TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

TRẦN XUÂN QUÝ – TRẦN HỮU CHÍ BẢO

KẾT HỢP CÁC ĐỐI TƯỢNG VÀ ĐẶC TRƯNG LÂN CẬN CHO BÀI TOÁN PHÂN LỚP ĐA NHÃN

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng 02/2022

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

TRẦN XUÂN QUÝ – 18120231

TRẦN HỮU CHÍ BẢO – 18120288

KẾT HỢP CÁC ĐỐI TƯỢNG VÀ ĐẶC TRƯNG LÂN CẬN CHO BÀI TOÁN PHÂN LỚP ĐA NHÃN

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN

GS.TS. Lê Hoài Bắc

Tp. Hồ Chí Minh, tháng 02/2022

Lời cảm ơn

Trước tiên, chúng em xin bày tỏ lòng biết ơn sâu sắc với GS.TS Lê Hoài Bắc, thầy đã trực tiếp hướng dẫn và hỗ trợ nhóm trong suốt quá trình thực hiện khóa luận. Khóa luận này chắc chắn không thể hoàn thành nếu không có sự tận tâm của thầy.

Chúng em chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Đại học Quốc gia Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng tôi trong 4 năm học tập tại trường và thực hiện khóa luận.

Chúng em cũng rất biết ơn gia đình và bạn bè, những người đã giúp đỡ, động viên chúng em rất nhiều trong lúc chúng tôi gặp khó khăn cũng như trong suốt quá trình thực hiện khóa luận.

Tuy đã cố gắng thực hiện hết sức mình trong phạm vi, kiến thức và khả năng cho phép nhưng các thiếu sót, sai lầm là điều không thể tránh khỏi. Chúng em mong nhận được các góp ý cho bài khóa luận này.

Một lần nữa, chúng em xin chân thành cảm ơn và mong nhận được sự chỉ bảo của quý thầy cô để khóa luận tốt nghiệp được hoàn chỉnh tốt nhất.

Thành phố Hồ Chí Minh, tháng 02 năm 2022

Nhóm sinh viên thực hiện

Trần Xuân Quý – Trần Hữu Chí Bảo

Đề cương chi tiết

Tên đề tài: Kết hợp các đối tượng và đặc trưng lân cận cho bài toán phân lớp đa nhãn (Combining instance and feature neighbours for multi-label classfication).
Giáo viên hướng dẫn: GS.TS Lê Hoài Bắc.
Thời gian thực hiện: 6 tháng.
Sinh viên thực hiện: Trần Xuân Quý – 18120231 Trần Hữu Chí Bảo – 18120288
Loại đề tài: Nghiên cứu
Nội dung đề tài: Bắt nguồn từ yêu cầu giải quyết bài toán phân loại văn bản - với mỗi văn bản thường thuộc nhiều hơn một phân lớp và bài toán chẩn đoán y khoa, gần đây ngày càng nhiều ứng dụng, bài toán có nhu cầu sử dụng các thuật toán phân lớp đa nhãn, như là phân loại chức năng protein, thể loại âm nhạc và ngữ nghĩa khung cảnh của ảnh. Hai hướng tiếp cận chính giải quyết các bài toán phân đa lớp hiện có gồm chia nhỏ bài toán thành tổ hợp những tác vụ phân lớp đơn nhãn và chỉnh sửa các thuật toán phân lớp đơn sao cho phù hợp với tác vụ đa lớp. Một số mô hình gần đây xét thêm yếu tố phụ thuộc tiềm tàng cũng như là phân bố lệch giữa các nhãn sử dụng phương pháp giảm chiều không gian hay xây dựng nhóm các mô hình cây có thứ bậc. Cách tiếp cận này cho độ chính xác và thời gian thực thi nhanh khi kiểm tra nhưng đòi hỏi lượng tài nguyên cho phần huấn luyện là rất lớn, cũng như yêu cầu phải tối ưu nhiều siêu tham số.
Kế hoạch thực hiện: Mỗi thành viên trong nhóm phải nắm được bản chất ý tưởng xây dựng mô hình KNN dựa trên đối tượng và đặc trưng cũng như thuật toán đề xuất LCIF. Các

thành viên phải có trách nhiệm truyền đạt kiến thức cho nhau mà mình tìm hiểu chính để mọi thành viên cùng nắm vững được vấn đề.

Thời gian và nội dung thực hiện cụ thể:

- Tháng 9/2021 – 10/2021: Tìm hiểu về bài toán, mục đích giải, mô hình được đề xuất và các tập dữ liệu.
- Tháng 10/2021 – 11/2021: Cài đặt các thuật toán instance-based và features-based KNN trên môi trường lập trình python.
- Tháng 11/2021 – 12/2022: Cài đặt thuật toán LCIF và instance-knn-fast sử dụng scipy.spare và thực hiện kiểm tra cài đặt trên các tập dữ liệu với các độ đo đánh giá và thời gian thực thi.
- Tháng 12/2021 – 02/2022: Tìm cách khắc phục tình trạng tràn bộ nhớ khi chạy với các tập dữ liệu cực nặng và đưa ra một số giải pháp nhằm cải thiện chất lượng của mô hình.
- Tháng 02/2022 – 03/2022: Viết báo cáo luận văn và chuẩn bị cho bảo vệ đề tài.

Xác nhận của GVHD	**NHÓM SINH VIÊN THỰC HIỆN** ***(Ký và ghi rõ họ tên)***

Mục lục

Danh mục hình

Danh mục bảng

Tóm tắt

Các vấn đề phân loại đa nhãn xảy ra tự nhiên trong nhiều lĩnh vực, bắt nguồn từ yêu cầu giải quyết bài toán phân loại văn bản - với mỗi văn bản yêu cầu là gán một tập hợp các chủ đề hoặc nhãn cho mỗi tài liệu kiểm tra trong một kho văn bản và hay như trong lĩnh vực sinh tin học, yêu cầu dự đoán tất cả chức năng có thể có của một mẫu gen... Tuy nhiên, so với học đơn nhãn thì học đa nhãn đòi hỏi những tiến hóa mới đối với các phương pháp học máy nhằm đưa ra giải pháp thích hợp với dữ liệu đa nhãn như vấn đề về mối quan hệ giữa các nhãn, chi phí tính toán của thuật toán, vấn đề mất cân bằng nhãn bởi vì nhãn chỉ xuất hiện ở một vài đối tượng, vấn đề đa chiều của dữ liệu... Một bài toán liên quan tới phân lớp đa nhãn là hệ thống tư vấn, thực hiện đề nghị một số sản phẩm cho một người dùng dựa trên sự tương đồng với những người khác hay những sản phẩm trong hệ thống. Điểm lợi của hệ thống tư vấn chính là chỉ cần một hệ thống là có thể dự đoán và đề nghị một số lượng sản phẩm đến người dùng. Áp dụng vào bài toán phân lớp đa nhãn, coi đối tượng là người dùng và mô hình đề nghị một số nhãn cho người dùng đó có thể phần nào giải quyết các vấn đề trên do không yêu cầu huấn luyện mô hình tổ hợp gồm nhiều thành phần cũng như việc tính độ tương đồng đã xét tới dữ liệu nhiều chiều và mối quan hệ giữa các chiều với nhau. Ở đây, nhóm áp dụng kỹ thuật phân lớp k lân cận gần nhất (kNN), lần lượt áp dụng thuật toán lọc cộng tác dựa trên người dùng (đối tượng) và dựa trên sản phẩm (đặc trưng) thay cho dự đoán dựa trên khoảng cách, sau cùng tổng hợp tuyến tính kết quả 2 thuật toán trên, thu được ma trận gồm các véc-tơ điểm dự doán các nhãn cho các đối tượng. Ngoài ra thành phần k lân cận dựa trên đối tượng còn áp dụng kỹ thuật xử lý từng đối tượng trong truy hồi thông tin, bỏ qua một số đối tượng có thể có độ tương đồng thấp mà không cần tính độ tương đồng. Nhìn chung 2 thuật toán kNN trên được đánh giá là hiệu quả hơn về mặt tính toán so với những mô hình xuất hiện trước cũng như có thể hoạt động tốt với phần cứng thông dụng.

Chương 1

Giới thiệu tổng quan về đề tài

1.1 Lý do nghiên cứu

Các vấn đề phân loại nhiều nhãn xảy ra tự nhiên trong nhiều lĩnh vực. Bắt nguồn từ yêu cầu giải quyết bài toán phân loại văn bản, yêu cầu của bài toán là gán một tập hợp các chủ đề hoặc nhãn cho mỗi tài liệu kiểm tra trong một kho văn bản. Trong phân loại scence, mục tiêu là gán một tập hợp các nhãn cho mỗi hình ảnh trong tập thử nghiệm chẳng hạn như hình ảnh một sân vườn có thể chứa một đứa trẻ, một cái xích đu và một cái cây, và nhiệm vụ là dự đoán tập hợp các đối tượng tương ưng với các nhãn này. Trong tin sinh học, nhiều ứng dụng yêu cầu mục tiêu là gắn một tập hợp các nhãn, chẳng hạn như dự đoán tất cả các chức năng của gen. Hai nhiệm vụ chính của bài toán phân lớp đa nhãn chính là xếp hạng các nhãn và phân loại chúng. Mô hình xếp hạng (multi-label ranking) đưa ra điểm tin cậy cho mỗi nhãn, trong khi thuật toán phân loại (classification algorithm) tạo ra sự phân chia hai nhóm nhãn: liên quan và không liên quan.

Bài toán phân lớp đa nhãn được chia thành hai hướng giải quyết chính là tiếp cận chuyển đổi bài toán và tiếp cận thích nghi thuật toán [1]. Tiếp cận chuyển đổi bài toán là phương pháp chia nhỏ bài toán đa nhãn thành một hoặc nhiều bài toán phân lớp đơn nhãn (phân lớp đa lớp hoặc phân lớp nhị phân) hoặc các bài toán hồi quy. Theo hướng tiếp cận này, các bộ phân lớp đơn nhãn được huấn luyện sau đó các dự đoán phân lớp đơn nhãn được kết hợp lại thành kết quả dự đoán đa nhãn. Phương pháp này khá phù hợp đối với các bài toán dữ liệu ít và nhãn không có mối liên quan và có thể sử dụng linh hoạt nhiều phương pháp để phân lớp đơn nhãn như SVM, Naive Bayes, kNN và Neural network. Tiếp cận thích nghi thuật toán là phương pháp mở rộng các thuật toán phân lớp đơn đã biết sao cho phù hợp với tác vụ đa lớp. Một số mô hình tiếp cận này xét thêm yếu tố phụ thuộc như sự phân bố lệch giữa các nhãn.

Cách tiếp cận này cho độ chính xác và thời gian thực thi nhanh khi kiểm tra nhưng lại đòi hỏi lượng tài nguyên cho chi phí tính toán ở quá trình huấn luyện là rất lớn, cũng như yêu cầu phải tối ưu nhiều siêu tham số hơn.

Trong bài khóa luận này, nhóm lựa chọn đề tài "kết hợp các đối tượng và đặc trưng lân cận cho bài toán phân lớp đa nhãn". Ở đây, nhóm áp dụng kỹ thuật phân lớp k-NN cho bài toán phân loại đa nhãn dựa trên kỹ thuật chọn phần tử láng giềng trên một số tập dữ liệu đời thực đã được tiền xử lý túi từ BoW (Bag of Word) nhằm giảm chi phí tính toán. Dựa trên đối tượng và đặc trưng, lần lượt áp dụng các tính năng của thuật toán lọc cộng tác dựa trên người dùng và dựa trên từng mục và sau cùng là tổ hợp tuyến tính của 2 thuật toán trên. Nhìn chung 2 thuật toán KNN trên được đánh giá là hiệu quả hơn về mặt tính toán so với những mô hình xuất hiện trước cũng như có thể hoạt động tốt với phần cứng thông dụng.

1.2 Mục tiêu nghiên cứu

Bài toán phân lớp đa nhãn đặt ra một số thách thức: số lượng phần tử nhãn lớn và mối quan hệ giữa các nhãn với nhau. Xuất pháp từ ý tưởng nêu trên, khóa luận này mục tiêu nghiên cứu chính mà nhóm nhắm đến là tập trung vào thực hiện việc tìm mối quan hệ sự phụ thuộc có thể có giữa các nhãn, đối phó với độ lệch nhãn bởi vì nhãn chỉ tập trung ở một số đối tượng. Ngoài ra, nhóm còn chú trọng vào việc cải thiện chi phí, tốc độ cũng như có thể tăng độ chính xác để tạo ra 1 mô hình khi số lượng nhãn rất lớn. Cụ thể sử dụng thuật toán k-NN dựa trên các đặc trưng đối tượng tiếp sau đó là thuật toán k-NN dựa trên các đặc trưng nhãn. Và cuối cùng sử dụng thuật toán sinh cắt tỉa [2] bằng cách kết hợp cả 2 phương pháp dựa trên đối tượng và đặc trưng.

Việc áp dụng ba thuật toán gặp phải một số thách thức như kích thước lớn, nhiều thuộc tính đặc trưng và nhãn gây khó khăn cho việc xử lý dữ liệu, phân tích và kiểm tra lại kết quả. Từ việc đánh giá hiệu quả cũng như mức độ ảnh hưởng đến việc xây dựng mô hình, nhằm tìm ra ưu điểm và nhược điểm của từng thuật toán. Từ đó, biết

cách kết hợp chúng sao cho đạt hiệu quả cao nhất mức độ đa dạng của các đối tượng để chọn một mô hình phù hợp nhất.

1.3 Cách tiếp cận

Nhóm trải qua các giai đoạn sau trong quá trình thực hiện đề tài:

Giai đoạn 1: Tìm hiểu về bài toán phân lớp đa nhãn. Ở phần này nhóm thực hiện tìm hiểu về đặc điểm của tập dữ liệu lớn và cực đoan cũng như cách lưu trữ thường dùng. Thêm vào đó nhóm cũng tìm hiểu về các mô hình hiện có để giải quyết bài toán này, các hướng tiếp cận được sử dụng cũng như những gì mô hình đã đạt được và những hạn chế của các phương pháp đó. Ngoài ra nhóm cũng tìm hiểu và tóm tắt những thông tin quan trọng của mô hình mới được đề xuất.

Giai đoạn 2: Nhóm tập trung tìm hiểu chi tiết và cài đặt các thuật toán k lân cận gần nhất dựa trên đối tượng và đặc trưng. Các thành viên tìm hiểu sâu hơn về các bước thực hiện của thuật toán trong bài báo [2] và mã nguồn của 2 mô hình [7], cũng như có thể thực hiện một số thay đổi cho phù hợp với ngôn ngữ lập trình Python.

Giai đoạn 3: Nhóm thực hiện kết hợp 2 thuật toán k lân cận trong giai đoạn 2 và cài đặt thêm thuật toán tính ngưỡng điểm dự đoán cũng như cài đặt các độ đo đánh giá. Thuật toán tính ngưỡng điểm dự đoán thực hiện loại bỏ các nhãn có điểm dự đoán thấp hơn hay bằng ngưỡng được chọn. Việc tìm ra ngưỡng phù hợp nhất dựa trên việc tối thiểu khác biệt giữa số lượng nhãn trung bình trong tập chân trị và tập dự đoán [2]. Sau đó nhóm kiểm tra kết quả trên các độ đo đánh giá được sử dụng bởi nhóm tác giá và thời gian thực thi chương trình.

Giai đoạn 4: Cài đặt các tối ưu mô hình bởi vì chạy các tập dữ liệu lớn tốn rất nhiều thời gian. Ngoài ra, nhóm đã cài đặt các hướng tiếp cận mới để cải thiện kết quả trên các phép đo của mô hình và thời gian huấn luyện trên các tập dữ liệu lớn và cực đoan.

1.4 Đóng góp

Nhóm kỳ vọng việc sử dụng phương pháp kết hợp các đối tượng và đặc trưng bằng cách cài đặt cách chuẩn hóa dữ liệu, sử dụng một hàm tối ưu mới để xét sự tương đồng giữa các đối tượng đã cải thiện được chất lượng của mô hình. Ngoài ra, nhóm còn tiến hành cài đặt việc thực thi song song đa tiến trình đã giảm thời gian chạy các mô hình rất nhiều và đối với những tập dữ liệu có số lượng đặc trưng và nhãn lớn thì đã mô hình đã chạy được và không còn hiện tượng tràn bộ nhớ mà không cần quá nhiều môi trường máy tính tính như ram và cpu. Thêm vào đó nhóm cũng thực hiện cài đặt một phương pháp tính độ tương đồng cộng hưởng cho dự đoán đa nhãn dựa trên đối tượng với hy vọng phần nào giải quyết một số vấn đề về số lượng và giá trị của những đặc trưng chung giữa các đối tượng có thể ảnh hưởng tới kết quả tương đồng truyền thống, từ đó cải thiện hiệu năng của mô hình cho bài toán phân lớp đa nhãn.

Chương 2

Tổng quan

2.1 Tổng quan về phân lớp dữ liệu

2.1.1 Giới thiệu

Ngày nay, phân lớp dữ liệu (classification) là một trong những hướng nghiên cứu chính của ngành khoa học dữ liệu. Trong thực tế, dữ liệu có từ rất nhiều nguồn khác nhau cộng với sự hỗ trợ của máy móc, con người có thể trích rút ra các quyết định nghiệp vụ một cách rất thông minh. Đối với việc học có giám sát, bài toán phân lớp dữ liệu về bản chất là xây dựng một hàm ánh xạ từ tập dữ liệu (dataset) của miền ứng dụng vào một tập nhãn cho trước. Trong đó, một đối tượng phân lớp hoặc một mẫu huấn luyện có thể được biểu diễn bằng các tập đặc trưng khác nhau. Trong những năm gần đây, phân lớp dữ liệu đã nhận được sự quan tâm rất lớn của các nhà nghiên cứu trong nhiều lĩnh vực khác nhau như học máy (machine learning), thống kê (statistics), hệ chuyên gia (expert system),... Công nghệ này cũng ứng dụng trong nhiều lĩnh vực khác nhau như: thương mại, maketing, nghiên cứu thị trường, bảo hiểm, y tế, giáo dục...

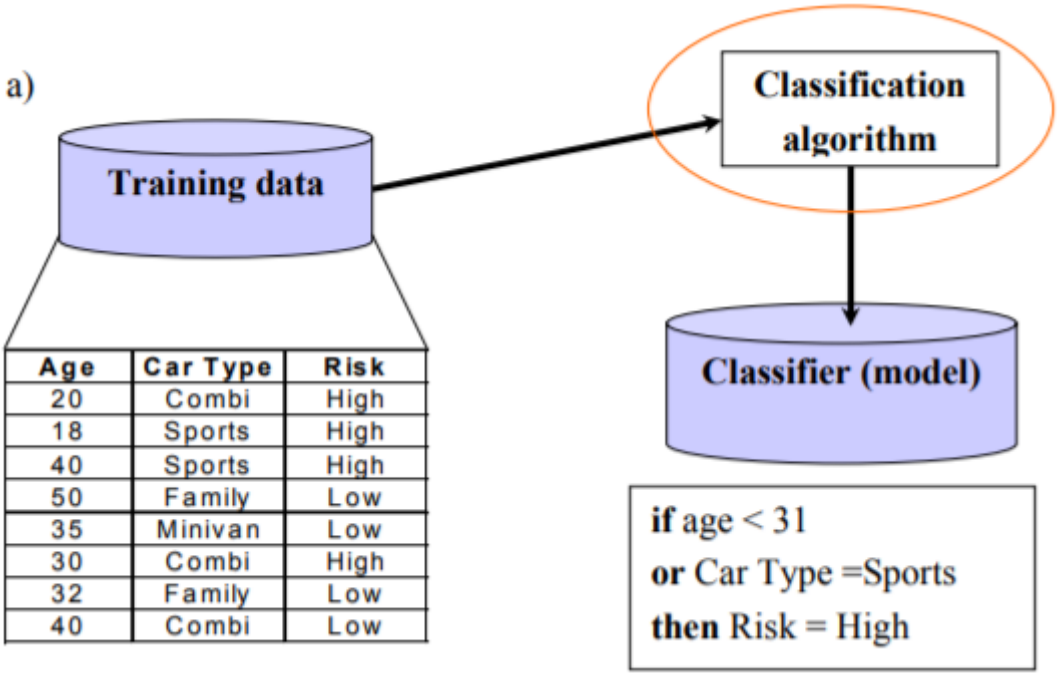
2.1.2 Quá trình phân lớp dữ liệu

Quá trình phân lớp dữ liệu cần trải qua nhiều bước khác nhau nhưng cơ bản gồm 2 bước chính [2]:

Bước thứ nhất: quá trình học (huấn luyện mô hình)

Quá trình học nhằm xây dựng một mô hình để có thể ánh xạ dữ liệu từ một tập các lớp dữ liệu vào các nhãn chính xác nhất. Đầu vào của quá trình này là một tập dữ liệu có cấu trúc được mô tả bằng các thuộc tính hoặc các đặc trưng từ tập các bộ giá trị của các thuộc tính gốc hoặc là qua quá trình chọn lọc dữ liệu. Trong tập dữ liệu này, mỗi đối tượng trong dữ liệu thuộc về một bộ lớp đã được gắn nhãn, ở đây thì lớp

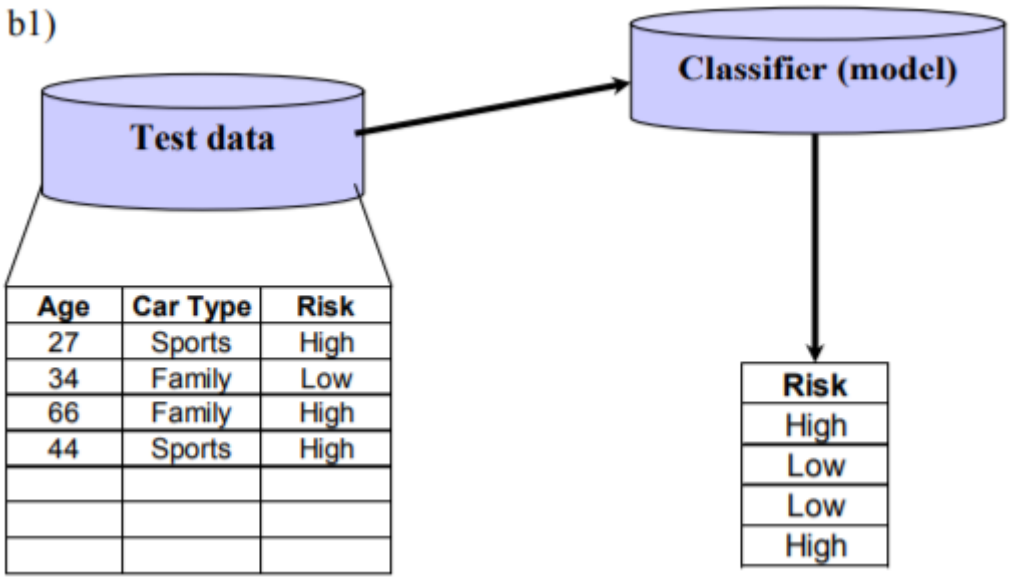
là giá trị của một hoặc nhiều thuộc tính được chọn làm thuộc tính phân lớp (class label attribute). Quá trình này được mô tả như trong hình bên dưới:



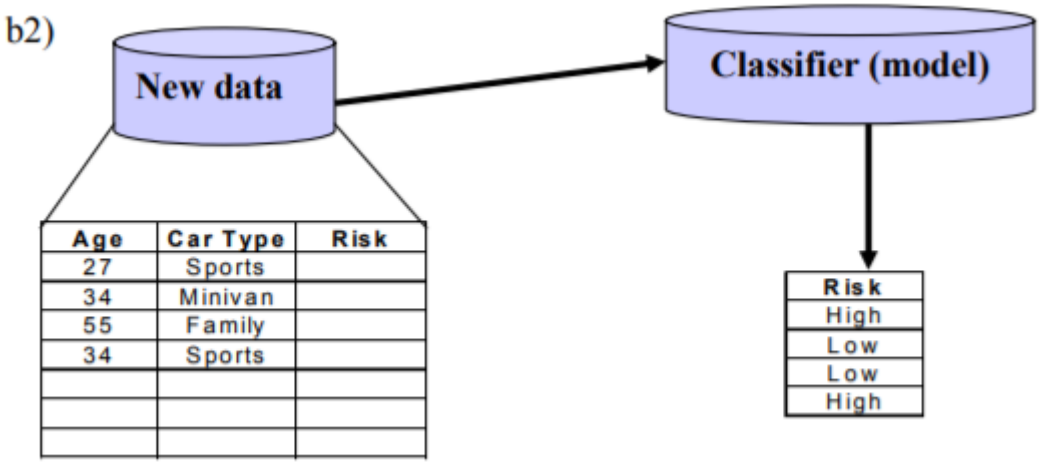
Hình 2.1 Xây dựng mô hình phân lớp

- **Bước thứ hai**: phân lớp (classification)

Bước thứ hai dùng mô hình đã được xây dựng để phân lớp dữ liệu mới. Trước tiên độ chính xác mang tính chất dự đoán của mô hình phân lớp vừa tạo ra được ước lượng. Sau đó, sử dụng các kỹ thuật khác nhau như gradient descent, … để ước tính tốt nhất các tham số có trong mô hình. Có rất nhiều cách đánh giá một mô hình phân lớp, tuỳ vào yêu cầu của từng bài toán khác nhau mà chúng ta sử dụng các phương pháp khác nhau. Độ chính xác của mô hình trên tập dữ liệu kiểm tra là tỉ lệ phần trăm các mẫu trong tập dữ liệu kiểm tra mà mô hình dự đoán đúng so với dữ liệu trong thực tế. Nếu độ chính xác của mô hình được ước lượng dựa trên tập dữ liệu huấn luyện cao thì kết quả phân lớp trên tập kiểm tra cũng là rất cao điều đó cho thấy khả năng mô hình có xu hướng được học tốt.



Hình 2.2 Quá trình phân lớp dữ liệu, ước lượng độ chính xác của mô hình trong quá trình huấn luyện



Hình 2.3 Quá trình phân lớp dữ liệu, dự đoán nhãn cho dữ liệu mới

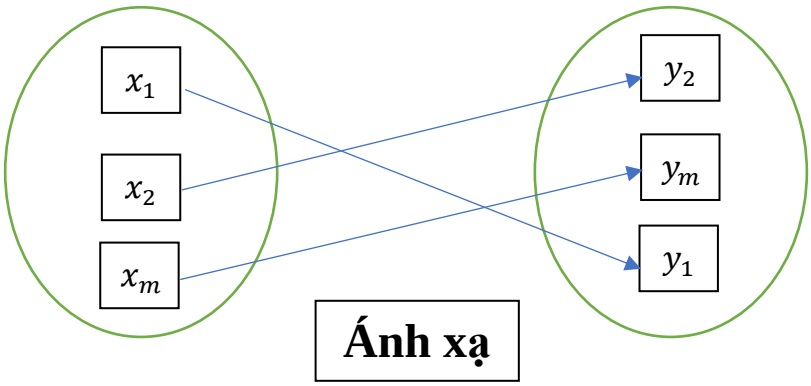
Đối với bài toán phân lớp thì các độ đo thường được sử dụng như: accuracy score, confusion matrix, ROC curve, Area Under the Curve, Precision and Recall, F1 score.

2.2 Tổng quan về bài toán phân lớp đơn nhãn

2.2.1 Mô tả bài toán

Input là tập không gian thể hiện (hoặc không gian thuộc tính) ký hiệu x và output là tập các nhãn ký hiệu là y. Bài toán phân lớp đơn nhãn được phát biểu như sau: Cho trước một tập mẫu huấn luyện D = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ trong đó $x_i \in x$ là một đối tượng và $y_i \in y$ là một nhãn tương ứng của đối tượng đó.

Output: Nhiệm vụ đặt ra là cần tạo ra được một hàm ánh xạ từ không gian thể hiện vào tập các nhãn.



Hình 2.4 Ánh xạ dữ liệu đơn nhãn từ input vào output

Người ta xây dựng các tập các đặc trưng khác nhau (ví dụ như mỗi tập đặc trưng biểu diễn cho một đoạn văn bản hay mỗi tập đặc trưng biểu diễn cho một câu trong văn bản), một văn bản được biểu diễn bởi các tập đặc trưng vừa xây dựng. Mô hình phân lớp đơn nhãn thể hiện đánh giá và kiểm tra mỗi đối tượng trong tập dữ liệu phân lớp có thuộc vào lớp đang xét hay không.

2.2.2 Một số kỹ thuật tiên tiến

Một số kỹ thuật phân lớp đơn nhãn hiện đại (state-of-the-art) đã được sử dụng trong những năm gần đây:

- Phân lớp cây quyết định (Decision tree classification).
- Máy vector hỗ trợ (support vector machine).

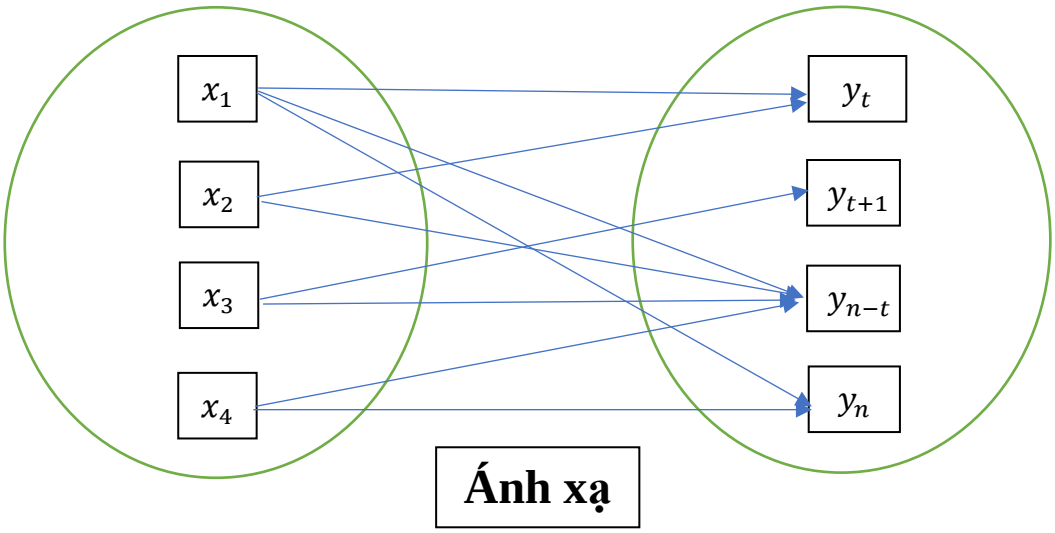
- Bộ phân lớp Bayesian (Bayesian classifier).
- Mô hình phân lớp K-láng giềng gần nhất (K-nearest neighbor classifier).
- Mạng nơron (neural network).
- Phân tích thống kê.
- Các thuật toán di truyền.

2.3 Tổng quan về bài toán phân lớp đa nhãn

Ta nhận thấy trong nhiều ứng dụng thực tế (phân lớp văn bản, gán nhãn ảnh, dự đoán chức năng gen, phân lớp video,…), một đối tượng có thể đồng thời được gán vào nhiều lớp khác nhau. Bài toán này là một sự tiến hóa của phân lớp đơn nhãn do đó nó tồn tại nhiều đặc trưng phức tạp hơn về dữ liệu đa nhãn, các phương pháp tiếp cận phân lớp đa nhãn và đánh giá học mình sau quá trình huấn luyện đa nhãn.

2.3.1 Mô tả bài toán

Cho trước một tập mẫu huấn luyện D = {(x_1, y_1), (x_2, y_2), …, (x_n, y_n) } trong đó $x_i \in x$ là một tập các thể hiện $\{x_{i1}, x_{i2}, \dots, x_{in}\}$ trong đó $x_{ij} \in x$ {j = 1,2,.., n} và $y_i \in y$ là một tập nhãn $\{y_{i1}, y_{i2}, \dots, y_{im}\}$ trong đó $y_{ik} \in y$ {= 1,2,.., m}. Nhiệm vụ đặt ra là cần tạo ra được một hàm ánh xạ f: $2^x \rightarrow 2^y$ từ không gian thể hiện vào tập các nhãn.



Hình 2.5 Ánh xạ dữ liệu đa nhãn từ input vào output

Trong phân lớp dữ liệu đa nhãn văn bản, các đối tượng phân lớp là tập các văn bản. Xây dựng các tập các đặc trưng khác nhau, mỗi văn bản được biểu diễn bởi các tập đặc trưng vừa xây dựng. Mô hình phân lớp đa nhãn đánh giá và gán cho văn bản phân lớp đồng thời vào nhiều lớp khác nhau.

2.3.2 Một số kỹ thuật tiên tiến

Một số kỹ thuật phân lớp đa nhãn hiện đại (state-of-the-art) đã được sử dụng trong những năm gần đây:

- Multi-label k-nearest neighbours (ML-kNN).
- instance-based logistic regression (IBLR).
- Binary relevance with support vector machines (BR-SVM).
- FastXml.

2.4 Giới thiệu một số kỹ thuật tiên tiến trong phân lớp đa nhãn

Để minh họa cho phương pháp bên dưới, nhóm sử dụng tập dữ liệu ở Bảng 1. Trong bảng dữ liệu có bốn mẫu được phân lớp vào tập nhãn lớp gồm bốn nhãn: $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

Bên dưới là phát biểu của bài toán phân lớp đa nhãn được sử dụng chung cho các phương pháp được trình bày ở các phần tiếp theo

Mẫu	Tập nhãn
1	$\{\lambda_1, \lambda_2, \lambda_3\}$
2	$\{\lambda_2, \lambda_4\}$
3	$\{\lambda_3\}$
4	$\{\lambda_2, \lambda_3, \lambda_4\}$

Bảng 2.1 Ví dụ về dữ liệu đa nhãn

2.4.1 Phương pháp Binary Relevance(BR)

Binary relevance là 1 phương pháp chuyển đổi đơn giản nhất của bài toán phân lớp đa nhãn. Theo Min-Ling ZHANG và các cộng sự [3], ý tưởng chính của phương pháp này là hoạt động bằng cách phân tách nhiệm vụ huấn luyện đa nhãn thành một số nhiệm vụ học nhị phân độc lập. Cụ thể, để xác định nhãn lớp thứ j (λ_j), đầu tiên thuật toán binary relevance sẽ xây dựng một tập huấn luyện nhị phân tương ứng bằng việc xem xét sự liên quan của mỗi mẫu huấn luyện với nhãn thứ j.

$$\lambda_j = \{ (x^i, y^i) \mid 1 \leq i \leq m \} \quad (1.1)$$

Trong đó, m là số lượng mẫu huấn luyện. Với mỗi mẫu huấn luyện (x^i, y^i) được chuyển đổi thành một mẫu huấn luyện nhị phân dựa trên λ_j. Xem xét về tập dữ liệu đa nhãn phía trên, khi đó ta sẽ xây dựng 4 tập huấn luyện cho 4 nhãn như sau:

Mẫu	**Nhãn**
1	λ_1
2	$\neg\, \lambda_1$
3	$\neg\, \lambda_1$
4	$\neg\, \lambda_1$
(a)	

Mẫu	**Nhãn**
1	λ_2
2	$\neg\, \lambda_2$
3	λ_2
4	λ_2
(b)	

Mẫu	**Nhãn**
1	λ_3
2	$\neg\, \lambda_3$
3	λ_3
4	$\neg\, \lambda_3$
(c)	

Mẫu	**Nhãn**
1	$\neg\, \lambda_4$
2	λ_4
3	$\neg\, \lambda_4$
4	λ_4
(d)	

Bảng 2.2 Phân tách đa nhãn thành đơn nhãn ở thuật toán binary relevance

Tuy nhiên, khi có 1 nhãn nào đó không thuộc mẫu huấn luyện nào trong tập huấn luyện thì kết quả của bộ phân lớp nhị phân đó đều mang giá trị âm nên tập nhãn dự đoán Y sẽ bị rỗng. Khi đó, luật T-Criterion có thể được áp dụng để tránh việc dự đoán tập nhãn rỗng:

$$Y^* = \{\lambda_i \mid g_j(x^*) > 0,\ 1 \leq j \leq q\} \quad (1.2)$$

Bảng bên dưới mô tả mã giả của thuật toán phân lớp Binary relevance

---

for j = 1 to q do:

Xây dựng tập huấn luyện nhị phân Dj theo công thức (1.1)

Gắn $g_j = B(Dj)$

Endfor

Trả kết quả $Y^* = \{\lambda_i \mid g_j(x^*) > 0,\ 1 \leq j \leq q\}$

Hình 2.6 Mã giả thuật toán phân lớp Binary relevance

Dựa theo đoạn chương trình mã giả, ta xác định được độ phức tạp thuật toán ở giai đoạn huấn luyện là $0(q.F'_B(m,d))$. Ngoài luật T-Criterion, một số quy luật khác cũng có thể được sử dụng trong việc dự đoán tập nhãn dựa trên đầu ra của mỗi bộ phân lớp nhị phân.

Ưu điểm của thuật toán binary relevance có thể dễ dàng điều chỉnh để học khi có các nhãn bị thiếu. Ngược lại, nhược điểm của binary relevance là đã bỏ qua các mối tương quan nhãn và bộ phân lớp nhị phân cho mỗi nhãn có thể rơi vào trạng thái mất cân bằng lớp khi lớn và mật độ nhãn thấp. Do sự mất mát thông tin này, bộ nhãn dự đoán của binary relevance có khả năng chứa quá nhiều hoặc quá ít nhãn, hoặc các nhãn không bao giờ có trong thực tế.

2.4.2 Phương pháp Classifier Chain (CC)

Theo Jeese Read và các cộng sự [4], ý tưởng của phương pháp này là chuyển bài toán học đa nhãn thành một chuỗi các bộ phân lớp nhị phân. Tuy nhiên, khác với thuật toán Binary revelance đã bỏ qua độ tương quan giữa các nhãn thì bộ phân lớp

nhị phân trong chuỗi được xây dựng dựa trên dự đoán của các bộ phân lớp trước đó. Cụ thể đi sâu vào chi tiết của bài toán, input của tập dữ liệu bao gồm $|L|$ bộ phân lớp nhị phân độc lập trong thuật toán binary relevance. Không gian đặc trưng của mỗi liên kết trong chuỗi được mở rộng với giá trị 0/1 so với liên kết nhãn của tất cả các liên kết trước đó.

Với mỗi mẫu huấn luyện (x, S) trong đó S ⊆ L biểu diễn 1 vector nhị phân $(l_1, l_2, \ldots, l_{|L|}) \in \{0,1\}^L$ và x là 1 $vector$ hiển thị input của mẫu huấn luyện đó. Ta xây dựng thuật toán phân lớp classifier chain như sau:

---

for j ∈ 1 ... $|L|$ do:

Biến đối nhãn đơn và training

$D' \leftarrow \{\}$

for (x,S) ∈ D:

$D' \leftarrow D' \cup ((x, l_1, .. , l_{j-1}), l_j)$

Huấn luyện C_j cho việc dự đoán l_j

C_j: $D' \rightarrow l_j \in \{0,1\}$

endfor

Endfor

Hình 2.7 Mã giả thuật toán phân lớp classifier chain

Với mỗi chuỗi nhị phân C_j sẽ đảm nhiệm việc học và dự đoán liên kết nhị phân của nhãn l_j sẽ được liên kết với các bộ nhị phân trước đó $l_1, .. , l_{j-1}$. Việc phân loại sẽ ở C_1 trước tiên, sau đó C_1 quyết định $P_r(l_1|x)$ và tương tự với các chuỗi $C_2 \ldots C_L$ và dự đoán $P_r(l_j|x_i, l_1, \ldots, l_{j-1})$.

Trong thuật toán này, mối quan hệ giữa các nhãn đã được xem xét không xét theo thứ thứ tự mà chọn một cách ngẫu nhiên. Dựa theo đoạn chương trình giả mã, tác giả xác định được độ phức tạp tính toán cho ở quá trình huấn luyện là O(|L| ×

f(|X| + |L|, |D|)). So với thuật toán binary relevance, thuật toán chuỗi bộ phân lớp (classifier chain) có ưu điểm là đã khai thác mối tương quan giữa các nhãn nhưng nhược điểm là không thực thi song song được do đặc điểm của chuỗi là khác nhau.

2.4.3 Phương pháp multi-label k-Nearest Neighbor (ML-kNN)

Theo Min-Ling Zhang và các cộng sự [5], ý tưởng chính của thuật toán ML-kNN là sử dụng kỹ thuật kNN (1 kỹ thuật học có giám sát) để xác định các láng giềng gần nhất của 1 đối tượng cần xác định nhãn, sau đó sử dụng các thông tin liên quan đến nhãn từ các láng giềng để đưa ra tập nhãn dự đoán. Với mỗi đối tượng x và tập nhãn tương ứng Y ⊆ L. 1 vector chỉ mục của vector x là $\overrightarrow{y_x}$, nếu tại đặc trưng thứ l của nhãn có xuất hiện giá trị thì lấy giá trị của $\overrightarrow{y_x}\,(l) = 1$ còn ngược lại thì giá trị của $\overrightarrow{y_x}\,(l) = 0$. Do đó, dựa trên các bộ nhãn, một vectơ đếm số lượng nhãn xuất hiện được định nghĩa như sau:

$$\overrightarrow{C_{x^{(l)}}} = \sum_{a \in N(x)} \overrightarrow{y_a}\,(l), \mathrm{l} \in L$$

Với mỗi đối tượng t trong tập test, thuật toán ML-KNN sẽ tìm k đối tượng đặc trưng lân cận N(t) trong tập huấn luyện. Sau đó, dựa vào số lượng nhãn trong vector $\overrightarrow{C_t}$, vector $\overrightarrow{y_t}$ được xác định bằng cách sử dụng nguyên tắc posteriori tối đa như sau:

$$\overrightarrow{y_t}\,(l) = arg\ max_{b \in \{0,1\}}\ P\left(H_b^l \mid E^l_{\overrightarrow{C_t}(l)}\right)$$

Trong đó, H_1^l thể hiện đối tượng t có nhãn thứ l, ngược lại H_0^l thể hiện đối tượng t không tồn tại nhãn thứ l. Ta xây dựng thuật toán Ml-kNN như sau:

[$\overrightarrow{y_t}$, $\overrightarrow{r_t}$] = ML-kNN (T, K, t, s)

// Tính toán xác suất ưu tiên P(H_b^l)

for l ∈ |L| do:

$P\left(H_1^l\right)$ = (s + $\sum_{i=1}^{m} \overrightarrow{y_{x_i}}(l)$ / (2. s + m)

$P\left(H_0^l\right)$ = 1 – $P\left(H_1^l\right)$

// Tính toán độ xác suất ưu tiên posterior $\mathrm{P}\left(\mathrm{E}_{\mathrm{j}}^{\mathrm{l}} \mid \mathrm{H}_{\mathrm{b}}^{\mathrm{l}}\right)$

Với mỗi đối tượng trong tập train N(x_i), i ∈ { 1, 2, . . , m}

for l ∈ |L| do:

for j ∈ { 0, 1, … , k} do

c|j| = 0, C'|j| = 0

for I ∈ { 1,2, … , m} do

$\delta = \overrightarrow{C_{x_i}}(l) = \sum_{a \in N(x)} \overrightarrow{y_a}(l)$;

$if\ (\ \overrightarrow{y_{x_i}}(l) == 1)$:

$C|\delta|$ += 1

else $C'|\delta|$ += 1

for j ∈ { 0, 1, …, k} do

$P(E_j^l \mid H_1^l) = (s + c|j|)/\ (s$ x (k+1) + $\sum_{p=0}^{k} c|p|)$;

$P(E_j^l \mid H_0^l) = (s + C'|j|)/\ (s$ x (k+1) + $\sum_{p=0}^{k} C'|p|)$;

// Tính toán $\overrightarrow{y_t}$ và , $\overrightarrow{r_t}$

Với mỗi đối tượng trong tập test N(t)

for l ∈ |L| do:

$\overrightarrow{C_t}(l) = = \sum_{a \in N(t)} \overrightarrow{y_a}(l)$

$\overrightarrow{y_t}(l) = \arg\ max_{b \in \{0,1\}} P\left(E^l_{\overrightarrow{C_t}(l)} \mid H_b^l \right) . P(H_b^l)$;

$\overrightarrow{r_t}(l) = P\left(H_1^l \mid E^l_{\overrightarrow{C_t}(l)} \right) . P(H_b^l)$;

endfor

Hình 2.8 Mã giả thuật toán phân lớp ML-kNN

Trong thuật toán, để đưa ra được dự đoán vector $\overrightarrow{y_t}$, ta cần phải biết được xác suất $P(H_b^l)$ (l ∈ |L|, b ∈ { 0,1}) và xác xuất posteriori $P\left(E_j^l \mid H_b^l\right)$ (j ∈ { 0,1, .., k } thông

qua quá trình huấn luyện dựa vào đếm tần suất xuất hiện. Thuật toán này có ưu điểm là ranh giới biên có thể được điều chỉnh một cách hợp lý dựa trên các láng giềng khác nhau được xác định cho mỗi mẫu chưa biết.

2.4.4 Phương pháp Rank - Support vector machine (Rank-SVM)

Theo Guoqiang Wu và các cộng sự [6], ý tưởng chính của thuật toán này là thích nghi cực đại biên để giải quyết bài toán học đa nhãn. Thuật toán còn nhằm mục đích giảm thiểu sự mất mát của hàm rank-loss khi biên được xác định dựa trên xếp hạng trên các lớp liên quan và không liên quan của mẫu. Với mỗi đối tượng x $\in R^m$, thuật toán dự đoán giá trị đạt được bằng hàm f = xW + b. Trong đó b = $[b_1, b_2, \dots, b_l] \in R^l$ là giá trị bias và W = $[w^1, w^2, \dots, w^l] \in R^{m.l}$ là tham số ma trận.

Xây dựng hệ thống phân lớp W = $\{(w_j, b_j)\}$ $(1 \leq j \leq q\}$ bằng việc giải quyết bài toán quy hoạch theo công thức

$$\text{Min(W)} = \frac{1}{2} \sum_{j=1}^{l} ||w^j||^2 + \lambda_2 \sum_{i=1}^{n} \frac{1}{|Y_i^+||Y_i^-|} \sum_{p \in Y_i^+} \sum_{p \in Y_i^-} \xi_{pq}^i$$

$$\text{Với giải thiết } (w_j - w_k, x_i) + b_j - b_k \geq 1 - \xi_{pq}^i$$

Trong đó, Y_i^+ và Y_i^- hiển thị chỉ số liên quan của nhãn đối tượng đang xét x_i, λ_2 là siêu tham số giúp điều chỉnh độ lỗi của mô hình. Ngoài ra, rank-SVM cần phải tính toán ngưỡng threshold một cách tự động. Đầu tiên, dựa vào việc học $f_i = [f_{i1}, f_{i2}, \dots, f_{il}] \epsilon R^L$ cho mỗi đối tượng x_i trong tập train, nó tìm giá trị threshold t(x_i) thông qua quy luật:

$$t(x_i) = arg \sum_{j=1}^{t} \{ |j \in Y_i^+| |f_{ij} \leq t| + |j \in Y_i^-| |f_{ij} \geq t| \}$$

Khi dự đoán với mỗi đối tượng trong tập test, sử dụng hàm f = xW, sau đó lấy giá trị của threshold t(x) = T(f) và cuối cùng đạt được kết quả của phân lớp đa nhãn h(x) = sign(| f > t(x) |).

Thuật toán có độ phức tạp tính toán cho pha huấn luyện là O($F_{QP}(dq + mq^2) + q^2(q + m)$). Ưu điểm của thuật toán là đã kết thừa được phương pháp nhân (kernel) để giải quyết vấn đề phân lớp không tuyến tính.

2.4.5 Phương pháp instance-based logistic regression (IBLR)

Theo Weiwei Cheng và Eyke Hüllermeier [7], ý tưởng chính của thuật toán là xem nhãn của các đối tượng lân cận là đặc điểm cho việc truy vấn x_0 có nhãn của mẫu được ước tính. IBLR là 1 phương pháp được cải thiện dựa trên các phương pháp hiện có là phương pháp ML-kNN. Ở đây, mỗi đối tượng $x_0 \in$ X nhãn $y_0 \in \{-1, +1\}$ sẽ được ước tính. Gọi P_o là xác xuất của y_o = +1 và lấy thông tin nhãn đã biết y_i làm cơ sở để tính toán nhãn chưa biết y_o, nó liên quan đến xác xuất posteri như sau:

$$\pi_o = \mathrm{P}\,(y_o = +1 \mid y_i)$$

Và sau đó áp dụng quy tắc Bayes:

$$\frac{\pi_o}{1-\pi_o} = \frac{P\,(y_i \mid y_o = +1)}{P\,(y_i \mid y_o = -1)} \cdot \frac{p_o}{1-p_o} = p \cdot \frac{p_o}{1-p_o}$$

Trên thực tế, p có giá trị lớn khi $\delta_i \to 0$ nếu y_i = +1 và $p \to 1$ khi $\delta_i \to \infty$. Sau đó, người ta xây dựng một hàm được tham số hóa để ước tính cho các tham số phù hợp là

$$p = p(\delta) \stackrel{\text{def}}{=} \exp\,(y_i \cdot \frac{a}{\delta})$$

Trong đó, α > 0 là một hằng số. $p(\delta)$ xác định xác suất để hai đối tượng có khoảng cách được cho bởi $\delta = \Delta(x_0, x_i)$ có cùng nhãn. Sau đó, xem xét các đối tượng lân cận cận của đối tượng x_0 là N(x_0). Dựa theo quy tắc Bayes làm cho giả định đơn giản hóa về tính độc lập có điều kiện, ta thu được kết quả:

$$\log\,(\frac{\pi_o}{1-\pi_o}) = w_0 + a \sum_{x_i \in N(x0)} \frac{y_i}{\delta_i} = w_0 + a\,.\,(w + x_o)$$

Về cơ bản có thể nhận ra một ước lượng lân cận có trọng số và được xem như một phiên bản "dựa trên mô hình" của việc học dựa trên phiên bản. Ưu điểm của thuật toán là đã phản ánh trực tiếp độ lớn của các hệ số hồi quy liên quan. Khả năng này

giúp phân biệt IBLR với các phương pháp dựa trên phiên bản hiện có để phân loại đa nhãn.

2.4.6 Phương pháp Fast-XML

Theo Yashoteja Prabhu và các cộng sự [8], ý tưởng chính của phương pháp này là xây dựng một bộ phân loại đa nhãn dựa trên cây. Với quá trình huấn luyện được hiển thị như sau $\{ (x_i, y_i) \}_{i=1}^{N}$ trong đó D là số chiều không gian đặc trưng ($x_i \in R^D$) và L là số chiều không gian nhãn ($y_i \in \{0,1\}^L$) mà tại đó $y_{il} = 1$ nếu đối tượng đó chứa nhãn thứ l và ngược lại bằng 0 nếu không chứa nhãn đó. Bên dưới là mã giả mô tả thuật toán thuật toán FastXML:

GrowNodeRecursive(n) do

If |n.Id| ≤ MaxLeaf:

n.P ← ProcessLeaf ($\{ x_i, y_i \}_{i=1}^{N}$, n)

else:

{n.W, n.left child, n.right child} ← split node ($\{ x_i, y_i \}_{i=1}^{N}$, n)

GrowNodeRecursive(n. left child)

GrowNodeRecursive(n. right child)

End

ProcessLeaf ($\{ x_i, y_i \}_{i=1}^{N}$, n) do

P ← top-k ($\frac{\sum_{i \in n.Id} y_i}{| n.Id |}$)

Return P

End

Parrell for i ∈ 1 ... $|T|$ do:

n^{root} ← new node

$n^{root}.Id$ ← { 1, …, N}

GrowNodeRecursive (n^{root})

$T_i \leftarrow$ new tree

$T_i.root \leftarrow n^{root}$

Return $T_1, \dots, T_T$

End

Hình 2.9 Mã giả thuật toán phân lớp FastXML

Sau khi đã huấn luyện mô hình FastXM, tiến hành dự đoán nhãn cho từng đối tượng như sau:

Parrell for i ∈ 1 ... $|T|$ do:

$n \leftarrow \mathcal{T}_i$.root
while n is not a leaf **do**
  $\mathbf{w} \leftarrow n.\mathbf{w}$
  if $\mathbf{w}^\top \mathbf{x} > 0$ **then**
    $n \leftarrow n$.left_child
  else
    $n \leftarrow n$.right_child
  end if
end while
$\mathbf{P}_i^{\text{leaf}}(\mathbf{x}) \leftarrow n.\mathbf{P}$
end for
$\mathbf{r}(\mathbf{x}) = \text{rank}_k\left(\frac{1}{T}\sum_{i=1}^{T} \mathbf{P}_i^{\text{leaf}}(\mathbf{x})\right)$

Return r(x)

End

Hình 2.10 Dự đoán nhãn cho mỗi đối tượng trong mô hình FastXML

Nhận thấy, việc dự đoán hiệu quả có thể được thực hiện bằng cách xác định điểm mới nằm trong đó bằng cách duyệt qua hệ thống phân cấp không gian đối tượng và sau đó tập trung vào tập hợp các nhãn đang hoạt động trong khu vực.

Ưu điểm của thuật toán là mô hình có thể chạy nhanh hơn đáng kể so với các phương pháp state-of-the-arts khác vì công thức phân vùng node được đề xuất có thể được tối ưu hóa hiệu quả hơn so với công thức dựa trên chỉ số entropy hoặc Gini.

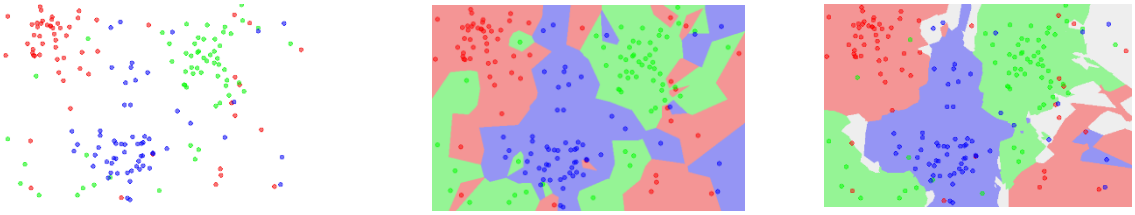
Chương 3

Cơ sở lý thuyết

3.1 Thuật toán K lân cận gần nhất

Trong các thuật toán học máy, K lân cận gần nhất (kNN) là một trong những thuật toán học có giám sát không tham số [9] đơn giản nhất mà hiệu quả trong một số trường hợp, được xếp vào nhóm thuật toán lười học [10]. Theo lý thuyết thì các thuật toán lười học sẽ trì hoãn việc tổng quát hóa trên dữ liệu huấn luyện đến khi có truy vấn gửi tới mô hình thay vì cố gắng xây dựng hàm mục tiêu với dữ liệu hiện có [11]. Phương pháp học trên hữu ích nhất với các tập dữ liệu có số lượng đối tượng cũng như số đặc trưng và nhãn lớn nhưng chỉ có một số đặc trưng được truy vấn thường xuyên, ngoài ra thường thay đổi, cập nhật liên tục [11]. Với bài toán phân lớp, đầu ra của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong tập huấn luyện, có thể được quyết định bằng bầu chọn số phiếu không trọng số hoặc có trọng số, như là sử dụng khoảng cách của lân cận để tính phần đóng góp vào dự đoán [10]. Việc sử dụng trọng số cho các lân cận là một trong những biện pháp đối phó với sự mất cân bằng của phân bố nhãn trong tập dữ liệu. việc lựa chọn số lân cận k tốt nhất thường dựa vào tập dữ liệu, với k lớn sẽ giúp làm giảm ảnh hưởng của các dữ liệu nhiễu nhưng ranh giới giữa các phân lớp khó phân biệt hơn [9].

Một độ đo khoảng cách phổ biến được sử dụng với các biến liên tục là phép đo khoảng cách euclid, trong khi các biến rời rạc, như ở bài toán phân loại văn bản thường dùng độ đo khoảng cách hamming. Các phép đo trên có thể không đưa ra giá trị hữu ích với dữ liệu có trên 10 chiều đặc trưng, khi đó mọi điểm dữ liệu gần như cách đều nhau [9]. Một số phương pháp giảm chiều dữ liệu và rút trích đặc trưng như phân tích thành phần chính (PCA), phân tích biệt thức tuyến tính (LDA) hay phân tích tương quan kinh điển (CCA) có thể được sử dụng để tiền xử lý tập dữ liệu [9].



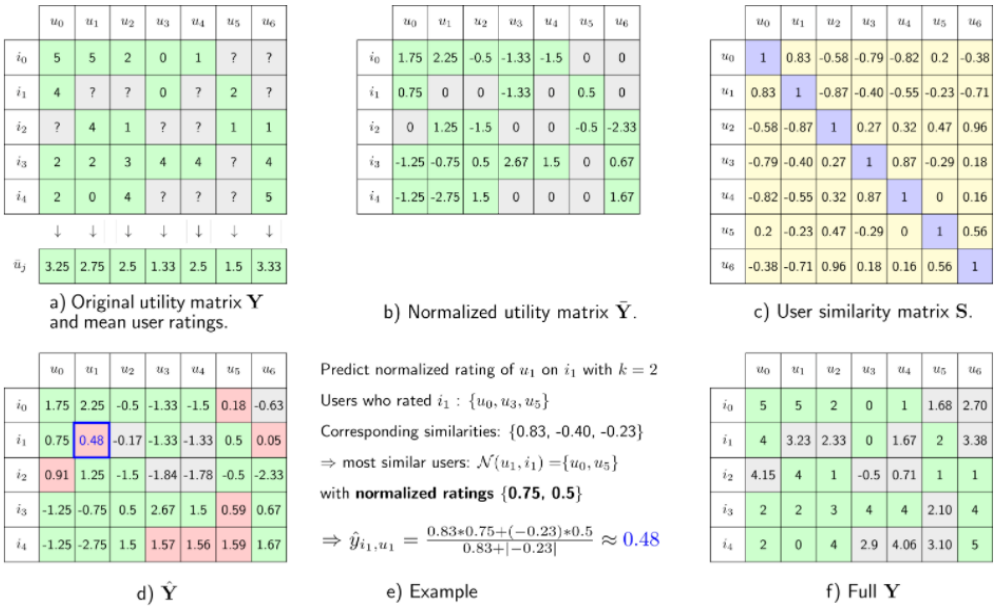
Hình 3.1 ví dụ một tập dữ liệu và kết quả phân lớp bởi k-NN lần lượt với k = 1 và k = 3 [9].

3.2 Lọc cộng tác dựa trên người dùng

3.2.1 Khái niệm

Lọc cộng tác là một trong hai nhóm lớn của hệ thống tư vấn, một mảng khá rộng của các thuật toán học máy được phát triển cùng với sự bùng nổ của Internet với khả năng tự động gợi ý cho người dùng những sản phẩm họ có thể thích [12]. Với hai thực thể chính cần quan tâm là tập những người dùng và sản phẩm, các thuật toán lọc cộng tác sẽ gợi ý những sản phẩm dựa trên tương quan giữa những người dùng và/hoặc những sản phẩm khác, như là việc một người dùng được gợi ý sản phẩm nào đó từ những người dùng khác có hành vi tương tự [12]. Tương quan giữa những người dùng có thể được xác định thông qua mức độ quan tâm của những người dùng này tới những sản phẩm khác mà hệ thống đã biết với dữ liệu hiện có là ma trận thể hiện mức độ quan tâm của người dùng với sản phẩm, bao gồm những giá trị chưa biết [13].

3.2.2 Minh hoạt thuật toán



Hình 3.2 Ví dụ mô tả User-user Collaborative Filtering [13]

Trong đó: a) Utility Matrix ban đầu. b) Utility Matrix đã được chuẩn hoá. c) User similarity matrix. d) Dự đoán các (normalized) *ratings* còn thiếu. e) Ví dụ về cách dự đoán normalized rating của u_1 cho i_1. f) Dự đoán các (denormalized) *ratings* còn thiếu.

Ví dụ như hình mô tả hoạt động của thuật toán lọc cộng tác, thực hiện dự đoán điểm số của u_1 dành cho i_1 với số lân cận gần nhất k = 2 theo các bước [13]:

- Xác định người dùng đã đánh giá i_1 là u_0, u_3, u_5.
- Tính độ tương đồng của u_1 với những người dùng trên, kết quả theo thứ tự là: 0.83, -0.40 và -0.23 với 2 giá trị lớn nhất là 0.83 và -0.23 ứng với u_0 và u_5.
- Xác định điểm đánh giá chuẩn hóa cho i_1 của u_0 và u_5 lần lượt là 0.75 và 0.5.
- Dự đoán điểm của u_1 dành cho i_1:

$$\hat{y}_{i_1,u_1} = \frac{0.83 \times 0.75 + (-0.23) \times 0.5}{0.83 \times |-0.23|} \approx 0.48$$

- Những dự đoán khác thực hiện tương tự các bước trên. Qua đó điểm dự đoán được xác định theo công thức:

$$\hat{y}_{i_j,u_k} = \frac{\sum_{u_l \in KNN(u_k)} r_{j,l} \times sim(u_k, u_l)}{\sum_{u_l \in KNN(u_k)} |sim(u_k, u_l)|}$$

Trong đó:

- $\hat{y}_{i_j,u_k}$: điểm dự đoán cho sản phẩm i_j của đối tượng u_k.
- u_l: người dùng có độ tương đồng cao nhất.
- $r_{j,l}$: đánh giá cho sản phẩm i_j của người dùng u_l
- $sim(u_k, u_l)$: độ tương đồng giữa người dùng u_k và u_l.

3.3 Độ tương đồng cosine

Độ tương đồng cosine là một cách thể hiện sự giống nhau giữa 2 dãy số. Ở đây các dãy số được coi như véc-tơ và độ đo được tính là giá trị hàm cos của góc giữa 2 véc-tơ đó, tức là tích vô hướng chia cho tích độ dài của 2 véc-tơ được xét, theo công thức [14]:

$$cosine\_similarity(A, B) = cos(A, B) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Trong đó: A_i, B_i là các thành phần trong véc-tơ A, B. Các kết quả độ đo có giá trị trong đoạn [-1, 1], giá trị -1 cho biết 2 véc-tơ hoàn toàn trái ngược nhau, còn khi giống nhau hoàn toàn thì giá trị hàm cos là 1.

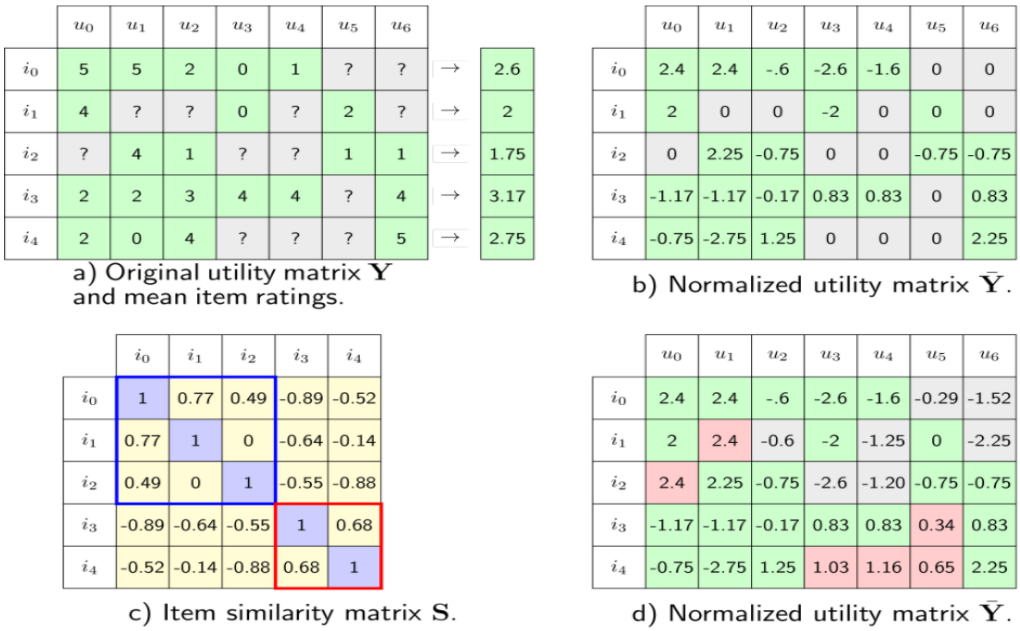
3.4 Lọc cộng tác dựa trên sản phẩm

3.4.1 Khái niệm

Kỹ thuật lọc cộng tác dựa trên người dùng thường gặp một số khó khăn khi sử dụng thực tế do số lượng người dùng thường lớn hơn rất nhiều so với số lượng sản phẩm, thêm vào đó người dùng không thường xuyên đánh giá sản phẩm khiến cho ma trận đánh giá chỉ có một số ít phần tử khác 0 dẫn đến khi có bất kỳ thay đổi nào về đánh giá đều làm cho trung bình cộng và véc-tơ chuẩn hóa của người dùng thay đổi nhiều

cũng như cần thực hiện lại việc tính độ tương đồng [13]. Mặt khác do số lượng sản phẩm không nhiều như người dùng nên một véc-tơ điểm đánh giá nếu xét theo sản phẩm sẽ có nhiều phần tử đã biết hơn, qua đó khi có cập nhật thì sự thay đổi về trung bình không nhiều, cho phép giảm mức độ thường xuyên cập nhật độ tương đồng.

3.4.2 Minh hoạ thuật toán



Hình 3.3 Ví dụ mô tả Item-Item Collaborative Filtering [13]

Trong đó a) Utility Matrix ban đầu. b) Utility Matrix đã được chuẩn hoá. c) User similarity matrix. d) Dự đoán các (normalized) *ratings* còn thiếu.

Ví dụ thực hiện dự đoán các đánh giá còn thiếu như hình trên có các bước tương tự lọc cộng tác dựa trên người dùng ngoại trừ việc sử dụng trung bình của từng sản phẩm để chuẩn hóa giá trị thay cho trung bình với từng người dùng và ma trận tương đồng tính theo những sản phẩm. Điểm dự đoán trên mỗi dòng được chuẩn hóa bằng cách trừ đi trung bình của dòng, sau đó thực hiện tính độ tương đồng cosine cho các cặp sản phẩm, tức các véc-tơ dòng thay cho véc-tơ cột người dùng, thu được ma trận S như hình 3.2C. Việc tính điểm dự đoán cũng thực hiện với các bước có phần giống như lọc cộng tác dựa trên người dùng, nhưng chỉ xét tại một người dùng và những đánh giá hiện có của người đó, dự đoán điểm số của u_6 dành cho i_0 với số lân cận gần nhất k = 2:

- Xác định những sản phẩm u_6 đã đánh giá, gồm i_2, i_3 và i_4.
- Độ tương đồng của i_0 với i_2, i_3 và i_4 lần lượt là 0.49, -0.89 và -0.52 với k = 2 giá trị lớn nhất là 0.49 và -0.52 cho i_2 và i_4.
- Dự đoán điểm của u_6 dành cho i_0:

$$\hat{y}_{i_0,u_6} = \frac{0.49 \times (-0.75) + (-0.52) \times 2.25}{0.49 \times |-0.52|} \approx -1.52$$

3.5 Giới thiệu các thuật toán dự đoán đa nhãn áp dụng kiến thức từ thuật toán lọc cộng tác

Để minh họa cho phương pháp bên dưới, nhóm sử dụng tập dữ liệu như sau: với tập dữ liệu đặc trưng X và nhãn Y của tập huấn luyện cùng tập kiểm tra X_q.

X

i	f_1	f_2	f_3	f_4	f_5	f_6
1	1	1	1	0	1	0
2	1	1	1	1	0	0

Y

i	l_1	l_2	l_3	l_4	l_5
1	1	1	1	0	0
2	0	1	1	1	0

X_q

q	f_1	f_2	f_3	f_4	f_5	f_6
1	1	1	1	0	1	0
2	1	1	1	1	0	1

3.5.1 Dự đoán đa nhãn dựa trên đối tượng

Nhóm nghiên cứu của Len Feremans và các cộng sự [15] nhận xét bài toàn phân lớp đa nhãn có sự liên quan nhất định đến các thuật toán hệ thống tư vấn, trong đó một công việc cần thực hiện ở cả hai bài toán chính là tìm được chính xác một tập k lân cận gần nhất. Từ đó nhóm tác giả đề xuất áp dụng quy trình thực hiện của thuật

toán lọc cộng tác dựa trên người dùng vào thuật toán phân lớp K lân cận gần nhất nhằm góp phần giải quyết một số vấn đề hiện có trong bài toán phân lớp đa nhãn như là sự phân bố các đối tượng về các nhãn không cân bằng hay yêu cầu lượng tài nguyên tính toán lớn khi huấn luyện mô hình. Có thể kể đến mô hình Classifier Chain hay FastXML cần huấn luyện lần lượt L và $\log(L)$ mô hình phân lớp đơn với L là số lượng nhãn và có nhiều siêu tham số cần tối ưu để đạt kết quả tốt [15].

3.5.1.1 Tìm K lân cận gần nhất – InstanceKNNSearch

Thuật toán phân lớp này sử dụng độ tương đồng cosine thay cho các độ đo khoảng cách thường áp dụng để làm tiêu chí xác định lân cận giữa các véc-tơ dữ liệu. Xét x_q, x_i lần lượt là véc-tơ đặc trưng được chuẩn hóa của đối tượng kiểm tra và huấn luyện, độ tương đồng giữa 2 véc-tơ tính theo công thức:

$$sim_{INS}(x_q, x_i) = \frac{x_q \cdot x_i}{\|x_q\|_2 \cdot \|x_i\|_2} = x_q \cdot x_i$$

Ví dụ minh họa

$\overline{X}$ =

i	f_1	f_2	f_3	f_4	f_5	f_6
1	0.5	0.5	0.5	0	0.5	0
2	0.5	0.5	0.5	0.5	0	0

$\overline{X_q}$ =

q	f_1	f_2	f_3	f_4	f_5	f_6
1	0.5	0.5	0.5	0	0.5	0
2	0.45	0.45	0.45	0.45	0	0.45

S = $X_q \cdot \overline{X}^T$ =

	i_1	i_2
q_1	1	0.75
q_2	0.675	0.9

3.5.1.2 Dự đoán – InstanceKNNPredict

Sau khi x_q đã có tập đối tượng lân cận gần nhất kNN(x_q) thì việc tính điểm dự đoán các nhãn cho x_q được thực hiện theo công thức:

$$\hat{y}_{q,j}^{INS} = \frac{\sum_{x_i \in kNN(x_q)} y_{i,j} \cdot sim_{INS}(x_q, x_i)^{\alpha}}{\sum_{x_i \in kNN(x_q)} sim_{INS}(x_q, x_i)^{\alpha}}$$

Trong đó:

- $\hat{y}_{q,j}^{INS}$: điểm dự đoán dựa trên đối tượng lân cận cho nhãn j của đối tượng kiểm tra x_q.
- $y_{i,j}$: giá trị tại nhãn j của đối tượng lân cận gần nhất x_i, $y_{i,j} \in \{0, 1\}$.
- $sim_{INS}(x_q, x_i)$: giá trị độ tương đồng của x_q và x_i.
- α: hệ số mũ, cho phép thay đổi đóng góp của các giá trị tương đồng lên điểm dự đoán, ví dụ $\alpha = 2$ thì $sim_{INS}(x_q, x_i)$ gần 1 nên có ảnh hưởng hơn so với $sim_{INS}(x_q, x_i)$ gần 0.01, với $\alpha = 0.5$ có hiệu ứng ngược lại.

Ví dụ minh họa

$S^{\alpha=2} =$

	i_1	i_2
q_1	1	0.5625
q_2	0.4556	0.81

$\sum S^{\alpha} =$

	$\sum_i x_i$
q_1	1.5625
q_2	1.2656

$\overline{Y^{INS}} = \hat{Y}^{INS} / \sum S^{\alpha} =$

q	l_1	l_2	l_3	l_4	l_5
1	1	1.5625	1.5625	0.5625	0
2	0.4556	1.2656	1.2656	0.81	0

$\hat{Y}^{INS} = S^{\alpha} \cdot Y =$

q	l_1	l_2	l_3	l_4	l_5
1	0.64	1	1	0.36	0
2	0.36	1	1	0.64	0

3.5.2 Dự đoán đa nhãn dựa trên các đặc trưng

3.5.2.1 Độ tương đồng giữa các đặc trưng và nhãn – CreateSimilMatrix.

Tiếp tục theo hướng tiếp cận áp dụng kỹ thuật lọc cộng tác vào mô hình phân lớp đa nhãn, Len Feremans và các cộng sự [15] đã áp dụng lọc cộng tác dựa trên sản phẩm sử dụng độ đo tương đồng cosine để xác định sự tương đồng giữa các đặc trưng và các nhãn trong tập dữ liệu với các bước thực hiện có phần tương tự thuật toán tính độ tương đồng giữa các đối tượng instanceKNNSearch cũng như yêu cầu các véc-tơ dữ liệu đã được chuẩn hóa thành véc-tơ đơn vị.

$$sim_{FL}(f_i, l_j) = \frac{f_i \cdot l_j}{\|f_i\|_2 \cdot \|l_j\|_2} = f_i \cdot l_j$$

Các bước thực hiện gồm:

Algorithm 4: CREATESIMILMATRIX($\mathcal{D}$) Computes similarities between all features and labels for feature-based kNN

Input: A dataset $\mathcal{D}$
Result: Similarity matrix S

```
   /* Create inverted index for labels */
1  IID ← EMPTY_HASH_MAP();
   /* For each instance */
2  for ⟨x_i, y_i⟩ in D do
   |  /* For each nonzero label value  */
3  |  for y_{i,j} ≠ 0 in y_i do
4  |  |  IID[j] ← IID[j] ∪ {x_i};
   /* Compute similarities */
5  S ← 0.0^{M×L};
   /* For each label j get instances from IID
      */
6  for y_j ≠ 0 in IID do
7  |  for x_i in IID[j] do
   |  |  /* For each nonzero feature k */
8  |  |  for x_{i,k} ≠ 0 in x_i do
   |  |  |  /* Compute partial dot product */
9  |  |  |  S_{j,k} ← S_{j,k} + x_{i,k} · y_{i,j};
10 return S;
```

Hình 3.4 Mã giả thuật toán createSimilMatrix

- Bước 1: Chuyển vị ma trận đặc trưng tập huấn luyện.
- Bước 2: Nhân ma trận giữa ma trận đặc trưng và ma trận nhãn của tập huấn luyện (1).
- Bước 3: Tính căn bậc 2 của tổng các giá trị trong tập nhãn theo từng cột (2).
- Bước 4: Chia giá trị ở (1) cho (2) theo từng cột.

3.5.2.2 Dự đoán dựa trên độ tương đồng đặc trưng - FeatureKNNPredict.

Ma trận độ tương đồng giữa các đặc trưng và các nhãn trong tập dữ liệu được sử dụng để tính điểm dự đoán cho từng nhãn như thuật toán dự đoán dựa trên đối tượng, tuy nhiên có thay đổi về việc sử dụng giá trị đặc trưng của đối tượng kiểm tra làm trọng số cho độ tương đồng và chuẩn hóa kết quả sang thay cho tổng độ tương đồng được biến đổi mũ [15]. Công thức để tính độ tin cậy ở đây là:

$$\hat{y}_{q,j}^{FL} = \frac{\sum_{i=1}^{M} x_{q,i} \cdot sim_{FL}(f_i, l_j)^{\beta}}{\sum_{i=1}^{M} x_{q,i}}$$

Trong đó:

- $\hat{y}_{q,j}^{FL}$: điểm dự đoán dựa trên đặc trưng cho nhãn j của đối tượng kiểm tra x_q.
- $x_{q,i}$: giá trị đặc trưng I của đối tượng x_q.
- $sim_{FL}(f_i, l_j)$: giá trị độ tương đồng của đặc trưng i và nhãn j.
- β: hệ số mũ.

Việc tìm kiếm siêu tham số tốt nhất cho siêu tham số k lân cận với việc dự đoán dựa trên đặc trưng không giúp cải thiện kết quả dự đoán, do đó thuật toán không lọc

ra k đặc trưng tương đồng lớn nhất cho mỗi nhãn trước khi tính điểm dự đoán mà sử dụng tất cả đặc trưng [15].

Algorithm 5: FEATUREKNNPREDICT(x_q, S, β) Computes feature-based confidence scores for labels

Input: A query instance x_q, a similarity matrix S, β for the power transform
Result: Prediction scores for labels

$\hat{\mathbf{y}} \leftarrow$ EMPTY_HASH_MAP();
/* For each nonzero feature i */
for $x_{q,i}$ ***in*** x_q **do**
    /* For each label j with nonzero similarity with feature i */
    for $S_{j,i} \neq 0 \in S_{*,i}$ **do**
        /* Compute partial confidence score */
        $\hat{\mathbf{y}}_j \leftarrow \hat{\mathbf{y}}_j + x_{q,i} \cdot S_{j,i}^{\beta}$;
normalise $\hat{\mathbf{y}}$ with $\sum x_{q,i}$;
return $\hat{\mathbf{y}}$;

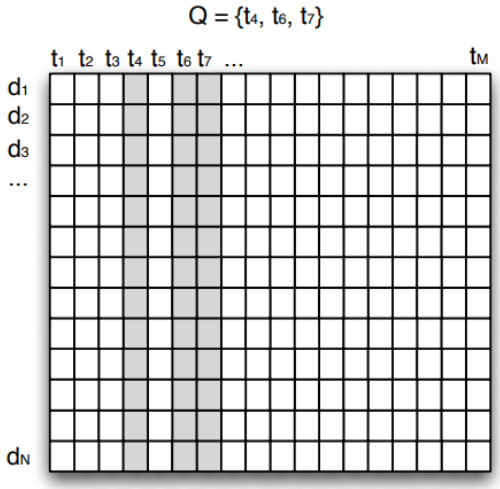
Hình 3.5 Mã giả mô tả thuật toán FeatureKnnPredict

Các bước thực hiện của thuật toán trong cài đặt của nhóm là:

- Bước 1: Áp dụng biến đổi mũ β lên các giá trị độ tương đồng.
- Bước 2: Nhân ma trận giữa ma trận đặc trưng tập kiểm tra và ma trận độ tương đồng.
- Bước 3: Tính tổng giá trị đặc trưng cho từng đối tượng tập kiểm tra.
- Bước 4: Chia giá trị điểm dự đoán cho tổng giá trị đặc trưng theo từng dòng.
- Bước 5: Lọc ra những nhãn có điểm dự đoán cao nhất, với số lượng nhãn tối đa tương ứng với số lượng nhãn trung bình của mỗi đối tượng trong tập huấn luyện.

3.6 Tìm kiếm K tài liệu tốt nhất trong truy hồi thông tin

Ngày nay, tìm kiếm K tài liệu tốt nhất là một tác vụ nền tảng trong nhiều ứng dụng, ví dụ phổ biến nhất là các công cụ tìm kiếm thực hiện thu thập, tính toán để trả về k trang web phù hợp nhất với truy vấn trên tập chỉ mục web nghịch đảo rất lớn và phân tán, ngoài ra còn được áp dụng với thư điện tử, ghi chú, tài liệu, … trong hệ thống quản lý thông tin doanh nghiệp [16], ngoài ra có thể có nhu cầu xử lý một lượng lớn tài liệu có tổng số cụm từ lên đến hàng tỷ cụm từ khác nhau hay yêu cầu truy vấn ngày càng phức tạp hay cần xếp hạng kết quả trả về [17]. Với một kho văn bản D, có thể được coi như ma trận thưa do đa số văn bản chỉ bao gồm một phần nhỏ của các cụm từ phân biệt trong kho, khi thực hiện truy vấn Q hệ thống tiến hành tính điểm số của các tài liệu và câu truy vấn dựa vào một hàm xác định $score(d, Q), d \in D$, thường dựa trên sự giống nhau giữa từ truy vấn và từ trong tài liệu. Cũng theo Fontoura và các cộng sự [16], hai phương pháp tìm kiếm tài liệu có trọng số khác 0 lần lượt là



Hình 3.6 Ví dụ thực hiện truy vấn Q trên ma trận văn bản

đánh giá từng dòng một, nói cách khác là xử lý từng tài liệu và cách thứ 2 là thực hiện theo từng cột, tức là xử lý từng cụm từ.

Qua cài đặt và so sánh với việc tính toàn bộ ma trận tương đồng giữa các đối tượng, nhóm nhận thấy việc áp dụng phương pháp xử lý từng đối tượng đã phần nào giảm được số lượng phép tính cần thực hiện, bỏ qua một số đối tượng có sự tương đồng thấp. như trong bảng dưới đây, thay vì tính toàn bộ ma trận tương đồng với

$\boldsymbol{train} \times \boldsymbol{test}$ phép tính thì việc áp dụng DAAT + Weak AND chỉ thực hiện $(\boldsymbol{test} \times \boldsymbol{N_{TAAT}}) + prod_{DAAT}$ phép tính tương đồng.

dataset	train	test	N_{TAAT}	$prod_{DAAT}$
medical	333	645	312	556
corel5k	5000	500	4500	500
bibtex	4880	2515	4880	0
delicious	12920	3185	12631	2440

Bảng 3. Số cặp đối tượng cần tinh tương đồng khi áp dụng DAAT + Weak AND so với không áp dụng

3.6.1 Xử lý truy vấn theo từng cụm từ phân biệt – Term-at-a-time (TAAT)

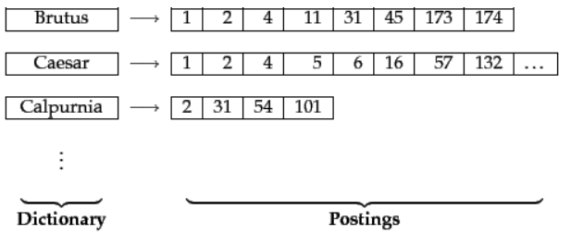
Cho một từ điển với khóa là những cụm từ phân biệt và giá trị là danh sách những văn bản có xuất hiện cụm từ đó, có thể kèm với giá trị thể hiện độ quan trọng của cụm từ trong văn bản, gọi là posting list, thuật toán TAAT duyệt qua từng cụm từ và tính tổng tích lũy điểm số của các văn bản ứng với các cụm từ. Kết quả tổng điểm sẽ được sắp xếp và chọn ra K tài liệu có điểm số cao nhất. Cụ thể khi duyệt đến cụm từ t thực hiện tính toán [16]:

$$A[d] \leftarrow A[d] + d_t q_t$$

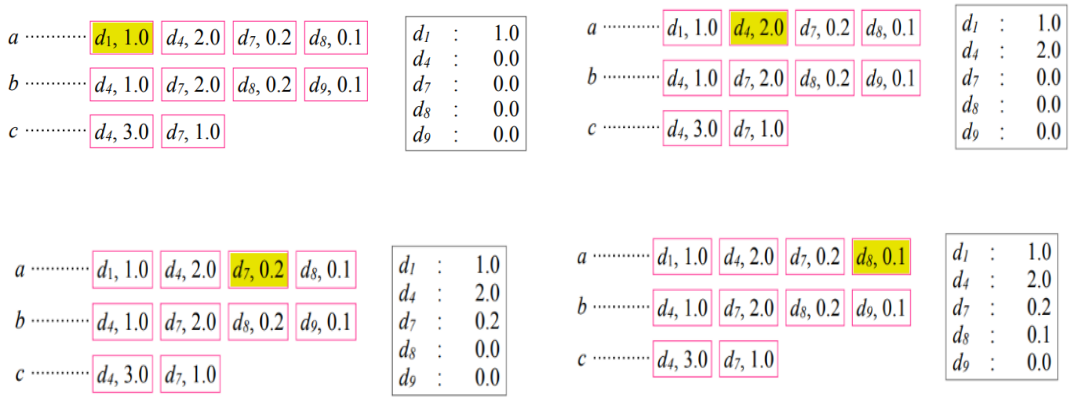
trong đó:

- $A[\dots]$: mảng lưu tổng tích lũy điểm số các tài liệu.
- $A[d]$: điểm số tích lũy của tài liệu d.
- d_t: điểm số của tài liệu d cho cụm từ t.
- q_t: trọng số của cụm từ truy vấn t.

- **Ví dụ posting list [18]**



Hình 3.7 Ví dụ posting list [18]



Hình 3.8 Một vài bước thực hiện thuật toán TAAT, tính tổng tích lũy điểm các tài liệu tại cụm từ "a" [19]

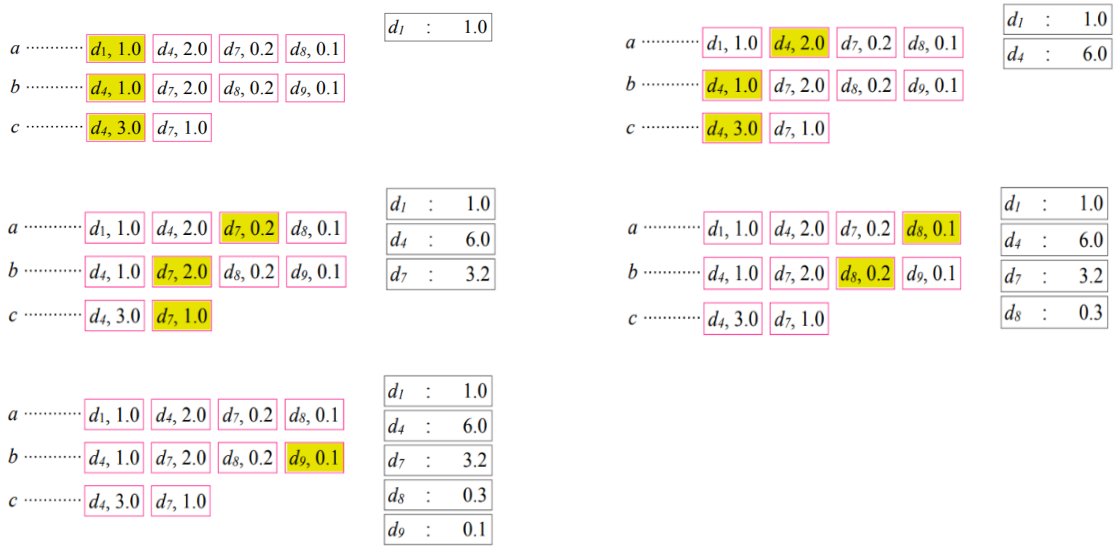
3.6.2 Xử lý truy vấn theo từng tài liệu – Document-at-a-time (DAAT)

Ý tưởng của thuật toán là duyệt đồng thời các posting list cho tất cả cụm từ có trong câu truy vấn. Ở phiên bản đơn giản nhất [16], thuật toán thực hiện tính tổng điểm tích lũy của những văn bản có xuất hiện các từ truy vấn, kết quả được lưu vào một min-heap để thuận tiện quản lý k văn bản có điểm số cao nhất. K văn bản có điểm số cao nhất được đảm bảo lưu trong heap khi quá trình xử lý kết thúc.

Giả sử có một kho văn bản với các posting list với các văn bản đã được sắp xếp, thuật toán DAAT thực hiện các bước [16] [19]:

- Đọc đồng thời các cụm từ trong câu truy vấn $Q = < t_1, \dots, t_q >$.
- Tính điểm của các văn bản cùng loại xuất hiện tai một hay nhiều posting list.

- So sánh điểm số vừa tính với điểm thấp nhất trong heap, nếu cao hơn thì thêm văn bản đang xét vào heap.



Hình 3.9 Ví dụ thực hiện xử lý truy vấn văn bản DAAT [19]

3.6.3 Toán tử Weak-AND – WAND

Ý tưởng của việc áp dụng toán tử WAND là để cải thiện hiệu quả truy vấn thông qua việc sử dụng một giá trị biên trên trong khi đánh giá điểm số của các văn bản đối với câu truy vấn [16]. Với mỗi danh sách văn bản có xuất hiện cụm từ truy vấn, ta có thể tính trước trọng số tối đa có thể có của những văn bản chứa cụm từ được xét. Ví dụ [16] cho truy vấn Q trên kho văn bản D, tại bước khởi động:

- Tính điểm các văn bản ta tính giá trị biên trên tối đa UB cho mỗi cụm từ $t \in Q$:

$$UB_t = D_t q_t$$

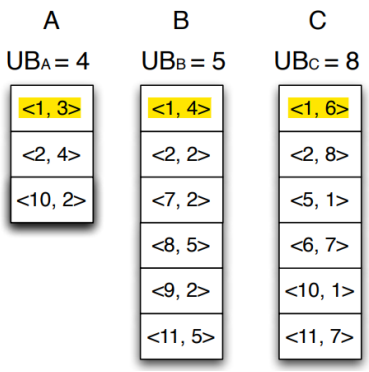
với D_t là giá trị quan trọng d_t lớn nhất của các tài liệu $d \in D$, còn q_t là trọng số của cụm từ truy vấn.

- Cụm từ phân vùng được xác định là cụm từ có chỉ mục p nhỏ nhất thỏa điều kiện:

$$\sum_{1 \le t \le p} UB_t > \theta$$

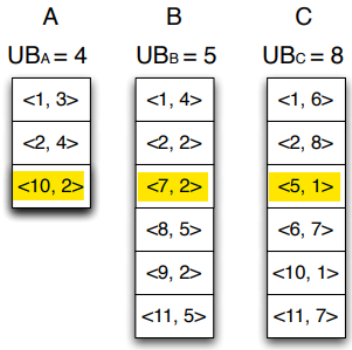
trong đó θ là điểm số thứ K của K văn bản có điểm số cao nhất với truy vấn Q.

- Kiểm tra chỉ mục $docid$ của văn bản cụm từ 1 và p đang xét, nếu bằng nhau văn bản đó được tính điểm, ngược lại thay đổi văn bản duyệt tiếp theo của một cụm từ t, $1 \leq t \leq p$ tới văn bản có chỉ mục $\geq docid$.



Heap	
$docid$	$score(d, Q)$
1	13 (θ)
2	14

	C	B	A
p	1	2	3
$docid$	5	7	10



Posting list và biên trên cho cụm từ truy vấn A, B và C.
$q_A = q_B = q_C = 1$

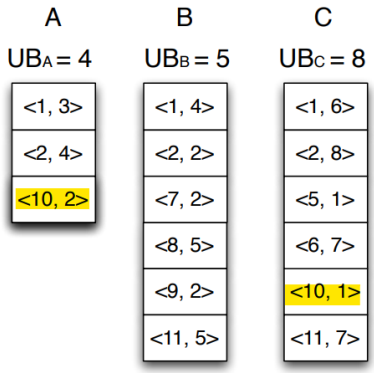
WAND quét mảng tổng hợp và chọn phân vùng:

$p = 1$: $UB_C = 8 < \theta = 13$.

$p = 2$: $UB_C + UB_B = 8 + 5 = \theta = 13$.

$p = 3$: $UB_C + UB_B + UB_A = 8 + 5 + 4 > \theta = 13$.

$k = 2$ tài liệu có điểm số cao nhất lưu trong Heap sau khi xử lý tài liệu 1 và 2. tổng hợp các tài liệu tiếp theo được xử lý trên các cụm từ sắp xếp theo thứ tự chỉ mục tăng dần.



Posting list cho cụm từ C di chuyển tới văn bản $docid$.

Posting list cho cụm từ B không có văn bản $docid$, duyệt tới văn bản 11 và tính ở bước sau.

Hình 3.10 Ví dụ thực hiện DAAT kết hợp WAND

Chương 4

Phương pháp đề xuất

4.1 Giới thiệu thuật toán kết hợp dựa trên đối tượng và đặc trưng

Từ cơ sở lý thuyết phía trên, nhóm tiến hành đề xuất phương pháp kết hợp dựa trên đối tượng và dựa trên đặc trưng. Nguyên nhân phương pháp này được sử dụng bởi vì có thể tận dụng được mối quan hệ giữa các đối tượng và các đặc trưng.

4.1.1 Mô tả thuật toán instance and feature-based k-nearest neighbours (LCIF)

LCIF là 1 thuật toán kết hợp từ lần lượt thực hiện dự đoán các đối tượng kiểm tra sử dụng phương pháp dựa trên đối tượng và dựa trên đặc trưng. Kết quả dự đoán cuối cùng là tổng có trọng số từ 2 độ tin cậy được tính ở trên với trọng số được tính theo siêu tham số λ, công thức:

$$\hat{y}_{q,i} = \lambda\ \hat{y}_{q,i}^{\mathrm{INS}} + (1 - \lambda)\ \hat{y}_{q,i}^{\mathrm{FL}}$$

Trong đó:

$\hat{y}_{q,i}$: điểm dự đoán sau cùng cho nhãn i của đối tượng x_q.

$\hat{y}_{q,i}^{INS}$: điểm dự đoán dựa trên đối tượng cho nhãn i của đối tượng x_q.

$\hat{y}_{q,i}^{FL}$: điểm dự đoán dựa trên đặc trưng cho nhãn i của đối tượng x_q.

λ: độ quan trọng của dự đoán thành phần.

4.1.2 Các bước thực hiện

Thuật toán LCIF: Hàm LCIF(D, X_{test}, k, a, β, λ, t) Dự đoán nhãn dựa vào sự kết hợp của instance-based và feature-based

Input: Tập dữ liệu training D, tập dữ liệu test X_{test}

k: k phần tử là chỉ số của nhãn có độ tương đồng lớn nhất

a: chỉ số tính giá trị đặc trưng của mỗi nhãn trong thuật toán instance-kNN.

β: chỉ số tính giá trị hàm mũ trong thuật toán featureKNNPredict

λ: tham số tính giá trị cho mỗi đặc trưng.

t: sử dụng ngưỡng đo để lấy các nhãn.

Output: Dự đoán nhãn cho mỗi đối tượng trong tập X_{test}

IID ←createIndex(D)

S ←CreateSimilMatrix(D)

$\hat{y} \leftarrow \emptyset$

for x_q in X_{test} do

kNN ← InstanceKnnSearch(x_q, k, IID)

$\hat{y}_q^{Ins}$ ← InstanceKnnPredict (x_q, kNN, a)

$\hat{y}_q^{FL}$ ← FeatureKnnPredict (x_q, S, β)

$\hat{y}_q^{LCIF}$ ß $\lambda\ \hat{y}_q^{Ins} + (1 - \lambda)\ \hat{y}_q^{FL}$

$\hat{y}_q$ ß $\hat{y}_{q,j} \mid \hat{y}_{q,j} \in\ \hat{y}_q^{LCIF} : \hat{y}_{q,j} \geq t$

$\hat{y}$ ß $\hat{y}\ \cup \hat{y}_q$

Return $\hat{y}$

Hình 4.1 Mã giả thuật toán LCIF [20]

Nhóm giả định các tập dữ liệu huấn luyện và kiểm tra đầu vào chưa qua bước tiền xử lý chuyển đổi các véc-tơ dữ liệu về véc-tơ đơn vị, vì vậy các bước thực hiện bao gồm:

Bước 1: Chuẩn hóa giá trị theo dòng cho tập huấn luyện và kiểm tra.

Bước 2: Tính độ tương đồng của các đối tượng kiểm tra với đối tượng huấn luyện, lọc ra k giá trị tương đồng cao nhất ở mỗi dòng.

Bước 3: Dự đoán nhãn dựa trên độ tương đồng giữa các đối tượng.

Bước 4: Chuẩn hóa giá trị theo cột cho tập huấn luyện và kiểm tra.

Bước 5: Tính độ tương đồng của các đặc trưng và nhãn với đối tượng huấn luyện

Bước 6: Tính số lượng nhãn trung bình ở mỗi đối tượng trong tập huấn luyện

Bước 7: Dự đoán nhãn dựa trên độ tương đồng giữa đặc trưng và nhãn.

Bước 8: Tổng hợp kết quả của hai phương thức dự đoán.

4.2 Phân vùng tập dữ liệu

Để giải quyết vấn đề phải duyệt qua tất cả đối tượng khi tính độ tương đồng, dẫn đến hiệu năng bị ảnh hưởng lớn khi số lượng đối tượng tăng lên rất cao, [15] đề xuất một thuật toán tìm kiếm k lân cận kết hợp 2 hướng duyệt qua các đối tượng liên quan nhất ứng với từng đối tượng kiểm tra. Theo đó, tập dữ liệu được phân chia thành 2 tập con rời nhau gồm tập term-at-a-time (TAAT) và document-at-a-time (DAAT) với TAAT lưu những đối tượng có bất kỳ giá trị đặc trưng nào thuộc vào nhóm m giá trị lớn nhất, xét theo từng đặc trưng, những đối tượng còn lại được lưu vào tập DAAT.

4.2.1 Mô tả thuật toán CreateIndexPartition

Thuật toán createIndexPartion: createIndexPartion(D,m) Xây dựng dữ liệu phân vùng cho 2 tập TAAT và DAAT.

Input: Tập dữ liệu training D

m: số lượng m phần tử lớn nhất.

Output: Dữ liệu phân vùng cho 2 tập TAAT và DAAT

IID ←createIndex(D)

I_{taat} ß ∅

for f_i in D do

{($x'_1, x'_{1,j}$), …, ($x'_m, x'_{m,j}$) ß HeapSortTopK(IID[f_i], m)

I_{taat} ß I_{taat} ∪ {x'_1, … , x'_m}

I_{daat} ß D \ I_{taat}

IID_{taat} ß CreateIndex(I_{daat})

IID_{daat} ß CreateIndex(I_{taat})

Sort giá trị đặc trưng theo IID_{taat}

Sort chỉ số đối tượng theo IID_{daat}

Max_{daat} ← Max giá trị for f_i in I_{daat}

∅ ← {I_{taat}, IID_{taat}, I_{daat}, IID_{daat}}

Return ∅

Hình 4.2 Mã giả thuật toán CreateIndexPartition

Đầu tiên, tạo mảng đánh dấu những dòng xếp vào tập TAAT.

Với mỗi cột đặc trưng:

- Sắp xếp và lấy ra m giá trị lớn nhất.
- Đánh dấu những dòng có giá trị thuộc nhóm trên lưu vào TAAT.
- Duyệt qua mảng đánh dấu, sắp xếp các đối tượng vào TAAT và DAAT tùy theo giá trị true/false tại phần tử tương ứng.
- Sắp xếp tập DAAT theo thứ tự chỉ mục tăng dần.

Tại tập DAAT, tìm giá trị lớn nhất ở mỗi cột.

Kết quả trả về là bộ 3 ma trận TAAT, DAAT và giá trị lớn nhất ở mỗi cột theo tập DAAT.

4.3 Dự đoán dựa trên đối tượng áp dụng kỹ thuật truy vấn trên từng đối tượng truy hồi thông tin

4.3.1 Mô tả thuật toán InstanceKNNFast

Việc phân chia tập dữ liệu thành 2 phần cho phép giảm số đối tượng cần thực hiện tính độ tương đồng khi chỉ cần tính trên toàn tập TAAT, trong khi các đối tượng trong

$$\sum_{x_{q,j} \in x_q \wedge x_{i,j} \neq 0} UB_{q,j} \leq \sum_{x_{q,j} \in x_q} UB_{q.j} \leq \theta$$

tập DAAT có thể được bỏ qua nếu tổng giá trị biên trên tính tới khi xét đối tượng thấp hơn độ tương đồng lớn thứ k, công thức:

với $$UB_{q,j} = max(\{x_{i,j} \mid x_i \in I_{daat}\}) \cdot x_{q,j}.$$

4.3.2 Các bước thực hiện

Thuật toán InstanceKNNFast: InstanceKNNFast(x_q, k, ∅) Tìm k đối tượng lân cận cho mỗi đối tượng trong tập test cho cả 2 phân vùng TAAT và DAAT.

Đầu vào: x_q: Mỗi đối tượng trong tập test.

k: số lượng k đối tượng lân cận.

∅: chỉ mục đảo ngược trong ∅.

Đầu ra: k đối tượng lân cận của x_q.

---

kNN_{taat} ß InstanceKnnSearch(x_q, k, IID_{taat})

Heap ← CreateHeap(kNN_{taat})

UB_q ← ∅

for $x_{q,j}$ in x_q do

if $x_{q,j} \neq 0$:

$UB_{q,j} = x_{q,j}\ Max_{daat[j]}$

Offsets ← ∅

While offsets ≠ [-1, …, -1] do

next ← {}

for $x_{q,j} \neq 0$ in x_q và $offsets_j \neq$ -1 do

next ← next ∪ IID_{taat}[j][$offsets_j$]

next ← sort tăng dần trên ID

pivot ← ϕ

$UB_{cur} \leftarrow 0$

for (x_i, $x_{i,j}$) in next do

$UB_{cur} \leftarrow UB_{cur} + UB_{b,j}$

if UB_{cur} > MinHeap(heap):

pivot $\leftarrow x_i$

break

$sim_{q,p} \leftarrow x_q$. pivot

heap $\leftarrow$ PushPop(heap, pivot, $sim_{q,p}$)

Return heap

Hình 4.3 Mã giả thuật toán InstanceKNNFast

Bước 1: Thực hiện tính độ tương đồng của tập kiểm tra với các đối tượng trong tập TAAT.

Nếu tập DAAT không có đối tượng nào thì trả về kết quả độ tương đồng nêu trên và kết thúc.

Bước 2: Tính tích vô hướng giữa các véc-tơ đặc trưng trong tập kiểm tra và tập giá trị đặc trưng lớn nhất (DAAT).

Bước 3: Duyệt từng đối tượng kiểm tra, với mỗi đối tượng lặp đến khi gặp điều kiện dừng.

Bước 4: Xác định các đối tượng ứng viên trong tập DAAT là đối tượng đầu tiên chia sẻ đặc trưng với đối tượng kiểm tra, ứng với mỗi đặc trưng.

Nếu không có ứng viên thì qua đối tượng kiểm tra kế tiếp.

Bước 5: Xác định đối tượng có độ tương đồng cao thứ k và giá trị.

Với mỗi ứng viên, lần lượt cộng điểm xét duyệt với giá trị trong ma trận ở bước 3 tại cột đặc trưng nơi ứng viên được chọn, dừng khi điểm xét duyệt cao hơn giá trị tương đồng nhỏ nhất và chọn ứng viên dang xét hoặc duyệt qua tất cả ứng viên.

Nếu không chọn được ứng viên thì qua đối tượng kiểm tra tiếp theo.

Bước 6: Tính độ tương đồng của đối tượng kiểm tra và ứng viên được chọn.

Nếu độ tương đồng lớn hơn độ tương đồng thứ k hiện có thì thay thế.

Với mỗi đặc trưng, tìm đối tượng đầu tiên có chỉ mục lớn hơn ứng viên được chọn, nếu không còn thì đánh dấu đã duyệt hết đối tượng.

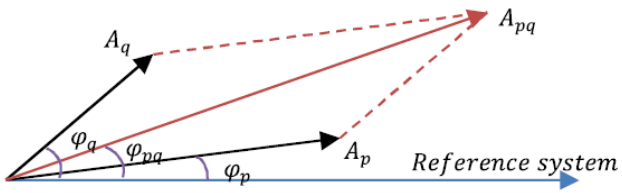
4.4 Độ tương đồng cộng hưởng

Theo như Zhenhua Tan và Liangliang He [21], độ tương đồng cosine thường dùng trong hệ thống tư vấn tồn tại một số hạn chế do tập trung chủ yếu vào hướng của các véc-tơ dữ liệu mà bỏ qua một số yếu về số phần tử tương quan và các giá trị thành phần:

Problem	Examples			PCC	COS	PIP	RES_Eq3	RES_Eq4
	ID	Vector u	Vector v				$\mathbb{K}$ = [1,1,1,1]	$\mathbb{K}$ = [1,1,1,1]
Equal-ratio	1	[1,2,1]	[1.5,3,1.5]	1.0	1.0	6015.4719	10.3781	0.9388
	2	[1,2,1]	[2.5,5,2.5]	1.0	1.0	2029.6406	2.2512	0.7339
	3	[1,2,1,1,4]	[0.5,1,0.5,0.5,2]	1.0	1.0	14913.2172	19.8975	0.9680
Unequal-length	4	[1,2]	[1,2]	1.0	1.0	5690.250	13.8312	0.9541
	5	[1,2,4,5,5,5,5,5]	[1,2,5,5,5,5,5,5]	0.9765	0.9971	28257.870	65.4555	0.9903
Flat-value	6	[1,1,1]	[1,1,1]	NaN	1.0	11911.860	25.6892	0.9752
	7	[1,1,1]	[3,3,3]	NaN	1.0	441.0	1.1164	0.5350
	8	[1,1,1]	[5,5,5]	NaN	1.0	3.0	0.0412	0.0262
Opposite-value	9	[1,5,1]	[5,1,5]	-1.0	0.4042	3.0	0.0125	0.0080
	10	[2,4,2]	[4,2,4]	-1.0	0.8165	75.0	0.2364	0.1478
	11	[2,3,2]	[4,3,4]	-1.0	0.9469	302.0	1.8632	0.6864
Single-value	12	[1]	[1]	NaN	1.0	5285.250	12.1825	0.9479
	13	[1]	[3]	NaN	1.0	147.0	0.5684	0.3291
	14	[1]	[5]	NaN	1.0	1.0	0.0137	0.0087
Cross-value	15	[1,5]	[5,1]	-1.0	0.3846	2.0	0.0002	0.0001
	16	[1,4]	[4,2]	-1.0	0.4706	31.0	0.0757	0.0481
	17	[2,4]	[3,2]	-1.0	0.8682	324.0	0.9626	0.4879
	18	[5,1]	[5,4]	1.0	0.8882	2585.250	3.6253	0.8287
	19	[5,1]	[5,2]	1.0	0.9833	3137.250	5.2020	0.8791
	20	[5,2]	[4,1]	1.0	0.9908	1536.0	1.9971	0.7045

Bảng 4.1 So sánh kết quả độ tương đồng PCC, COS, PIP và RES ở một số ví dụ véc-tơ dữ liệu [21]

Phần nào giải quyết những vấn đề trên, nhóm tác giả Zhenhua Tan và Liangliang



Hình 4.4 Ví dụ hiện tượng cộng hưởng giữa 2 dao động điều hòa [21]

He đã đề xuất một phương pháp tính độ tương đồng mới dựa vào hiện tượng cộng hưởng trong vật lý, tại đó hai giao động điều hòa đơn giản có hướng gần nhau sẽ có độ đồng nhất cao hơn và đạt được biên độ dao động cộng hưởng lớn hơn. Độ tương

đồng này bao gồm 3 thành phần gồm độ nhất quán (C), khoảng cách (D) và hệ số tương tự Jaccard (J). Kết quả độ tương đồng giữa hai đối tượng u, v sẽ là tổng các tích của 3 thành phần nêu trên [21]:

$$RES(u,v) = \sum_{I_{u,v}} C(u,v,k_1) * D(u,v,k_2,k_3) * J(u,v,k_4)$$

với $I_{u,v}$ là tập những sản phẩm u, v cùng đánh giá. Các tham số $k_1 \sim k_4$ được học từ tập dữ liệu.

4.4.1 Độ nhất quán giữa các đối tượng – Consistency (C)

Trong một hệ quy chiếu vật lý, hai dao động diều hòa p và q có pha ban đầu φ_p và φ_q càng gần nhau thì càng có sự đồng nhất cao, dẫn tới giá trị biên độ cộng hưởng A_{pq} càng cao [21]. Hiện tượng trên dẫn tới đề xuất của nhóm tác giả rằng 2 người dùng có những đánh giá sản phẩm gần giống nhau có thể có sở thích giống nhau, qua đó thực hiện tính độ đồng nhất giữa 2 véc-tơ điểm đánh giá của 2 người dùng.

Nhiệm vụ chính của độ đồng nhất là biểu diễn đánh giá của người dùng bằng giá trị tương tự pha ban đầu φ của dao động, trong đó xem xét hai yếu tố [21]:

- Khoảng cách giữa đánh giá người dùng u và trung vị R_{med} của tập dữ liệu, được gọi là giá trị khởi tạo của pha ban đầu φ^{base}:

$$\varphi_u^{base} = r_{u,i} - R_{med}$$

với $r_{u,i}$ là đánh giá của sản phẩm i của u và $R_{med} = \frac{R_{max} - R_{min}}{2}$.

- Khoảng cách giữa đánh giá người dùng và trung bình $\bar{r}_u$ đánh giá của người dùng, được gọi là sở thích đánh giá được cá nhân hóa. Xét điều kiện:

$$condition = \varphi_u^{base} \times (\bar{r}_u - \mu)$$

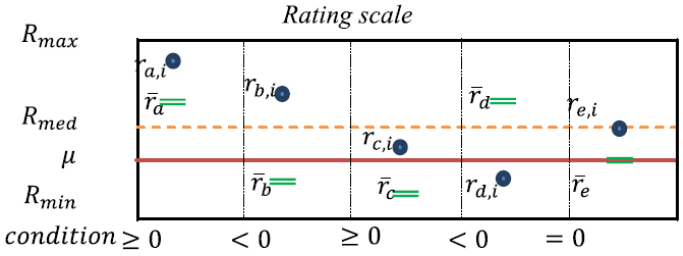
với μ là điểm đánh giá trung bình trên toàn tập dữ liệu. Giá trị $condition$ quyết định giá trị pha ban đầu sau cùng của người dùng u theo hai trường hợp:

$$\varphi_u = \begin{cases} \frac{\pi}{R_{max} - R_{min}} \cdot \varphi_u^-, & condition < 0 \\ \frac{\pi}{R_{max} - R_{min}} \cdot \varphi_u^+, & condition \geq 0 \end{cases}$$

$$\varphi_u^+ = \frac{1}{1+|\bar{r}_u - \mu|} \cdot \varphi_u^{base};\ \varphi_u^- = 1 + \frac{|\bar{r}_u - \mu|}{R_{med}} \cdot \varphi_u^{base}$$

- Cuối cùng độ đồng nhất giữa 2 người dùng là:

$$C(u, v, k_1) = \left(\sqrt{0.5 + 0.5\cos(\varphi_u - \varphi_v)}\right)^{k_1}$$

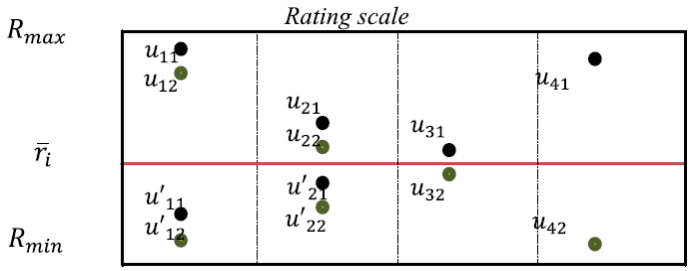


Hình 4.5 Ví dụ kết quả condition của một số người dùng

4.4.2 Hệ số khoảng cách – Distance (D)

Nhận thấy đánh giá của người dùng trên cùng một sản phẩm thường khác so với đánh giá trung bình của sản phẩm đó, Tan và He [21] đã đề xuất hệ số khoảng cách để phản ánh sự tương đồng trong ý kiến của những người dùng trên sản phẩm cùng đánh giá. Hệ số trên xét biểu thức:

$$(r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i)$$



Hình 4.6 Ví dụ về khác biệt trong đánh giá giữa những người dùng

với $r_{u,i}, r_{v,i}$ là đánh giá sản phẩm i của người dùng u và v, còn $\bar{r}_i$ là đánh giá trung bình của i và chia ra 2 trường hợp, tóm tắt như sau:

$$D(u, v, k_2, k_3) = \begin{cases} D^-(u, v, k_3), & (r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i) < 0 \\ D^+(u, v, k_2), & (r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i) \geq 0 \end{cases}$$

- $(r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i) \geq 0$: đánh giá của u và v cho i đều trên hoặc dưới trung bình. Lúc này hệ số khoảng cách được xác định bởi hai thành phần:

- $d_1^+(u,v) = e^{-|r_{u,i} - r_{v,i}|}$: $|r_{u,i} - r_{v,i}|$ càng lớn cho thấy độ tương đồng u và v càng thấp.

- $d_2^+(u,v) = e^{\frac{1}{2} \times (|r_{u,i} - \bar{r}_i| + |r_{v,i} - \bar{r}_i|)}$: xét yếu tố trung bình khoảng cách giữa $r_{u,i}, r_{v,i}$ và $\bar{r}_i$. $(|r_{u,i} - \bar{r}_i| + |r_{v,i} - \bar{r}_i|)$ càng lớn thể hiện sự tương đồng cao giữa u và v.
  ví dụ trong hình 4.6 với cặp (u_{11}, u_{12}) và (u_{21}, u_{22}), mặc dù $d_1^+(u_{11}, u_{12}) \approx d_1^+(u_{21}, u_{22})$ nhưng điểm số của (u_{11}, u_{12}) cao hơn, qua đó thể hiện ý kiến rõ ràng hơn so với (u_{21}, u_{22}) nên được coi có tính tương đồng hơn.

- Hệ số khoảng cách trong trường hợp này là:

$$D^+(u, v, k_2) = \left(d_1^+(u,v) \times d_2^+(u,v)\right)^{k_2}$$

- $(r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i) < 0$: cho thấy cặp người dùng có ý kiến khác nhau về sản phẩm cùng đánh giá, do đó chỉ cần sử dụng khác biệt trong đánh giá để biểu diễn khoảng cách theo hàm $d_1^-(u,v)$ với ý nghĩa tương tự $d_1^+(u,v)$:

$$d_1^-(u,v) = e^{-|r_{u,i} - r_{v,i}|}$$

ví dụ cặp (u_{31}, u_{32}) và (u_{41}, u_{42}) cho đánh giá không cùng phía so với trung bình $\bar{r}_i$, nhưng (u_{31}, u_{32}) có khác biệt trong điểm đánh giá ít hơn nên có tính tương đồng hơn cặp(u_{41}, u_{42})

- Hệ số khoảng cách trong trường hợp này là:

$$D^-(u, v, k_3) = d_1^-(u,v)^{k_3}$$

4.4.3 Chỉ số Jaccard (J)

Cho ví dụ về các cặp người dùng như bảng:

	$\lvert I_u \rvert$	$\lvert I_v \rvert$	$\lvert I_{u,v} \rvert$
(u_1, v_1)	15	50	2
(u_2, v_2)	15	50	15
(u_3, v_3)	150	500	15

Cặp (u_1, v_1) thường có độ tương đồng cosine cao do chỉ có ít đặc trưng chung. Với cặp (u_2, v_2) và (u_3, v_3), độ tương đồng cosine là tương đương cho 2 cặp do có cùng số đặc trưng chung, nhưng cặp(u_2, v_2) nên có độ tương đồng cao hơn do tỷ lệ đặc trưng chung trên tổng số đặc trưng cao hơn so với cặp (u_3, v_3). Hệ số Jaccard được áp dụng để thể hiện yếu tố trên trong độ tương đồng giữa hai đối tượng, được tính theo công thức:

$$J(u, v, k_4) = \left(\frac{\lvert I_{u,v} \rvert}{\lvert I_u \rvert + \lvert I_v \rvert - \lvert I_{u,v} \rvert} \right)^{k_4}$$

trong đó:

- $\lvert I_u \rvert$, $\lvert I_v \rvert$ và $\lvert I_{u,v} \rvert$ lần lượt là số đặc trưng của u, v và số đặc trưng chung.
- $\lvert I_u \rvert \cup \lvert I_v \rvert = \lvert I_u \rvert + \lvert I_v \rvert - \lvert I_{u,v} \rvert$: tổng số đặc trưng của u và v.

4.5 Tối ưu toàn cục sử dụng thuật toán Basin-hopping.

Basin-hopping là một thuật toán ngẫu nhiên cố gắng tìm kiếm cực tiểu toàn cục của một hàm số một hay nhiều biến [22]. Tại mỗi lượt chạy, thuật toán thực hiện tạo một tập gồm các thay đổi ngẫu nhiên giá trị ứng viên khởi tạo hay ứng viên tốt từ lần chạy trước, sau đó tìm cực tiểu cục bộ cho mỗi ứng viên [23]. Việc thay đổi các giá

trị biến để tạo ứng viên mới cho phép thuật toán di chuyển đến những vùng mới trong không gian tìm kiếm và có thể tìm được một cực tiểu toàn cục mới. cực tiểu tốt nhất từ những ứng viên có thể được giữ lại hoặc loại bỏ sử dụng hàm quyết định ngẫu nhiên [22]

$$e^{\frac{-func(xnew)-func(xold)}{T}}$$

với func() là hàm mục tiêu cần tối ưu và T là tham số "nhiệt độ" tương tự như trong thuật toán simulated annealing [24]. Tham số T có thể được điều chỉnh qua những lượt chạy thuật toán, cho phép ứng viên bất kỳ có thể được chấp nhận là lời giải tối ưu ở một vài lượt chạy đầu và chỉ chấp nhận những lời giải tốt hơn về sau.

Như đã trình bày tại mục độ tương đồng cộng hưởng, kết quả các thành phần đồng nhất, khoảng cách và hệ số Jaccard giữa 2 đối tượng được áp dụng hàm mũ với các số mũ $k_1 \sim k_4$ và các tham số này được học từ tập dữ liệu. với miền giá trị khá lớn, $k_1 \sim k_4 \geq 0$, việc lựa chọn một số lượng nhỏ giá trị để tìm kiếm lưới không phải là cách phù hợp để xác định tham số tối ưu nên nhóm đã áp dụng thuật toán tối ưu basin-hopping vào việc tìm tham số $k_1 \sim k_4$ tốt nhất. với hàm mục tiêu như sau:

Đầu vào:

- k_power: bộ tham số $k_1 \sim k_4$.
- X_val, Y_val: dữ liệu đặc trưng và nhãn của tập đánh giá.
- Y_train: dữ liệu nhãn của tập huấn luyện.
- DAAT: phân vùng dữ liệu đặc trưng của tập huấn luyện được xử lý truy vấn từng đối tượng – document-at-a-time.
- Max_w: mảng điểm số tối đa có thể có cho mỗi đặc trưng sử dụng trong WAND.
- Metric: độ đo đánh giá.
- K_rows, alpha: số lân cận và hệ số mũ cho việc dự đoán.

Thực hiện:

- Áp dụng biến đối mũ cho các thành phần của độ tương đồng cộng hưởng đã tính trước đó, nhận được ma trận tương đồng giữa các đối tượng huấn luyện và đánh giá.
- Xét duyệt và tính độ tương đồng của các đối tượng thỏa điều kiện trong tập DAAT theo phương pháp document-at-a-time.
- Dự đoán nhãn cho các đối tượng đánh giá.
- Đánh giá kết quả theo độ đo được chọn.
- Nếu độ đo đánh giá không phải hamming loss thì đổi dấu kết quả, ngược lại giữ nguyên và trả về kết quả.

Nhóm thực hiện áp dụng basin-hopping để tìm bộ tham số $k_1 \sim k_4$ cho kết quả hàm mục tiêu tốt nhất.

4.6 Các hàm hỗ trợ

4.6.1 Chuẩn hóa các giá trị

Một yêu cầu đối với dữ liệu trong thuật toán lọc cộng tác dựa trên người dùng cũng như là dựa trên sản phẩm đó là các dữ liệu về điểm đánh giá cần được chuẩn hóa, và yêu cầu trên cũng áp dụng cho thuật toán phân lớp đa nhãn dựa trên đối tượng được phát triển từ thuật toán lọc cộng tác. ở đây các giá trị trên từng dòng/cột sẽ được chuẩn hóa bằng cách chia cho tổng giá trị đặc trưng trên dòng/cột đó.

Ví dụ có các dòng dữ liệu $X =$

i	f_1	f_2	f_3	f_4	f_5	f_6
1	1	1	1	0	1	0
2	1	1	1	1	0	0

Các bước thực hiện gồm:

Tính $\frac{1}{\|x_i\|_2} = \frac{1}{\sqrt{\sum_{i=1}^{n} A_i^2}}$

x_i	$\|x_i\|_2$
1	$\frac{1}{\sqrt{1^2+1^2+1^2+0+1^2+0}} = \frac{1}{2}$
2	$\frac{1}{\sqrt{1^2+1^2+1^2+0+1^2+0}} = \frac{1}{2}$

tính $\overline{X} = X \cdot \frac{1}{\|x_i\|_2}$

i	f_1	f_2	f_3	f_4	f_5	f_6
1	1	1	1	0	1	0
2	1	1	1	1	0	0

×

x_i	$\|x_i\|_2$
1	0.5
2	0.5

=

i	f_1	f_2	f_3	f_4	f_5	f_6
1	0.5	0.5	0.5	0	0.5	0
2	0.5	0.5	0.5	0.5	0	0

việc thực hiện chuẩn hóa trên các cột cũng tương tự nhưng tập kiểm tra sẽ sử dụng chuẩn các véc-tơ đặc trưng của những cột đặc trưng trong tập huấn luyện.

4.6.2 Lọc ra k giá trị lớn nhất trên mỗi dòng

Do việc lưu trữ chỉ bao gồm các chỉ mục của những phần tủ có giá trị thay vì là mảng liên tục các giá trị nên việc lọc ra một số lượng giá trị lớn nhất cần phải thực hiện duyệt qua và tổng hợp các cặp (chỉ mục, giá trị) khác nhau cho từng dòng dữ liệu. thao tác này không phỏ biến nên khó có phương thức hỗ trợ sẵn, vì vậy nhóm có cài đặt hàm hỗ trợ để thực hiện việc lọc dữ liệu trên. Các bước thực hiện trên mỗi dòng dữ liệu bao gồm:

- Xác định các chỉ mục cột và giá trị phần tử tương ứng.
- Phân vùng các giá trị tại phần tử thứ k từ cuối, có dạng: < $\leq k$ >, k, < $\geq k$ >.
- Lấy ra các chỉ mục và giá trị từ vị trí k như trên đến cuối.

Ví dụ thực hiện lọc ra k = 2 giá trị lớn nhất trên mảng.

x	l_1	l_2	l_3	l_4	l_5
1	1	1.5625	1.5625	0.5625	0
2	0.4556	1.2656	1.2656	0.81	0

→

x	l_1	l_2	l_3	l_4	l_5
1	1	1.5625	0	0	0
2	0	0	1.2656	0.81	0

Các bước trên có thể không ổn định với những ô có giá trị giống nhau trên từng hàng. Một kết quả khác có thể xảy ra tại ví dụ trên là cột l_3 tại dòng 1 có thể được chọn thay cho cột l_2, tương tự cho cột l_2 ở hàng 2.

Chương 5

Thực nghiệm và kết quả

5.1 Môi trường lập trình

5.1.1 Thư viện SciPy (Scientific Python)

SciPy – được xây dựng dựa trên Numpy – là một thư viện tổng hợp những thuật toán và các hàm chức năng tiện dụng, cung cấp khả năng đáng kể trong việc sử dụng ngôn ngữ lập trình Python phát triển ứng dụng chuyên dụng như tính toán khoa học và toán học một cách hiệu quả [25]. Trong quá trình thực hiện đề tài, nhóm nhận thấy có một số tập dữ liệu có số lượng rất lớn về đối tượng, đặc trưng cũng như nhãn, lên tới hàng trăm ngàn, có thể hàng triệu phần tử [15]. Len Feremans và các cộng sự [15] đã đề xuất cách biểu diễn dữ liệu dạng từ diển với khóa là các chỉ mục đặc trưng hoặc nhãn và giá trị là véc-tơ các cặp (đối tượng, giá trị) thể hiện đối tượng nào có giá trị tại đặc trưng hay nhãn nào, ngoài ra những bước tính toán trên tập dữ liệu ở các thuật toán cũng có thể được biểu diễn dưới dạng phép toán trên ma trận. SciPy với lớp đối tượng ma trận thưa tại mô-đun scipy.sparse chỉ lưu thông tin của những ô có giá trị thể hiện tốt những thuộc tính của biểu diễn từ điển nêu trên, cùng với sự hỗ trợ các phương thức tính toán ma trận giúp cài đặt việc đọc và tính toán trên dữ liệu với ngôn ngữ Python có thể đạt hiện quả như cài đặt C/C++ [15].

5.1.2 Thư viện mpire (Multi Processing Is Really Easy)

Một số công việc như lọc ra một số lượng giá trị lớn nhất ở mỗi dòng dữ liệu hay xét duyệt các đối tượng trong một phân vùng dựa theo các đặc trưng của đối tượng kiểm tra cần có vòng lặp để duyệt qua từng dòng, ngoài ra các công việc trên là độc lập với mỗi dòng nên có thể xử lý song song với các dòng dữ liệu mà không gặp phải ảnh hưởng lớn đến kết quả. Việc thực hiện chạy song song đa luồng trên Python với các chức năng có nhiều bước tính toán, xử lý không có tác dụng do ảnh hưởng từ khóa trình phiên dịch toàn cục và một giải pháp thay thế là cài đặt đa tiến trình [26].

Một số thư viện đa tiến trình như Joblib và Ray có giao diện lập trình với cú pháp có nhiều khác biệt so với thư viện multiprocessing tiêu chuẩn, thêm vào đó là chức năng tính toán phân tán như của thư viện Ray khiến cho việc học và tối ưu khả năng của các thư viện trên gặp không ít trở ngại [27]. Được xây dựng dựa trên cũng như có cú pháp tương tự với thư viện multiprocessing nhưng hiệu năng được cải thiện nhiều, cùng với đó là thông tin chi tiết, bao gồm những tham số truyền vào khi xảy ra lỗi và dễ dàng khai báo đối tượng chia sẻ giữa các tiến trình chính là lý do nhóm sử dụng thư viện MPIRE cho một số tác vụ cần song song hóa.

5.1.3 Google Colab và Intel Devcloud

Đây đều là những môi trường điện toán đám mây đi cùng với máy chủ Jupyter hoặc tương đương cùng với những thư viện phổ biến trong tính toán khoa học hiệu năng cao, học máy, học sâu, … cho phép người lập trình giảm bớt việc cài đặt các môi trường thư viện liên quan hay những yêu cầu cao về tài nguyên tính toán cần có và tập trung vào xây dựng, phát triển những ứng dụng hay những mô hình học máy [28] [29]. Không bị giới hạn ở những thư viện có sẵn, người dùng hoàn toàn có thể cài đặt thêm một số thư viện khác cần cho ứng dụng đang thực hiện, hay như với Devcloud thì có thể cài đặt hệ thống quản lý gói Conda với phiên bản Python và các thư viện mới hơn tùy theo nhu cầu, với Colab khả năng này có một số hạn chế khi chỉ có thể cài đặt gói tương thích với phiên bản Python có sẵn và không dễ để cập nhật Python. Devcloud còn có điểm lợi khác về tài nguyên tính toán được phép sử dụng với những lựa chọn cấu hình vi xử lý, bộ nhớ trong, … khác nhau, tùy tài khoản người dùng được cấp phép sử dụng cho các cấu hình khác nhau [30]. Một máy chỉ nhận và thực thi một công việc, chương trình từ một người dùng tại một thời điểm, trong đó môi trường thực thi và các thư viện liên quan được khởi tạo và dọn dẹp trước và sau mỗi công việc [31]. Tuy nhiên qua sử dụng nhóm nhận thấy tuy có lựa chọn cấu hình để có thể chạy tốt với các tập dữ liệu cực lớn trong đề tài nhưng mỗi công việc gửi lên bị giới hạn thời gian chạy tối đa là 20 phút, rất ít so với thời gian 12 giờ liên tục của một tiến trình Colab, mặc dù sức mạnh xử lý không cao và có thể bị điều chỉnh.

Qua đó thấy được rằng các công cụ nêu trên không cái nào thay thế hoàn toàn cái còn lại, nhóm tùy chọn công cụ phù hợp nhất tại mỗi công việc để hoàn thành Khóa luận.

5.1.4 Thư viện Numba

Numba [32] là một trình biên dịch tại thời gian chạy hỗ trợ một phấn những kiểu dữ liệu, một số hàm tích hợp và trong thư viện chuẩn đi kèm cũng như thư viện tính toán mảng số Numpy của ngôn ngữ lập trình Python. Hoạt động tốt nhất với những hàm, phương thức sử dụng mảng Numpy cùng các phương thức hỗ trợ và vòng lặp, Numba thực hiện biên dịch và tối ưu hóa mã nguồn tùy theo kiểu dữ liệu của mảng sử dụng thư viện LLVM, giúp quá trình thực thi hàm được biên dịch có tốc độ tiệm cận với ngôn ngữ C hoặc Fortran. Trong quá trình cài đặt độ tương đồng cộng hưởng [21] nhóm nhận thấy một số thành phần so sánh kết quả của biểu thức điều kiện với 2 trường hợp ≥ 0 và < 0 để xác định kết quả. Kiểu dữ liệu ma trận thưa của scipy không cho phép so sánh giá trị phần tử = 0 do đây là giá trị ngầm định các ô không chứa dữ liệu nên cần thực hiện thêm một số bước để ghi nhận những giá trị điều kiện = 0. Ngược lại, việc cài đặt các bước tính toán chỉ xét từng giá trị, theo sát mô tả [21] sẽ không gặp phải vấn đề trên so với cài đặt tập trung sử dụng kiểu ma trận và các phương thức được hỗ trợ nhưng thời gian chạy không tốt. do đó trong cài đặt của mình, nhóm đã áp dụng thư viện Numba hỗ trợ biên dịch, tối ưu hiệu năng các hàm tính toán thành phần của độ tương đồng cộng hưởng nhằm đảm bảo hiệu năng của mô hình.

5.2 Tập dữ liệu

Các thực nghiệm được thực hiện trên miền dữ liệu bao gồm 10 tập dữ liệu bao gồm 5 tập dữ liệu lớn và 2 tập dữ liệu cực đoan (số lượng đặc trưng và nhãn vô cùng lớn khoảng 300.000 danh mục, kích thước lưu trữ dữ liệu từ 1gb trở lên).

Với 5 tập dữ liệu lớn:

- Tập dữ liệu medical bao gồm gần 1000 tài liệu chứa văn bản lâm sàng miễn phí được thu thập tại khoa X quang của trung tâm y tế bệnh viện dành cho

trẻ em. Vấn đề được giao là tạo thêm những chuẩn đoán y khoa hay thủ tục đã mã hóa bằng CD-9-CM dựa trên các xét nghiệm lâm sàng miễn phí.

- Tập dữ liệu Corel5K bao gồm 499 đặc trưng nhị phân hiển thị việc phân loại ngữ cảnh như rừng, biển, bầu trời, …
- Tập dữ liệu Bibtex hiển thị cho một vấn đề về việc gán thẻ. Tập dữ liệu Delicious tương tự như Bibtex. Đối với 2 tập dữ liệu này, các nhãn (hoặc thẻ) đã được chỉ định bằng cách sử dụng các trang web gắn thẻ xã hội Bibsonomy và Del.icio.us.
- Tập dữ liệu IMDB-F hiển thị thể loại phim dựa trên văn bản tóm tắt phim.

Với 2 tập dữ liệu cực đoan:

- Bộ dữ liệu Eurlex là một bộ sưu tập các tài liệu về luật Châu Âu và có gần 4000 danh mục.
- Wiki10 tương ứng với 20.000 bài viết trên Wikipedia.

5.3 Các độ đo đánh giá

Hiệu năng của một mô hình được đánh giá dựa trên tập dữ liệu kiểm thử (test data). Cụ thể, giả sử đầu ra của mô hình khi đầu vào là tập kiểm thử được mô tả bởi vector $\hat{y}_i$ là vector dự đoán phân lớp cho với mỗi đối tượng trong tập kiểm thử. Nhiệm vụ của ta cần so sánh giữa vector dự đoán $\hat{y}_i$ này với vector đúng trong thực tế là vector y_i. Có rất nhiều cách đánh giá một mô hình phân lớp, tuỳ vào những bài toán khác nhau mà chúng ta sử dụng các phương pháp khác nhau. Trong khóa luận này, nhóm sử dụng 5 độ đo bao gồm example based-accuracy, F1-score, Haming loss, micro F1 và macroF1.

5.3.1 Độ chính xác dựa trên đối tượng (Example based-accuracy)

Based-accuracy là độ đo thể hiện trung bình mỗi đối tượng kiểm tra có tỷ lệ dự đoán có nhãn i và dự đoán đúng là bao nhiêu.

$$\text{Example based-accuracy} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{y_i} \cap \hat{\mathbf{y}_i}|}{|\mathbf{y_i} \cup \hat{\mathbf{y}_i}|}$$

Với mỗi đối tượng:

- Tìm phần giao và hội giữa tập nhãn chân trị và dự đoán.
- Nếu phần hội không có phần tử nào thì điểm đánh giá là 1, ngược lại tính điểm theo tỷ lệ số phần tử phần giao trên phần hội.

Kết quả là trung bình cộng điểm số của các đối tượng. Nếu kết quả càng gần đến 1 thì bộ phân lớp càng chính xác => mô hình được huấn luyện càng tốt.

5.3.2 F1-score, micro-F1 và macro-F1

Precision xác định với mỗi nhãn i, có bao nhiêu dự đoán đúng các đối tượng nhận nhãn này trên tổng dự đoán có nhãn i.

$$\text{Precision} = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive\ (FP)}$$

Recall xác định với mỗi nhãn i, có bao nhiêu nhãn dự đoán positive trên tổng số nhãn đúng.

$$\text{Recall} = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative\ (FN)}$$

F1-score là trung bình điều hòa sự cân bằng giữa precision và recall. Do đó nó đại diện hơn trong việc đánh giá độ chính xác trên đồng thời precision và recall.

$$\text{F1-score} = 2 \text{ x } \frac{recall * precision}{recall + precision} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

F1-score có giá trị nằm trong nửa khoảng (0,1]. F1-score càng cao, bộ phân lớp càng tốt. trong bài toán phân lớp đa nhãn với số lượng nhãn có thể lên đến hàng trăm

ngàn, nhóm tập trung sử dụng độ đo microF1 và macroF1. Với microF1 là F1-score mà các giá trị TP, FP và FN được tính tổng trên tất cả giá trị thành phần tương ứng của các nhãn trong tập dữ liệu, khi đó công thức F1-score là:

$$F1_{micro} = \frac{\sum_{j=1}^{L} tp_j}{\sum_{j=1}^{L} tp_j + \frac{1}{2}\left(\sum_{j=1}^{L} fp_j + \sum_{j=1}^{L} fn_j\right)}$$

với tp_j, fp_j và fn_j lần lượt là dự đoán đúng, sai lầm loại I và II của nhãn j trong tập có L nhãn. Mặt khác, độ đo marcoF1 chỉ lấy trung bình không trọng số của các giá trị F1 tại mỗi nhãn, khi đó độ đo này được tính:

$$F1_{macro} = \frac{1}{L} \times \sum_{j=1}^{L} \frac{tp_j}{tp_j + \frac{1}{2}\left(fp_j + fn_j\right)}$$

5.3.3 Hamming loss

Hamming loss là phần của các nhãn sai trên tổng số nhãn.

Trong phân loại đa nhãn, hamming loss tính toán số lượng phủ định sai và phủ định sai trên mỗi trường hợp và sau đó lấy trung bình cộng trên tổng số trường hợp đào tạo.

$$Hamming\ Loss = \frac{1}{nL} \sum_{i=1}^{n} \sum_{j=1}^{L} [I(y_j^{(i)} \neq \hat{y}_j^{(i)}$$

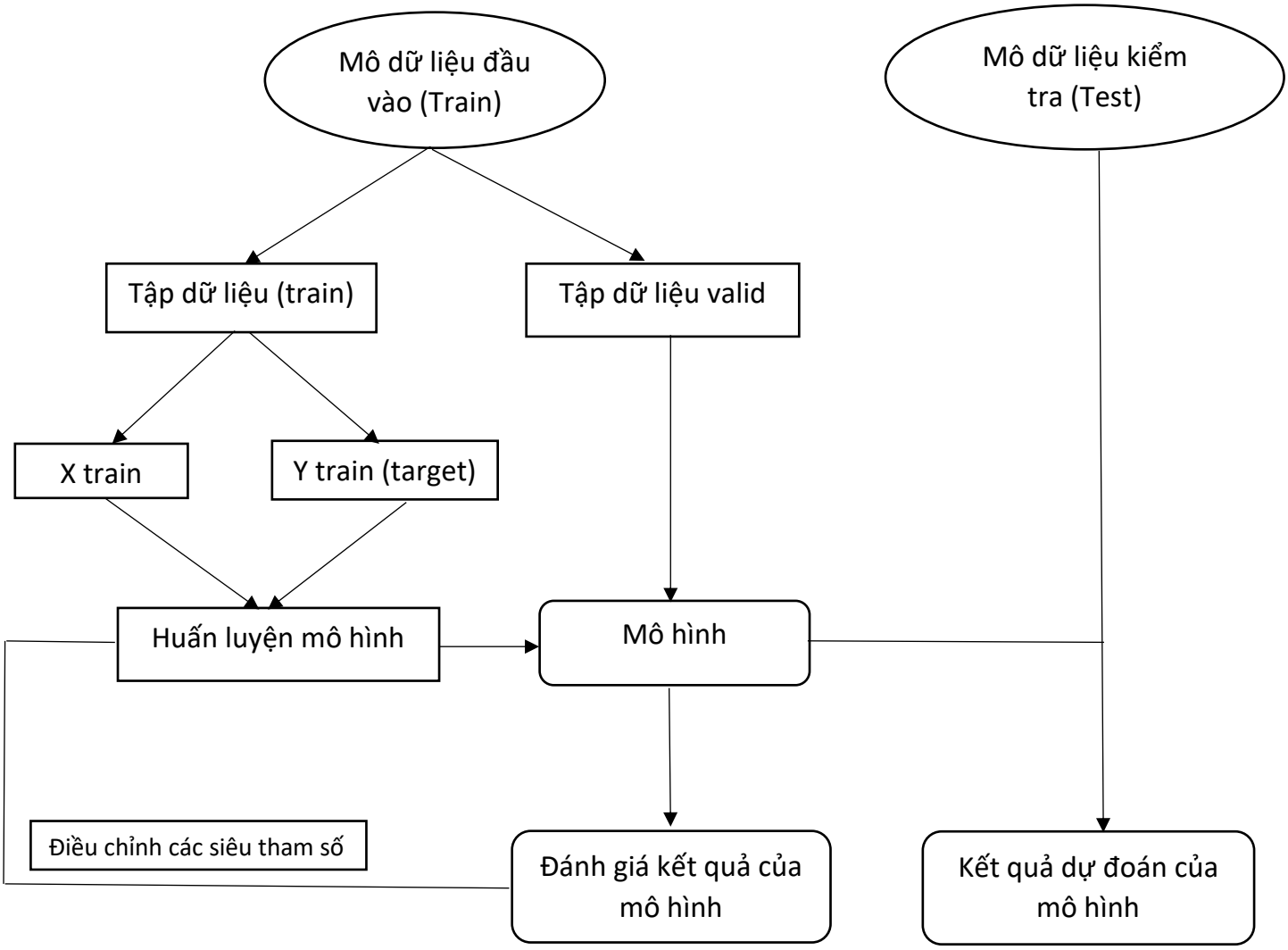
Trong đó: n là số lượng đối tượng của tập huấn luyện.

$y_j^{(i)}$: nhãn đúng cho mẫu đối tượng thứ i và phân lớp thứ j.

$\hat{y}_j^{(i)}$: nhãn dự đoán cho mẫu đối tượng thứ I và phân lớp thứ j.

Nếu kết quả càng nhỏ thì bộ phân lớp càng chính xác => mô hình được huấn luyện càng tốt

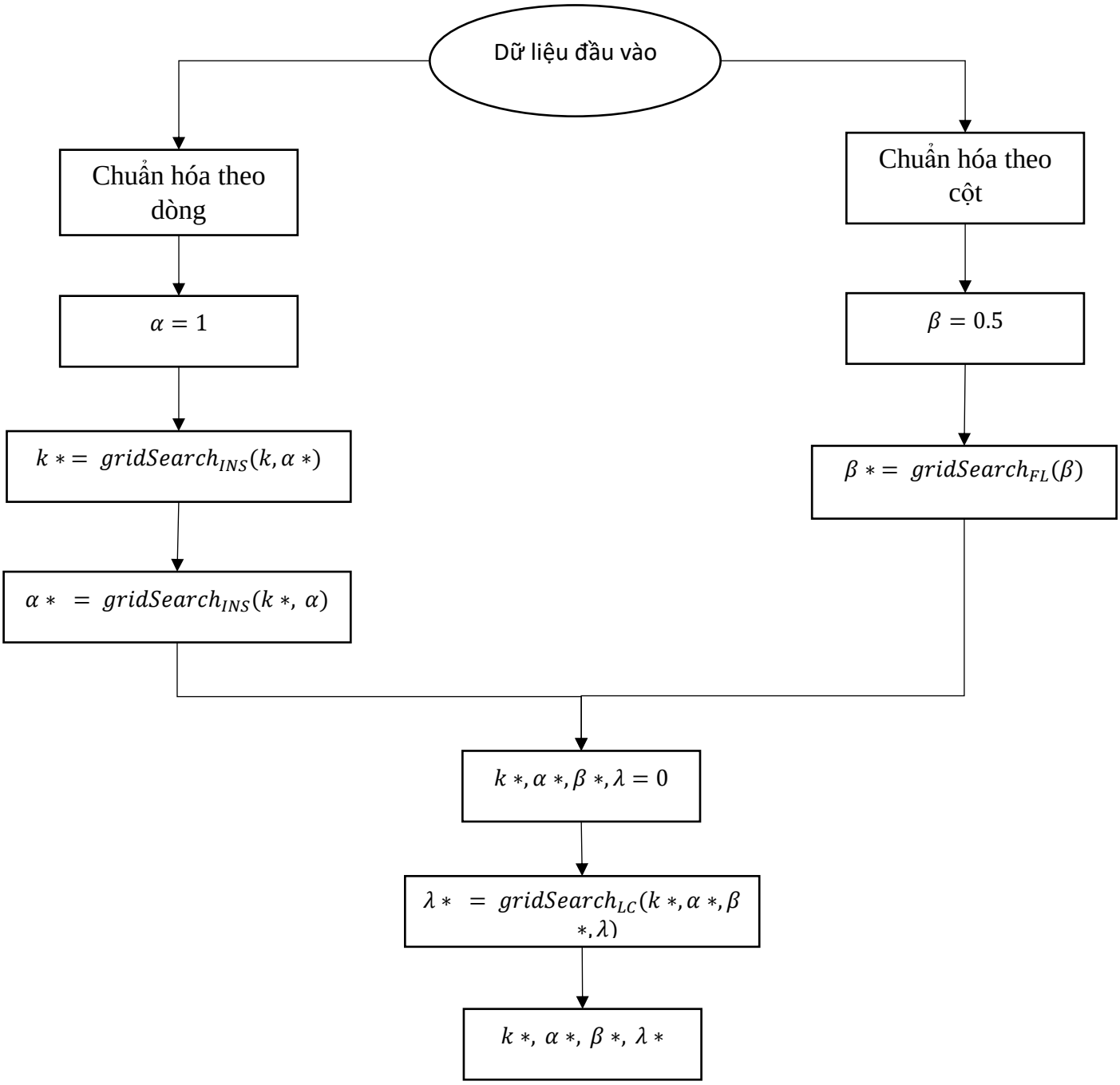
5.4 Quy trình thực nghiệm



Hình 5.1 Quy trình thực nghiệm

Quy trình thực nghiệm tổng quan được phân chia thành 2 bộ thực nghiệm: tập huấn luyện và tập kiểm tra. Bộ dữ liệu huấn luyện được chia làm 2 tập là tập train và tập validation. Sau khi đã huấn luyện mô hình, ta sử dụng tập validation để đánh giá kết quả. Nếu kết quả chưa tốt phải điều chỉnh các siêu tham số và huấn luyện lại mô hình đó đến khi mô hình đạt được hiệu quả tốt nhất. Sau khi đạt được mô hình tốt nhất, ta lấy các tham số trong mô hình đó đi phân lớp cho tập dữ liệu thực tế.

5.4.1 Huấn luyện mô hình



Hình 5.2 Quy trình huấn luyện mô hình

Do mô hình đề xuất là tổ hợp của hai mô hình phân lớp đa nhãn nên quá trình huấn luyện và tìm kiếm siêu tham số tốt nhất cần có 2 tác vụ con thực hiện tối ưu các siêu tham số cho mỗi mô hình thành phần. Các siêu tham số được thực hiện tối ưu hóa bao gồm kRows, α, β, λ:

Siêu tham số	Ý nghĩa	Miền giá trị
kRows	Số lân cận gần nhất cho dự đoán dựa trên đối tượng	{1,5,50,100,150,200,250, 300, 350}
α	Hệ số mũ cho độ tương đồng trên đối tượng	{0.5, 1, 1.5, 2}
β	Hệ số mũ biến đổi độ tương đồng trên đặc trưng	{0.5, 1, 1.5, 2}
λ	Trọng số kết hợp 2 thành phần dự đoán	{0, 0.1, …, 0.9}

Bảng 5.1 Giá trị các siêu tham số sử dụng quá trình huấn luyện và dự đoán

Theo đó việc tìm kiếm siêu tham số tốt nhất bắt đầu với tập dữ liệu huấn luyện và đánh giá chưa qua bước chuẩn hóa dữ liệu, bao gồm 3 thành phần:

Quá trình tìm kiếm lưới siêu tham số cho mô hình phân lớp dựa trên đối tượng diễn ra như sau:

Chuẩn hóa dữ liệu theo dòng.

Khởi tạo kRows = 1, $\alpha = 1$.

Giữ cố định α, thực hiện dự đoán với các giá trị k và chọn ra k* có kết quả tốt nhất trên độ đo xác dịnh.

Từ k* cố định, thực hiện dự đoán với các giá trị α, thu được α*, đồng thời lưu lại tập dự đoán là kết quả của k* và α*.

Quá trình tìm kiếm lưới siêu tham số cho mô hình phân lớp dựa trên đặc trưng:

Chuẩn hóa dữ liệu theo cột.

Khởi tạo $\beta = 0.5$.

dự đoán với các giá trị β và chọn ra β* có kết quả tốt nhất trên độ đo xác dịnh, cũng như lưu lại tập dự đoán tương ứng.

sau cùng tìm kiếm hệ số kết hợp tốt nhất cho hai dự đoán thành phần trên, thu được $\lambda*$. Kết quả là bộ 4 siêu tham số ($k*$, $\alpha*$, $\beta*$, $\lambda*$) tốt nhất cho tập dữ liệu với độ đo được chọn.

5.5 Kết quả

Bảng 5.2 Kết quả chạy thực nghiệm trên các độ đo

	INS.KNN – RES	**INS.KNN - cosine**	**LCIF – RES**	**LCIF - cosine**
Accuracy ↑				
medical	0.585	0.563	**0.671**	0.636
corel5k	0.187	0.163	**0.195**	0.17
bibtex	0.341	**0.347**	0.332	0.341
delicious	0.23	0.23	**0.24**	0.23
IMDB-F	0.25	0.247	**0.253**	0.25
Hamming Loss ↓				
medical	0.02	0.025	**0.018**	0.021
corel5k	0.012	0.014	**0.007**	0.01
bibtex	0.02	0.017	**0.01**	0.014
delicious	0.025	0.025	**0.016**	0.024
IMDB-F	0.09	0.094	**0.082**	**0.082**
Micro F1 ↑				
medical	0.656	0.622	**0.705**	0.69
corel5k	**0.293**	0.266	0.291	0.274
bibtex	0.417	0.426	0.42	**0.427**
delicious	0.368	0.368	**0.371**	0.369
IMDB-F	0.35	0.341	**0.351**	0.346
Eurlex	0.505	0.506	**0.528**	0.517
Wiki10	0.358	**0.359**	0.358	**0.359**

Macro F1 ↑				
medical	0.349	0.339	0.407	**0.492**
corel5k	0.315	0.315	**0.354**	0.315
bibtex	0.32	0.321	**0.328**	**0.328**
delicious	**0.181**	0.18	**0.181**	**0.181**
IMDB-F	0.085	**0.09**	0.08	0.084
Eurlex	0.471	0.509	0.5	**0.512**
Wiki10	0.314	**0.324**	0.314	**0.324**

Nhóm đã tập hợp tất cả các số liệu trên tất cả các độ đo và thể hiện so sánh kết quả dưới dạng bảng. Cụ thể, về độ đo accuracy thì đa phần các kết quả trên các tập dữ liệu dựa trên phương pháp đề xuất của nhóm đều tốt hơn phương pháp của tác giả fereman ngoại trừ tập bibtex lại cho ra kết quả thấp hơn 1 chút. Còn về độ đo hamming loss và độ đo micro1 thì phương pháp đề xuất của nhóm đa phần đều cho ra nhỉnh hơn so với bài báo gốc của tác giả. Cuối cùng về độ đo macroF1 thì cho ra kết quả khá tương đồng, riêng tập corel5K thì kết quả của nhóm cho ra tốt hơn, còn tập Eurlex và wiki10 thì lại cho ra kết quả thấp hơn một chút. Nhìn chung, trên hầu hết các tập dữ liệu, phương pháp kết hợp dựa trên đối tượng và đặc trưng (lcif) cho ra kết quả tốt hơn 2 phương pháp lọc cộng tác dựa trên người dùng và đặc trưng. Vì vậy, việc áp dụng phương phương kết hợp dựa trên đối tượng và đặc trưng (lcif) trong huấn luyện đã chứng minh được hiệu quả của nó.

Chương 6
Kết luận và hướng phát triển

5.6 Kết luận

Trong khóa luận này, từ kết quả nghiên cứu của Len Feremans và cộng sự [15] kết hợp với quy trình chỉnh sửa, nhóm đã đề xuất thuật toán kết hợp dựa trên đối tượng và dựa trên đặc trưng đã cho thấy kết quả được tốt hơn.

Thuật toán LCIF là thuật toán đã xử lý một số vấn đề của quá trình phân lớp dữ liệu đa nhãn: chuẩn hoán dữ liệu, lựa chọn thuộc tính tốt nhất, mối quan hệ giữa các nhãn, .. Quá trình triển khai, cài đặt thử nghiệm cùng với các đánh giá hiệu năng mô hình phân lớp LCIF đã được tiến hành. Và đã thu được nhiều kết quả có ý nghĩa thực tiễn, cũng như các kết quả mở ra những hướng nghiên cứu tiếp theo. Thêm vào đó nhóm cũng thực hiện cài đặt một phương pháp tính độ tương đồng cộng hưởng cho dự đoán đa nhãn dựa trên đối tượng với hy vọng phần nào giải quyết một số vấn đề về số lượng và giá trị của những đặc trưng chung giữa các đối tượng có thể ảnh hưởng tới kết quả tương đồng truyền thống, từ đó cải thiện hiệu năng của mô hình cho bài toán phân lớp đa nhãn.

5.7 Hướng phát triển

Nhóm mong muốn tiếp túc áp dụng những tiến bộ đạt được trong các thuật toán lọc cộng tác, có thể là một thuật toán tính độ tương đồng tiên tiến hơn nhằm tiếp tục cải thiệu hiệu quả của mô hình trên các tập dữ liệu có kích thước lớn và cực lớn. Ngoài ra nhóm cũng hi vọng nâng cao độ chính xác của mô hình thông qua việc xây dựng một tổ hợp lớn hơn bao gồm nhiều mô hình kết hợp hiện có, hay là thử nghiệm các phương pháp tiền xử lý dữ liệu khác, qua đó tìm ra được quy trình áp dụng cho ra kết quả tốt nhất có thể.

Tài liệu tham khảo

[1] P. T. Ngan, “NGHIÊN CỨU CẢI TIẾN PHÂN LỚP ĐA NHÃN VĂN BẢN,” Pham Thi Ngan, 2017.

[2] N. T. T. Linh, “NGHIÊN CỨU CÁC THUẬT TOÁN PHÂN LỚP DỮ LIỆU,” Nguyen Thi Thuy Linh, 2015.

[3] M.-L. ZHANG, “Binary Relevance for Multi-Label Learning,” ZHANG, Min-Ling, 2017.

[4] J. Read, “Classifier Chains for Multi-label Classification,” Jesse Read, 2009.

[5] M.-L. Zhang, “Ml-knn: A Lazy Learning Approach to,” Min-Ling Zhang, 2007.

[6] G. Wu, “Joint Ranking SVM and Binary Relevance with Robust,” Guoqiang Wu, 2019.

[7] E. H. Weiwei Cheng, “Combining instance-based learning and logistic regression for multilabel classification,” Weiwei Cheng, Eyke Hüllermeier, 2019.

[8] Y. Prabhu, “FastXML: A Fast, Accurate and Stable Tree-classifier for,” Yashoteja Prabhu, 2014.

[9] "k-nearest neighbors algorithm," [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[10] "Machine Learning cơ bản - Bài 6: K-nearest neighbors," [Online]. Available: https://machinelearningcoban.com/2017/01/08/knn/.

[11] "Lazy learning," [Online]. Available: https://en.wikipedia.org/wiki/Lazy_learning.

[12] V. H. Tiệp, "Machine Learning cơ bản - Bài 23: Content-based Recommendation Systems," [Online]. Available: https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/.

[13] T. H. Vũ, "Machine Learning cơ bản - Bài 24: Neighborhood-Based Collaborative Filtering," [Online]. Available: https://machinelearningcoban.com/2017/05/24/collaborativefiltering/.

[14] "Cosine similarity," [Online]. Available: https://en.wikipedia.org/wiki/Cosine_similarity.

[15] L. Feremans, B. Cule, C. Vens and B. Goethal, "Combining instance and feature neighbours for extreme multi-label classification," *International Journal of Data Science and Analytics,* no. 3, pp. 215-231, 2020.

[16] M. a. J. V. a. L. J. a. V. S. a. Z. X. a. Z. J. Fontoura, "Evaluation strategies for top-k queries over memory-resident inverted indexes," *Proceedings of the VLDB Endowment,* vol. 4, no. 12, pp. 1213-1224, 2011.

[17] P. R. S. Christopher Manning, “Boolean retrieval,” trong *Introduction to Information Retrieval*, Cambridge University Press, 2008, p. 3.

[18] P. R. S. Christopher Manning, "An example information retrieval problem," in *Introduction to Information Retrieval*, Cambridge University Press, 2008, pp. 6-7.

[19] Max Planck Institute for Informatics, "Information Retrieval & Data Mining," Saarbrücken, 2013/14.

[20] L. Feremans, "lcif - src," [Online]. Available: https://bitbucket.org/len_feremans/lcif/src/master/src/.

[21] Z. a. H. L. Tan, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle," *IEEE Access,* vol. 5, pp. 27211-27228, 2017.

[22] "scipy.optimize.basinhopping," [Online]. Available: https://scipy.github.io/devdocs/reference/generated/scipy.optimize.basinhopping.html#scipy.optimize.basinhopping.

[23] J. Brownlee, "Basin Hopping Optimization in Python," [Online]. Available: https://machinelearningmastery.com/basin-hopping-optimization-in-python/.

[24] "Simulated annealing," [Online]. Available: https://en.wikipedia.org/wiki/Simulated_annealing.

[25] "SciPy - Introduction," [Online]. Available: https://scipy.github.io/devdocs/tutorial/general.html.

[26] A. Ajitsaria, "What Is the Python Global Interpreter Lock (GIL)?," [Online]. Available: https://realpython.com/python-gil/#the-impact-on-multi-threaded-python-programs.

[27] S. Jansen, "MPIRE for Python: MultiProcessing Is Really Easy," [Online]. Available: https://towardsdatascience.com/mpire-for-python-multiprocessing-is-really-easy-d2ae7999a3e9.

[28] "Welcome to Colab!," [Online]. Available: https://colab.research.google.com/.

[29] "Intel® DevCloud," [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/devcloud/overview.html.

[30] "Intel® DevCloud for the Edge - Run Your Code," [Online]. Available: https://www.intel.com/content/www/us/en/secure/developer/devcloud/edge/run-your-code.html.

[31] "Intel® DevCloud for the Edge - Manage Jobs," [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/devcloud/edge/build/manage-jobs.html.

[32] "Numba," [Online]. Available: https://numba.pydata.org/.

[33] e. M.-l. Learning, “FastXML: A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning,” eXtreme Multi-label Learning, 2014.