**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN TỐT NGHIỆP**

**Nghiên cứu đề xuất mô hình học tăng cường và giải pháp mất cân bằng dữ liệu cho bài toán định vị lỗi phần mềm**

**NGUYỄN HỮU KHẢI**

Khai.n168712@sis.hust.edu.vn

**Ngành: Công nghệ thông tin**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Bùi Thị Mai Anh |
| **Bộ môn:** | Công nghệ phần mềm |
| **Viện:** | Công nghệ thông tin – Truyền thông  Chữ ký của GVHD |
| **HÀ NỘI, 01/2022** | |

# Lời cam kết

Họ và tên sinh viên: Nguyễn Hữu Khải. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Điện thoại liên lạc: .0389764848 . . . . . . . . . . Email: khai.nh168712@sis.hust.edu.vn

Lớp: VUW IT 15. . . . . . .. . . . . . . . . . Hệ đào tạo: Kỹ sư chính quy. . . . . . . . . . . . . .

Tôi – *Nguyễn Hữu Khải* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *TS. Bùi Thị Mai Anh*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 15 tháng 1 năm 2022*  Tác giả ĐATN  Nguyễn Hữu Khải |

## ĐỀ TÀI TỐT NGHIỆP

Biểu mẫu của Đề tài/khóa luận tốt nghiệp theo qui định của viện, tuy nhiên cần đảm bảo giáo viên giao đề tài ký và ghi rõ họ và tên.

Trường hợp có 2 giáo viên hướng dẫn thì sẽ cùng ký tên.

![Shape

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAjUAAABaCAMAAAEA8WeaAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAADUExURQAAAKd6PdoAAAABdFJOUwBA5thmAAAACXBIWXMAACHVAAAh1QEEnLSdAAAASUlEQVR4Xu3BAQ0AAADCoPdPbQ43IAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADO1QDHTAAB9wbESAAAAABJRU5ErkJggg==)Giáo viên hướng dẫn

Ký và ghi rõ họ tên

# Lời cảm ơn

Đầu tiên, em xin được gửi lời cảm ơn chân thành đến các thầy giáo, cô giáo thuộc trường Đại học Bách Khoa Hà Nội, đặc biệt là các thầy giáo, cô giáo thuộc Viện Công nghệ Thông tin và Truyền thông đã tận tình dạy dỗ và trang bị cho em những kiến thức nền tảng của đồ án tốt nghiệp trong suốt những năm qua. Em xin gửi lời cảm ơn chân thành nhất đến TS Bùi Thị Mai Anh, giảng viên Viện Công nghệ Thông tin và Truyền thông người đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình làm đồ án tốt nghiệp. Cô không chỉ giỏi về chuyên môn, mà còn rất thân thiện với sinh viên, giúp em tiếp thu những kiến thức mới, chia sẻ những kinh nghiệm để em có thể hoàn thành đồ án tốt nghiệp một cách tốt nhất. Cuối cùng, em xin chân thành cảm ơn gia đình, bạn bè, tập thể lớp VUW IT 15, và các thành viên ISE lab đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên em trong suốt quá trình học tập và hoàn thành đồ án tốt nghiệp.

# Mục lục

[Lời cam kết 2](#_Toc97670063)

[Lời cảm ơn 3](#_Toc97670064)

[Mục lục 1](#_Toc97670065)

[Danh mục công thức 5](#_Toc97670066)

[Tóm tắt 8](#_Toc97670067)

[Chương 1 Giới thiệu đề tài 9](#_Toc97670068)

[1.1 Đặt vấn đề 9](#_Toc97670069)

[1.2 Các giải pháp hiện tại và hạn chế 10](#_Toc97670070)

[1.3 Mục tiêu và định hướng giải pháp 10](#_Toc97670071)

[1.4 Đóng góp của đồ án 11](#_Toc97670072)

[Chương 2 Những nghiên cứu liên quan 12](#_Toc97670073)

[2.1 Những hướng tiếp cận giải quyết mất cân bằng trước đây: 12](#_Toc97670074)

[2.1.1 DNN (Deep neural network): 12](#_Toc97670075)

[2.1.2 RUS (Re-sampling under-sampling) của nhóm tác giả Drummond C, Holte RC,2003: 12](#_Toc97670076)

[2.1.3 ROS (Re-sampling over-sampling) của nhóm tác giả Drummond C, Holte RC,2003: 12](#_Toc97670077)

[2.1.4 .Học tăng cường (Deep Reinforcement learning) của nhóm tác giả Enlu Lin, Qiong Chen, Xiaoming Qi: 13](#_Toc97670078)

[2.1.5 Ưu và nhược điểm của phương pháp tiếp cận: 13](#_Toc97670079)

[2.2 Thách thức đối với bài toán Bug Localization: 13](#_Toc97670080)

[Chương 3 Các kiến thức nền tảng 15](#_Toc97670081)

[3.1 Word Embedding 15](#_Toc97670082)

[3.1.1 Tần suất xuất hiện từ - Nghịch đảo tần suất văn bản (TF-IDF): 16](#_Toc97670084)

[3.1.2 Glove: 17](#_Toc97670089)

[3.2 Mạng Reinforcement Learning 18](#_Toc97670090)

[3.2.1 Tổng quan về mạng Reinforcement Learning 18](#_Toc97670091)

[3.2.2 Các thuật ngữ thông dụng trong RL 20](#_Toc97670094)

[3.2.3 Markov Decision Process (MDP) 21](#_Toc97670096)

[3.2.4 Q-value 22](#_Toc97670098)

[3.2.5 Hàm loss: 23](#_Toc97670102)

[3.2.6 Hàm kích hoạt: 24](#_Toc97670106)

[Chương 4 Giải pháp đề xuất 27](#_Toc97670111)

[4.1 Tổng quan áp dụng học tăng cường vào bài toán phân loại: 27](#_Toc97670112)

[4.2 Xử lý dữ liệu 28](#_Toc97670114)

[4.3 Chiến lược phân loại: 30](#_Toc97670115)

[4.4 Agent và cơ chế dự đoán khi tiếp nhận dữ liệu đầu vào: 31](#_Toc97670117)

[4.5 Chi tiết quá trình huấn luyện và giải pháp mất cân bằng dữ liệu: 32](#_Toc97670120)

[Chương 5 Đánh giá thực nghiệm 35](#_Toc97670123)

[5.1 Các phương pháp đánh giá độ chính xác của mô hình: 35](#_Toc97670124)

[5.1.1. Top-k Accuracy 35](#_Toc97670125)

[Công thức 13 Giá trị Top-k-accurancy 35](#_Toc97670126)

[5.1.2. Mean Reciprocal Rank (MRR) 35](#_Toc97670127)

[5.1.2. Mean Average Precision (MAP) 35](#_Toc97670129)

[5.1.4. Ví dụ minh họa 36](#_Toc97670133)

[5.2 Dữ liệu thử nghiệm: 37](#_Toc97670134)

[5.3 Quá trình thực nghiệm huấn luyện và dự đoán: 38](#_Toc97670137)

[5.3.1. Quá trình huấn luyện 38](#_Toc97670138)

[5.3.2. Quá trình dự đoán 38](#_Toc97670139)

[5.3.3. Thiết lập tham số cấu hình. 39](#_Toc97670140)

[5.4 Kết quả thực nghiệm: 40](#_Toc97670141)

[**Bảng 5** Kết quả mô hình trên các tập dữ liệu 40](#_Toc97670142)

[Chương 6 Kết luận 41](#_Toc97670143)

[6.1 Kết luận 41](#_Toc97670144)

[6.2 Hướng phát triển trong tương lai 41](#_Toc97670145)

[Tài liệu tham khảo 42](#_Toc97670146)

# Danh mục công thức

[**Công thức 1** Tần suất xuất hiện của một từ trong văn bản 17](#_Toc95807018)

[**Công thức 2** Giá trị idf của một từ trong tập văn bản 18](#_Toc95807019)

[**Công thức 3** Giá trị tf-idf của một từ trong tập văn bản 18](#_Toc95807020)

[**Công thức 4** Véc-tơ biểu diễn văn bản d trong mô hình không gian véc-tơ 18](#_Toc95807021)

[Công thức 5 Giá trị Q 24](#_Toc95807032)

[Công thức 6 Giá trị Temporal Difference 24](#_Toc95807033)

[Công thức 7 Giá trị Q-learning 24](#_Toc95807034)

[Công thức 8 Công thức tính hàm loss 24](#_Toc95807036)

[Công thức 9 Công thức tính y 25](#_Toc95807037)

[Công thức 10 Đạo hàm của hàm loss 25](#_Toc95807038)

[Công thức 11 Công thức hàm sigmoid 26](#_Toc95807041)

[Công thức 12 Công thức hàm ReLU 27](#_Toc95807043)

[Công thức 13 Giá trị Top-k-accurancy 36](#_Toc95807059)

[Công thức 14 Giá trị MRR 36](#_Toc95807061)

[**Công thức 15** Giá trị Mean Average Precision (MAP) 37](#_Toc95807064)

[**Công thức 16** Giá trị độ chính xác trung bình AvgPre 37](#_Toc95807065)

[**Công thức 17** Giá trị độ chính xác truy xuất trên top-k source files dự đoán 37](#_Toc95807066)

**Danh mục hình vẽ**

[**Hình 1** Các từ có liên hệ được biểu diễn gần nhau trong không gian véc-tơ 16](#_Toc95807016)

[**Hình 2** Mối liên hệ giữa AI, Machine learning và Deep learning (Ảnh: intel) 20](#_Toc95807025)

[Hình 3 3 loại chính của Học máy (Ảnh: Google) 21](#_Toc95807026)

[Hình 4 Tổng quan quá trình học tăng cường 22](#_Toc95807028)

[Hình 5 Mô phỏng quá trình Markov 23](#_Toc95807030)

[Hình 6 Biểu đồ tính toán hàm Sigmoid 26](#_Toc95807040)

[Hình 7 Biểu đồ tính toán hàm Relu 27](#_Toc95807042)

[Hình 8 Tổng quát quá trình áp dụng học tăng cường vào bài toán phân loại 29](#_Toc95807046)

[Hình 9 Chiến lược giải quyết mô hình học tăng cường 31](#_Toc95807049)

[Hình 10 Cấu trúc mạng nơ ron của Agent 32](#_Toc95807051)

[Hình 11 Quá trình lưu trữ và học tập của Agent 35](#_Toc95807055)

**Danh mục bảng biểu**

[**Bảng 1** Mã giả kỹ thuật sử dụng đồng thời explore và exploite 33](#_Toc95807492)

[**Bảng 2** Mã giả hàm tính Reward cho bài toán mất cân bằng dữ liệu 34](#_Toc95807494)

[**Bảng 3** Số lượng các bug reports trong bài toán 38](#_Toc95807509)

[**Bảng 4** Số lượng các mã nguồn trong bài toán 39](#_Toc95807510)

[**Bảng 5** Kết quả mô hình trên các tập dữ liệu 41](#_Toc95807516)

# Tóm tắt

Gỡ lỗi và định vị các tệp nguồn bị lỗi là những công việc tốn nhiều thời gian và phức tạp. Để cải thiện năng suất và giúp các nhà phát triển tập trung vào các tệp quan trọng, các mô hình định vị bug tự động đã được đề xuất trong nhiều năm. Các mô hình này đề xuất các tệp nguồn có lỗi bằng cách xếp hạng chúng theo mức độ liên quan của chúng với một báo cáo lỗi nhất định. Có hai thách thức đáng kể trong lĩnh vực nghiên cứu này: thu hẹp khoảng cách từ vựng giữa các báo cáo lỗi thường được mô tả bằng ngôn ngữ tự nhiên và tệp nguồn được viết bằng ngôn ngữ lập trình; giảm tác động của việc phân phối dữ liệu không cân bằng trong đào tạo mô hình vì ngày càng ít tệp nguồn liên quan đến một báo cáo lỗi nhất định trong khi phần lớn chúng không liên quan. Trong bài báo này, Em đề xuất một mô hình mạng học tăng cường để áp dụng vào bài toán nhằm giải quyết 2 vấn đề: trích xuất đặc trưng của dữ liệu văn bản và giải quyết vấn đề mất cân bằng. Để giải quyết sự mất cân bằng, em đã áp dụng mô hình học tăng cường, áp dụng cơ chế điểm thưởng để cân bằng độ quan trọng của dữ liệu giúp giải quyết vấn đề mất cân bằng. Em đã đánh giá hiệu suất của phương pháp tiếp cận của mình trên sáu dự án mã nguồn mở Java quy mô lớn.

# Giới thiệu đề tài

## Đặt vấn đề

Trong quá trình phát triển và bảo trì phần mềm, lỗi là không thể tránh khỏi. Việc quản lý hiệu quả các lỗi đóng một vai trò thiết yếu trong việc đảm bảo chất lượng phần mềm. Các hệ thống theo dõi sự cố như Jira \*, Bugzilla †, v.v. thường được sử dụng lý các để báo lưu cáo trữ lỗi và quản cung cấp thông tin quan trọng về lỗi phần mềm bao gồm mã hóa triệu chứng, bối cảnh môi trường cũng như các ràng buộc cụ thể để tái tạo lỗi. Để sửa một lỗi mới được báo cáo, các nhà phát triển phải phân tích cẩn thận báo cáo lỗi và xem xét một lượng lớn các tệp mã nguồn để tìm các tệp có lỗi tiềm ẩn. mong muốn giảm chi phí bảo trì và tăng tốc năng suất của cả nhóm phát triển phần mềm. Do đó, các hoạt động sửa lỗi đòi hỏi thời gian và công sức đáng kể, đặc biệt là trong các dự án phần mềm quy mô lớn với hàng nghìn tệp nguồn. Do đó, bản địa hóa lỗi tự động mong muốn giảm chi phí bảo trì và tăng tốc năng suất của cả nhóm phát triển phần mềm. Do đó, các hoạt động sửa lỗi đòi hỏi thời gian và công sức đáng kể, đặc biệt là trong các dự án phần mềm quy mô lớn với hàng nghìn tệp nguồn. Một số kỹ thuật bản địa hóa lỗi đã được đề xuất, chúng có thể được phân loại thành hai hướng: động và bản địa hóa lỗi tĩnh. Khác với các phương pháp tiếp cận động, trong đó các tệp bị lỗi được định vị bằng cách phân tích dấu vết thực thi và kiểm tra trạng thái thời gian chạy của hệ thống phần mềm các phương pháp tĩnh khoanh vùng lỗi bằng cách khám phá mối quan hệ giữa báo cáo lỗi và tệp nguồn mà không yêu cầu phiên bản có thể chạy được của hệ thống.

Xem xét báo cáo lỗi như một truy vấn và tệp nguồn là một tập hợp tài liệu, bản định vị lỗi có thể được giải quyết bằng cách sử dụng kỹ thuật Infor mation Retrieval (IR). Các cách tiếp cận này đo lường sự giống nhau về mặt từ vựng giữa một báo cáo lỗi đã cho và tất cả các tệp nguồn và tạo ra một tài liệu được xếp hạng cao nhất làm kết quả. Tuy nhiên, sự không khớp về mặt từ vựng giữa các ngôn ngữ tự nhiên được sử dụng để mô tả báo cáo lỗi và ngôn ngữ lập trình được sử dụng để viết mã nguồn là một trong những yếu tố chính hạn chế hiệu suất của các phương pháp dựa trên IR và gây ra thách thức lớn cho các mô hình bản địa hóa lỗi [7]. Để khắc phục vấn đề này, các nghiên cứu gần đây tập trung khai thác không chỉ thông tin từ vựng mà còn cả mối quan hệ ngữ nghĩa giữa báo cáo lỗi và tệp nguồn. Với việc áp dụng thành công các mô hình học sâu trong các lĩnh vực khác nhau, mạng nơ ron sâu (DNN), mạng nơ ron tái diễn (RNN), mạng nơ ron phức hợp (CNN) [7], [11] đã được sử dụng để trích xuất thông tin ngữ nghĩa từ các báo cáo lỗi và tệp nguồn

## Các giải pháp hiện tại và hạn chế

Trong dữ liệu của các dự án, với mỗi báo cáo lỗi sẽ liên quan tới nhiều tệp mã nguồn do đó bài toán định vị lỗi có thể được xem như là một bài toán phân loại và được chia làm 2 loại: có liên quan và không liên quan đến bug report.

Để giải quyết bài toán này, đa phần các nhà nghiên cứu đã đưa ra rất nhiều phương pháp và chủ yếu dựa trên nền tảng DNN (Deep Neural Network), mô hình mạng nhiều tầng. Và hướng tiếp cận chính được đưa ra là học giám sát. Dữ liệu đầu vào chia làm 2 cụm có liên quan và không liên quan tới báo cáo lỗi. Sau khi quá trình huấn luyện kết thúc, bài toán sẽ xếp hạng các cặp dữ liệu có liên quan với độ chính xác cao nhất để đánh giá mô hình chung. Với những dữ liệu có liên quan nhãn đầu vào sẽ là 1 còn đối với những cặp dữ liệu không liên quan nhãn sẽ là 0.

Đã có rất nhiều mô hình được đề xuất để giải quyết bài toán định vị lỗi phần mềm và hầu như đã đạt được hiệu quả khá cao trong việc xử lý dữ liệu đầu vào trích chọn đặc trưng dữ liệu. Dữ liệu đầu vào đã được thông qua các mô hình biểu diễn vector với những đặc trưng rất chặt chẽ mang đầy đủ tính chất của 1 cặp bug\_source. Nhưng còn 1 thách thức khác đó chính là sự mất cân bằng dữ liệu. Trong bài toán số lượng cặp không liên quan nhiều hơn rất nhiều so với những cặp có liên quan. Để giải quyết bài toán mất cân bằng, các bài toán trước đây đã cố gắng thay đổi số lượng dữ liệu đầu vào hoặc cải thiện hàm mất mát để làm sao có thể cải thiện được tham số cân bằng dữ liệu. Nhưng điều này gây ra hệ lụy khi ta thay đổi số lượng dữ liệu đầu vào có thể sẽ đánh mất đi những dữ liệu có tính chất quan trọng, có thể dẫn đến trạng thái overfitting và việc huấn luyện sẽ không được hiệu quả. Với vấn đề mất cân bằng này, em đã tập trung nghiên cứu mô hình học tăng cường dựa trên bài báo của Enlu Lin và đồng nghiệp về cách sử dụng mô hình học tăng cường cho bài toán phân loại cũng như sẽ lợi dụng cơ chế tính điểm của mô hình để làm cân bằng dữ liệu.

## Mục tiêu và định hướng giải pháp

Qua tìm hiểu và phân tích em đã đưa ra giải pháp sử dụng mô hình học tăng cường để đồng thời giải quyết 2 vấn đề: trích chọn đăng trưng của từng mẫu phân loại có liên quan, không liên quan và làm cân bằng dữ liệu theo cách tính độ quan trọng cho từng nhóm dữ liệu. Cụ thể, em đề xuất xây dựng mô hình học tăng cường để giải quyết các thách thức đã đề cập ở phần trước.

* Thứ nhất, liên quan đến kiến trúc mô hình, em đề xuất mô hình mạng học tăng cường, đây là cách tiếp cận mới, gồm các thành phần chính là môi trường (enviroment), chủ thể (agent), hành động đưa ra của chủ thể (action), mô hình mạng nơ-ron sẽ được tích hợp ở thành phần agent với cơ chế học q-value giúp tối ưu việc hiệu quả của việc học máy.
* Thứ hai, để giải quyết vấn đề mất cân bằng dữ liệu, em hướng tới xử lý theo hai hướng, thay đổi cơ chế đưa ra điểm thưởng (reward) cho từng action được chọn ra bởi agent để cân bằng mức độ quan trọng cho bộ dữ liệu.

## Đóng góp của đồ án

Đồ án tốt nghiệp có 2 đóng góp chính như sau:

1. Đồ án đề xuất mô hình học tăng cường để giải quyết đồng thời trích chọn đặc trưng và mất cân bằng dữ liệu.
2. Đồ án đề xuất thay đổi cơ chế tính điểm trong mô hình học tăng cường nhằm giải quyết việc mất cân bằng dữ liệu.­­­­­

# Những nghiên cứu liên quan

## Những hướng tiếp cận giải quyết mất cân bằng trước đây:

### DNN (Deep neural network):

Đây là giải pháp chung và đầu tiên khi tiếp cận bài toán phân loại. Ta sẽ sử dụng mạng nơ ron nhiều lớp với các hàm kịch hoạt để thay đổi số chiều của data đầu vào, sau đó sử dụng làm loss Cross Entropy để tiến hành huấn luyện mô hình. Mô hình này sẽ không sử dụng bất cứ phương pháp nào để làm cân bằng bộ dữ liệu đầu vào.

### RUS (Re-sampling under-sampling) của nhóm tác giả Drummond C, Holte RC,2003:

Under sampling là việc ta giảm số lượng các quan sát của nhóm đa số để nó trở nên cân bằng với số quan sát của nhóm thiểu số. Ưu điểm của under sampling là làm cân bằng mẫu một cách nhanh chóng, dễ dàng tiến hành thực hiện mà không cần đến thuật toán giả lập mẫu.

Tuy nhiên nhược điểm của nó là kích thước mẫu sẽ bị giảm đáng kể. Gỉa sử nhóm thiểu số có kích thước là 500, như vậy để tạo ra sự cân bằng mẫu giữa nhóm đa số và thiểu số sẽ cần giảm kích thước mẫu của nhóm đa số từ 10000 về 500. Tổng kích thước tập huấn luyện sau under sampling là 1000 và chiếm gần 1/10 kích thước tập huấn luyện ban đầu. Tập huấn luyện mới khá nhỏ, không đại diện cho phân phối của toàn bộ tập dữ liệu và thường dễ dẫn tới hiện tượng overfitting.

### ROS (Re-sampling over-sampling) của nhóm tác giả Drummond C, Holte RC,2003:

Over sampling là các phương pháp giúp giải quyết hiện tượng mất cân bằng mẫu bằng cách gia tăng kích thước mẫu thuộc nhóm thiểu số bằng các kĩ thuật khác nhau. Có 2 phương pháp chính để thực hiện over sampling đó là:

* Lựa chọn mẫu có tái lập.
* Mô phỏng mẫu mới dựa trên tổng hợp của các mẫu cũ.

.

### .Học tăng cường (Deep Reinforcement learning) của nhóm tác giả Enlu Lin, Qiong Chen, Xiaoming Qi:

Các thuật toán phân loại thông thường không hiệu quả trong trường hợp phân phối dữ liệu không cân bằng và có thể thất bại khi dữ liệu. phân phối mất cân đối cao. Để giải quyết vấn đề này, họ đã đề xuất một mô hình phân loại mất cân bằng chung dựa trên học tăng cường, trong đó chúng tôi hình thành vấn đề phân loại như một quá trình ra quyết định tuần tự và giải quyết nó bằng một mạng Q-learning sâu. Trong mô hình của chúng tôi, tác nhân thực hiện hành động phân loại trên một mẫu trong mỗi bước thời gian và môi trường đánh giá hành động phân loại và trả lại phần thưởng cho tác nhân. Phần thưởng từ tầng lớp thiểu số mẫu lớn hơn, do đó tác nhân nhạy cảm hơn với lớp thiểu số. Đại lý cuối cùng đã tìm ra một chính sách phân loại tối ưu trong dữ liệu không cân bằng dưới sự hướng dẫn của chức năng phần thưởng cụ thể và môi trường mô phỏng có lợi.

### Ưu và nhược điểm của phương pháp tiếp cận:

DNN được sử dụng làm mô hình cơ sở cho các phương pháp tiếp cận học tập về sự mất cân bằng. Ví dụ, ROS và RUS đã sử dụng DNN làm mô hình phân loại bằng cách lấy mẫu lại dữ liệu thông qua lấy mẫu quá mức lớp thiểu số hoặc lấy mẫu dưới lớp đa số. CSM đã đào tạo mô hình DNN bằng cách ấn định các chi phí phân loại sai khác nhau cho các lớp khác nhau trong đó biểu diễn chi phí sử dụng tần số lớp nghịch đảo. DTA đã đào tạo mô hình DNN và điều chỉnh ngưỡng quyết định của mô hình trong thời gian thử nghiệm. MFE, FL và CRL là một số cách tiếp cận học tập không cân bằng nổi bật được kết hợp với học sâu trong những năm gần đây. MFE và FL đã định hình lại tổn thất entropy chéo của mô hình DNN theo cách khác và CRL thực hiện khai thác mẫu cứng cho lớp thiểu số trong quá trình đào tạo mô hình DNN.

Những phương pháp trên chủ yếu sẽ thay đổi số lượng data trong quá trình huấn luyện nhằm cân bằng dữ liệu hoặc sẽ phải thay đổi hàm loss để đánh giá mức độ dộ quan trọng của bộ dữ liệu thiểu số so với đa số. Nếu như thay đổi quá nhiều ở bộ dữ liệu sẽ dẫn đến việc thiếu đi những tính đặc trưng của bộ data hoặc sẽ dẫn đến tình trạng overfitting trong quá trình huấn luyện. Riêng chỉ có hướng giải quyết của nhóm tác giả Enlu Lin đã sủ dụng mô hình học tăng cường, lợi dụng cơ chế đánh giá chỉ số quan trọng của dữ liệu để giải quyết bài toán mất cân bằng dữ liệu trong học mát mà không cần thay đổi số lượng dữ liệu ban đầu hay tinh chỉnh hàm loss của bài toán.

## Thách thức đối với bài toán Bug Localization:

Thách thức lớn đối với bài toán định vị lỗi đó chính là sự mất cân bằng trong dữ liệu. Trong một dự án sẽ có hàng ngàn những tệp mã nguồn và mỗi 1 báo cáo lỗi. Các mã nguồn có liên quan đến các file báo cáo lỗi sẽ ít hơn rất nhiều so với những file không liên quan và điều này sẽ gây ra sự mất cân bằng dẫn tới quá trình huấn luyện sẽ không được hiệu quả. Để giải quyết vấn đề trên, em đã dựa trên hướng nghiên cứu học tăng cường của nhóm tác giả Enlu Lin. Bằng phương pháp này, mô hình sẽ giải quyết đồng thời hai vấn đề: trích chọn đặc trưng của dữ liệu và giải quyết vấn đề mất cân bằng của dữ liệu mà không phải can thiệp vào số lượng dữ liệu ban đầu hay thay đổi hàm mất mát của mô hình.

# Các kiến thức nền tảng

## Word Embedding

Word Embedding là một không gian vector dùng để biểu diễn dữ liệu có khả năng miêu tả được mối liên hệ, sự tương đồng về mặt ngữ nghĩa, văn cảnh(context) của dữ liệu. Không gian này bao gồm nhiều chiều và các từ trong không gian đó mà có cùng văn cảnh hoặc ngữ nghĩa sẽ có vị trí gần nhau. Ví dụ như ta có hai câu: “Hôm nay ăn táo” và “Hôm nay ăn xoài “. Khi ta thực hiện Word Embedding, “táo” và “xoài” sẽ có vị trí gần nhau trong không gian chúng ta biễu diễn do chúng có vị trị giống nhau trong một câu.

Diagram

Description automatically generated

**Hình 1** Các từ có liên hệ được biểu diễn gần nhau trong không gian véc-tơ

Ví dụ minh họa như trong Hình 1, các cặp từ “Con trai” và “Con gái”, “Học sinh” và “Trường học” có mỗi liên quan gần tới nhau, do đó chúng được biểu diễn gần nhau trong không gian véc-tơ. Do đó khi sử dụng word embedding, tất cả các từ riêng lẻ được biểu diễn như các véc-tơ có giá trị thực trong một không gian véc-tơ xác định. Mô hình word embedding sẽ được huấn luyện với một tập từ vựng cho trước. Có hai mô hình được sử dụng rộng rãi các năm gần đây: Word2Vec và Glove. Ngoài ra còn có một số phiên bản khác như Sentence2Vec, Doc2Vec nhưng em sẽ không trình bày trong chương này vì nằm ngoài bài toán định vị lỗi phần mềm. Gần đây nhất mô hình BERT. được giới thiệu như là một mô hình mới nhất để biểu diễn véc-tơ từ nhưng do quy mô của BERT khá lớn và không phù hợp để ứng dụng vào bài toán định vị lỗi nên em cũng không đề cập. Lí do BERT không phù hợp là do lượng từ vựng trong các bug report và source files không lớn và khá đặc trưng do mang sắc thái của dự án phần mềm. Việc dùng BERT là không cần thiết vì các mô hình Word2Vec hay Glove cũng đã được huấn luyện trên các tập từ vựng tương đối lớn và đủ để có thể sử dụng trong bài toán định vị lỗi.

### Tần suất xuất hiện từ - Nghịch đảo tần suất văn bản (TF-IDF):

TF-IDF (Term Frequency – Inverse Document Frequency) là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu. Một vài biến thể của tf-idf thường được sử dụng trong các hệ thống tìm kiếm như một công cụ chính để đánh giá và sắp xếp văn bản dựa vào truy vấn của người dùng.  Tf-idf cũng được sử dụng để lọc những từ stopwords trong các bài toán như tóm tắt văn bản và phân loại văn bản.

Term Frequency(Tần suất xuất hiện của từ) là số lần từ xuất hiện trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, term frequency thường được chia cho độ dài văn bản( tổng số từ trong một văn bản):

**Công thức 1** Tần suất xuất hiện của một từ trong văn bản

Trong đó:

tf(t,d) là tần suất xuất hiện từ t trong văn bản d,

là số lần xuất hiện từ t trong văn bản d,

là tổng số từ trong văn bản d.

IDF: Inverse Document Frequency(Nghịch đảo tần suất của văn bản), giúp đánh giá tầm quan trọng của một từ . Khi tính toán TF , tất cả các từ được coi như có độ quan trọng bằng nhau. Nhưng  một số từ như “is”, “of” và “that” thường xuất hiện rất nhiều lần nhưng độ quan trọng là không cao. Như thế chúng ta cần giảm độ quan trọng của những từ này xuống.

**Công thức 2** Giá trị idf của một từ trong tập văn bản

Trong đó:

idf(t,D) là giá trị idf của từ t trong tập văn bản

|D| là tổng số văn bản trong tập D

là số văn bản trong tập D chứa từ t

Cơ số logarit trong công thức này không thay đổi giá trị idf của từ mà chỉ thu hẹp khoảng giá trị của từ đó. Vì thay đổi cơ số sẽ dẫn đến việc giá trị của các từ thay đổi bởi một số nhất định và tỷ lệ giữa các trọng lượng với nhau sẽ không thay đổi. (nói cách khác, thay đổi cơ số sẽ không ảnh hưởng đến tỷ lệ giữa các giá trị IDF). Việc sử dụng logarit nhằm giúp giá trị tf-idf của một từ nhỏ hơn, do chúng ta có công thức tính tf-idf của một từ trong 1 văn bản là tích của tf và idf của từ đó.

Cụ thể, chúng ta có công thức tính tf-idf hoàn chỉnh như sau:

**Công thức 3** Giá trị tf-idf của một từ trong tập văn bản

Những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó).

Giá trị của TF-IDF nằm trong khoảng từ 0 tới 1. Sau khi tính toán, những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này mà ít xuất hiện trong văn bản khác. Các giá trị TF-IDF của các từ trong văn bản d tạo thành véc-tơ biểu diễn văn bản:

**Công thức 4** Véc-tơ biểu diễn văn bản d trong mô hình không gian véc-tơ

Trong đó: là giá trị tf-tdf của từ t trong văn bản d được tính theo Công thức .

### Glove:

Glove (Global Vector for Word Representation) là một phương pháp phổ biến tương tự Word2Vec để biểu diễn các véc-tơ từ. Mô hình Glove biểu diễn phân phối các từ. Mô hình là một thuật toán học không giám sát để đạt được biểu diễn véc-tơ cho các từ **Error! Reference source not found.**. Điều này đạt được bởi việc ánh xạ các từ trong một không gian véc-tơ mà trong đó khoảng cách giữa các từ liên quan tới độ tương quan ngữ nghĩa giữa chúng. Quá trình huấn luyện được thực hiện trên một tập dữ liệu thống kê đồng xuất hiện (co-occurrence) của các từ. Nó dựa trên cơ sở của các kỹ thuật phân tích nhân tử ma trận (matrix factorization) trên ma trận ngữ cảnh từ (word-context). Một ma trận lớn của thông tin đồng xuất hiện (co-occurrence information) được xây dựng trong đó mỗi hàng gồm các từ và tần suất thấy từ này trong một số ngữ cảnh (ở các cột) trong tập dữ liệu lớn. Thông thường, mô hình quét tập dữ liệu theo cách thức như sau: với mỗi từ, nó tìm kiếm các từ ngữ cảnh trong một số khu vực xác định bởi một cửa sổ trước từ và một cửa sổ sau từ. Ngoài ra, các từ ở xa sẽ có trọng số thấp hơn. Vì số lượng ngữ cảnh là lớn, nên mô hình cần thiết tổ hợp các cửa sổ theo kích thước. Vì vậy, ma trận này được phân tích thành các ma trận số chiều nhỏ hơn, trong đó mỗi hàng biểu diễn véc-tơ của mỗi từ. Mô hình Glove được phát triển như một dự án mã nguồn mở tại đại học Stanford, nước Mỹ **Error! Reference source not found.**.

Glove có thể được sử dụng để tìm sự liên quan giữa các từ đồng nghĩa, các mối quan hệ giữa các từ như trường học – sinh viên, … Glove được sử dụng bên trong mô hình Spacy**Error! Reference source not found.** để xây dựng véc-tơ ngữ nghĩa word embedding hay các véc-tơ đặc trưng. Glove cũng được sử dụng để tính toán các từ đầu danh sách (top list) bởi độ đo khoảng cách như độ tương quan hàm cosine hoặc khoảng cách Euclidean.

## Mạng Reinforcement Learning

### Tổng quan về mạng Reinforcement Learning

“Machine learning (ML) is a process whereby a computer program learns from experience to improve its performance at a specified task” (Kiran, 2020). Hiểu đơn giản machine learning (học máy) là kỹ thuật giúp cho máy tính có thể tự học và đưa ra quyết định mà không cần phải cài đặt các quy tắc, luật lệ. Machine learning đang là lĩnh vực công nghệ được quan tâm nhiều nhất hiện nay. Nó ngày càng được ứng dụng vào thực tế cuộc sống, từ các ứng dụng mạng xã hội, thương mại điện tử hay marketing… tạo ra những giá trị to lớn cho các dịch vụ này.

Diagram

Description automatically generated

**Hình 2** Mối liên hệ giữa AI, Machine learning và Deep learning (Ảnh: intel)

Các thuật toán học máy thường được phân thành 3 loại lớn: supervised learning (học có giám sát), unsupervised learning (học không giám sát) và reinforcement learning (học tăng cường). Nếu như supervised learning là học tập từ một tệp các dữ liệu được gắn nhãn để suy luận ra quan hệ giữa đầu vào và đầu ra, thì unsupervised learning không được cung cấp các dữ liệu được gắn nhãn ấy, thay vào đó chỉ được cung cấp dữ liệu mà thuật toán tìm cách mô tả dữ liệu và cấu trúc của chúng. Loại thứ 3 là reinforcement learning - phương pháp tập trung vào việc làm thế nào để cho một tác tử trong môi trường có thể hành động sao cho lấy được phần thưởng nhiều nhất có thể. Khác với học có giám sát, học tăng cường không có cặp dữ liệu gán nhãn trước làm đầu vào và cũng không có đánh giá các hành động là đúng hay sai.

“Reinforcement learning là đào tạo các mô hình học máy để đưa ra một chuỗi các quyết định. Tác tử học cách đạt được mục tiêu trong một môi trường không chắc chắn, có thể là phức tạp.”

Diagram, venn diagram

Description automatically generatedĐến đây ta có thể thấy, reinforcement learning là một nhánh của machine learning.

### Hình 3 3 loại chính của Học máy (Ảnh: Google)

### ****Các thuật ngữ thông dụng trong RL****

Trong reinforcement learning có rất nhiều các thuật ngữ khác nhau.

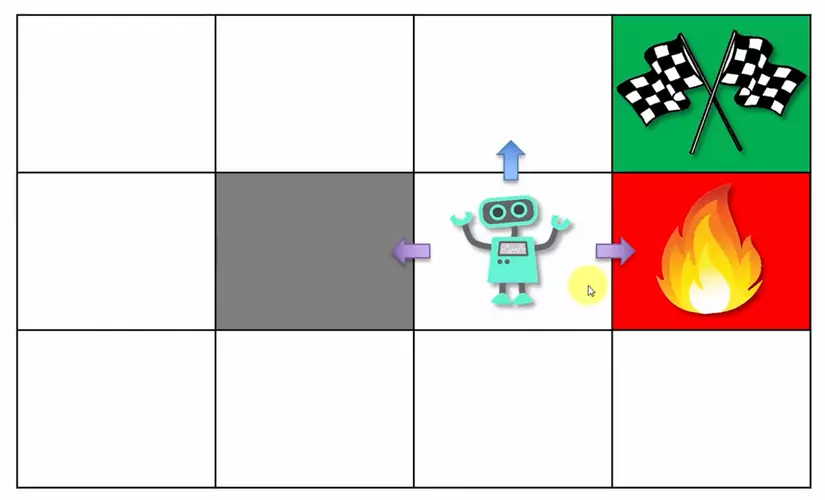
* **Agent** - được định nghĩa là “anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators” (máy quan sát môi trường và sinh ra hành động tương ứng).
* **Environment** - Môi trường là không gian xung quanh của agent, nơi mà agent tồn tại và tương tác
* Action - Hành động là phương thức của agent cho phép nó tương tác với môi trường và thay đổi môi trường. Dựa trên State S(t) của environment hiện tại mà agent sẽ đưa ra action a(t).
* **Observation** - Sau khi nhận được sự tương tác từ agent thì environment có sự chuyển đổi trạng thái đối với agent
* **State** - trạng thái của môi trường mà agent nhận được
* **Policy** - Chính sách là yếu tố xác định cách thức hoạt động của agent tại một thời điểm nhất định. Nói cách khác, chính sách là một ánh xạ từ các trạng thái (state) của môi trường đến các hành động sẽ được thực hiện khi ở trong các trạng thái đó. Chính sách là cốt lõi của agent trong việc xác định hành vi. Trong một số trường hợp, chính sách có thể là một hàm hoặc bảng tra cứu đơn giản. Trong một số trường hợp khác, chính sách có thể liên quan đến tính toán mở rộng, ví dụ như quá trình tìm kiếm.
* **Diagram, timeline

  Description automatically generatedReward** - Ở mỗi hành động, môi trường gửi đến cho agent một phần thưởng xác định. Mục tiêu của agent là tối đa hóa tổng phần thưởng mà nó nhận được trong một thời gian dài. Tín hiệu phần thưởng (reward signal) giúp xác định đâu là sự kiện tốt và xấu đối với agent, đồng thời nó cũng là cơ sở chính để thay đổi chính sách. Nếu một hành động được lựa chọn bởi chính sách mang đến phần thưởng thấp, thì chính sách đó có thể bị thay đổi. Agent sẽ lựa chọn các hành động khác trong các tình huống tương tự ở tương lai

### Hình 4 Tổng quan quá trình học tăng cường

### Markov Decision Process (MDP)

MDP là một framework giúp agent đưa ra quyết định tại một state nào đó. Để áp dụng được framework này, ta giả sử các states có thuộc tính Markov (Markov Property): mỗi state chỉ phụ thuộc vào state trước nó vào xác suất chuyển đổi giữa 2 states này. Giả sử agent đang ở vị trí như hình dưới, việc nó đến đó bằng cách nào không quan trọng; nó có thể vòng từ trên bức tường xuống hoặc chui từ dưới lên hoặc thậm chí chạy sang trái sang phải 1000 lần trước khi đến nơi. Dù là cách nào thì khi agent đã ở đó, xác suất đi lên, sang trái/phải vẫn giống nhau. Có thể hiểu đơn giản Markov property là tính "không nhớ" (memoryless).



### Hình 5 Mô phỏng quá trình Markov

Với Markov property, chúng ta có thể áp dụng MDP để biểu diễn bài toán dưới dạng:

(S, A, R, P)

Trong đó: S là tập các states, A là tập các actions, R là reward nhận được khi chuyển state, P là phân bố xác suất chuyển đổi.

Việc biểu diễn này rất quan trọng, khi mà ta không phải lưu một chuỗi các states trước đó để biểu diễn state hiện tại khiến cho việc tính toán trở nên phức tạp và tiêu tốn bộ nhớ.

### ****Q-value****

Q-learning là một thuật toán học-tăng-cường off policy. Nó được gọi là off policy vì thuật toán q learning học từ các hành động được thực hiện ngẫu nhiên, tức là không có trước một policy (\*\*\* policy hiểu theo thuật toán Markov thì có thể dịch là chính sách hoặc chiến lược) nào cả. Cụ thể, q-learning học để tìm ra được một chiến lược (policy) có thể tối đa hoá được tổng phần thưởng (maximizes the total reward).

Chữ Q đại diện cho chất lượng (Quality). Chất lượng trong trường hợp này là sự biểu thị của độ tốt của việc thực hiện một hành động (action) tới việc làm tăng phần thưởng trong dài hạn.

### Công thức 5 Giá trị Q

Trong đó Q(s, a) là Q-value khi thực hiện action a tại state s; r(s, a) là reward nhận được; s' là state kế tiếp. γ là hệ số discount, đảm bảo càng "xa" đích Q-value càng nhỏ

Công thức này cho thấy Q-value của action a tại state s bằng reward r(s,a) cộng với Q-value lớn nhất của các states s' tiếp theo khi thực hiện các action a. Vậy đó, chỉ với công thức đơn giản kia chúng ta có thể tạo ra một ma trận state-action như một lookup table. Từ đó với mỗi state agent chỉ cần tìm action nào có Q-value lớn nhất là xong. Tuy nhiên, như mình đã nói ở trên thì RL là một stochastic process nên Q-value ở thời điểm trước và sau khi thực hiện action sẽ khác nhau. Khác biệt này gọi là Temporal Difference:

### Công thức 6 Giá trị Temporal Difference

Như vậy, ma trận Q(s, a) cần phải cập nhật trọng số dựa trên TD:

Trong đó α là learning rate. Qua các lần agent thực hiện actions, Q(s, a) sẽ dần hội tụ. Quá trình này chính là Q-Learning:

### Công thức 7 Giá trị Q-learning

### ****Hàm loss:****

### Công thức 8 Công thức tính hàm loss

Trong đó y là mục tính toán của hàm Q, cách tính y theo công thức dưới đây:

### Công thức 9 Công thức tính y

Trong đó, s’ là state tiếp theo của s, a’ là hành được thực hiện bởi agent ở state s’, t=1 nếu như terminal = True, ngược lại t = 0

Đạo của hàm của hàm loss với công thức thứ 8 sẽ là:

### Công thức 10 Đạo hàm của hàm loss

Bây giờ chúng ta có thể có được hàm Q ∗ tối ưu bằng cách giảm thiểu hàm mất mát, và quy luật giảm dần theo hàm Q ∗ tối ưu sẽ nhận được phần thưởng tích lũy tối đa. Vì vậy quy luật phân loại tối ưu π ∗: S → A cho ICMDP đạt được.

### ****Hàm kích hoạt:****

Một hàm kích hoạt là một hàm ánh xạ các đầu vào của một nút với đầu ra tương ứng của nó. Điều này có ý nghĩa với hình minh họa chúng ta đã thấy trong bài trước về các [layers](https://tek4.vn/layers-trong-mang-neural/). Chúng ta lấy tổng trọng số của mỗi kết nối đến cho mỗi node trong layer và truyền tổng trọng số đó cho một hàm kích hoạt. Hàm kích hoạt thực hiện một số loại hoạt động để biến đổi tổng thành một số. Phép biến đổi này thường là một phép biến đổi phi tuyến tính.

Trong RL các hàm kích hoạt được sử dụng để cấu tạo nên các lớp mạng trong thành phần agent.

#### Chart, line chart Description automatically generatedHàm sigmoid

### Hình 6 Biểu đồ tính toán hàm Sigmoid

Sigmoid là một hàm phi tuyến với đầu vào là các số thực và cho kết quả nằm trong khoảng [0,1] và được xem là xác xuất trong một số bài toán. Trong hàm Sigmoid, một sự thay đổi nhỏ trong input dẫn đến một kết quả output ko mấy thay đổi. Vì vậy, nó đem lại một đầu ra "mượt" hơn và liên tục hơn so với input.

Công thức của hàm Sigmoid:

### Công thức 11 Công thức hàm sigmoid

Hàm sigmoid là một hàm liên tục và đạo hàm của nó cũng khá đơn giản, dẫn đến việc áp dụng hàm vào mô hình mạng đem lại sự dễ dàng trong việc xây dựng mô hình và cập nhật tham số dựa trên back-propagation. Một điểm đáng chú ý của hàm Sigmoid, khiến nó trở nên nổi bật trong thời gian gần đây lại là điểm "bất lợi" của nó.

#### Hàm relu

### Hình 7 Biểu đồ tính toán hàm Relu

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron. ReLU đơn giản lọc các giá trị < 0. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó. Một số ưu điểm khá vượt trội của nó so với Sigmoid và Tanh:

* Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh ([Krizhevsky et al.](http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf)). Điều này có thể do ReLU không bị bão hoà ở 2 đầu như Sigmoid và Tanh.
* Tính toán nhanh hơn. Tanh và Sigmoid sử dụng hàm exp và công thức phức tạp hơn ReLU rất nhiều do vậy sẽ tốn nhiều chi phí hơn để tính toán.
* Tuy nhiên ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là “Dying ReLU“. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent. => Leaky ReLU ra đời.
* Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neuron dừng việc cập nhật.

### Công thức 12 Công thức hàm ReLU

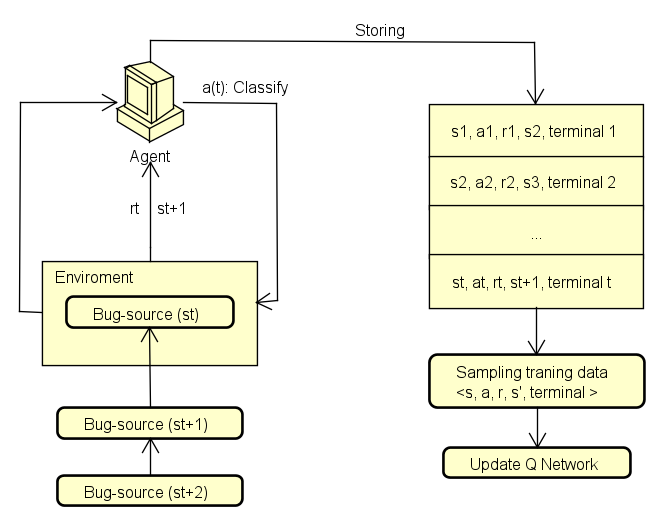
# Giải pháp đề xuất

## Tổng quan áp dụng học tăng cường vào bài toán phân loại:

Các nghiên cứu trước đây về phân loại dữ liệu không cân bằng chủ yếu tập trung vào hai cấp độ: cấp độ dữ liệu và cấp độ thuật toán.

Các phương pháp mức dữ liệu nhằm mục đích cân bằng sự phân bố lớp bằng cách thao tác với các mẫu huấn luyện, bao gồm lấy nhiều mẫu hơn với lớp thiểu số, lấy ít mẫu hơn với lớp đa số và sự kết hợp của hai phương pháp trên. Tuy nhiên, việc lấy mẫu quá nhiều có thể dẫn đến việc overft khi học máy trong khi lấy mẫu quá ít có thể làm mất thông tin có giá trị về nhóm đa số.

Các phương pháp cấp độ thuật toán nhằm mục đích nâng cao tầm quan trọng của lớp thiểu số bằng cách cải thiện các thuật toán hiện có, bao gồm học tập nhạy cảm với chi phí, học tập theo nhóm và điều chỉnh ngưỡng quyết định. Các phương pháp học tập nhạy cảm với chi phí ấn định các chi phí phân loại sai khác nhau cho các lớp khác nhau bằng cách sửa đổi hàm mất mát, trong đó chi phí phân loại sai của lớp thiểu số cao hơn chi phí phân loại sai của lớp đa số. Các phương pháp dựa trên học tập tổng hợp đào tạo nhiều bộ phân loại phụ riêng lẻ, sau đó sử dụng biểu quyết hoặc kết hợp để có được kết quả tốt hơn. Các phương pháp điều chỉnh ngưỡng huấn luyện bộ phân loại trong dữ liệu không cân bằng ban đầu và thay đổi ngưỡng quyết định trong thời gian thử nghiệm. Một số phương pháp dựa trên học sâu gần đây đã được đề xuất để phân loại dữ liệu không cân bằng.

Học tăng cường gần đây đã đạt được kết quả xuất sắc trong các nhiệm vụ phân loại vì nó có thể hỗ trợ các nhà phân loại tìm hiểu các tính năng có lợi hoặc chọn các phiên bản chất lượng cao từ dữ liệu nhiễu. Trong, nhiệm vụ phân loại được xây dựng thành một quy trình ra quyết định tuần tự, sử dụng đa các tác nhân để tương tác với môi trường để tìm hiểu chính sách phân loại tối ưu. Tuy nhiên, mô phỏng thực tế giữa các tác nhân và môi trường gây ra độ phức tạp về thời gian cực kỳ cao. Em đề xuất một mô hình dựa trên học tập củng cố sâu để tìm hiểu sự phân loại mối quan hệ trong dữ liệu văn bản nhiễu. Mô hình được chia thành bộ chọn phiên bản và bộ phân loại quan hệ.  Dựa trên nghiên cứu của Enlu Lin và đồng nghiệp em đã mô phỏng lại với bài toán nhận diện Bug (Bug Localiztion):

### Hình 8 Tổng quát quá trình áp dụng học tăng cường vào bài toán phân loại

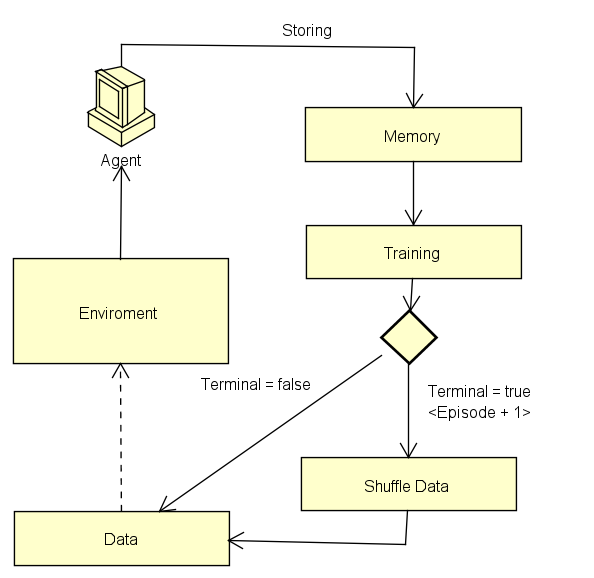
## Xử lý dữ liệu

Dữ liệu ban đầu sẽ có 2 phần: source code và bug report. Một báo cáo lỗi thu được từ hệ thống theo dõi lỗi thường bao gồm một bản tóm tắt và một mô tả. Trước tiên, Họ kết hợp chúng thành một tài liệu dưới dạng trình bày văn bản của báo cáo lỗi, sau đó được phân tích cú pháp thành các từ sử dụng khoảng trắng và được lọc để loại bỏ dấu chấm câu và các từ dừng tiêu chuẩn (ví dụ: “a”, “the”). Việc xử lý trước các tệp nguồn phải loại bỏ các từ khóa của ngôn ngữ lập trình (tức là từ khóa java), số và dấu chấm câu. Để giảm bớt sự dư thừa của một khối lượng lớn các dòng mã, Họ xây dựng mô tả văn bản của tệp nguồn bằng cách chỉ trích xuất comments và mã định danh của các lớp / phương thức cũng như thuộc tính. Ví dụ, UserName nên được chia thành User và Name, sau đó, tất cả các từ được chuyển sang dạng gốc để giảm thiểu các từ có liên quan đến nghĩa gốc.

Gọi B = {r1, r2, ..., rN} biểu thị tập các báo cáo lỗi và S = {s1, s2, ..., sM} là tập hợp các tệp nguồn của dự án phần mềm trong đó N và M là tổng số báo cáo lỗi và tệp nguồn, tương ứng. Họ trích xuất năm đặc điểm sau cho một cặp (r, s) với r ∈ B và s ∈ S.

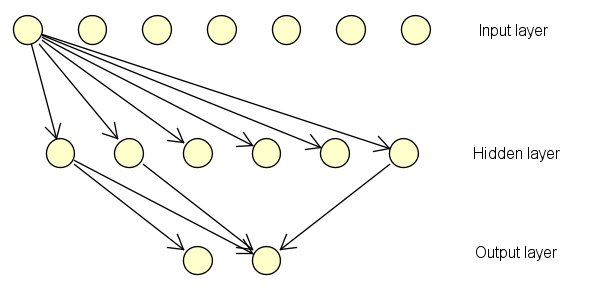
1. Lexical Similarity: Lấy cảm hứng từ kỹ thuật Mô hình không gian vectơ (VSM), tính toán sự giống nhau về mặt từ vựng dựa trên tần suất các mã thông báo xuất hiện trong báo cáo lỗi (Tần suất thuật ngữ-TF) và số lượng tệp nguồn nơi các mã thông báo này xuất hiện (Tần suất tài liệu nghịch đảo IDF). Sự giống nhau về mặt từ vựng giữa hai vectơ VSM Vs và Vr.
2. Semantic Similarity: Để giải quyết sự không khớp về mặt từ vựng giữa các báo cáo lỗi và các tệp nguồn liên quan của chúng, áp dụng kỹ thuật nhúng từ để khám phá mối quan hệ ngữ nghĩa giữa các từ dựa trên ngữ cảnh mà chúng xuất hiện. Mô hình GloVe được đào tạo trước để xây dựng các vectơ nhúng từ cho các báo cáo lỗi và tệp nguồn.
3. Similar Bug Reports: Như đã chỉ ra trong các nghiên cứu gần đây, các lỗi tương tự có thể liên quan đến các tệp nguồn tương tự. Do đó, sự giống nhau giữa các báo cáo lỗi có thể được sử dụng như một chỉ báo để tìm các tệp nguồn bị lỗi liên quan.
4. Code Change History: Các nghiên cứu gần đây đã thừa nhận rằng những thay đổi được thực hiện trong các tệp nguồn có lỗi để giải quyết các báo cáo lỗi trước đó có thể trở thành lý do của lỗi mới sắp xuất hiện các báo cáo. Điều này thúc đẩy em sử dụng lịch sử thay đổi mã nguồn để dự đoán hiệu quả các tệp nguồn dễ bị lỗi. Thật vậy, một tệp nguồn đã được sửa rất lâu trong quá khứ hoặc không bao giờ thay đổi ít có khả năng chứa lỗi hơn một tệp đã được thay đổi gần đây. Sau đó, Họ đánh giá tầm quan trọng của tệp nguồn s ∈ S trong điều kiện sửa chữa lần truy cập gần đây cho một báo cáo lỗi r ∈ B nhất định bằng cách tính thời gian đã trôi qua (tính bằng ngày) giữa ngày cam kết cuối cùng của s và ngày mà r được báo cáo.
5. Bug Fixing Frequency: Tần suất mà tệp nguồn đã được sửa cũng có thể được sử dụng như một chỉ báo để đo lường mức độ dễ xảy ra lỗi của nó. Với một cặp <r, s> của báo cáo lỗi r và tệp nguồn s, điểm tần suất sửa lỗi của nó được xác định là số lần tệp nguồn s đã được sửa trong một khoảng thời gian của µ ngay trước khi xuất hiện báo cáo lỗi r.

## Chiến lược phân loại:

Trong mô hình học, số lượng data theo thứ tứ ban đầu sẽ được đưa vào qua mỗi Episode. Các phần tử con trong data sẽ được đẩy vào lần lượt từ phần tử đầu tiên vào Enviroment để Agent dự đoán. Cứ mỗi lần Agent dự đoán sai vòng lặp sẽ dừng lại và chuyển sang Episode tiếp theo, data lúc này sẽ được thay đổi thứ theo cơ chế ngẫu nhiên và quá trình học tập sẽ được diễn ra từ phần tử ban đầu của data theo thứ tự mới. Bằng phương pháp này agent sẽ thích ứng được nhiều trạng thái mới giúp cải thiện sự hiệu quả trong quá trình học. Dưới đây là mô hình chiến lược tổng quát:

### Hình 9 Chiến lược giải quyết mô hình học tăng cường

## Agent và cơ chế dự đoán khi tiếp nhận dữ liệu đầu vào:

Bản chất Agent sẽ giống như cấu trúc mạng nơ ron thông thường cấu tạo gồm có các tầng mạng nhiều lớp với các hàm kích hoạt. Dưới đây là cấu trúc mô tả mô hình mạng trong Agent.

### Hình 10 Cấu trúc mạng nơ ron của Agent

Đầu vào của Agent sẽ là 6 trích xuất đặc trưng mà em vừa đề cập tới ở phần trước, các tham số sẽ thông qua các hớp mạng và cho đầu ra là 1 mảng 2 phần tử. Khác với các bài toán phân loại thông thường, mô hình học tăng cường luôn luôn cho đầu ra là 1 mảng 2 phần tử và sẽ sử dụng giá trị cao hơn làm kết quả dự đoán, tương ứng với số lượng hành động của agent khi lựa chọn để dự đoán cho bộ dữ liệu đầu vào.

Mô hình reinforment sẽ đề xuất cho hành động theo 2 cách: thăm dò (explore) và khai thác (exploit)

Một tác nhân tương tác với môi trường (environment) theo 1 trong 2 cách. Tác nhân sẽ chọn hành động dựa trên giá trị tối đa của những hành động đó. Đây được gọi là khai thác (exploit) vì chúng ta sử dụng thông tin có sẵn để đưa ra một quyết định (make a decision).

Cách thứ hai để hành động là hành động ngẫu nhiên. Đây được gọi là thăm dò (explore). Thay vì chọn các hành động dựa trên phần thưởng tối đa trong tương lai, chúng ta chọn một hành động một cách ngẫu nhiên. Hành động ngẫu nhiên rất quan trọng vì nó cho phép tác nhân thăm dò và khám phá các trạng thái mới mà không được chọn trong quá trình khai thác. Em sẽ cân bằng việc thăm dò / khai thác bằng cách sử dụng epsilon  ϵ và đặt giá trị của tần suất thăm dò/khai thác. Dưới đây là 1 đoạn code mô tả cách mà agent thực hiện hành động khi được cân bằng cho cả explore và exploit.

**Bảng 1** Mã giả kỹ thuật sử dụng đồng thời explore và exploite

|  |
| --- |
| 1.Khởi tạo giá trị Epsilon =0.5  2. if random(0,1) < Epsilon  3. Chọn ra hành động ngẫu nhiên không phụ thuộc vào dữ liệu  4. else:  5. Chọn ra hành dộng với giá trị cao nhất phụ thuộc vào bộ dữ liệu trong tương lai  6. ….  7. Epsilon = Epsilon \* Episilon |

Sau khi hoàn tất 1 quá trình học tập cho 1 trạng thái, tham số epsilon này sẽ được tự nhân với chính nó, vì là số thập phân nên các qua trình học xảy ra, tham số này càng giảm, trường hợp đưa ra tham số ngẫu nhiên sẽ ít được xảy ra hơn và càng phụ thuộc vào tác nhân đã thiết lập sẵn trong mô hình.

## Chi tiết quá trình huấn luyện và giải pháp mất cân bằng dữ liệu:

Sau khi tác nhân đưa ra được động dự đoán cho trạng thái ở thời điểm hiện tại, môi trường đánh giá xem dữ liệu mà tác nhân đưa ra đúng hay sai. So với dữ liệu test, nếu tác nhân dự đoán sẽ được điểm cộng và sai sẽ bị điểm trừ. Đặt vào trường hợp của bài toán hiện tại đang mất cân bằng dữ liệu, cụ thể là số lượng dữ liệu nhãn có liên quan ít hơn số lượng dữ liệu không liên quan khá nhiều nên em đã đề xuất 1 cơ chế tính điểm mới nhằm nâng tầm quan trọng của nhãn có liên quan phù hợp với tỉ lệ mất cân bằng dữ liệu có trong bài toán. Dưới đây là mô tả cơ chế tính điểm của mô hình.

Ω: tỉ lệ mất cân bằng dữ dữ liệu thiểu số và đa số

Dp: đại diện cho dữ liệu thiểu số

**Bảng 2** Mã giả hàm tính Reward cho bài toán mất cân bằng dữ liệu

|  |
| --- |
| 1.Khởi tạo giá trị terminalt = false  2. if(st ϵ Dp):  3. if at = lt then  4. set rt = 1  5 else  6. set rt = -1  7. terminal = True  8. else:  9. if at  = lt then  10. set rt = Ω  11. else:  12. set rt = -Ω  14. return rt, terminalt |

Tham số terminal có kiểu boolean sẽ là 1 phần cần thiết cho việc chắt lọc các dữ liệu tốt để đưa vào học máy phía sau.

Khi môi trường nhận thấy tác nhân đánh giá 1 dữ liệu thiểu số, môi trường sẽ đưa ra số điểm tối đa khi tác nhân dự đoán đúng còn khi dự đoán sai sẽ trừ đúng với số điểm tối đa. Với dữ liệu đa số số điểm được đánh giá bằng với tỉ lệ giữa dữ liệu đúng với dữ liệu sai. Điều này sẽ giúp độ quan trọng của dữ liệu thiểu số sẽ được tăng lên đáng kể và phù hợp với tỉ lệ ban đầu của tập dữ liệu.

Khi terminal = true tức là đã kết thúc 1 episode, dữ liêu trước đó sẽ được thêm vào memory để tiến hành huấn luyện. Các dữ liệu sau này, môi trường sẽ thay vị trí của data sau mỗi episode để có thể tạo ra nhiều trạng thái khác nhau nhằm tối ưu khả năng học tập của agent. Khi môi trường đã cung cấp tới data cuối cùng mà số lượng vòng lặp episode vẫn còn, môi trường sẽ thực hiện lại quá trình reset và sẽ lại tạo ra bộ data mới với thứ tự mới.

Diagram

Description automatically generatedMôi trường mô phỏng theo định nghĩa của ICMDP. Kiến trúc của mạng Q phụ thuộc vào độ phức tạp và số lượng của tập dữ liệu huấn luyện. Đầu vào của mạng Q phù hợp với cấu trúc của các mẫu huấn luyện, và số lượng đầu ra bằng số loại mẫu. Trên thực tế, mạng Q là một bộ phân loại mạng nơ-ron không có lớp softmax cuối cùng. Thuật toán Q-learning sâu sẽ chạy khoảng 500 lần lặp (cập nhật các tham số mạng θ). Mạng Q hội tụ sau đó thêm một lớp softmax có thể được coi là một bộ phân loại mạng nơ-ron được đào tạo bởi dữ liệu không cân bằng. Dưới đây là mô tả chi tiết quá trình huấn luyện.

### Hình 11 Quá trình lưu trữ và học tập của Agent

Dữ liệu sẽ được lưu trữ tại 1 mảng memory có kích thước đủ lớn để tạo ra Q-table nhằm thực hiện tính Q-value ở các bước sau. Sau đó, mô hình sẽ lấy các giá trị trong bộ nhớ để xem xét tính toán cho Agent. Với những kết quả sai với dữ liệu gốc, em sẽ lấy ngay điểm đánh giá của dữ liệu đó và gán trực tiếp cho tham số y. Đối với những dữ liệu đúng và gần đúng, em sẽ đem tính Q-value nhằm tối ưu hóa việc học máy cho các dữ liệu sau này. Tất cả quá trình trên sẽ được diễn ra cho đến hết 500 episodes.

# Đánh giá thực nghiệm

## Các phương pháp đánh giá độ chính xác của mô hình:

### 5.1.1. Top-k Accuracy

Top-k-accurancy là chỉ số biểu hiện phần trăm các báo cáo lỗi mà chúng xuất hiện ít nhất 1 lần dự đoán đúng trong k tệp được dự đoán.

### Công thức 13 Giá trị Top-k-accurancy

là số lượng dữ liệu báo cáo lỗi được sử dụng để thử nghiệm mô hình.

Ví dụ: Top 1 accuracy là kết quả những file cao nhất trong các file đã được dự đoán bởi mô hình

### 5.1.2. Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) là một trong những metrics đơn giản nhất trong việc đánh giá các ranking models. MRR tính trung bình của các thức hạng tương ứng của mục liên quan đầu tiên đối với tập các truy vấn B, có thể địn nghĩa nó như sau

### Công thức 14 Giá trị MRR

Trong đó là số lượng tệp báo cáo lỗi có trong mục dự đoán,

là vị trí hạng đầu tiên tệp các mã nguồn liên quan đầu trong danh sách k tệp trong mục dự đoán tương ứng với file báo cao lỗi thứ i.

### 5.1.2. Mean Average Precision (MAP)

Mean Average Precision: Trung bình cộng của tất cả các kết quả thử nghiệm của mô hình

**Công thức 15** Giá trị Mean Average Precision (MAP)

**Công thức 16** Giá trị độ chính xác trung bình AvgPre

là số lượng các file báo cáo lỗi trong quá trình đánh giá

K là tập các file tệp mã nguồn có liên quan trong danh sách k

là độ chính xác truy xuất (retrieval precision) trên top-k source files trong danh sách dự đoán theo công thức:

**Công thức 17­­­­­** Giá trị độ chính xác truy xuất trên top-k source files dự đoán

Mỗi file báo cáo lỗi sẽ có 1 hoặc nhiều hơn hơn các tệp mã nguồn có liên quan, MAP sẽ cho ra các tệp mã nguồn này ở vị trí đầu tiên trong danh sách dự đoán của mô hình không, giá trị này sẽ giao động từ 0 đến 1 và giá trị càng cao thì sự hiệu quả của mô hình càng tốt.

### 5.1.4. Ví dụ minh họa

Trong phần này em sẽ đưa ra ví dụ cụ thể cho cách đánh giá của mô hình dựa trên các công thức trên. Vì 1 báo cáo lỗi sẽ liên quan đến nhiều tệp mã nguồn và giả sử mô hình đã dự đoán được 10 cặp báo cáo lỗi và tệp mã nguồn và kết quả trong đó sẽ cho ra như sau:

A\_result = {a, X, X, X, a, X, X, X, X, X}

B\_result = {X, b, X, b, X, b, X, X, X, X}

Các mảng dự đoán trên đã được sắp xếp theo thứ tự từ cao nhất đến thấp nhất.

A\_result là các dữ liệu dự đoán của các cặp với báo cáo lỗi A và các tệp liên quan với A, trong đó có a là cặp dự đoán có liên quan. Tương tự như B\_result với b là cặp dự đoán có liên quan

Các tham số đánh giá sẽ được tính như sau:

Top\_1\_acurancy: Giá trị cao nhất của báo cáo lỗi A là 1 cặp bug\_src có liên quan đến nhau còn đối với B thì không cho nên giá trị top\_1\_accurancy = ½ =50%

Top\_2\_acurancy: Giá trị cao nhất của 2 báo cáo lỗi A có chứa ít nhất 1 cặp bug\_src có liên quan đến nhau còn đối với B cũng như vậy cho nên giá trị top\_2\_accurancy = 1/1 =100%

Top\_5\_acurancy: Giá trị cao nhất của 5 báo cáo lỗi A có chứa ít nhất 1 cặp bug\_src có liên quan đến nhau còn đối với B thì không cho nên giá trị top\_5\_accurancy = 1/1 = 100%

MRR: Được tính toán dựa theo giá trị cao nhất có liên quan của 2 báo cáo lỗi A và B: (1/2 +1/1)/2 = 0,75

MAP: Được tính dựa theo tất cả các giá trị có liên quan của cả 2 mảng giá trị A và B:

((1/2+2/5)/2 + (1/1 +2/4 +3/6)/3 = 0,56

## Dữ liệu thử nghiệm:

Dữ liệu huấn luyện mô hình em sẽ sử dụng của 4 dự án lớn thống kê dưới đây:

|  |  |  |
| --- | --- | --- |
| Tên dự án | Thời gian triển khai | Số lượng báo cáo lỗi |
| **AspectJ** | 01/2003-12/2013 | 593 |
| **Tomcat** | 10/2004-01/2014 | 1056 |
| **Eclipse UI** | 10/2001-12/2013 | 6495 |
| **SWT** | 02/2002-01/2014 | 4151 |

**Bảng 3** Số lượng các bug reports trong bài toán

Ta có thể thấy được dữ liệu báo cáo lỗi của dự án Eclipse là lớn nhất (6495) hơn gần gấp 10 lần so với dữ liệu báo cáo lỗi của AspectJ (593)

Các tệp mã nguồn của 4 dự án trên cũng được sử dụng và dưới đây là thống kê:

|  |  |  |
| --- | --- | --- |
| Tên dự án | Thời gian triển khai | Số lượng mã nguồn |
| **AspectJ** | 03/2002-01/2014 | 4439 |
| **Tomcat** | 07/2002-01/2014 | 1552 |
| **Eclipse UI** | 10/2001-01/2014 | 3454 |
| **SWT** | 02/2002-01/2014 | 2056 |

**Bảng 4** Số lượng các mã nguồn trong bài toán

Ta có thể thấy số lượng tệp mã nguồn nhiều nhất có ở trong dự án AspectJ (4439) và số lượng ít nhất thuộc về dự án (Tomcat)

## Quá trình thực nghiệm huấn luyện và dự đoán:

### 5.3.1. Quá trình huấn luyện

Từ kiến trúc mô hình em đã đề cập, dữ liệu đầu vào cho mồ hình sẽ là các cặp báo cáo lỗi và mã nguồn liên quan. Thông qua các thuật toán tính véc tơ đặc trưng và độ đo gốc tương tổng em đã đề cập ở chương 4 để đưa vào mô hình huấn luyện. Với mỗi cặp bug\_src có liên quan đầu ra mong muốn sẽ là 1 còn với những cặp không liên quan sẽ là 0.

Với những cặp bug\_src có liên quan, em chỉ cần lấy các thông tin bên trong báo cáo lỗi với những tệp mã nguồn tương ứng sẽ cho ra những cặp bug\_src có liên quan, Với nhưng cặp không liên quan em sẽ cho báo cáo lỗi đó ghép cặp với những tệp mã nguồn khác. Vì số lượng ghép cặp khác tương đối lớn cho nên em sẽ lấy ra 20 cặp không liên quan có chỉ số vec tơ đặc trưng thấp nhất để cho ra dữ liệu sai.

### 5.3.2. Quá trình dự đoán

Mô hình sẽ cho ra kết quả là 1 mảng có 2 phần tử tương ứng với số lượng mà hành động mà Agent đưa ra cụ thể là 0 và 1. Đầu ra của lớp mạng là hàm kích hoạt Softmax cho nên sẽ đưa ra 2 tham số xác xuất, nếu hành động nào có xác suất cao hơn sẽ là hành động chính để đưa ra kết quả dự đoán. Với bài toán này em chỉ quan tâm đến xác xuất đưa ra hành động đúng (hành động xác xuất bằng 1). Nếu như tham số xác suất này gần 1 thể hiện dữ liệu dự đoán là đúng và ngược lại nếu gần 0 sẽ đưa ra cặp dữ liệu này là không đúng.

### 5.3.3. Thiết lập tham số cấu hình.

Em đã sử dụng bộ dữ liệu đã đề cập ở trên để tiến hành huấn luyện mô hình. Sau khi tiền xử lý dữ liệu, bộ dữ liệu có khối lượng khá lớn nên em đã sử dụng phương pháp chia fold để chia dữ liệu ra các phần nhỏ. Mục đích sẽ lấy ra 1 fold để đánh giá và 9 fold còn lại để huấn luyện mô hình và cách làm này sẽ đi từ fold 1 cho đến fold 10. Bằng phương pháp này em có thể đánh giá tổng thể mô hình và tất cả các dữ liệu đều được kiểm thử.

Kiến trúc mạng nơ ron được cấu hình dựa trên thư viện Keras. Keras là thữ viện hỗ trợ sử dụng thiết lập các lớp mạng có trong mô hình mạng 1 cách dễ dàng. Đầu vào của Agent là lớp input với 5 tham số tương ứng với 5 tính đặc trưng của bộ data. Tiếp theo là 2 lớp mạng với số nút lần lượt là 300 và 150, 2 lớp mạng này sử dụng hàm kích hoạt ReLu. Lớp mạng cuối cùng là gồm 2 nút tương ứng với 2 hành đồng mà Agent cần đưa ra sử dụng hàm kích hoạt Softmax để tính xác suất cho ra của từng hành động. Khi agent sẽ có hàm loss là common loss em đã định nghĩa ở trên.

Mô hình sẽ được huấn luyện theo cơ chế vòng lặp với số lượng Episode là 500, con số này là lý tưởng để khám phá và khai thác đủ lớn cho toàn bộ dữ liệu. Imbalance rate sẽ được tính toán cho mỗi 1 fold theo tỷ lệ giữa dữ liệu đúng và sai.

Em đã sử dụng server chạy trên nền tảng Linux với chip 64 nhân và ram 128gb

## Kết quả thực nghiệm:

Bảng dưới đây sẽ hiển thị các kết quả của mô hình với các tập dữ liệu em đã nêu ở trên:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tên dự án | Top-k Accuracy (%) | | | | | | | MRR | MAP |
| **1** | **2** | **3** | **4** | **5** | **10** | **15** |
| **AspectJ** | 51.4 | 67.6 | 76.1 | 80 | 82.7 | 88 | 90.4 | 0.65 | 0.50 |
| **Tomcat** | 52.1 | 64.4 | 70 | 74.9 | 77.2 | 84.5 | 87.8 | 0.63 | 0.58 |
| **Eclipse UI** | 47.0 | 61.1 | 67.7 | 72.9 | 75.6 | 83.3 | 86.7 | 0.62 | 0.53 |
| **SWT** | 40.1 | 53.7 | 63.1 | 68.2 | 72.3 | 83.7 | 87.1 | 0.54 | 0.50 |

**Bảng 5** Kết quả mô hình trên các tập dữ liệu

Theo như kết quả ở bảng trên ta có thể thấy kết quả top 1 cao nhất với project Tomcat là 52,1% các cặp báo cáo đúng và kết quả top 1 thấp nhất của project SWT là 40,1%. Đối với top 15 thì kết quả cao nhất ở project Aspect là 90,4% và top-15 thấp nhất là 87,1% của project SWT. Như vậy ta có thể thấy sự hiệu quả của mô hình là khá cao, để so sánh với những dữ liệu đúng của tập test và dữ liệu dự đoán của mô hình, những dữ liệu dự đoán đúng mà mô hình dự đoán đã có khoảng 50% kết quả nằm ở những trị và cao nhất.

# Kết luận

## Kết luận

Gỡ lỗi và định vị các tệp nguồn bị lỗi là những công việc tốn nhiều thời gian và phức tạp. Để cải thiện năng suất và giúp các nhà phát triển tập trung vào các tệp quan trọng, các mô hình định vị bug tự động đã được đề xuất trong nhiều năm. Các mô hình này đề xuất các tệp nguồn có lỗi bằng cách xếp hạng chúng theo mức độ liên quan của chúng với một báo cáo lỗi nhất định. Có hai thách thức đáng kể trong lĩnh vực nghiên cứu này: thu hẹp khoảng cách từ vựng giữa các báo cáo lỗi thường được mô tả bằng ngôn ngữ tự nhiên và tệp nguồn được viết bằng ngôn ngữ lập trình; giảm tác động của việc phân phối dữ liệu không cân bằng trong đào tạo mô hình vì ngày càng ít tệp nguồn liên quan đến một báo cáo lỗi nhất định trong khi phần lớn chúng không liên quan. Trong bài báo này, Họ đề xuất một mô hình mạng nơron sâu để điều tra thông tin thiết yếu ẩn trong các báo cáo lỗi và tệp nguồn thông qua việc thu thập không chỉ các quan hệ từ vựng mà còn cả các chi tiết ngữ nghĩa cũng như các tính năng kiến thức miền như sửa lỗi lịch sử, lịch sử thay đổi mã. Để giải quyết sự phân bố lớp bị lệch, em đã áp dụng mô hình học tăng cường, áp dụng cơ chế điểm thưởng để cân bằng độ quan trọng của dữ liệu giúp giải quyết vấn đề mất cân bằng.

Trong quá trình làm đồ án, em đã nắm được nhiều kinh nghiệm và kiến thức quan trọng về cách giải quyết bài toán định vị lỗi phần mềm và các cách chắt lọc trích xuất đặc trưng của dữ liệu văn bản. Về quá trình thực nghiệm, em đã bắt gặp vấn đề mất cân bằng dữ liệu và đã được tiếp cận sâu hơn về mô hình học tăng cường, áp dụng nó cho bài toán phân loại và cụ thể hơn là giải quyết bài toán mất cân bằng dữ liệu. Những kinh nghiệm này đã giúp đỡ em rất nhiều cho các công việc và dự án trong tương lai.

## Hướng phát triển trong tương lai

Hiện tại trong quá trình huấn luyện còn hạn chế rất nhiều về mặt thời gian, do đó trong tương lai em muốn cải thiện hơn về tốc độ huấn luyện và tối ưu hóa về mặt phần cứng. Về mặt mô hình cũng chưa thực sự triệt để nên em muốn cải thiện độ chính xác hơn để có thể so sánh với các mô hình phân loại và các phương pháp giải quyết mất cân bằng khác.

# Tài liệu tham khảo

1. Drummond C, Holte RC et al (2003) C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Workshop on Learning from Imbalanced datasets II, vol 11, Citeseer, pp 1–8
2. Wang S, Liu W, Wu J, Cao L, Meng Q, Kennedy PJ (2016) Training deep neural networks on imbalanced data sets, in Neural Networks (IJCNN). In: 2016 International Joint Conference on. IEEE, pp 4368–4374
3. Zhou Z-H, Liu X-Y (2006) Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans Knowl Data Eng 18(1):63–77
4. Chen J, Tsai C-A, Moon H, Ahn H, Young J, Chen C-H (2006) Decision threshold adjustment in class prediction. SAR QSAR Environ Res 17(3):337–352
5. Lin T-Y, Goyal P, Girshick R, He K, Dollar P (2017) Focal loss for ´ dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp 2980–2988
6. Dong Q, Gong S, Zhu X (2018) Imbalanced deep learning by minority class incremental rectification. IEEE Transactions on Pattern Analysis and Machine Intelligence
7. Enlu Lin, Qiong Chen, Xiaoming Qi (2020) Deep reinforcement learning for imbalanced classification