vụ mã hóa (code&comment-to-code), chúng tôi quan sát thấy rằng 15% (24%) dự đoán hoàn hảo được chia sẻ bởi cả hai cách tiếp cận (nghĩa là cả hai đều thành công), 65% (70%) là dự đoán hoàn hảo chỉ cho T5 và 20% (6%) chỉ cho đường cơ sở.

Bảng 4: RQ5: Thử nghiệm của McNemar (adj. - Gia´ trị va` hoặ C)

Nhiệm vụ	Thử nghiệm	giá trị xác suất	HOẶC
mã hóa thành mã	T5 trước khi tập luyện vs [46]	<0,01	2.90
	T5 không được đào tạo trước so với T4 [46]	<0,01	1,69
	T5 được đào tạo trước so với T5 không được đào tạo trước	<0,01	2,50
ma˜ & to´ i ma˜ só´	T5 trước khi tập luyện vs [46]	<0,01	11,48
	T5 không được đào tạo trước so với T4 [46]	<0,01	2.38
	T5 được đào tạo trước so với T5 không được đào tạo trước	<0,01	5,69%

5 MỐI ĐE DỌA ĐỐI VỚI HIỆU LỰC

Khái niệm giá trị Như đã giải thích trong Phần 2, chúng tôi đã làm sạch các bộ dữ liệu được sử dụng trong nghiên cứu của chúng tôi bằng cách loại bỏ các bản sao và các điểm dữ liệu nhiễu trong phạm vi có thể. Tuy nhiên, chúng tôi biết rằng các trường hợp có vấn đề có thể có mặt, đặc biệt là trong bộ dữ liệu mới (lớn) mà chúng tôi đã xây dựng. Điều này thể hiện, ví dụ, trong các nhận xét không phải bằng tiếng Anh, hoặc trong một số "liên kết" sai giữa nhận xét và lập kế hoạch thực hiện (ví dụ, chúng tôi giả định rằng đã thực hiện một thay đổi được mô tả trong khi, trên thực tế, nó thực hiện một thay đổi khác).

Giá trị bên trong. Chúng tôi đã không khám phá đầy đủ vai trò của các thông số T5 đối với hiệu suất của nó. Thật vậy, điều chỉnh siêu thông số của chúng tôi bị giới hạn ở các biến thể trong tỷ lệ học tập, như đã làm trong công việc trước đây [28]. Đối với các thông số khác, chúng tôi dựa trên kiến trúc tốt nhất được xác định

6 |||UNTRANSLATED_CONTENT_START|||RELATED WORK|||UNTRANSLATED_CONTENT_END|||

Công việc của chúng tôi liên quan đến ba lĩnh vực nghiên cứu: (i) các kỹ thuật DL để tự động hóa các nhiệm vụ liên quan đến phần mềm, (ii) các nghiên cứu thực nghiệm về xem xét mã và (iii) cung cấp các khuyến nghị về cách tối ưu hóa quy trình xem xét mã và/hoặc trình bày các kỹ thuật để tự động hóa một phần quy trình. Ở đây chúng tôi tập trung vào lĩnh vực nghiên cứu thứ ba, trong khi đối với hai cuốn sách đầu tiên, chúng tôi hướng người đọc đến các đánh giá tài liệu có hệ thống của Watson và cộng sự [49] (học sâu về kỹ thuật phần mềm) và của Davila và Nunes [20] (đánh giá mã hiện đại).

Tối ưu hóa/tự động hóa quy trình xem xét mã. Bằng cách nghiên cứu các công cụ và kỹ thuật hỗ trợ xem xét mã, Tymchuk et al.
bởi Raffel et al. .35 Chúng tôi thừa nhận rằng việc điều chỉnh bổ sung có thể dẫn đến cải thiện hiệu suất.

Giá trị bên ngoài RQ1 -RQ 4 đã được trả lời bằng cách sử dụng một tập dữ liệu có độ lớn lớn hơn so với công việc trước đây của chúng tôi về tự động hóa các nhiệm vụ xem xét mã [46]. Tuy nhiên, những phát hiện của chúng tôi chỉ giới hạn ở Java. Liên quan đến RQ5 trong đó chúng tôi so sánh với đường cơ sở [46], chúng tôi chỉ sử dụng bộ dữ liệu được trình bày trong [46]. Điều này là do thực tế là cách tiếp cận trước đây của chúng tôi [46] yêu cầu trừu tượng hóa mã và, như đã giải thích trước đây, không thể làm việc trên các trường hợp có các định danh và chữ mới được chèn vào trong quá trình xem xét mã. Bộ dữ liệu mới được sử dụng trong bài báo này chưa được xây dựng với một hạn chế như vậy trong tâm trí và do đó, nó không phù hợp để so sánh trực tiếp.

[47] kết luận rằng các nền tảng xem xét mã phổ biến (ví dụ: Gerrit, Code Flow, Phabricator) hầu hết cung cấp các chức năng cơ bản giống nhau với ít hỗ trợ để tự động hóa các nhiệm vụ. Phát hiện như vậy đã được xác nhận bởi Pascarella et al. 34. Ngoài ra, trong một nghiên cứu được thực hiện bởi Lewis et al. [27] tại Google, các tác giả cho thấy rằng trong khi các nhà phát triển hào hứng với ý tưởng nắm bắt các giải pháp tự động để xem xét mã, họ tìm thấy các giải pháp hiện tại không sẵn sàng để sử dụng hàng ngày. Bắt đầu từ những quan sát này, các nhà nghiên cứu đã nghiên cứu các op-timization có thể có của quá trình xem xét: Baum et al. [14] nghiên cứu ảnh hưởng của việc sắp xếp các thay đổi được đệ trình theo cách thay thế hơn là theo thứ tự bảng chữ cái mà, như được hiển thị bởi Barnett et al. [12] và Baum và Schneider [13], là dưới tối ưu. Baum et al. [14] kết luận rằng việc đặt hàng thông minh hơn là cần thiết khi kích thước của bản vá tăng lên, và đề xuất tổng hợp các bộ phận được thay đổi bởi sự liên quan.

Di Biase và cộng sự [21] đã nghiên cứu tác động của kích thước bản vá đối với hiệu quả của đánh giá, phát hiện ra rằng các bản vá nhỏ hơn, trong khi không làm giảm các khiếm khuyết được tìm thấy, ảnh hưởng đến cách người đánh giá tiếp cận nhiệm vụ của họ. Spadini et al. [43] so sánh hiệu quả của quy trình xem xét mã tiêu chuẩn với xem xét mã theo hướng thử nghiệm (TDR), tức là, người xem xét kiểm tra mã thử nghiệm đã thay đổi trước khi mã sản xuất. Họ cho thấy rằng TDR không thúc đẩy hiệu quả xem xét mã.

Một số nhà nghiên cứu [23, 33, 51] đề xuất khai thác các mô hình khiếm khuyết trong quá trình xem xét mã. Tương tự, Balachandran [11] và Singh et al. [42] đề nghị sử dụng các công cụ phân tích tĩnh để phát hiện các vi phạm tiêu chuẩn mã hóa tự động và các khiếm khuyết phổ biến.

Liên quan đến việc tự động hóa các nhiệm vụ rà soát mã cụ thể, các tác giả đã đề xuất các kỹ thuật để tối ưu hóa nhiệm vụ của người rà soát. Ví dụ, Al-Zubaidi et al. [9] trong mã nguồn mở và Chouchen et al. [18] trong bối cảnh công nghiệp cho thấy cách tiếp cận dựa trên tìm kiếm đa mục tiêu có thể đơn giản hóa quá trình phân tích xem xét mã.

Shi et al. [41] và Chouchen et al. [18] xem xét việc tự động hóa xem xét mã từ một quan điểm tương tự. Shi và cộng sự [41] trình bày một mô hình DL lấy làm đầu vào cho mã được gửi để xem xét và mã sửa đổi thực hiện các thay đổi được khuyến nghị bởi người xem xét và cung cấp đầu ra cho dù thay đổi có thể được chấp nhận hay không. Lưu ý rằng (các) thay đổi được yêu cầu bởi (các) người đánh giá không được xem xét bởi mô hình. Chouchen và cộng sự. [18] thay vào đó sử dụng một tập hợp các số liệu chất lượng làm tính năng cho các thuật toán máy học để phân loại chất lượng của mã được gửi để xem xét. Gần đây, Hellendoorn et al. [22] tập trung vào dự đoán vị trí của nhận xét của người đánh giá, cho thấy rằng ngay cả nhiệm vụ đơn giản này cũng khó tự động hóa.

Các kỹ thuật được thảo luận ở trên [18, 22, 41] là bổ sung cho phương pháp mà chúng tôi đã trình bày trong [46] (và kết quả là, các mô hình được thử nghiệm trong công việc này).

Trong khi Shi et al. [41] và Chouchen et al. [18] đánh giá mã đang được xem xét thông qua một "câu trả lời boolean" (tức là chấp nhận/từ chối hoặc viết tốt/viết xấu), chúng tôi cố gắng tự động hóa các thay đổi mã được thực hiện trong xem xét mã. Ngoài ra, cách tiếp cận của Hel-lendoorn et al. có thể được kết hợp với việc tự động hóa nhiệm vụ code-to-comment mà chúng tôi đã trình bày.

7KẾT LUẬN VÀ CÔNG VIỆC TRONG TƯƠNG LAI

Bài báo của chúng tôi bắt đầu bằng việc thảo luận về những hạn chế trong cách tiếp cận mà chúng tôi đã đề xuất để tự động hóa các nhiệm vụ rà soát mã [46]. Chúng tôi nhấn mạnh rằng việc sử dụng trừu tượng hóa mã không cho phép hỗ trợ các kịch bản xem xét mã phi tầm thường yêu cầu thay đổi mã dẫn đến việc giới thiệu các định danh/chữ viết mới. Do đó, chúng tôi đã đề xuất sử dụng mô hình T5 được đào tạo trước [35] dựa vào một Mảnh ghép Câu [26] tokenizer để khắc phục hạn chế như vậy và làm việc trực tiếp trên mã nguồn thô. Đánh giá thực nghiệm của chúng tôi, được thực hiện trên một bộ dữ liệu đánh giá mã lớn hơn và thực tế hơn nhiều, cho thấy những cải tiến do mô hình T5 mang lại thể hiện một bước tiến như được chuẩn bị cho hiện đại [46] cả về khả năng ứng dụng (tức là, các kịch bản trong đó nó có thể được áp dụng) và hiệu suất. Tuy nhiên, mức độ hiệu suất thực tế được quan sát làm cho các kỹ thuật này không thể triển khai trong thực tế, kêu gọi nghiên cứu nhiều hơn trong tự động hóa xem xét mã.

Chương trình nghiên cứu trong tương lai của chúng tôi sẽ tập trung vào việc thiết kế các giải pháp đã được chứng minh để tăng cường độ chính xác dự đoán của các sản phẩm công nghệ này (ví dụ: bằng cách kết hợp các đại diện khác nhau của mã [17] và/hoặc bằng cách khai thác sự tự tin của mô hình như một bộ lọc có thể để chỉ chọn các khuyến nghị chất lượng cao).

Mã̃ va̋ dữ liệ u đượ c sử dụ ng trong nghiên cứ u củ a chú ng tôi đượ c công bố́ công khai [8].

ĐƠN XÁC NHẬN

Dự án này đã nhận được tài trợ từ Hội đồng Nghiên cứu Châu Âu (ERC) theo chương trình nghiên cứu và đổi mới Horizon 2020 của Liên minh Châu Âu (thỏa thuận tài trợ số 851720). W&M đã được hỗ trợ một phần bởi các khoản tài trợ NSF CCF-1955853 và CCF-2007246. Bất kỳ ý kiến, phát hiện và kết luận nào được trình bày ở đây là của các tác giả và không nhất thiết phản ánh ý kiến của các nhà tài trợ.

TÀI LIỆU THAM KHẢO

[1] [n.d.]. Gerrit. https://www.gerritcodereview.com/.
[2] [n.d.]. GitHub. https://github.com/.
[3] [n.d.]. langdetect. https://pypi.org/project/langdetect/.
[4] [n.d.]. Lizard. https://github.com/terryyin/lizard/.
[5] [n.d.]. Nền tảng khai thác MSR. https://seart-ghs.si.usi.ch.
[6] [n.d.]. pycld3. https://pypi.org/project/pycld3/.
[7] [n.d.]. Stack Exchange Dumps. https://archive.org/details/stackexchange.
[8] 2021. Gói sao chép. https://github.com/RosaliaTufano/code_review_ tự động hóa.
[9] Wisam Haitham Abbood Al-Zubaidi, Patanamon Thongtanunam, đập Hòa Khánh, Chakkrit Tantithamthavorn, và Aditya Ghose. 2020. Khuyến nghị của người đánh giá nhận thức khối lượng công việc sử dụng phương pháp tiếp cận dựa trên tìm kiếm đa mục tiêu. Trong Kỷ yếu của Hội nghị Quốc tế ACM lần thứ 16 về Mô hình Dự đoán và Phân tích Dữ liệu trong Kỹ thuật Phần mềm. 21-30
[10] Alberto Bacchelli và Christian Bird. 2013. Kỳ vọng, kết quả và kết quả của đánh giá mã hiện đại. Trong Kỷ yếu hội nghị quốc tế về kỹ thuật phần mềm năm 2013. IEEE Press, 712–721.
[11] Vipin Balachandran. 2013. Giảm nỗ lực của con người và cải thiện chất lượng trong đánh giá mã ngang hàng bằng cách sử dụng phân tích tĩnh và khuyến nghị người đánh giá tự động. Năm 2013, Hội nghị quốc tế lần thứ 35 về kỹ thuật phần mềm (ICSE). 931–940. https://doi.org/10.1109/ICSE.2013.6606642

[12] Mike Barnett, Christian Bird, João Brunet, và Shuvendu K. Lahiri. 2015. Giúp các Nhà phát triển Trợ giúp Themselves: Tự động Phân tích các bộ Thay đổi Đánh giá Mã. Trong Kỷ yếu của Hội nghị Quốc tế lần thứ 37 về Kỹ thuật Phần mềm - Tập 1 (ICSE '15). 134–144.

[13] Tobias Baum và Kurt Schneider. 2016. Về sự cần thiết của một thế hệ mới của các công cụ xem xét mã. Trong Hội nghị quốc tế về cải tiến quy trình phần mềm tập trung vào sản phẩm. Springer, 301–308.

[14] Tobias Baum, Kurt Schneider, và Alberto Bacchelli. 2017 Về thứ tự tối ưu của việc đọc các thay đổi mã nguồn để xem xét. Năm 2017, IEEE đã tổ chức Hội nghị quốc tế về bảo trì và tiến hóa phần mềm (ICSME). IEEE, 329–340.

[15] Gabriele Bavota và Barbara Russo. 2015. Bốn con mắt tốt hơn hai: Về tác động của đánh giá mã đối với chất lượng phần mềm. Trong IEEE International Conference on Software Maintenance and Evolution, (ICSME). 81–90.

[16] A. Bosu và J. C. Carver. 2013. Tác động của Đánh giá Mã Đồng đẳng đối với Hình thành Ấn tượng Đồng đẳng: Khảo sát. Trong Hội thảo quốc tế ACM / IEEE về Kỹ thuật và Đo lường phần mềm thực nghiệm năm 2013. 133–142.

[17] Saikat Chakraborty và Baishakhi Ray. 11111111111111 Trên học đa phương thức của chỉnh sửa mã nguồn. arXiv: 2108.06645 [cs.SE]

[18] Moataz Chouchen, Ali Ouni, Mohamed Wiem Mkaouer, Raula Gaikovina Kula, và Katsuro Inoue. 11111111111111 WHOReview: Cách tiếp cận dựa trên tìm kiếm đa mục tiêu cho khuyến nghị của người đánh giá mã trong đánh giá mã hiện đại. Applied Soft Com-puting 100 (2021), 106908.

[19] Ozren Dabic, Emad Aghajani, và Gabriele Bavota. 11111111111111 Các dự án lấy mẫu trong GitHub cho các nghiên cứu MSR. Tại Hội nghị quốc tế IEEE/ACM lần thứ 18 về Kho phần mềm khai thác, MSR 2021. IEEE, 560–564.

[20] Nicole Davila và Ingrid Nunes. 11111111111111 Đánh giá tài liệu có hệ thống và phân loại đánh giá mã hiện đại. Tạp chí Hệ thống và Phần mềm (2021), 110951.

[21] Marco di Biase, Magiel Bruntink, Arie van Deursen, và Alberto Bacchelli. năm 2019. Ảnh hưởng của sự phân hủy thay đổi đối với việc xem xét mã - một thí nghiệm có kiểm soát. PeerJ Computer Science 5 (2019), e193.

[22] Vincent J Hellendoorn, Jason Tsay, Manisha Mukherjee, và Martin Hirzel. 11111111111111 Hướng tới tự động hóa rà soát mã ở quy mô lớn. Trong Kỷ yếu của cuộc họp chung ACM lần thứ 29 về Hội nghị kỹ thuật phần mềm châu Âu và Hội nghị chuyên đề về nền tảng của kỹ thuật phần mềm. 1479–1482.

[23] Hoàng Thông, Đàm Khánh Hòa, Yasutaka Kamei, David Lo, và Naoyasu Ubayashi. năm 2019. DeepJIT: một khuôn khổ học sâu từ đầu đến cuối để dự đoán khiếm khuyết chỉ trong thời gian ngắn. Năm 2019, IEEE/ACM đã tổ chức Hội nghị quốc tế lần thứ 16 về Kho phần mềm khai thác (MSR). IEEE, 34–45.

[24] Sture Holm. 1979 Một quy trình kiểm tra nhiều lần từ chối tuần tự đơn giản. Scan-dinavian journal of statistics (1979), 65–70.

[25] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, và Marc Brockschmidt. năm 2019. Thử thách CodeSearchNet: Đánh giá Trạng thái Tìm kiếm Mã Ngữ nghĩa. CoRR abs/1909.09436 (2019). http://arxiv.org/abs/1909.09436

[26] Taku Kudo và John Richardson. NĂM 2018. SentencePiece: Một tokenizer và detokenizer từ phụ ngôn ngữ đơn giản và độc lập cho Xử lý văn bản thần kinh. CoRR (2018). arXiv:1808.06226

[27] Chris Lewis, Zhongpeng Lin, Caitlin Sadowski, Xiaoyan Zhu, Rong Ou và E James Whitehead. 2013. Dự đoán lỗi có hỗ trợ các nhà phát triển con người không? kết quả từ một nghiên cứu điển hình của google. Năm 2013, Hội nghị quốc tế lần thứ 35 về kỹ thuật phần mềm (ICSE). IEEE, 372–381.

[28] Antonio Mastropaolo, Simone Scalabrino, Nathan Cooper, David Nader Palacio, Denys Poshyvanyk, Rocco Oliveto và Gabriele Bavota. 11111111111111 Nghiên cứu việc sử dụng máy biến áp chuyển tiếp văn bản để hỗ trợ các tác vụ liên quan đến mã. Năm 2021, IEEE/ACM đã tổ chức Hội nghị quốc tế lần thứ 43 về Kỹ thuật phần mềm (ICSE). IEEE, 336–347.

[29] Shane McIntosh, Yasutaka Kamei, Bram Adams, và Ahmed E. Hassan. 2014. Tác động của Bảo hiểm Đánh giá Mã và Tham gia Đánh giá Mã về Chất lượng Phần mềm: Nghiên cứu điển hình về các Dự án Qt, VTK và ITK. Kỷ yếu Hội nghị làm việc lần thứ 11 về Kho phần mềm khai thác (MSR 2014). 192–201.

[30] Quinn McNemar. 1947 Lưu ý về sai số lấy mẫu của sự khác biệt giữa tỷ lệ tương quan hoặc tỷ lệ phần trăm. Psychometrika 12, 2 (1947), 153–157.

[31] Rodrigo Morales, Shane McIntosh, và Foutse Khomh. 2015. Thực tiễn đánh giá quy tắc có tác động đến chất lượng thiết kế không? Nghiên cứu điển hình về các Dự án Qt, VTK và ITK. Trong Proc. của 22 Int'l Conf. về phân tích phần mềm, tiến hóa và tái kỹ thuật (SANER). 171–180.

[32] Kishore Papineni, Salim Roukos, Todd Ward và Wei-Jing Zhu. 2002 BLEU: Phương pháp đánh giá tự động dịch máy. Trong Kỷ yếu của Đại hội thường niên lần thứ 40 về Hiệp hội Ngôn ngữ học Tính toán (ACL '02). 311–318.

[33] Luca Pascarella, Fabio Palomba, và Alberto Bacchelli. năm 2019. Dự đoán khiếm khuyết vừa kịp thời của hạt mịn. Tạp chí Hệ thống và Phần mềm 150 (2019), 22–36.

[34] Luca Pascarella, Davide Spadini, Fabio Palomba, Magiel Bruntink, và Alberto Bacchelli. NĂM 2018. Nhu cầu thông tin trong xem xét mã đương đại. Kỷ yếu của ACM về tương tác con người - máy tính 2, CSCW (2018), 1–27.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li và Peter J. Liu. 2020. Khám phá các giới hạn của việc học chuyển giao với một máy biến đổi văn bản tiếp theo thống nhất. Tạp chí Nghiên cứu Học máy 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[36] Veselin Raychev, Martin Vechev, và Eran Yahav. 2014. Hoàn thành Mã với Mô hình Ngôn ngữ Thống kê. Trong Kỷ yếu của Hội nghị ACM SIGPLAN lần thứ 35 về Thiết kế và Thực hiện Ngôn ngữ Lập trình (PLDI '14). ACM, 419–428.

[37] Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco và Shuai Ma. 2020. CodeBLEU: một phương pháp tự động đánh giá tổng hợp mã. arXiv:2009.10297 [cs.SE]

[38] Peter C. Rigby và Christian Bird. 2013. Thực hành đánh giá ngang hàng phần mềm đương đại hội tụ. Trong Kỷ yếu của Cuộc họp chung lần thứ 9 năm 2013 về Sáng lập Kỹ thuật Phần mềm (ESEC/FSE 2013). 202–212.

[39] Peter C. Rigby, Daniel M. German, Laura Cowen, và Margaret-Anne Storey. 2014. Đánh giá ngang hàng về các dự án phần mềm nguồn mở: Thông số, Mô hình thống kê và lý thuyết. ACM Trans. Mềm mại. Động cơ Methodol. 23, 4 (2014).

[40] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, và Alberto Bacchelli. NĂM 2018. Đánh giá mã hiện đại: Nghiên cứu điển hình tại Google. Kỷ yếu Hội nghị Quốc tế lần thứ 40 về Kỹ thuật Phần mềm: Kỹ thuật Phần mềm trong Thực hành (ICSE-SEIP '18). 181190

[41] Shu-Ting Shi, Ming Li, David Lo, Ferdian Thung và Xuan Hoo. năm 2019. Xem xét mã tự động bằng cách tìm hiểu việc sửa đổi mã nguồn. Trong Kỷ yếu Hội nghị AAAI về Trí tuệ nhân tạo, Tập 33. 4910–4917.

[42] Devarshi Singh, Varun Ramachandra Sekar, Kathryn T Stolee, và Brittany John-son. 2017 Đánh giá cách các công cụ phân tích tĩnh có thể giảm nỗ lực xem xét mã. Năm 2017, IEEE đã tổ chức Hội thảo chuyên đề về Ngôn ngữ trực quan và tính toán con người - trung tâm (VL/HCC). IEEE, 101–105.

[43] Davide Spadini, Fabio Palomba, Tobias Baum, Stefan Hanenberg, Magiel Bruntink và Alberto Bacchelli. năm 2019. Đánh giá mã theo hướng thử nghiệm: một nghiên cứu thực nghiệm. Năm 2019, IEEE/ACM đã tổ chức Hội nghị quốc tế lần thứ 41 về Kỹ thuật phần mềm (ICSE). IEEE, 1061–1072.

[44] Michele Tufano, Dawn Drain, Alexey Svyatkovskiy, Shao Kun Deng, và Neel Sundaresan. 2020. Phát sinh ca kiểm tra đơn vị với máy biến áp. CoRR abs/2009.05617 (2020). https://arxiv.org/abs/2009.05617

[45] Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, và Denys Poshyvanyk. năm 2019. Một nghiên cứu thực nghiệm về tìm hiểu các bản vá lỗi cố định trong tự nhiên thông qua dịch máy thần kinh. ACM Trans. Mềm mại. Động cơ Methodol. 28, 4 (2019), 19:1–19:29.

[46] Rosalia Tufano, Luca Pascarella, Michele Tufano, Denys Poshyvanyk, và Gabriele Bavota. 11111111111111 Hướng tới Tự động hóa các Hoạt động Đánh giá Mã. Trong Hội nghị Quốc tế lần thứ 43 về Kỹ thuật Phần mềm, ICSE'21. https://arxiv.org/abs/ 2101.02518

[47] Yuriy Tymchuk, Andrea Mocci, và Michele Lanza. 2015. Đánh giá mã: Veni, vidi, vici. Năm 2015, IEEE đã tổ chức Hội nghị Quốc tế lần thứ 22 về Phân tích Phần mềm, Tiến hóa và Tái tạo (Saner). IEEE, 151–160.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, và Illia Polosukhin. 2017 Chú ý là tất cả những gì anh cần. Những tiến bộ trong hệ thống xử lý thông tin thần kinh. 5998–6008.

[49] Cody Watson, Nathan Cooper, David Nader Palacio, Kevin Moran, và Denys Poshyvanyk. 2020. A Systematic Literature Review on the Use of Deep Learning in Software Engineering Research. arXiv preprint arXiv:2009.06520 (2020).

[50] Cody Watson, Michele Tufano, Kevin Moran, Gabriele Bavota, và Denys Poshyvanyk. 2020. Về việc học các câu khẳng định ý nghĩa cho các trường hợp kiểm tra đơn vị. trong
ICSE '20: Hội nghị Quốc tế lần thứ 42 về Kỹ thuật Phần mềm, Seoul, Hàn Quốc, 27 tháng 6 - 19 tháng 7 năm 2020, Gregg Rothermel và Doo-Hwan Bae (Eds.). ACM, 1398–1409.

[51] Supatsara Wattanakriengkrai, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Hideaki Hata, và Kenichi Matsumoto. 2020. Dự đoán các đường bị lỗi bằng cách sử dụng mô hình - kỹ thuật chẩn đoán. CoRR (2020).