ĐẠI HỌC QUỐC GIA TP HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

KHÓA LUẬN TỐT NGHIỆP

XÂY DỰNG HỆ THỐNG LỌC RÁC CHO HỆ THỐNG PHÁT HIỆN TIN NÓNG TỪ CÁC TRANG TIN TỨC

Giảng viên hướng dẫn: **TS. Huỳnh Ngọc Tín**

Sinh viên 1: **Hoàng Anh Minh**

MSSV: **13520505**

Lớp: **CNPM08**

Khóa: **2013-2017**

Sinh viên 2: **Lâm Tuấn Anh**

MSSV: **13520020**

Lớp: **CNPM08**

Khóa: **2013-2017**

TP HỒ CHÍ MINH – Tháng 6, năm 2017

ĐẠI HỌC QUỐC GIA TP HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

KHÓA LUẬN TỐT NGHIỆP

XÂY DỰNG HỆ THỐNG LỌC RÁC CHO HỆ THỐNG PHÁT HIỆN TIN NÓNG TỪ CÁC TRANG TIN TỨC

Giảng viên hướng dẫn: **TS. Huỳnh Ngọc Tín**

Sinh viên 1: **Hoàng Anh Minh**

MSSV: **13520505**

Lớp: **CNPM08**

Khóa: **2013-2017**

Sinh viên 2: **Lâm Tuấn Anh**

MSSV: **13520020**

Lớp: **CNPM08**

Khóa: **2013-2017**

Đơn vị đồng hành: **VCCorp**

TP HỒ CHÍ MINH – Tháng 12, năm 2017

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. ……. – Chủ tịch.
2. ……. – Thư ký.
3. ……. – Ủy viên.
4. ……. – Ủy viên.

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến TS. Huỳnh Ngọc Tín, người đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn em trong suốt quá trình thực hiện khóa luận tốt nghiệp này. Những kinh nghiệm, lời nhận xét và chia sẻ của thầy truyền đạt cho em thật sự rất quý báu và ý nghĩa.

Em cũng gửi lời cảm ơn đến quý thầy cô đang công tác tại Trường Đại học Công Nghệ Thông Tin nói chung và Khoa Khoa học máy tính nói riêng đã dạy dỗ, truyền đạt những kiến thức, kinh nghiệm vô cùng quý báu, giúp em vận dụng trong quá trình thực tập.

Ngoài ra, xin cảm ơn sự hỗ trợ quý giá của các bạn trong nhóm AdTech tại công ty VCCorp trong quá trình xây dựng hệ thống.

Cuối cùng, em xin chúc thầy luôn mạnh khỏe và đạt nhiều thành công trong cuộc sống.

Tp. HCM, tháng 12 năm 2017

Sinh viên thực hiện

Hoàng Anh Minh

Lâm Tuấn Anh

Mục lục

Danh mục các ký hiệu, thuật ngữ và chữ viết tắt

Topic Detection and Tracking	: Phát hiện và theo dõi sự kiện
First Story	: Bài viết đầu tiên về một sự kiện
First Story Detection	: Phát hiện bài viết đầu tiên
Nearest Neighbor Search	: tìm láng giềng gần nhất
Locality Sensitive Hashing	: thuật toán Locality Sensitive Hashing
Document	: một bài viết/điểm dữ liệu trong hệ thống
Tweet	: một bài đăng bởi bất kỳ người dùng nào trên Twitter
Merge Threshold	: Giá trị ngưỡng dùng khi xét hai document có đủ tương đồng để gom vào một cụm hay không

Danh sách bảng

Danh sách hình vẽ

TÓM TẮT KHÓA LUẬN

Với sự phát triển chóng mặt của công nghệ số, chúng ta có thể tiếp xúc với một lượng thông tin khổng lồ mỗi ngày. Năm 2016, cứ mỗi phút có gần 150 triệu email được gửi đi, hơn 500 giờ video được đăng tải lên YouTube, và 3.3 triệu bài viết được đăng trên Facebook [1]. Tương tự, trong lĩnh vực truyền thông báo chí, cứ mỗi giờ có khoảng 2,118 bài được đăng trên các trang báo mạng Việt Nam [2]. Điều này dẫn đến tình trạng quá tải thông tin, và đặt ra vấn đề đối với nhà báo: Làm sao để thu hút người đọc khi tin tức được cung cấp tràn lan khắp nơi?

Các kênh thông tin trong lĩnh vực truyền thông rất đa dạng: báo chí, đài truyền hình, đài phát thanh, các trang báo mạng chính thống, mạng xã hội, blog. Những nguồn chính thống thường đưa thông tin chuẩn xác, chất lượng bài viết được kiểm duyệt tốt. Ngược lại các nguồn như mạng xã hội, đặc biệt là Twitter, thường có thông tin chưa được xác thực, kiểm chứng. Tuy nhiên không thể chối cãi tốc độ đưa tin nhanh chóng của các trang mạng xã hội, nhờ vào lượng người dùng rộng rãi khắp nơi.

Về phía biên tập viên, nhà báo, họ cần phải dành thêm nhiều công sức, nguồn lực cho việc tìm kiếm, sàng lọc các nguồn tin, cũng như rất nhiều thời gian để kiểm chứng độ tin cậy của các tin đó. Việc khai thác hiệu quả các nguồn tin không chính thống như các trang mạng xã hội Twitter, Facebook là một vấn đề đáng quan tâm, nghiên cứu.

Hiểu được nhu cầu trên, em thực hiện khóa luận này với mục tiêu nghiên cứu và đánh giá các phương pháp phát hiện tin nóng, cụ thể là trên dữ liệu từ Twitter, và xây dựng một hệ thống có khả năng phát hiện tin nóng trong thực tiễn.

Trong phạm vi của khóa luận, em đã nghiên cứu một số phương pháp gom cụm: k-láng giềng gần nhất, gom cụm có tăng trọng số cho thực thể có tên, và gom cụm

[1] http://www.smartinsights.com/internet-marketing-statistics/happens-online-60-seconds/

[2] Theo thống kê từ dữ liệu thu thập bởi công ty VCCorp tính đến tháng 7/2017.

dùng locality sensitive hashing. Áp dụng một công thức xếp hạng để sắp xếp các tin theo độ nóng, và xây dựng hệ thống phát hiện tin nóng dựa trên các kiến thức đã tìm hiểu được.

Sau quá trình thực hiện, đề tài khóa luận thu thập được các kết quả như sau:

- Thu thập bộ dữ liệu gồm các bài đăng tiếng Việt từ nguồn Twitter.
- Đánh giá và so sánh các phương pháp: k-láng giềng gần nhất, gom cụm có tăng trọng số cho thực thể có tên, và gom cụm dùng locality sensitive hashing.
- Xây dựng được hệ thống phát hiện tin nóng.

Chương 1

MỞ ĐẦU

1.1 Dẫn nhập

Việc đọc báo cập nhật tin tức là nhu cầu thiết yếu của nhiều người. Ngày nay, với sự phát triển mạnh mẽ của các dịch vụ chia sẻ thông tin trên internet, ta có rất nhiều nguồn tin tức dồi dào để lựa chọn. Tuy nhiên, điều này cũng dẫn đến tình trạng quá tải thông tin, và việc tìm ra những "tin nóng" càng trở nên khó khăn. Tin nóng có thể hiểu là những tin có nội dung quan trọng, đáng chú ý, thu hút sự quan tâm của cộng đồng, và có thể gây ảnh hưởng rộng rãi.

Ngoài các nguồn tin tức truyền thống như báo giấy, báo mạng, và các kênh truyền hình, các trang mạng xã hội cũng có thể đóng vai trò là một nguồn tin tức lớn, với các tin tức được chia sẻ bởi chính những người dùng. Twitter là một mạng xã hội hướng tới việc chia sẻ thông tin một cách nhanh chóng, ngắn gọn, và có lượng người dùng toàn cầu hàng tháng lên đến 328 triệu [1]. Thống kê cho thấy: trong thời điểm đầu năm 2017, trên Twitter cứ mỗi phút có khoảng 350,000 tweet được đăng tải[2]. Với những con số trên, ta có thể thấy Twitter cũng như những mạng xã hội khác có tiềm năng trở thành một nơi cung cấp tin tức rất nhanh chóng, tuy rằng tính xác thực nội dung có thể không bằng các kênh tin truyền thống.

Bài toán đặt ra là làm thể nào để có thể khai thác nguồn tin từ mạng xã hội, nhằm đưa tin tức đến cho người đọc một cách chính xác và sớm nhất. Với nhu cầu và điều kiện thuận lời từ công ty VCCorp, khóa luận này hướng đến việc xây dựng một hệ thống có khả năng lọc và phát hiện tin nóng để hỗ trợ cho các biên tập viên báo chí

[1] https://about.twitter.com/company
[2] http://www.internetlivestats.com/twitter-statistics/

trong quá trình tìm kiếm tư liệu để viết bài.

1.2 Mục tiêu đề tài

- Tìm và chọn được phương pháp phù hợp nhất để nhận biết tin nóng, là tin về những sự kiện mới, có khả năng thu hút sự chú ý, quan tâm của nhiều người.
- Xây dựng hệ thống áp dụng phương pháp trên để hỗ trợ biên tập viên trong việc viết bài.

1.3 Nội dung thực hiện

- Tìm hiểu bài toán phát hiện tin nóng: Đọc và thảo luận các bài báo nghiên cứu về phát hiện tin tức trên mạng xã hội.
- Thử nghiệm đánh giá các phương pháp đã tìm hiểu:
  - Thu thập dữ liệu Twitter thông qua Twitter Search API và Streaming API.
  - Tiến hành một số thống kê trên dữ liệu thu thập được.
  - Cài đặt và so sánh các thuật toán: k-láng giềng gần nhất, gom cụm có tăng trọng số cho thực thể có tên, và gom cụm dùng Locality Sensitive Hashing.
- Xây dựng hệ thống:
  - Tìm hiểu về MongoDB, framework Struts 2, thư viện Apache Lucene.
  - Xây dựng kiến trúc hệ thống.
  - Thiết kế chức năng, giao diện hệ thống.
  - Cài đặt hệ thống.

1.4 Phạm vi đề tài

- Nguồn dữ liệu: các bài viết từ mạng xã hội Twitter.
- Ngôn ngữ: tiếng Việt.

- Các phương pháp tiếp cận: k-láng giềng gần nhất, gom cụm có tăng trọng số cho thực thể có tên, và gom cụm dùng Locality Sensitive Hashing. Kết hợp với xếp hạng cụm.

1.5 Cấu trúc báo cáo

Luận văn được bố cục thành chương mục như sau:

- **Chương 1**: Mở đầu: Giới thiệu về đề tài.
- **Chương 2**: Bài toán phát hiện tin nóng và các phương pháp tiếp cận phổ biến: Trình bày cơ sở lý thuyết, các khái niệm, phương pháp tiếp cận liên quan đến bài toán phát hiện tin nóng.
- **Chương 3**: Hiện thực hệ thống phát hiện tin nóng: Trình bày về kiến trúc, cài đặt hệ thống phát hiện tin nóng.
- **Chương 4**: Thực nghiệm và đánh giá: Trình bày về bộ dữ liệu thu thập được, đánh giá và so sánh các thuật toán.
- **Mục Tài liệu tham khảo**
- **Phụ lục. Giới thiệu về thư viện Apache Lucene**

Chương 2

BÀI TOÁN PHÁT HIỆN TIN NÓNG VÀ CÁC PHƯƠNG PHÁP TIẾP CẬN PHỔ BIẾN

2.1 Mở đầu

Chương này sẽ giới thiệu bài toán phát hiện và theo dõi tin tức. Trình bày cơ sở lý thuyết và phát biểu bài toán phát hiện tin nóng. Cuối cùng trình bày một số phương pháp tiếp cận bài toán phát hiện tin nóng và các kiến thức liên quan.

2.2 Giới thiệu bài toán

2.2.1 Các khái niệm cơ bản

Để có thể xác định khái niệm tin nóng, trước hết ta cần xem xét định nghĩa của "sự kiện" và "mẩu tin". Fiscus và Doddington [1] đã tóm tắt định nghĩa về event và story như sau:

- *Sự kiện (event)* là một sự việc bất kì, xảy ra tại một địa điểm cụ thể vào thời điểm xác định, cùng với những điều kiện dẫn đến nó và các hậu quả kéo theo.
- *Mẩu tin (story)* là một bài viết liên quan đến một chủ đề nhất định. Có ít nhất 2 câu trần thuật độc lập với nhau.

Dựa trên quan điểm trên, ta định nghĩa tin nóng như sau:

Định nghĩa: Tin nóng là những tin viết về một sự kiện mới xảy ra, có tính thời sự, có tầm ảnh hưởng rộng, thu hút được sự chú ý, quan tâm của cộng đồng.

2.2.2 Bài toán Topic Detection and Tracking

Topic Detection and Tracking (TDT) là một dự án bao quát được khởi xướng từ năm 1996, với mục tiêu nghiên cứu, phát triển công nghệ cho việc lưu trữ, tổ chức, tìm kiếm dữ liệu từ các nguồn *tin tức* [1].

Nhiệm vụ của TDT là xử lý luồng dữ liệu văn bản liên tục (từ báo chí, từ đài phát thanh, đài truyền hình thông qua các bộ chuyển giọng nói thành văn bản), từ đó theo dõi và phát hiện được các sự kiện đang diễn ra, cũng như tổ chức các bài viết thành những nhóm cùng bàn về một sự kiện nào đó.

Theo Fiscus và Doddington [1], TDT được chia làm 5 tác vụ chính:

- Story segmentation: phân đoạn dữ liệu từ đài phát thanh thành những mẩu tin riêng biệt.
- Topic detection: Gom nhóm các bài viết theo sự kiện, chủ đề chúng đề cập tới.
- Topic tracking: Theo dõi những chuyển biến của sự kiện đã diễn ra.
- First story detection: Phát hiện tin tức đầu tiên nói về sự kiện vừa xảy ra.
- Link detection: Xác định xem một cặp bài viết ngẫu nhiên có đang đề cập về cùng sự kiện không.

2.2.3 Bài toán phát hiện tin nóng

Bài toán phát hiện tin nóng có thể xem là sự kết hợp giữa hai bài toán Topic Detection và First Story Detection (FSD) của TDT. Theo James Allan [2], bài toán First Story Detection, hay còn gọi là New Event Detection, được đặt ra với mục tiêu phát hiện các tin tức, bài viết đầu tiên trên báo chí về các sự kiện vừa xảy ra, chưa từng được báo cáo trước đó. Ví dụ như bài viết đầu tiên đưa thông tin về một vụ tai nạn, khủng bố, hay một vụ scandal chính trị nào đó. Một hệ thống có khả năng phát hiện các first story có thể cung cấp thông tin hữu ích cho các nhà phân tích, các biên tập viên báo chí một cách nhanh chóng, kịp thời.

Đối với bài toán Topic Detection, nhiệm vụ chính là gom các bài viết về cùng một sự kiện thành từng cụm, mỗi cụm là đại diện cho một sự kiện cụ thể nào đó. Do tính chất của bài toán, ta không được biết trước số lượng sự kiện (số cụm) hay nội dung

từng sự kiện như thế nào, điều này hoàn toàn bởi hệ thống tự động xác định. So với FSD, bài toán này tập trung nhiều hơn vào việc nhóm được phần lớn các bài viết liên quan lại với nhau, hơn là việc phát hiện bài viết đầu tiên. Tuy nhiên trên thực tế, nhiều nhà nghiên cứu sử dụng chung một hướng giải quyết cho cả 2 bài toán này [2].

Ngoài nguồn dữ liệu từ báo đài, các bài viết từ mạng xã hội cũng là một nguồn tin có tiềm năng khai thác rất lớn. Tuy chất lượng bài viết và tính xác thực về nội dung có thể không tốt bằng các nguồn chính thống, các tin tức từ mạng xã hội thường có lợi thế về mặt tốc độ đưa tin. Một ví dụ nổi bật cho tính chất này là sự kiện Osama Bin Laden bị ám sát vào năm 2011, khi đó một người dùng là Keith Urbahn đã đưa thông tin lên Twitter nhanh hơn giới truyền thông đến 21 phút.

2.2.4 Phát biểu bài toán phát hiện tin nóng từ Twitter

Cho trước $D = (d_1, d_2, d_3, ...d_n)$: chuỗi các bài viết từ Twitter được sắp xếp theo thứ tự thời gian.

Mục tiêu của bài toán là phát hiện được các cụm tin $C = (c_1, c_2, ...c_m)$, với mỗi phần tử $c_i = (d_{i1}, d_{i2}, ...d_{ik})$ là chuỗi các bài viết bàn về cùng một sự kiện cụ thể, và d_{i1} là bài viết đầu tiên viết về sự kiện đó. Danh sách các chuỗi tin C được sắp xếp theo thứ tự độ nóng giảm dần, với độ nóng được tính toán dựa trên thời gian đăng tin, tầm ảnh hưởng và mức độ quan tâm cộng đồng đối với sự kiện được đề cập.

2.3 Thuật toán Naive Bayes

2.3.1 Giới thiệu thuật toán Naive Bayes

Thuật toán Naive Bayes là một thuật toán phân lớp xác suất đơn giản dùng để tính một tập các xác suất bằng cách đếm tần suất và kết hợp của các giá trị trong một tập cho trước. Thuật toán sử dụng định luật Bayes sẽ giả định rằng tất cả các chiều(attributes) của dữ liệu là độc lập với nhau. Các giả định rằng các chiều là độc lập với nhau rất khó để có thể xuất hiện trong thực tế. Tuy nhiên giả thiết ngây ngô này lại mang lại những kết quả phân lớp tốt cho nhiều bài toán phân lớp [3].

2.3.2 Lý thuyết Bayes

Định lý Bayes cho phép tính xác suất xảy ra của một sự kiện ngẫu nhiên A khi biết sự kiện liên quan B đã xảy ra. Xác suất này được ký hiệu là P(A|B), và đọc là "xác suất của A nếu có B". Đại lượng này được gọi xác suất có điều kiện hay xác suất hậu nghiệm vì nó được rút ra từ giá trị được cho của B hoặc phụ thuộc vào giá trị đó.

Theo định lí Bayes, xác suất xảy ra A khi biết B sẽ phụ thuộc vào 3 yếu tố:

Xác suất xảy ra A của riêng nó, không quan tâm đến B. Kí hiệu là P(A) và đọc là xác suất của A. Đây được gọi là xác suất biên duyên hay xác suất tiên nghiệm, nó là "tiên nghiệm" theo nghĩa rằng nó không quan tâm đến bất kỳ thông tin nào về B.

Xác suất xảy ra B của riêng nó, không quan tâm đến A. Kí hiệu là P(B) và đọc là "xác suất của B". Đại lượng này còn gọi là hằng số chuẩn hóa (normalising constant), vì nó luôn giống nhau, không phụ thuộc vào sự kiện A đang muốn biết.

Xác suất xảy ra B khi biết A xảy ra. Kí hiệu là P(B|A) và đọc là "xác suất của B nếu có A". Đại lượng này gọi là khả năng (likelihood) xảy ra B khi biết A đã xảy ra. Chú ý không nhầm lẫn giữa khả năng xảy ra B khi biết A và xác suất xảy ra A khi biết B.

Tóm lại định lý Bayes sẽ giúp ta tính ra xác suất xảy ra của một giả thuyết bằng cách thu thập các bằng chứng nhất quán hoặc không nhất quán với một giả thuyết nào đó. Khi các bằng chứng tích lũy, mức độ tin tưởng vào một giả thuyết thay đổi. Khi có đủ bằng chứng, mức độ tin tưởng này thường trở nên rất cao hoặc rất thấp, tức là xác xuất sảy ra giả thuyết sẽ thay đổi thì các bằng chứng liên quan đến nó thay đổi.

Công thức của định luật Bayes được phát biểu như sau:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{2.1}$$

Trong đó:

- P(A|B) là xác suất xảy ra của một sự kiện ngẫu nhiên A khi biết sự kiện liên quan B đã xảy ra.
- P(B|A) là xác suất xảy ra B khi biết A xảy ra
- P(A) là xác suất sảy ra của riêng A mà không quan tâm đến B.

- P(B) là xác suất xảy ra của riêng B mà không quan tâm đến A.

Ở trên ta có thể thấy xác suất sảy ra của giả thuyết A phụ thuộc và xác suất của giả thuyết B, nhưng trong thực tế xác suất A có thể phụ thuộc vào xác suất của nhiều các giác thuyết khác có thể là B1, B2, B3 ... Bn. Vậy định luật Bayes có thể được mở rộng bằng công thức sau:

$$P(A \mid B) = \frac{(P(B_1 \mid A) \times P(B_2 \mid A) \times P(B_3 \mid A) ... \times P(B_n \mid A)) \times P(A)}{P(B_1) \times P(B_2) \times P(B_3) ... \times P(B_n)} \tag{2.2}$$

2.3.3 Naive Bayes Classifier

Xét bài toán classification với C classes 1,2,...,C. Giả sử có một điểm dữ liệu $x \in \mathbb{R}^d$. Hãy tính xác suất để điểm dữ liệu này rơi vào class c. Nói cách khác, hãy tính:

$$p(y = c \mid x) \tag{2.3}$$

hoặc viết gọn là $p(c \mid x)$ Tức tính xác suất để đầu ra là class c biết rằng đầu vào là vector x. Biểu thức này, nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi class. Từ đó có thể giúp xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:

$$c = \arg \max_{c \in \{1,..,C\}} p(c \mid x) \tag{2.4}$$

Biểu thức (2) thường khó được tính trực tiếp. Thay vào đó, quy tắc Bayes thường được sử dụng:

$$c = \arg \max_c p(c \mid x) \tag{2.5}$$

$$= \arg \max_c \frac{p(x \mid c)p(c)}{p(x)} \tag{2.6}$$

$$= \arg \max_c p(x \mid c)p(c) \tag{2.7}$$

Từ (2.5) sang (2.6) là vì quy tắc Bayes. Từ (2.6) sang (2.7) là vì mẫu số p(x) không phụ thuộc vào c. Tiếp tục xét biểu thức (2.7), p(c) có thể được hiểu là xác suất để một điểm rơi vào class c. Giá trị này có thể được tính bằng MLE, tức tỉ lệ số điểm dữ liệu trong tập training rơi vào class này chia cho tổng số lượng dữ liệu trong tập traing; hoặc cũng có thể được đánh giá bằng MAP estimation. Trường hợp thứ nhất thường

được sử dụng nhiều hơn.

Thành phần còn lại p(x|c), tức phân phối của các điểm dữ liệu trong class c, thường rất khó tính toán vì x là một biến ngẫu nhiên nhiều chiều, cần rất rất nhiều dữ liệu training để có thể xây dựng được phân phối đó. Để giúp cho việc tính toán được đơn giản, người ta thường giả sử một cách đơn giản nhất rằng các thành phần của biến ngẫu nhiên x là độc lập với nhau, nếu biết c (given c..) Tức là:

$$p(x \mid c) = p(x_1, x_2, ..., x_d \mid c) = \prod_{i=1}^{d} p(x_i \mid c) \tag{2.8}$$

Khi một dữ liệu mới được thêm vào, Naive Bayes sẽ xác định lớp của điểm dữ liệu x bởi:

$$c = \arg \max_{c \in \{1,...,C\}} p(c) \prod_{i=1}^{d} p(x_i \mid c) \tag{2.9}$$

Khi d lớn và các xác suất nhỏ, biểu thức ở vế phải của 2.9 sẽ là một số rất nhỏ, khi tính toán có thể gặp sai số. Để giải quyết việc này, 2.9 thường được viết lại dưới dạng tương đương bằng cách lấy log của vế phải:

$$c = \arg \max_{c \in \{1,...,C\}} = \log(p(c) + \sum_{i=1}^{d} \log(p(x_i \mid c))) \tag{2.10}$$

Việc này không ảnh hưởng tới kết quả vì log là một hàm đồng biến trên tập các số dương.

Mặc dù giả thiết mà Naive Bayes Classifiers sử dụng là quá phi thực tế, chúng vẫn hoạt động khá hiệu quả trong nhiều bài toán thực tế, đặc biệt là trong các bài toán phân loại văn bản, ví dụ như lọc tin nhắn rác hay lọc email spam. Trong phần sau của bài viết, chúng ta cùng xây dựng một bộ lọc email spam tiếng Anh đơn giản.

Cả việc training và test của NBC là cực kỳ nhanh khi so với các phương pháp classification phức tạp khác. Việc giả sử các thành phần trong dữ liệu là độc lập với nhau, nếu biết class, khiến cho việc tính toán mỗi phân phối p(xi|c) trở nên cực kỳ nhanh.

Mỗi giá trị p(c),c=1,2,...,C có thể được xác định như là tần suất xuất hiện của class c trong training data.

Việc tính toán p(xi|c) phụ thuộc vào loại dữ liệu. Có ba loại được sử dụng phổ biến

là: Gaussian Naive Bayes, Multinomial Naive Bayes, và Bernoulli Naive.

2.3.4 Một số phân phối thường dùng

2.3.4.1 Gaussian Naive Bayes

Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục. Với mỗi chiều dữ liệu i và một class c, xi tuân theo một phân phối chuẩn có kỳ vọng μ_{ci} và phương sai σ_{ci}^2:

$$p(x_i \mid c) = p(x_i \mid \mu_{ci}, \sigma_{ci}^2) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp(\frac{(x_i - \mu_{ci})^2}{2\sigma_{ci}^2}) \tag{2.11}$$

Trong đó, bộ tham số $\theta = \mu_{ci}, \sigma_{ci}^2$ được xác định bằng Maximum Likehood

$$(\mu_{ci}, \sigma_{ci}^2) = \arg\max_{\mu_{ci}, \sigma_{ci}^2} \prod_{i=1}^{n} p(x_i^(n) \mid \mu_{ci}, \sigma_{ci}^2) \tag{2.12}$$

2.3.4.2 Multinominal Naive Bayes

Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó, p(xi|c) tỉ lệ với tần suất từ thứ i (hay feature thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của class c. Giá trị này có thể được tính bằng cách:

$$\lambda_{ci} = p(x_i \mid c) = \frac{N_c i}{N_c} \tag{2.13}$$

Trong đó:

- $N_c i$ là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c
- N_c là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Có thể suy ra rằng N_c=$\sum_{i=1}^{d} = N_c i$, từ đó =$\sum_{i=1}^{d} \lambda_{ci} = 1$.

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức (10) sẽ bằng 0, điều này dẫn đến vế phải của (7) bằng 0 bất kể các

giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác (xem thêm ví dụ ở mục sau). Để giải quyết việc này, một kỹ thuật được gọi là Laplace smoothing được áp dụng:

$$\hat{\lambda_{ci}} = \frac{N_{ci} + \alpha}{N_c + d_\alpha} \tag{2.14}$$

Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với d_α để đảm bảo tổng xác suất $\sum_{i=1}^{d} \hat{\lambda_{ci}} = 1$. Như vậy, mỗi class c sẽ được mô tả bởi bộ các số dương có tổng bằng 1: $\hat{\lambda_c} = \left\{\hat{\lambda_{c1}}+, ..., +\hat{\lambda_{cd}}\right\}$.

2.3.4.3 Bernoulli Naive

Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary - bằng 0 hoặc 1. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không.

Khi đó, $p(x_i \mid c$ được tính bằng:

$$p(x_i \mid c) = p(i \mid c)x_i + 1 - p(i \mid c)(1 - x_i) \tag{2.15}$$

với p(i|c) có thể được hiểu là xác suất từ thứ i xuất hiện trong các văn bản của class c.

2.3.5 Ưu điểm thuật toán Naive Bayes

2.3.6 Nhược điểm điểm thuật toán Naive Bayes

2.3.7 Naive Bayes với bài toán lọc rác tin tức

2.4 Thuật toán J48

2.4.1 Giới thiệu thuật toán J48

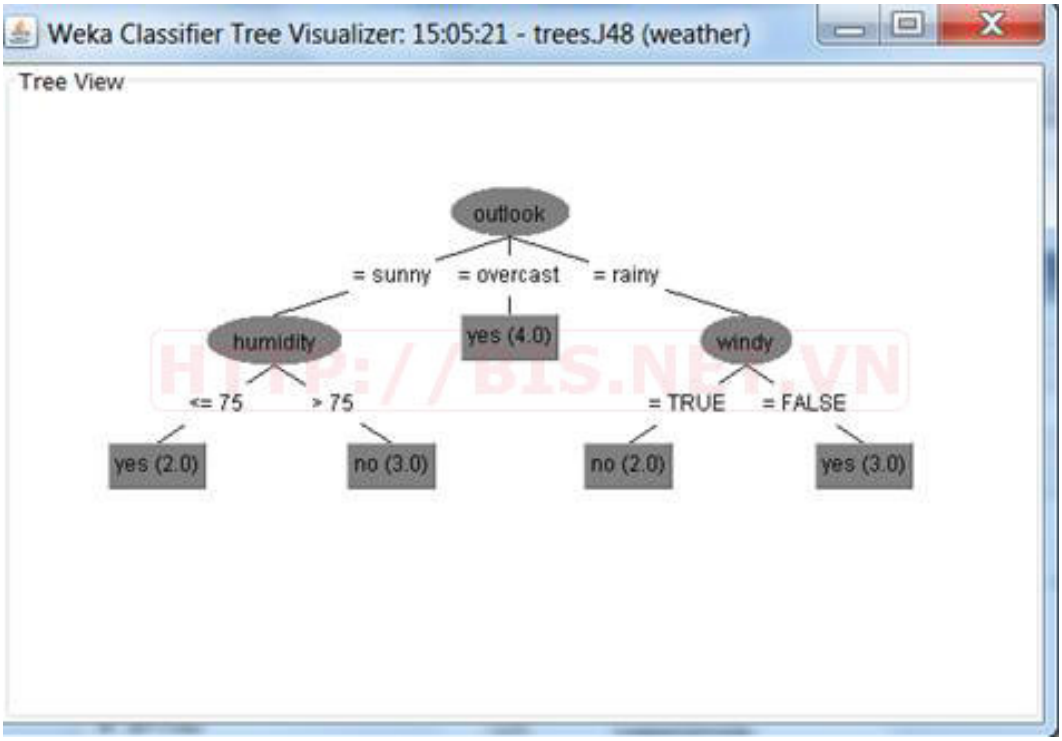
Thuật toán J48(C4.5) là một thuật toán sử dụng cây quyết định(decision tree) cho việc phân lớp. Thuật toán sẽ tạo ra môn cây nhị phân. Bằng cách sử dụng cây quyết định, cách tiếp cận này cũng thường đước sử dụng trong bài toán phân lớp. Cây quyết định sẽ được xây dựng để mô hình hóa quá trình phân lớp.

2.4.2 Lý thuyết cây quyết định

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật (series of rules). Các thuộc tính của đối

tượng (ngoại trừ thuộc tính phân lớp – Category attribute) có thể thuộc các kiểu dữ liệu khác nhau (Binary, Nominal, ordinal, quantitative values) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal. Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các đối tượng chưa biết (unseen data)

2.4.3 Ví dụ về cây quyết định J48



Hình 2.1: Ví dụ về cây quyết định

2.5 Tiền xử lý

2.5.1 Khái niệm

Tiền xử lý dữ liệu là quá trình lọc và chuẩn bị dữ liệu cho quá trình phân lớp văn bản. Thường các văn bản đầu vào đa số chứa nhiều thông tin gây nhiễu và các thông tin không có giá trị như HTML tags, địa chỉ trang web, quảng cáo... Vì vậy, ở mức độ từ, nhiều từ không có ý nghĩa cho ý chung của câu.

2.5.2 Vì sao phải tiền xử lý

Giữ những từ có khả năng gây nhiễu sẽ làm tăng số chiều làm quá trình phân lớp trở nên khó hơn vì mỗi từ được xem như một chiều cần phải xem xét. Đây là lý do chúng ta cần tiền xử lý dữ liệu: để giảm nhiễu trong văn bản sẽ giúp tăng hiệu năng của bộ phân lớp(classfier) và làm giảm thời gian phân lớp.

2.5.3 Một số cách tiếp cận

Quá trình tiền xử lý dữ liệu thường sẽ gồm các bước:

- Loại bỏ một số thông tin không cần thiết (Ví dụ như HTML tags,...)
- Loại bỏ một số dấu câu không cần thiết (khoảng trắng hoặc dấu câu)
- Loại bỏ stop-word: Một số từ thường xuất hiện trong văn bản nhưng không mang nhiều ý nghĩa. Thường sẽ được lưu thành một stop-word list. Ở tiếng Việt, một số stop word là(à, á, á à, ạ,....)
- Ghép từ một số từ ghép được ghép lại để biểu diễn cho một chiều trong phân lớp (Ví dụ máy bay sẽ ghép thành máybay hoặc máy_bay)

2.5.4 Tiền xử lý trong bài toán lọc rác tin tức

Ở bài toán lọc rác tin tức, dữ liệu đầu vào là tin tức nên đa số thông tin gây nhiễu như quảng cáo, HTML tags đã được giảm thiểu, chúng ta chỉ cần xử lý văn bản và lọc một số thông tin gây nhiễu. Ví dụ như:

- Link(URL) nếu có.
- Ghép từ để tăng độ chính xác (ví dụ máy bay sẽ được ghép thành máy_bay)
- Loại bỏ stop-word

2.6 Các nghiên cứu liên quan

Twitter đã thu hút sự chú ý của nhiều nhà nghiên cứu trong thời gian gần đây. Nhờ vào sự phổ biến rộng rãi trên nhiều quốc gia, cùng với vai trò là một mạng xã

hội cho phép người dùng chia sẻ thông tin một cách nhanh chóng và ngắn gọn. Jagan Sankaranarayanan và cộng sự [4] đã xây dựng một hệ thống xử lý tin tức trên Twitter, với tên là Twitter Stand. Hệ thống biểu diễn dữ liệu dưới dạng tf-idf, và sử dụng thuật toán phân lớp Naive Bayes để lọc bỏ các tin rác. Sau đó dùng một thuật toán online clustering để gom các tin thành từng câu chuyện. Phuvipadawat và cộng sự [5] đưa ra phương pháp phát hiện tin nóng trên Twitter dựa vào tiếp cận gom cụm. Các tweet được thu thập thông qua Twitter Search API với một số từ khóa (VD: "breaking news", "#breakingnews"). Độ tương đồng giữa các bài viết được tính theo công thức dựa trên tf-idf, với các danh từ riêng, các hashtag và tên người dùng được tăng trọng số. Tác giả nhấn mạnh việc tăng trọng số như vậy giúp tăng độ chính xác của thuật toán, bởi độ dài tweet giới hạn.

2.7 Giới thiệu một số độ đo khoảng cách/sự tương đồng

Một thành phần tất yếu của tiếp cận gom cụm là tiêu chí, độ đo được sử dụng để định lượng sự tương đồng giữa các đối tượng. Dưới đây là một số độ đo phổ biến.

Euclidean Distance

Euclidean Distance là độ đo khoảng cách tiêu chuẩn trong các vấn đề liên quan đến hình học, và cũng thường được sử dụng trong các bài toán gom cụm. Công thức tính Euclidean distance [6]:

$$D_{Euclidean}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.16}$$

với n là số chiều của vector biểu diễn document, x_i và y_i lần lượt là giá trị tọa độ thứ i của document x và y

Cosine Similarity

Cosine Similarity phản ánh góc chênh lệch giữa 2 vector, mà không cân nhắc đến độ lớn của vector. Độ đo này được áp dụng rộng rãi đối với dữ liệu văn bản. Cách tính

[6]:

$$Similarity_{Cosine}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \cdot ||\vec{y}||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.17}$$

với x_i và y_i lần lượt là giá trị tọa độ thứ i của document x và y

2.8 Các phương pháp tiếp cận phổ biến

Gom cụm, gom nhóm là quá trình nhóm các đối tượng thành những nhóm/cụm/lớp, qua đó phát hiện được cấu trúc, ý nghĩa tiềm ẩn của dữ liệu. Các đối tượng trong cùng một nhóm có nhiều tính chất chung và có những tính chất khác với các đối tượng ở nhóm khác. Đây là một trong những tác vụ chính của ngành khai thác dữ liệu, và được dùng trong nhiều lĩnh vực như: máy học, nhận diện mẫu, phân tích ảnh,... Bài toán gom cụm bao gồm nhiều phương pháp khác nhau như: phân hoạch, phân cấp, dựa trên mật độ, dựa trên mô hình,... Trong khóa luận này chỉ sử dụng đến *phương pháp phân hoạch*.

2.8.1 Thuật toán gom cụm k-láng giềng gần (k-Nearest Neighbor)

2.8.1.1 Ý tưởng

Hướng tiếp cận truyền thống của bài toán FSD là sử dụng phương pháp gom cụm k-láng giềng gần nhất, với k = 1 [7]. Theo trực quan ta có thể thấy nếu một bài viết có nội dung tương tự nhiều với những bài có sẵn, thì khả năng nó là một first story rất thấp. Ngược lại, khi nội dung của một bài viết mới lạ, khác hẳn những bài viết trước đây, thì có thể xem đó là một first story.

Thuật toán gom cụm này hoạt động như sau: Mỗi document mới sẽ được so sánh với tất cả các document hiện có trong hệ thống và tìm ra một document tương đồng nhất. Nếu độ tương đồng của chúng cao hơn một ngưỡng cho trước, thì document mới này sẽ được gán vào cụm của document tương đồng nhất, ngược lại tạo cụm mới và xem document mới như một first story.

2.8.1.2 Minh họa thuật toán

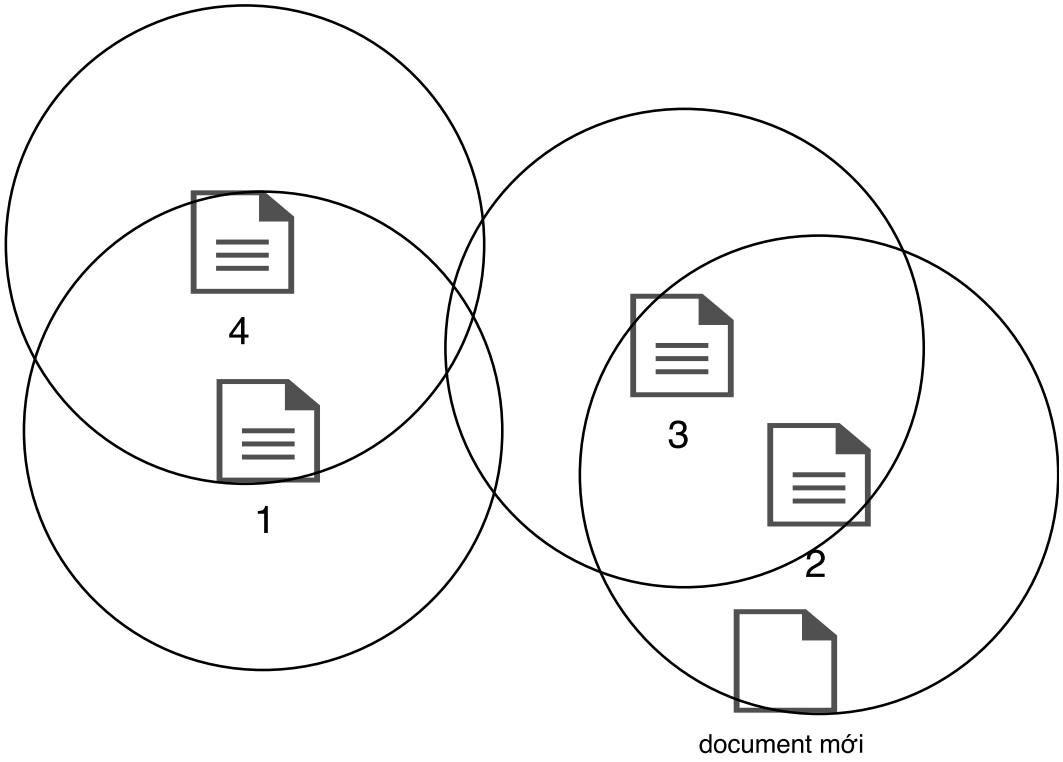
Giả sử ta có dữ liệu như hình vẽ. Đường tròn thể hiện ngưỡng khoảng cách để một document được xem là first story hay không, chọn ngưỡng giá trị 0.8. Ta có sẵn 4 document và 1 document mới vào hệ thống như sau:

$$\text{doc1} = (3,4,1,2)$$
$$\text{doc2} = (1,1,5,6)$$
$$\text{doc3} = (1,2,4,4)$$
$$\text{doc4} = (3,5,1,1)$$
$$\text{newDoc} = (2,1,4,5)$$

	doc1	doc2	doc3	doc4	**newDoc**
doc1	1	0.55202	0.6903	0.973729	0.646058
doc2		1	0.973479	0.398962	0.984525
doc3			1	0.575396	0.969572
doc4				1	0.491473
newDoc					1

Bảng 2.1: Độ tương đồng cosine giữa các document

Ta có thể thấy document tương tự nhất của newDoc là doc2 (0.984525), lớn hơn ngưỡng cho trước (0.8), do đó newDoc được cho vào chung cụm với doc2 và không phải là một first story.



Hình 2.2: Cách xử lý một document mới trong thuật toán Nearest Neighbor Search

2.8.1.3 Mã giả

Algorithm 1 Phát hiện tin nóng dựa trên k-Nearest neighbor

Input: luồng bài viết từ Twitter, giá trị ngưỡng MergeThreshold
Output: các cụm bài viết

foreach document d trong toàn bộ dữ liệu **do**
  $S(d) \leftarrow \emptyset$
  foreach term t trong document d **do**
    foreach document d' trong $index[t]$ **do**
      cập nhật độ tương đồng $similarity(d, d')$
      $S(d) \leftarrow S(d) \cup d'$
    end for
    $index[t] \leftarrow index[t] \cup d$ (thêm document d vào index của term t)
  end for
  similarity score của d: $similarity_{max}(d) \leftarrow 0$
  foreach document d' trong S(d) **do**
    if $similarity(d, d') > similarity_{max}(d)$ **then**
      $similarity_{max}(d) \leftarrow similarity(d, d')$
    end if
  end for
  if $similarity_{max}(d) < MergeTheshold$ **then**
    d là một first story
  else
  end if
end for

Chú thích thuật toán:

- $MergeThreshold$: ngưỡng để xét một document có phải là first story hay không
- $S(d)$: tập document có ít nhất 1 term chung với d
- $similarity(d, d')$: độ tương đồng giữa document d và d', có thể sử dụng các độ đo ở mục 2.7
- $index(t)$: danh sách cách document có chứa term t
- $dis_{min}(d)$: khoảng cách giữa document d với document gần nhất

2.8.1.4 Ưu điểm, hạn chế

Ưu điểm:

- Đơn giản và dễ cài đặt.

- Có thể chọn nhiều độ đo khoảng cách khác nhau.
- Thích nghi tốt với nhiều loại dữ liệu.

Nhược điểm:

- Chi phí tính toán cao do phải lưu trữ và tính toán trên toàn bộ dữ liệu, không thể áp dụng cho luồng dữ liệu không liên tục.
- Độ chính xác giảm khi số chiều của dữ liệu tăng cao.

2.8.2 Thuật toán gom cụm có Boost trọng số cho Named Entity

2.8.2.1 Ý tưởng

Cách tiếp cận được đề xuất bởi Phuvipadawat là gom cụm các bài viết theo độ tương tự nội dung, với điểm nhấn là tăng giá trị tương đồng của các bài viết nếu chúng cùng đề cập đến một thực thể có tên (Named entity) nào đó [5]. Mỗi cụm thu được sẽ ứng với một sự kiện phát hiện được, với document cũ nhất làm đại diện cho cụm đó.

Theo Phuvipadawat, một đặc điểm của các bài viết trên Twitter là thường có nội dung khá ngắn (tối đa 140 kí tự, và ngắn hơn nữa sau khi tiền xử lý loại bỏ stop words). Việc sử dụng phương pháp TF-IDF truyền thống để tính độ tương đồng có thể không đạt được kết quả tốt do không có nhiều term để tìm ra sự tương đồng giữa các document. Vì thế tác giả đã đưa ra phương pháp tăng trọng số cho các danh từ riêng (Named Entity), qua đó tăng độ tương đồng giữa những bài viết cùng thảo luận về một sự vật, sự việc cụ thể nào đó.

Khi một document d mới được đưa vào hệ thống, ta so sánh độ tương đồng giữa d với tất cả cụm hiện có thông qua document đầu tiên trong cụm. Ta chọn ra cụm có độ tương đồng với d cao nhất, gán d vào cụm đó nếu độ tương đồng vượt ngưỡng định trước, ngược lại ta tạo cụm mới với d là document đầu tiên của cụm.

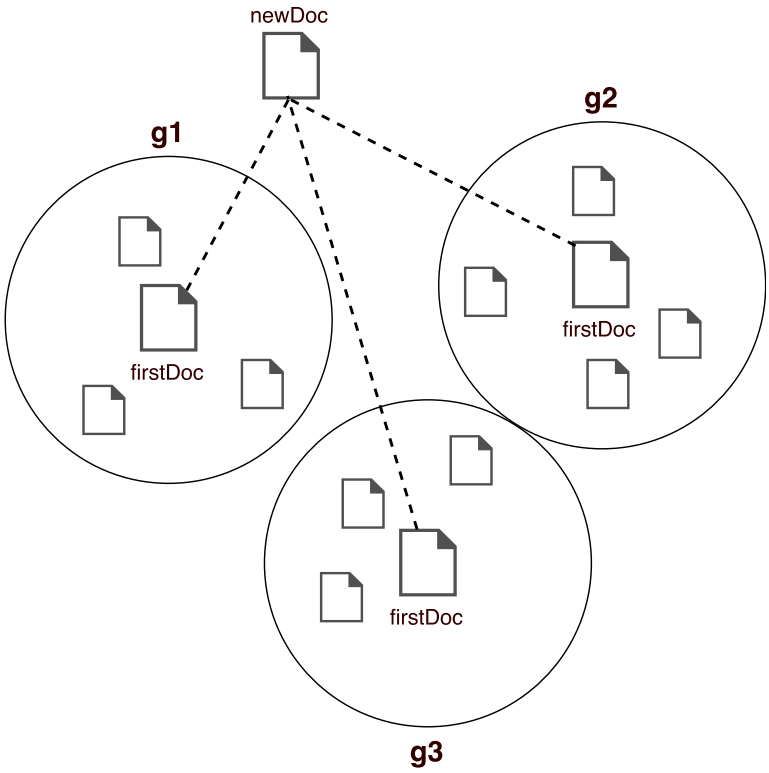
2.8.2.2 Minh họa thuật toán

Giả sử ta có một số document đã được gom nhóm sẵn thành 3 cụm như hình, ngưỡng MergeThreshold = 0.7, mỗi cụm có một firstDoc làm đại diện. Đường tròn thể

hiện ngưỡng khoảng cách để một document được xem là first story hay không.

Khi một document mới (newDoc) vào hệ thống:

- Bước 1: So sánh newDoc với từng firstDoc của từng cụm:
  $similarity(newDoc, g1.firstDoc) = 0.6$
  $similarity(newDoc, g2.firstDoc) = 0.5$
  $similarity(newDoc, g3.firstDoc) = 0.35$
- Bước 2: Tìm cụm có firstDoc tương tự với newDoc nhất: Chọn được cụm $g1$
- Bước 3: $similarity(newDoc, g1.firstDoc) = 0.6$ nhỏ hơn MergeThreshold: Tạo cụm g4 với $g4.firstDoc = newDoc$.



Hình 2.3: Cách xử lý một document mới theo thuật toán Boost Named Entity

2.8.2.3 Mã giả

Algorithm 2 Phát hiện tin nóng sử dụng gom cụm theo nội dung, boost Named Entity

Input: luồng bài viết từ Twitter, giá trị ngưỡng MergeThreshold
Output: các cụm bài viết ứng với mỗi story phát hiện được, có thứ tự xếp hạng giữa các cụm

foreach document d trong toàn bộ dữ liệu **do**
  $MaxScore_d \leftarrow 0$
  $IDMaxScore_d \leftarrow \emptyset$
  foreach cụm g trong tập hợp cụm G **do**
    tính $Score_d[g] \leftarrow similarity(d, g.firstDoc)$
    if $MaxScore_d < Score_d[g]$ **then**
      $MaxScore_d \leftarrow Score_d[g]$
      $IDMaxScore_d \leftarrow g.groupID$
    end if
  end for
  if $MaxScore_d >$ MergeThreshold **then**
    gán d vào cụm $IDMaxScore_d$
  else
    tạo cụm g_{new}
    $g_{new}.firstDoc \leftarrow d$
  end if
end for

Chú thích thuật toán:

- $MaxScore_d$: chứa độ tương đồng lớn nhất giữa document d và các cụm đã có
- $IDMaxScore_d$: ID của cụm tương đồng nhất với document d
- $Score_d[g]$: độ tương đồng giữa document d với cụm g
- $MergeThreshold$: ngưỡng để xét một document có thuộc về cụm/có là first story hay không

Độ tương đồng giữa 2 document được tính bằng các công thức sau:

$$similarity(d_1, d_2) = \sum_{t \in d_1,\ t \in g.topTerms} [tf(t, d_2) \times idf(t) \times boost(t)] \tag{2.18}$$

$$tf(t, d) = \frac{count(t \in d)}{size(d)} \tag{2.19}$$

$$idf(t) = 1 + \log \frac{N}{count(m\ has\ t)} \tag{2.20}$$

Các kí hiệu:

- $similarity(d_1, d_2)$: độ tương đồng giữa document d_1 và document d_2
- $boost(t)$: giá trị boost cho term t, nếu t là một danh từ riêng (Named Entity) thì boost(t) > 1, ngược lại boost(t) $= 1$
- $tf(t, d)$: tần suất xuất hiện (theo phần trăm) của term t trong document d
- $count(t \in d)$: tần suất term t xuất hiện trong trong document d
- $size(d)$: số lượng term của document d
- $idf(t)$: giá trị inverse document frequency của term t
- N: tổng số document trong hệ thống
- $count(m\ has\ t)$: số document có chứa term t

2.8.2.4 Ưu điểm, hạn chế

Ưu điểm:

- Dễ gom nhóm được các sự kiện bàn về cùng thực thể nào đó.

Nhược điểm:

- Chất lượng kết quả gom cụm phụ thuộc một phần vào việc phát hiện được thực thể có tên.

2.8.3 Thuật toán Locality Sensitive Hashing

2.8.3.1 Ý tưởng

Thuật toán tìm kiếm láng giềng gần nhất tốn rất nhiều chi phí khi lượng dữ càng lớn. Thay vào đó, ta có thể giải bài toán tìm *xấp xỉ* láng giềng gần nhất. Một thuật toán để giải quyết bài toán này là *Locality Sensitive Hashing (LSH)* được đề xuất bởi Piotr Indyk và cộng sự [8].

LSH hoạt động bằng cách chia không gian biểu diễn dữ liệu thành nhiều vùng riêng biệt bằng một tập các siêu phẳng ngẫu nhiên. Ta có thể xem mỗi cách chia không

gian này ứng với một hash table, và số bit của hash code bằng với số lượng siêu phẳng đã dùng để chia không gian. Mỗi khi một document mới vào hệ thống, ta tính hash code của nó. Khi cần tìm láng giềng cho một document, ta chỉ cần so sánh nó với các document thuộc chung vùng không gian (ứng với một bucket trong hash table), nhờ đó giảm đáng kể chi phí tính toán.

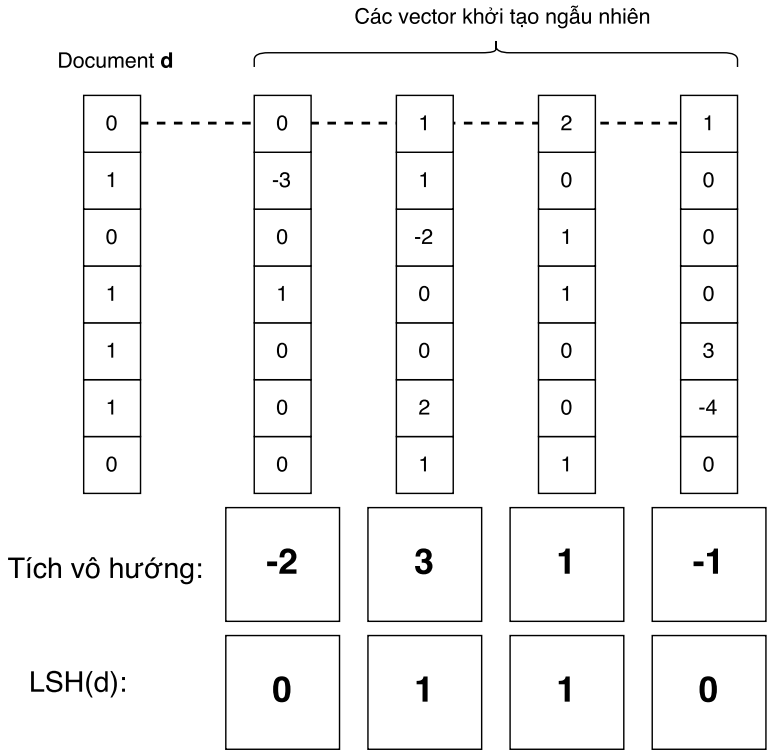
Vì các siêu phẳng được chọn một cách ngẫu nhiên, nên đôi khi các document dù gần nhau vẫn có thể bị phân vào vùng khác nhau. Do đó ta thường dùng cùng lúc nhiều hash table để tăng thêm khả năng tìm được láng giềng gần nhất.

2.8.3.2 Minh họa cách tính hash code cho LSH

Giả sử ta có không gian biểu diễn document gồm 7 từ/term khác nhau (8 chiều). Xét một document d biểu diễn dưới dạng vector $d = (0, 1, 0, 1, 1, 1, 0)$, và ta chọn sử dụng hash code có 4 bit (dùng 4 siêu phẳng để chia không gian dữ liệu).

Mỗi hash table được khởi tạo bằng cách: tạo 4 vector ngẫu nhiên ứng với 4 siêu phẳng, số chiều bằng 7, ứng với số chiều của vector d.

Cách tính hash code của document d trong một hash table như sau: lần lượt tính tích vô hướng của vector d với từng vector của các siêu phẳng, nếu tích dương thì bit đó có giá trị 1, nếu tích âm thì có giá trị 0.



Hình 2.4: Cách tính hash code cho một document trong một hash table

2.8.3.3 Mã giả

Algorithm 3 Phát hiện tin nóng sử dụng Locality Sensitive Hashing

Input: luồng bài viết từ Twitter, giá trị ngưỡng t
Output: những cụm bài viết

```
1: Khởi tạo L hash tables
2: foreach document d mới do
3:    foreach hash table l trong L do
4:       Tính hash code cho document d: LSH_l(d)
5:       Thêm tất cả các documents d' có LSH_l(d') = LSH_l(d) vào tập S
6:    end for
7:    dis_min(d) = 1
8:    foreach document d' trong S do
9:       if distance(d, d') < dis_min(d) then
10:         dis_min(d) = distance(d, d')
11:      end if
12:   end for
13:   if dis_min(d) >= t then
14:      tạo cụm mới chứa d
15:   else
16:      thêm d vào cụm chứa hàng xóm gần nhất của d
17:   end if
18: end for
```

Chú thích thuật toán:

- t: ngưỡng để xét một document có phải là first story hay không
- $LSH_l(d)$: hash code của document d trong hash table thứ l
- $dis_{min}(d)$: khoảng cách giữa document d với document gần nhất

Ưu điểm:

- Chi phí tính toán thấp do không cần so sánh toàn bộ các document với nhau.
- Độ chính xác tương đối tốt

Nhược điểm:

- Cần tìm chọn các thông số như: số lượng hash table, số lượng bit trong hash code,... tốt để cho kết quả chính xác.

- Có thể không tìm được document thật sự tương đồng nhất, trong trường hợp các điểm không được chia vào cùng vùng trong không gian do cách thiết lập của các siêu phẳng trong hash tables.

2.8.3.4 LSH cải tiến

Dựa vào dữ liệu thực nghiệm của mình, Sasa Petrovic và cộng sự [9] đưa ra nhận định rằng việc áp dụng LSH thuần túy vào bài toán FSD cho ra kết quả chưa tốt. Trong trường hợp điểm dữ liệu cách xa tất cả điểm còn lại, LSH không tìm được láng giềng gần nhất, do không có điểm nào khác rơi vào cùng bucket trong các hash table, dẫn đến không có điểm dữ liệu khác để so sánh.

Để giải quyết vấn đề này, Petrovic đưa ra phương án: mỗi khi tìm ra document d có $dis_{min}(d)$ lớn hơn ngưỡng, ta áp dụng thuật toán tìm láng giềng gần nhất giữa document d và khoảng 1000 tới 3000 document có thời gian gần đây nhất trong hệ thống, cập nhật giá trị $dis_{min}(d)$ và xét xem nó vẫn vượt ngưỡng hay không, nếu có thì ta kết luận d là một first story, ngược lại ta cho d vào cụm của láng giềng gần nó nhất.

Algorithm 4 Phát hiện tin nóng sử dụng Locality Sensitive Hashing kết hợp với Nearest Neighbor Search

Input: luồng bài viết từ Twitter, giá trị ngưỡng t
Output: những cum bài viết

Khởi tạo L hash tables
foreach document d mới **do**
  foreach hash table l trong L **do**
    Tính hash code cho document d: $LSH_l(d)$
    Thêm tất cả các documents d' có $LSH_l(d') = LSH_l(d)$ vào tập S
  end for
  $dis_{min}(d) = 1$
  foreach document d' trong S **do**
    if $distance(d, d') < dis_{min}(d)$ **then**
      $dis_{min}(d) = distance(d, d')$
    end if
  end for
  if $dis_{min}(d) >= t$ **then**
    Tính khoảng cách giữa d với một lượng (1000-3000) documents mới nhất trong bộ dữ liệu, cập nhật $dis_{min}(d)$ nếu có thay đổi.
  end if
  if $dis_{min}(d) >= t$ **then**
    tạo cụm mới chứa d
  else
    thêm d vào cụm chứa hàng xóm gần nhất của d
  end if
end for

2.9 Giới thiệu một số độ đo đánh giá phân lớp

Đánh giá chất lượng của kết quả gom cụm là nhiệm vụ khó khăn và phức tạp, do bản chất không hoàn toàn rõ ràng về định nghĩa của một "cụm". Theo tác giả Pang-Ning Tan và cộng sự [10], có 3 loại phương pháp để đánh giá chất lượng cụm:

- Đánh giá ngoại, có giám sát (External evaluation): Các độ đo có sử dụng thông tin bên ngoài như nhãn lớp của dữ liệu để so sánh với kết quả gom cụm. VD: Purity, Rand measure,...
- Đánh giá nội, không giám sát (Internal evaluation): Chỉ dựa vào các thông tin, thuộc tính có sẵn trong cụm để đánh giá. VD: Silhouette index, Dunn index, Davies–Bouldin index...

- Đánh giá tương đối (Relative evaluation): So sánh kết quả giữa các phương pháp gom cụm hoặc giữa các cụm, có thể dùng độ đo ngoại lẫn nội.

2.9.1 Khoảng cách cục bộ và toàn cục (Intra-cluster và Inter-cluster distance)

Hai độ đo này thuộc loại đánh giá nội, chỉ dựa vào chính dữ liệu được gom cụm để tính giá trị của chúng. Khoảng cách cục bộ của một cụm thể hiện mức độ các điểm dữ liệu trong một cụm tương đồng với nhau thế nào. Dưới đây là 3 cách để tính khoảng cách cục bộ:

- Dùng khoảng cách giữa 2 điểm xa nhau nhất trong cụm:
  $intraclusterDistance(C_i) = \ max_{x,y \in C_i} distance(x, y)$
- Dùng trung bình khoảng cách giữa tất cả cặp điểm trong cụm:
  $intraclusterDistance(C_i) = \frac{1}{size(C_i) \ * \ [size(C_i)-1]} \sum_{x,y \in C_i, x \neq y} distance(x, y)$
- Dùng trung bình khoảng cách giữa từng điểm với tâm cụm:
  $intraclusterDistance(C_i) = \frac{\sum_{x \in C_i} distance(x,\mu)}{size(C_i)}$, $\mu = \frac{\sum_{x \in C_i} x}{size(C_i)}$

Khoảng cách toàn cục là giá trị cho thấy mức độ rời rạc giữa tất cả các cụm với nhau, được tính bằng trung bình khoảng cách giữa các cụm:

$$interclusterDistance = \frac{1}{size(C) * [size(C) - 1]} \sum_{C_i, C_j \in C} distance(C_i, C_j) \qquad (2.21)$$

Trong đó, khoảng cách giữa 2 cụm $distance(C_i, C_j)$ cũng có thể được xác định bằng nhiều cách như:

- Dùng khoảng cách giữa 2 điểm xa nhau nhất trong 2 cụm:
  $distance(C_i, C_j) = \ max_{x \in C_i, y \in C_j} distance(x, y)$
- Dùng trung bình khoảng cách giữa tất cả cặp điểm trong 2 cụm:
  $distance(C_i, C_j) = \frac{1}{size(C_i)+size(C_j)} \sum_{x \in C_i, y \in C_j} distance(x, y)$

- Dùng khoảng cách giữa 2 tâm cụm:
  $distance(C_i, C_j) = distance(\mu_{C_i}, \mu_{C_j})$, với $\mu_{C_i} = \frac{\sum_{x \in C_i} x}{size(C_i)}$

Chú thích kí hiệu:

- C_i: cụm dữ liệu thứ i
- x, y: 2 điểm dữ liệu bất kì
- $distance(x, y)$: khoảng cách giữa 2 điểm dữ liệu, dùng các khoảng cách ở mục 2.7
- $size(C_i)$: số lượng điểm dữ liệu được gom vào cụm i

2.9.2 Độ đo Dunn index

Dunn index được đề xuất bởi J. C. Dunn [11], độ đo này thể hiện mức độ gắn kết của từng cụm lẫn độ rời rạc giữa các cụm. Giá trị Dunn index được tính theo công thức sau:

$$DI = \frac{min_{1 \leq i \leq j \leq n} distance(C_i, C_j)}{max_{1 \leq k \leq n} intraclusterDistance(k)} \tag{2.22}$$

Với n là tổng số cụm thu được, và intraclusterDistance được tính theo một trong những cách ở mục 2.9.1.

Giá trị Dunn index càng cao thì các cụm càng tách biệt nhau và mỗi cụm thì có những điểm dữ liệu gom sát với nhau. Tuy vậy, mẫu số trong công thức (2.22) lấy khoảng cách cục bộ của cụm rời rạc nhất chứ không lấy trung bình tất cả cụm, nên giá trị Dunn index là trường hợp xấu nhất chứ không phải trường hợp trung bình.

2.9.3 Hệ số Silhouette (Silhouette coefficient)

Silhouette là một độ đo xem xét các đối tượng trong dữ liệu có được gán vào cụm hợp lý hay không, thông qua việc so sánh cả độ gắn kết của từng cụm và độ rời rạc giữa các cụm. Với mỗi điểm dữ liệu, giá trị silhouette coefficient nằm trong khoảng từ -1 đến 1, giá trị này cao khi điểm dữ liệu tương đồng với cụm của nó và ít tương đồng với cụm khác, và ngược lại.

Cách tính hệ số Silhouette cho một điểm dữ liệu **i** như sau:

1. Tính khoảng cách trung bình giữa i với các điểm trong cùng cụm với i. Gọi giá trị này là a_i.

2. Với mỗi cụm g không chứa i, tính khoảng cách của i đến cụm g bằng trung bình khoảng cách giữa i với từng điểm trong cụm g, chọn cụm có khoảng cách đến i *nhỏ nhất*. Gọi giá trị này là b_i

3. Tính hệ số Silhouette theo công thức sau:

$$s(i) = \frac{b_i - a_i}{max\{a_i, b_i\}} \tag{2.23}$$

Ở đây a_i đại diện cho độ bất phù hợp của điểm i với cụm của nó, và b_i đại diện cho độ tương đồng giữa điểm i với cụm gần nhất không chứa i. Vì vậy khi a_i nhỏ, b_i lớn, giá trị hệ số silhouette tiến tới 1, nghĩa là điểm dữ liệu i rất phù hợp với cụm của nó và không phù hợp với cụm khác. Trong trường hợp ngược lại, giá trị silhouette tiến tới -1.

Giá trị hệ số Silhouette cho một kết quả phân cụm được tính bằng trung bình hệ số Silhouette của tất cả điểm dữ liệu.

2.9.4 Độ đo Purity

Đây là một độ đo đánh giá ngoại. Dựa trên nhãn lớp của dữ liệu, purity là độ đo về tính thuần khiết/độ đồng nhất của các cụm. Giá trị này tính bằng cách lấy tần số của lớp chiếm số lượng nhiều nhất trong cụm (tính theo phần trăm kích cỡ cụm). Purity càng cao khi cụm chứa càng nhiều điểm dữ liệu thuộc cùng một nhãn lớp.

Giá trị purity của một cụm C_i được tính bằng công thức:

$$purity(C_i) = max_j(\frac{m_{ij}}{m_i}) \tag{2.24}$$

Trong đó:

- m_i: số điểm dữ liệu thuộc cụm i
- m_{ij}: số điểm dữ liệu thuộc lớp j và nằm trong cụm i

Giá trị purity của tất cả cụm được tính như sau:

$$purity = \sum_{i=1}^{K} \frac{m_i}{m}\ purity(C_i) \tag{2.25}$$

Trong đó:

- K: tổng số cụm
- m: tổng số điểm dữ liệu

2.10 Xếp hạng cụm

Một yêu cầu của bài toán phát hiện tin nóng là các chuỗi tin (ứng với các cụm) phải được sắp xếp theo mức độ nóng giảm dần. Hầu hết các phương pháp đã khảo sát trong khóa luận này chỉ tập trung vào việc gom cụm và cải thiện chất lượng cụm, mà không đề cập đến việc sắp xếp cụm theo thứ tự nào đó.

Tác giả Phuvipadawat đưa ra một số công thức để tính điểm lượng giá độ nóng của chuỗi tin, và các chuỗi tin được sắp xếp theo giá trị điểm giảm dần [5].

Điểm này cân nhắc tầm ảnh hưởng của những người viết bài trong cụm, cũng như mức độ lan tỏa của chính các bài viết trong cụm. Giá trị điểm cũng được cập nhật thường xuyên và sẽ giảm dần theo thời gian nếu không có những bài viết mới làm tăng điểm cho nó. So với công thức của Phuvipadawat, công thức 2.12 bổ sung thêm yếu tố lượng favorite của cụm, do đây cũng là một thành phần ghi nhận phản ứng từ người dùng Twitter.

Giá trị điểm xếp hạng của một cụm C_i được cho bằng công thức sau:

$$Score(C_i) = \frac{1}{Z} \sum_{j=1}^{k} \frac{S_i}{log_2(\Delta_j + 2)} \tag{2.26}$$

Với S_i tính bằng:

$$S_i = w_1 \sum_{u_i \in C_i} FollowerCount(u_i) + w_2\ TotalRetweetCount(C_i) + w_3\ TotalFavoriteCount(C_i) \tag{2.27}$$

Trong đó:

- $\frac{1}{Z}$: giá trị để chuẩn hóa điểm cụm, Z là tổng điểm của các cụm khi chưa chuẩn hóa
- k: số lượng document trong cụm

- Δ_j: chênh lệch thời gian giữa thời điểm hiện tại với thời gian bài viết j được đăng
- w_1, w_2: trọng số cho hai vế của biểu thức
- u_i: người dùng thứ i
- $FollowerCount(u_i)$: số người follow người dùng u_i trên Twitter
- $TotalRetweetCount(g_i)$: tổng số retweet của các bài đăng trong cụm c_i

2.11 Kết chương

Chương này đã phát biểu bài toán phát hiện tin nóng và đưa ra 4 thuật toán tiếp cận theo hướng gom cụm. Đồng thời trình bày về các độ đo tương đồng, độ đo đánh giá chất lượng cụm và cách xếp hạng cụm. Chương tiếp theo sẽ trình bày về việc xây dựng hệ thống phát hiện tin nóng.

Chương 3

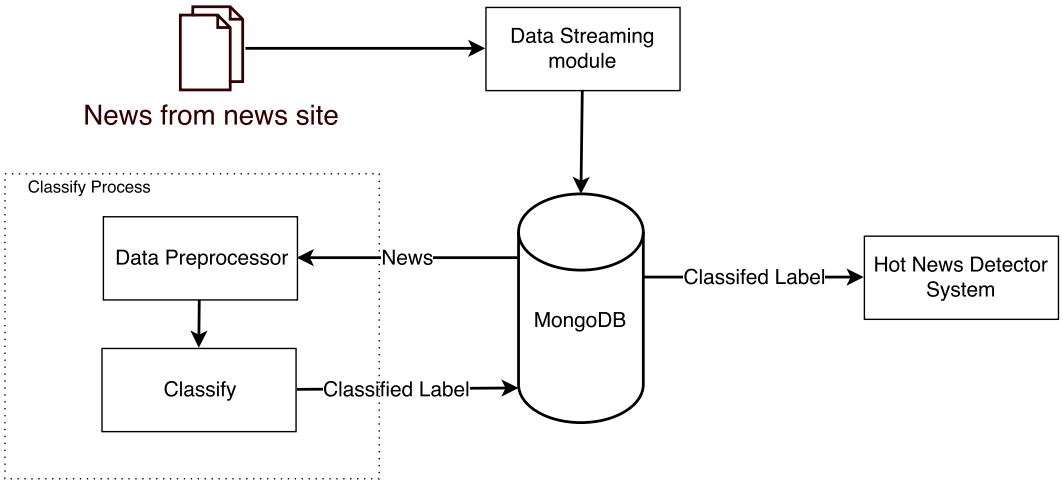
HIỆN THỰC HỆ THỐNG PHÁT HIỆN TIN NÓNG

3.1 Mở đầu

Chương này sẽ trình bày về thiết kế, chi tiết cài đặt, các thư viện và framework được sử dụng để xây dựng hệ thống lọc rác tin tức.

3.2 Mô hình hệ thống

Dưới đây là mô hình của các thành phần chính hệ thống:



Hình 3.1: Các thành phần chính của hệ thống

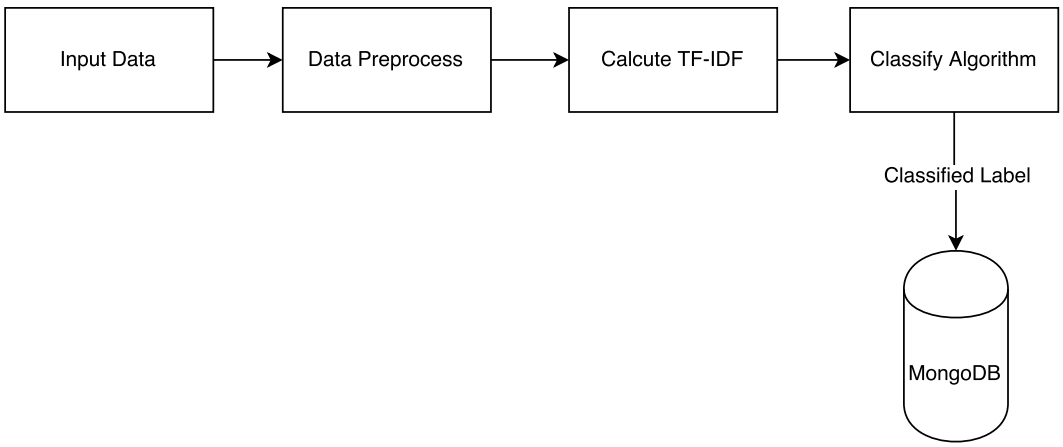
3.2.1 Luồng xử lý dữ liệu

Hệ thống phân loại tin tức được chia thành 2 hệ thống nhỏ hơn, đó là hệ thống Crawler và hệ thống Spam Filtering:

- Hệ thống Crawler có nhiệm vụ đồng bộ hóa tin tức đã được lấy về từ các trang tin tức và thực hiện tiến trình phân lớp(classification process) cho các tin tức đó.
- Hệ thống Spam Filtering cung cấp các API: phân loại tin tức bằng tiêu đề hoặc bằng URL của tin đó, cung cấp dữ liệu thống kê cho các tin đã phân lớp ở hệ thống Crawler, API feedback khi các tin dự đoán sai mong muốn của bình luận viên.

Hệ thống Crawler phục vụ cho bài toán Hot News Detection, sau khi tin tức được lọc rác, các tin tức được đánh giá là rác sẽ bị bỏ qua khi đưa vào hệ thống Hot News Detection. Hệ thống Spam Filtering chủ yếu phục vụ cho biên tập viên. Biên tập viên sử dụng hệ thống thông qua giao diện Spam Filtering để sử dụng chức năng phân lớp tin tức hoặc xem các số liệu thống kê về tin tức đã phân lớp lưu trong cơ sở dữ liệu.

3.3 Tiến trình phân lớp



Hình 3.2: Tiến trình phân lớp

Tiến trình phân lớp sẽ bao gồm Phân hệ tiền xử lý dữ liệu (Data Preprocessor) và phân hệ Classify. Phân hệ Classify bao gồm: Phân hệ phân loại tin tức và Phân hệ phân loại loại rác. Luồng xử lý của tiến trình phân lớp sẽ thực thi theo hình 3.2 :

- Input data: Dữ liệu đầu vào, gồm các document bao gồm Id và title của tin được lưu dưới định dạng csv. Mỗi dòng trong file csv bao gồm 2 cột: Id và title của tin tức đó.
- Data Preprocess: Dữ liệu ở cột title sẽ thực thi theo Tài liệusec:DataPreprocessor. Output của quá trình này là file csv với cột title đã được xử lý.
- Calculate TF-IDF: Dữ liệu title sau khi được tiền xử lý sẽ được tính TF-IDF. Output của quá trình tính tf-idf sẽ là file csv chứa các kết quả tf-idf theo attribute của tập từ điển.
- Classify Algorithm: Hệ thống sẽ sử dụng thư viện Weka để phân lớp dữ liệu. Output của quá trình này sẽ là nhãn(Label) của tin. Sau đó, nhãn sẽ được lưu xuống cơ sở dữ liệu MongoDB.

3.4 Phân hệ thu thập dữ liệu (Data Streaming)

Module này sử dụng hệ thống Crawler để lấy các bài đăng từ nhiều nguồn tin tức lưu trữ ở cơ sở dữ liệu MySQL và đưa qua cơ sở dữ liệu MongoDB, hệ thống sẽ thực hiện các chức năng:

- Connect vào cơ sở dữ liệu mySQL để lấy các bài bài báo.
- Xóa các bài viết trùng.
- Lưu trữ xuống cơ sở dữ liệu MongoDB

3.5 Phân hệ tiền xử lý dữ liệu (Data Preprocessor)

Module tiền xử lý có nhiệm vụ chính gồm loại bỏ URL, thực hiện tách từ, loại bỏ stopwords và biểu diễn dữ liệu thành vector trọng số tf-idf. Trước khi chạy thuật toán phân loại, dữ liệu được lấy ra từ MongoDB và tiền xử lý phần nội dung của bài báo bằng các bước sau:

1. Loại bỏ URL bằng regular expression.

2. Tách từ sử dụng thư viện TPSegmenter. Bước này dùng để biểu diễn các từ ghép trong tiếng Việt bằng cách thêm gạch nối giữa các tiếng của từ.
   Ví dụ: *"Vụ tai nạn 13 người chết: Đã giám định mẫu máu tài xế xe tải"* qua bộ tách từ sẽ thành *"Vụ tai_nạn 13 người chết : Đã giám_định mẫu máu tài_xế xe_tải"*.

3. Loại bỏ các từ trong danh sách gồm 813 stopwords.

4. Tính và biểu diễn dữ liệu thành vector tf-idf và metadata.

3.6 Phân hệ phân loại tin tức

Sử dụng model đã train trước đó để phân loại tin tức. Model sử dụng thuật toán SVM đã trình bày ở chương 2 làm thuật toán chính để phân loại tin tức. Hiện tại hệ thống có 2 nhãn đã được định nghĩa trước đó:

- *K*hông rác
- *R*ác

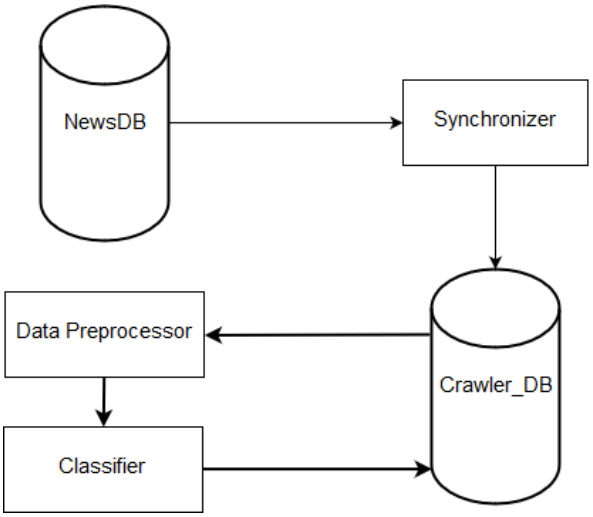
Sau khi đã phân loại, hệ thống sẽ lưu nhãn đã gán cho tin tức xuống cơ sở dữ liệu MongoDB.

3.7 Phân hệ phân loại loại rác

Hệ thống sẽ lấy những tin được gán nhãn "Rác" ở bước phân loại tin tức ở trên để phân loại loại rác cho tin đó. Hiện tại có 3 loại rác đã được định nghĩa trước:

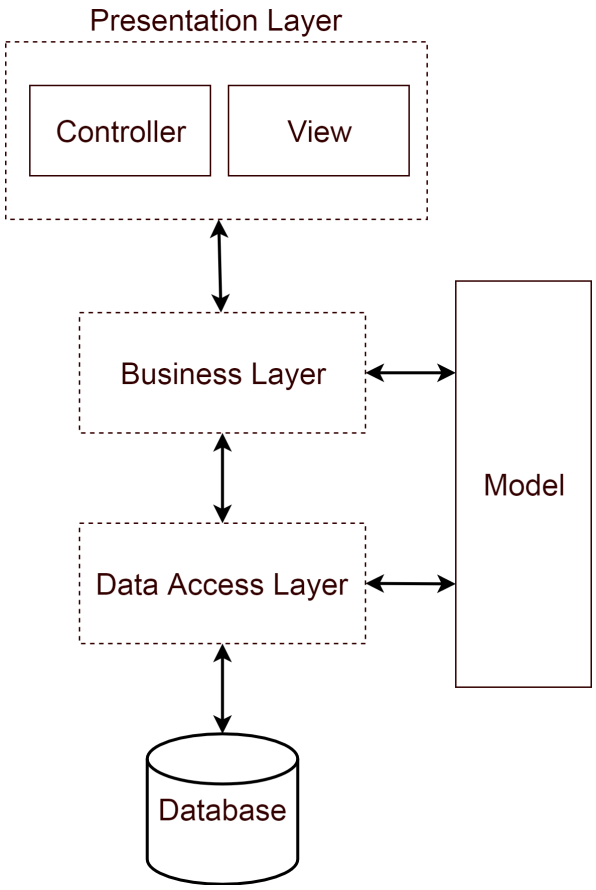
- *Q*uảng cáo
- *T*uyển dụng
- *C*hia sẻ

3.8 Thiết kế hệ thống



Hình 3.3: Kiến trúc hệ thống crawler tin tức

Hệ thống Crawler được xây dựng độc lập bao gồm phân hệ thu thập dữ liệu, phân hệ tiền xử lý và phân hệ phân lớp để chạy ngầm nhằm thu thập và phân loại thông tin



Hình 3.4: Kiến trúc hệ thống Spam Filter kết hợp Multilayer architecture kết hợp với mô hình MVC

Hệ thống xây dựng theo kiến trúc 3 tầng gồm: Presentation Layer, Business Logic Layer và Data Access Layer. Trong đó, Presentation Layer áp dụng mô hình Model - View - Controller (MVC). Cụ thể:

- Presentation Layer: Có trách nhiệm hiển thị thông tin, tương tác với người dùng hệ thống. Gồm 2 thành phần:
  - Controller: Điều khiển các luồng của hệ thống web, nhận các tín hiệu từ người dùng và xử lý tương ứng.
  - View: Có nhiệm vụ hiển thị các giao diện hệ thống cho người dùng, hệ thống sử dụng React để gọi api trả về dữ liệu cho giao diện người dùng.
- Model: Đối tượng chứa dữ liệu để xử lý và hiển thị.
- Business Layer: Chứa các nghiệp vụ của hệ thống. Bao gồm các bước xử lý dữ liệu, thuật toán gom cụm, các tác vụ thống kê,...

- Data Access Layer: Có nhiệm vụ giao tiếp với các hệ cơ sở dữ liệu.

3.9 Cài đặt hệ thống

Hệ thống ứng dụng được xây dựng trên nền tảng Java EE với các thành phần sau:

- Ngôn ngữ: Java, HTML, CSS, JavaScript, React.
- Hệ cơ sở dữ liệu: MongoDB và MySQL.
- Thư viện, framework: Apache Struts 2, Apache Lucene, TPSegmenter, Weka
- Server: Apache Tomcat

3.9.1 Các package

Source code chương trình được tổ chức thành các package như sau:
Hệ thống crawler:

- vn.vccorp.crawler.bo: Chứa các business object của hệ thống
- vn.vccorp.crawler.config: Các file config cho hệ thống
- vn.vccorp.crawler.constant: Các file constant của hệ thống
- vn.vccorp.crawler.dao: Data Access Object, thực hiện các tác vụ đọc, ghi database
- vn.vccorp.crawler.dbconnection: Cung cấp kết nối đến database
- vn.vccorp.crawler.dto: Data Transfer Object, các đối tượng để vận chuyển dữ liệu từ database
- vn.vccorp.crawler.main: Các lớp bao đóng để tạo thread chạy song song các tác vụ
- vn.vccorp.crawler.thread: Các lớp bao đóng để tạo thread chạy song song các tác vụ
- vn.vccorp.crawler.util: Các công cụ hỗ trợ trong hệ thống

Hệ thống Spam Filter được tích hợp vào hệ thống HotNewsDetector:

- vn.vccorp.hotnewsdetector.action.general: lớp ảo chứa các phương thức của action.
- vn.vccorp.hotnewsdetector.action.news: chứa các action cho tin tức
- vn.vccorp.hotnewsdetector.action.twitter: chứa các action cho twitter
- vn.vccorp.hotnewsdetector.bo: Chứa các business object của hệ thống
- vn.vccorp.hotnewsdetector.config: Các file config cho hệ thống
- vn.vccorp.hotnewsdetector.constant: Các file constant của hệ thống
- vn.vccorp.hotnewsdetector.context: chứa các action context của hệ thống
- vn.vccorp.hotnewsdetector.crawler.news: Dùng để lấy dữ liệu từ trang web tin tức
- vn.vccorp.hotnewsdetector.dao: Data Access Object, thực hiện các tác vụ đọc, ghi database
- vn.vccorp.hotnewsdetector.dbconnection: Cung cấp kết nối đến database
- vn.vccorp.hotnewsdetector.dto: Data Transfer Object, các đối tượng để vận chuyển dữ liệu từ database
- vn.vccorp.hotnewsdetector.exception: Quản lý lỗi và đưa ra thông báo của hệ thống
- vn.vccorp.hotnewsdetector.thread
- vn.vccorp.hotnewsdetector.utils

3.9.2 Cơ sở dữ liệu MongoDB

Hệ thống sử dụng MongoDB để lưu trữ dữ liệu tin tức và quản lý kết quả phân lớp. Đây là một hệ cơ sở dữ liệu NoSQL, cung cấp khả năng mở rộng, sao lưu, phân mảnh dữ liệu tốt, và có thể thay đổi cấu trúc dữ liệu một cách linh hoạt.

Dưới đây là bảng so sánh một số thuật ngữ cơ bản giữa các cơ sở dữ liệu SQL truyền thống và MongoDB:

Thuật ngữ SQL	Thuật ngữ MongoDB
database	database
table	collection
row	document hoặc BSON document
column	field
index	index
table joins	$lookup, embedded documents

Bảng 3.1: So sánh các thuật ngữ giữa SQL và MongoDB

3.9.2.1 Collection News

Collection này chứa thông tin về các tin lấy từ cơ sở dữ liệu MySQL, cũng là nơi lưu trữ tất cả thông tin về tweet trong hệ thống. Khi dữ liệu được stream từ MySQL về, mỗi tin chỉ có 12 trường, các trường khác sẽ được thêm vào trong quá trình hệ thống xử lý.

Thuộc tính	Loại	Ý nghĩa
_id	ObjectID	ID trong MongoDB của document
Id	Int	ID của tin tức lấy về
Title	String	Title của tin tức
Content	String	Nội dung của tin tức
Source	String	Nguồn trang báo điện tử của tin tức
CreateTime	Date	Thời gian đăng của tin
GetTime	Date	Thời gian tin tức được thu thập vào cơ sở dữ liệu MySQL
CollectDate	Date	Thời gian tweet được thu thập vào cơ sở dữ liệu MongoDB
Author	String	Người viết bài báo(nếu có)
Category	Integer	Chủ đề của bài báo
SpamLabel	String	Nhãn đã gán cho tin tức
SpamCategory	String	Nhãn đã gán cho loại tin rác
SpamLabelFeedback	Integer	Feedback của nhãn đã gán cho tin tức đó

Bảng 3.2: Các trường của collection News

3.9.2.2 Collection HashCode

Collection cluster chứa các hash code cho bài báo để kiểm tra trùng cho tin tức đó.

Thuộc tính	Loại	Ý nghĩa
_id	ObjectID	ID trong MongoDB của cụm
HashCode	String	Hash code của tin tức

Bảng 3.3: Các trường của collection HashCode

3.10 Các API hệ thống HotNewsDetector

Hệ thống HotNewsDetector cung cấp các API:

3.10.1 Classified News List

Lấy và hiển thị các tin đã được phân loại và gán nhãn "Không rác" hoặc "Rác" trong một ngày

URL:

/HotNewsDetector/news/spam/list

Request Method:

GET

Data Params:

- startDate (String): chuỗi thể hiện ngày cần lấy tin với định dạng "dd-MM-yyyy". Lưu ý ngày trong CSDL được lưu theo múi giờ UTC+0:00.
- offset (Integer, optional): số lần lượt bỏ các tin đầu tiên theo limit (số tin lượt bỏ = offset * limit). Mặc định là 0. Giá trị phải là số nguyên không âm.
- limit (Integer, optional): số tin tối đa cần lấy từ danh sách trả về (sau khi đã sắp xếp và lược bỏ). Giá trị phải là số nguyên không âm và nhỏ hơn 50. Giá trị mặc định là 10.

Success Response:

```
{
    "error": false,
    "data": "[{
        "title": "...",
        "postdate": "...",
```

```
        "id": "...",
        "source": "...",
        "SpamLabel": "...",
        "spamCategory": "...",
        "spamLabelFeedback": "...",
        "url": "..."
    },...]"
}
```

Error Response

```
    "error": true,
    "message": "..."
```

Attributes

- id(String): ID của tin.
- title(String): tiêu đề tin.
- source(String): nguồn post tin.
- postDate(String): ngày đăng tin.
- spamLabel(String): nhãn của tin đã được phân loại, nhãn của tin có thể có 3 giá trị: Rác, Không rác hoặc null nếu chưa phân loại.
- spamCategory(String): nhãn của loại rác mà tin rác được phân loại, nhãn của loại rác có thể có 4 giá trị: Quảng cáo, Tuyển dụng, Chia sẻ hoặc null nếu tin được phân loại là "Tin" hoặc chưa phân loại.
- spamLabelFeedback(String): phản hồi của người dùng về nhãn mà tin được tiên đoán. Có thể là "Tin", "Rác" hoặc null.
- url(String): đường dẫn tới tin.

Sample Call

```
axios.get("/HotNewsDetector/news/spam/list", {params:{startDate="11
.then(response => { console.log(response);
});
```

3.10.2 Spam Label Feedback

Cho phép người dùng phản hồi về nhãn của một tin (Rác, Tin)

URL:

/HotNewsDetector/news/spam/feedback

Request Method:

POST

Data Params:

- spamFeedback(Integer, optional): giá trị feedback người dùng. Các giá trị có thể sử dụng: 0 (Rác), 1 (Tin).
- id(String): ID của tin.

Success Response:

```
{
    "error": false,
    "message": "Thao tác thành công"
}
```

Error Response

```
    "error": true,        '
    "message": "..."
```

Attributes

- error(Boolean): Kết quả trả về, true nếu có lỗi và false nếu thành công.
- message(String): Thông báo thông tin cho người dùng.

Sample Call

```
axios.get("/HotNewsDetector/news/spam/list",
{params:{startDate="11-12-2017", offset=2, limit=5}})
.then(response => { console.log(response);
});
```

3.10.3 Spam Filter

Cho phép người dùng phân loại một tin. Dầu vào của API có thể là danh sách chứa các URL của các tin hoặc một chuỗi String là title của các bài viết.

URL:

/HotNewsDetector/news/spam

Request Method:

POST

Data Params:

- data(String[]): mảng chứa nội dung của các tin cần phân lớp, có thể là URL hoặc title của tin. Có thể nhận vào một hoặc nhiều tin cùng một lúc.
- type(int): xác định thuật toán sử dụng để phân lớp, có 2 giá trị có thể sử dụng: 0(Sử dụng thuật toán SVM), 1(Sử dụng Doc2Vec).

Success Response:

```
{
    error: false,
    data: [{
        "label": "...",
        "type": "...",
        "content": "..."}
    },...]
}
```

Error Response

```
    "error": true,
    "message": "..."
```

Attributes

- label(String): nhãn của tin đã phân lớp: Tin, Rác(Quảng cáo, tuyển dụng, chia sẻ) nếu loại tin là URL, nhãn của tin có thể là Undefined nếu không lấy được tin về từ web.
- type(String): loại tin đã nhập (URL hoặc Text)
- content (String): Nội dung đã nhập (URL hoặc nội dung tin)

Sample Call

axios.get("/HotNewsDetector/news/spam?data= Trà Ngọc Hằng Thà cô đơn chứ không đổi tự trọng lấy tình yêu"&type=0")
.then(response => { console.log(response);
});

3.10.4 Spam statistic

Lấy các thông số thống kê về số lượng tin đã phân lớp theo khoảng thời gian và theo nguồn tin

URL:

/HotNewsDetector/news/spam/statistics

Request Method:

GET

Data Params:

dateRange(Integer): số ngày cần lấy dữ liệu thống kê, chỉ nhận các giá trị 1, 7, 30, và 365 **Success Response:**

```
"error": false,
"data": [
{
    "bySource": [{
    "source": ...,
    "news": ...,
    "spams": ...
```

```
    },...],
        "byTime": [{
        "categories": ...,
        "news": ...,
        "spam": ...
    }, ...]}
]
```

Error Response

```
    "error": true,
    "message": "..."
```

Attributes

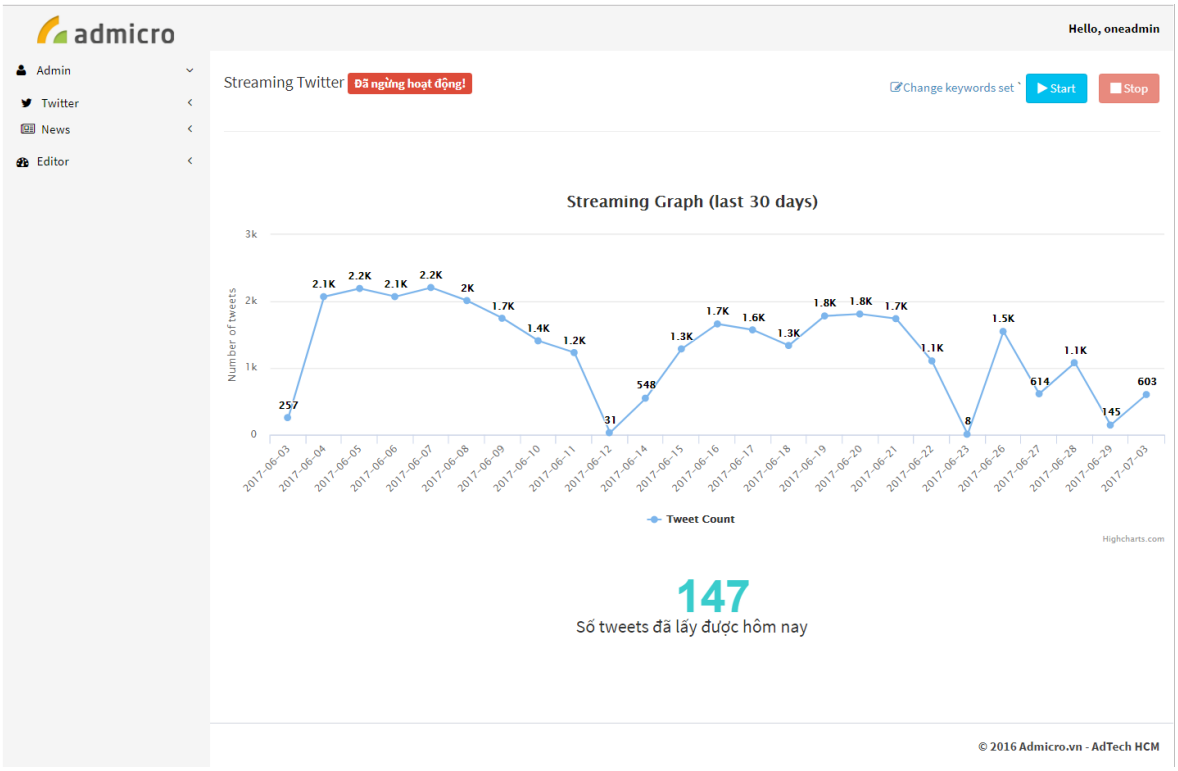
- (String): dữ liệu thống kê theo nguồn đưa tin.
- byTime(String): dữ liệu thống kê theo khoảng thời gian.
- source(String): tên nguồn tin.
- categories(String): khoảng thời gian đã phân chia, khoảng thời gian có thể có 4 giá trị tùy theo dateRange đã nhập:
  - Date range = 1: khoảng thời gian sẽ được chia làm 12 phần, mỗi phần tương ứng 2 tiếng
  - Date range = 7: khoảng thời gian được chia làm 7 phần, mỗi phần tương ứng 1 ngày.
  - Date range = 30: khoảng thời gian được chia làm 15 phần, mỗi phần tương ứng 2 ngày.
  - Date range = 365: khoảng thời gian được chia làm 12 phần, mỗi phần tương ứng 1 tháng.
- news(String): số lượng tin đã phân lớp "Không rác"
- spam(String): số lượng tin đã phân lớp là "Rác"

Sample Call

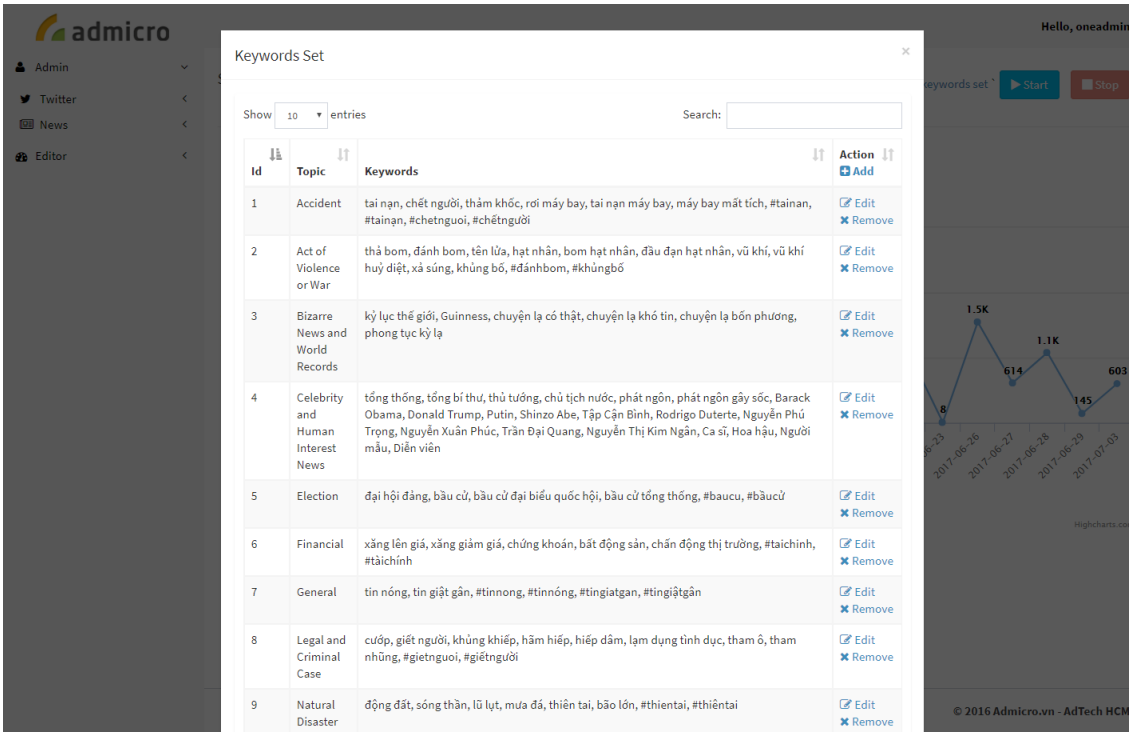
```
axios.get("/HotNewsDetector/news/spam/statistics", { params:{dateRa
.then(response => { console.log(response);
});
```

3.11 Kết quả

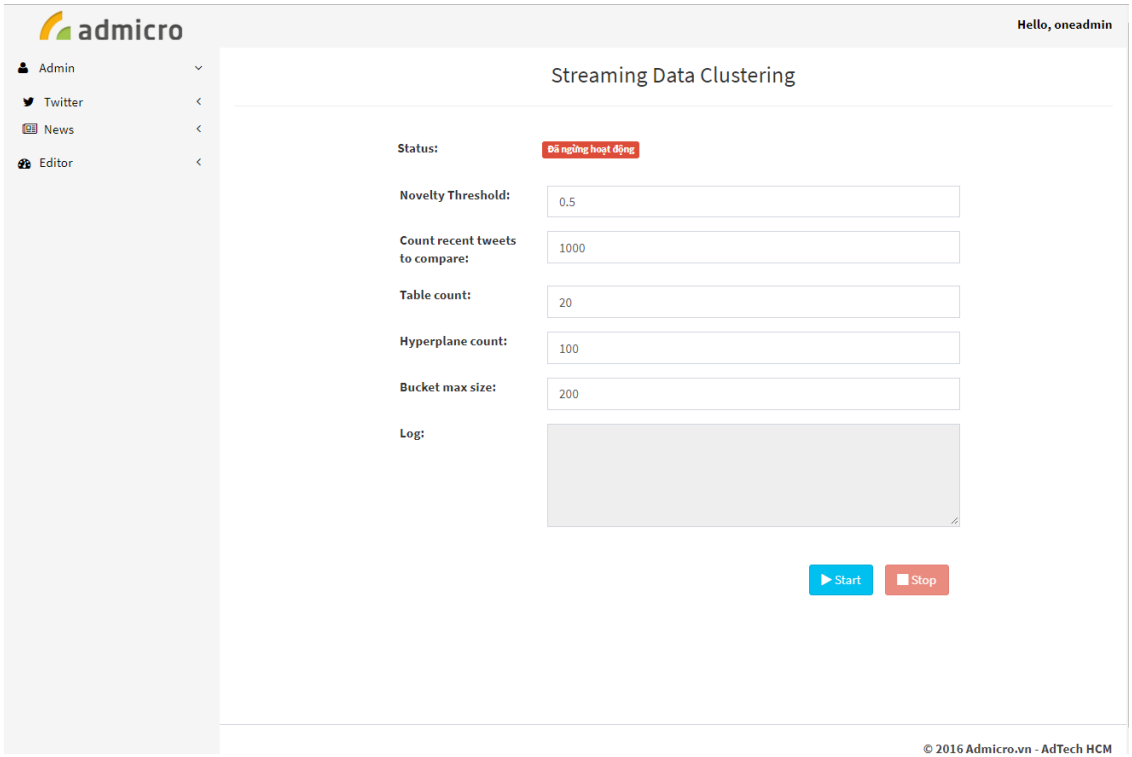
Hệ thống đã hoàn thiện các chức năng: Thu thập dữ liệu từ trang tin tức, điều khiển chạy luồng của thuật toán phân loại tin nóng, và hiển thị thống kê cho biên tập viên, hệ thống còn cung cấp giao diện phân loại tin tức bằng url hoặc title của tin tức. Dưới đây là một số hình ảnh của hệ thống:



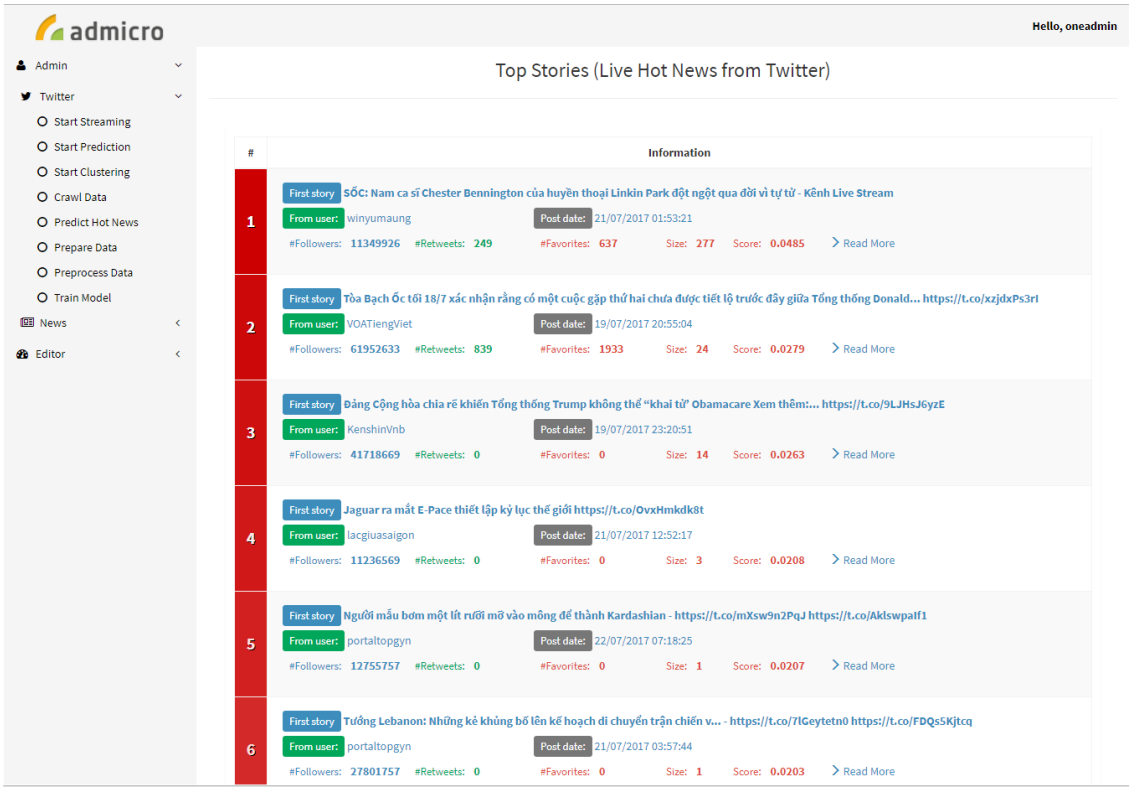
Hình 3.5: Giao diện điều khiển thu thập dữ liệu Twitter



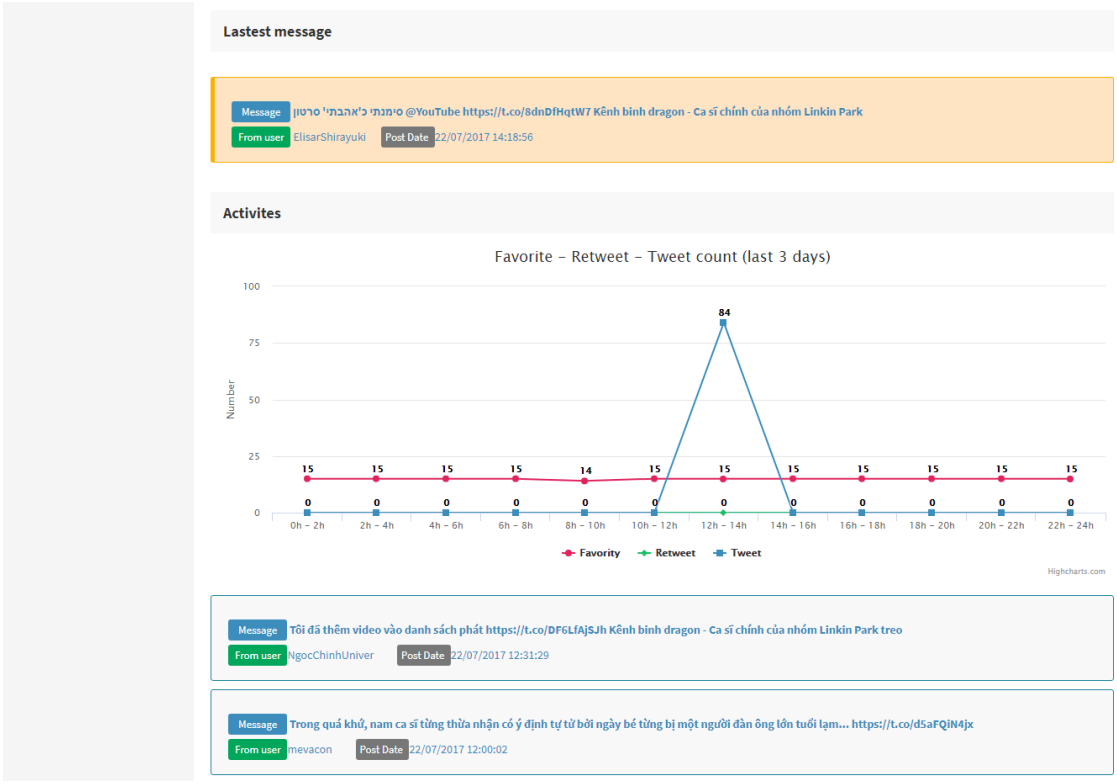
Hình 3.6: Giao diện thêm xóa từ khóa để thu thập dữ liệu Twitter



Hình 3.7: Giao diện điều khiển thuật toán phát hiện tin nóng



Hình 3.8: Giao diện hiển thị các cụm bài viết cho biên tập viên



Hình 3.9: Chi tiết một cụm bài viết

3.12 Kết chương

Chương này đã trình bày về các thành phần chính hệ thống, kiến trúc phân tầng, các hệ cơ sở dữ liệu được sử dụng và cách tổ chức, cùng một số kết quả cài đặt hệ thống.

Chương 4

THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Mở đầu

Mục đích của chương này là trình bày một số kết quả thực nghiệm trên bộ dữ liệu thu thập được. Qua đó đánh giá, nhận định và so sánh các thuật toán phân lớp.

4.2 Tổng quan về bộ dữ liệu

Bộ dữ liệu gồm các bài viết từ nhiều nguồn tin tức. Dữ liệu được lấy thông qua luồng: Crawler . Một danh sách hơn 700 nguồn được sử dụng để lấy dữ liệu thông tin từ trên mạng.

Mỗi tin trong tập dữ liệu bao gồm các thông tin sau: Id , title của tin tức, nội dung tin tức, nguồn đăng tin tức, thời gian tin tức được post, thời gian lấy về cơ sở dữ liệu MySQL, thời gian tin tức được thu thập vào hệ thống, description của bài báo, đường dẫn link của tin tức, tác giả.

Bộ dữ liệu thử nghiệm gần nhất gồm 15,612 tin tiếng Việt, được thu thập thông qua Crawler trong khoảng thời gian 11 ngày, từ 31/1/2017 đến 10/2/2017.

Dữ liệu thống kê cho tập dữ liệu sử dụng để train các model:

Dữ liệu gán nhãn	Số lượng
Không rác	7331
Rác	8281
Tổng	15612

Bảng 4.1: Thống kê dữ liệu gán nhãn

Dữ liệu gán nhãn	Số lượng
Quảng cáo	4627
Chia sẻ	3279
Tuyển dụng	375
Tổng	8281

Bảng 4.2: Thống kê loại rác

Accident	tai nạn, chết người, thảm khốc, rơi máy bay, tai nạn máy bay, máy bay mất tích, #tainan, #tainạn, #chetnguoi, #chếtngười
Act of War or Violence, Military News	thả bom, đánh bom, tên lửa, hạt nhân, bom hạt nhân, đầu đạn hạt nhân, vũ khí, vũ khí huỷ diệt, xả súng, khủng bố, #đánhbom, #khủngbố
Bizarre News and World Records	kỷ lục thế giới, Guinness, chuyện lạ có thật, chuyện lạ khó tin, chuyện lạ bốn phương, phong tục kỳ lạ
Celebrity and Human Interest News	tổng thống, tổng bí thư, thủ tướng, chủ tịch nước, phát ngôn, phát ngôn gây sốc, Barack Obama, Donald Trump, Putin, Shinzo Abe, Tập Cận Bình, Rodrigo Duterte, Nguyễn Phú Trọng, Nguyễn Xuân Phúc, Trần Đại Quang, Ca sĩ, Hoa hậu, Người mẫu, Diễn viên
Election	đại hội đảng, bầu cử, bầu cử đại biểu quốc hội, bầu cử tổng thống, #baucu, #bầucử
Financial	xăng lên giá, xăng giảm giá, chứng khoán, bất động sản, chấn động thị trường, #taichinh, #tàichính
General	tin nóng, tin giật gân, #tinnong, #tinnóng, #tingiatgan, #tingiậtgân
Legal and Criminal Case	cướp, giết người, khủng khiếp, hãm hiếp, hiếp dâm, lạm dụng tình dục, tham ô, tham nhũng, #gietnguoi, #giếtngười
Natural Disaster	động đất, sóng thần, lũ lụt, mưa đá, thiên tai, bão lớn, #thientai, #thiêntai
New Law	dự thảo luật, điều luật mới, chính sách mới
Political Meeting and Statement	tổng thống, tổng bí thư, thủ tướng, chủ tịch nước, hội nghị, cuộc gặp gỡ, gặp mặt, họp mặt
Scandal and Hearing	bê bối, quấy rối, quấy rối tình dục, hầu tòa, ra tòa, scandal, #scandal
Science and Discovery	phát minh, giải nobel, giải fields, khám phá mới, phát hiện mới
Sport	đội tuyển bóng đá quốc gia, đội tuyển U19, U19 Hoàng Anh Gia Lai, SeaGames, AFC Suzuki Cup, #bongda, #bóngđá

Bảng 4.3: Danh sách từ khóa để thu thập dữ liệu

Bộ dữ liệu có thể được tải về từ địa chỉ: https://drive.google.com/open?id=1xZVBcaVtZAmQ4xU0KKPvB5ZRAUoKc2_v

4.3 Thiết lập thực nghiệm, cách đánh giá

Sử dụng 3 thuật toán phân lớp: thuật toán Naive Bayes, thuật toán J48, thuật toán Support Vector Machine.

Ta đánh giá và so sánh kết quả về thời gian xử lý và chất lượng phân lớp thông qua một số độ đo đã trình bày ở mục 2.9: Precision, Recall, F-Measure, ROC Area, confusion matrix.

4.4 Kết quả thực nghiệm

4.4.1 Kết quả train model phân loại tin tức

Dưới đây là kết quả các thuật toán và kết quả của một số độ đo :

4.4.1.1 Kết quả dựa trên nội dung của tin của tập mẫu

Dữ liệu gán nhãn	Số lượng
Mẫu dữ liệu	15612
Số chiều	111363
Cross-validation:	10

Bảng 4.4: Thông số cơ bản của tập train

Time build model(seconds):		
SVM	NaiveBayes	J48
382.85	853.67	43530.21

Bảng 4.5: Thời gian train model

Class	Precision			Recall			F-Measure			ROC Area		
	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes
Rác	0.845	0.876	0.847	0.838	0.915	0.874	0.842	0.895	0.860	0.823	0.884	0.854
Tin	0.819	0.898	0.852	0.827	0.854	0.821	0.823	0.876	0.836	0.823	0.884	0.871
	0.832	0.887	0.849	0.833	0.886	0.849	0.832	0.886	0.849	0.823	0.884	0.862

Bảng 4.6: Kết quả train model dựa trên một số độ đo

4.4.1.2 Kết quả dựa trên tiêu đề của tin của tập mẫu

Dưới đây là kết quả các thuật toán và kết quả của một số độ đo :

Dữ liệu gán nhãn	Số lượng
Mẫu dữ liệu	15612
Số chiều	15133
Cross-validation:	10

Bảng 4.7: Thông số cơ bản của tập train

Time build model(seconds):		
SVM	NaiveBayes	J48
66.23	130.78	5895.41

Bảng 4.8: Thời gian train model

Class	Precision			Recall			F-Measure			ROC Area		
	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes
Rác	0.845	0.876	0.847	0.838	0.915	0.874	0.842	0.895	0.860	0.823	0.884	0.854
Tin	0.819	0.898	0.852	0.827	0.854	0.821	0.823	0.876	0.836	0.823	0.884	0.871
	0.832	0.887	0.849	0.833	0.886	0.849	0.832	0.886	0.849	0.823	0.884	0.862

Bảng 4.9: Kết quả train model dựa trên một số độ đo

4.4.2 Kết quả train model phân loại loại tin rác

Dữ liệu gán nhãn	Số lượng
Mẫu dữ liệu	8281
Số chiều	9526
Cross-validation:	10

Bảng 4.10: Thông số cơ bản của tập train

Time build model(seconds):		
SVM	NaiveBayes	J48
14.46	19.39	590.78

Bảng 4.11: Thời gian train model

Class	Precision			Recall			F-Measure			ROC Area		
	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM	NaiveBayes	J48	SVM
Chia sẻ	0.97	0.952	0.96	0.730	0.985	0.986	0.833	0.968	0.973	0.973	0.978	0.980
Quảng cáo	0.931	0.990	0.99	0.945	0.968	0.975	0.938	0.979	0.982	0.962	0.980	0.981
Tuyển dụng	0.335	0.989	1	0.992	0.952	0.955	0.5	0.970	0.977	0.954	0.987	0.977
	0.919	0.975	0.979	0.862	0.974	0.979	0.877	0.974	0.979	0.966	0.980	0.981

Bảng 4.12: Kết quả train model dựa trên một số độ đo

4.5 Nhận xét

4.5.1 Nhận định về các thuật toán phân lớp cho bài toán

Dựa vào bảng 4.6, 4.9 và 4.12, ta thấy rằng với độ đo và tập dữ liệu mẫu, thuật toán Support Vector Machine cho kết quả tốt với mọi độ đo so với thuật toán Naive Bayes và J48.

Dựa vào bảng 4.5, 4.8 và 4.11, ta thấy rằng thời gian để train model cho thuật toán Support Vector Machine tốn ít thời gian hơn thuật toán Naive Bayes và J48.

4.6 Kết chương

Qua các kết quả thử nghiệm, chương này đã thể hiện được một số tính chất của các thuật toán được đánh giá như thời gian train, một số độ đo thường dùng để đánh giá một hệ phân lớp. Ta thấy thuật toán SVM có thời gian train nhanh trong khi vẫn giữ được độ chính xác tốt so với thuật toán Naive Bayes và J48. Riêng thuật toán J48 có hạn chế về thời gian train quá lớn so với hai thuật toán còn lại.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết quả đạt được

Mục tiêu chính của đề tài là xây dựng hệ thống phát hiện tin nóng để hỗ trợ biên tập viên trong việc viết bài. Một số nội dung thực hiện được đề ra ban đầu là: tìm hiểu bài toán liên quan, thu thập dữ liệu, cài đặt và đánh giá một số thuật toán, và xây dựng hệ thống.

Sau quá trình nghiên cứu và thực hiện, khóa luận đã thu được một số kết quả sau:

- Kiến thức:
  - Tìm hiểu về bài toán phát hiện tin tức, tin nóng trên dữ liệu từ mạng xã hội.
- Sản phẩm:
  - Thu thập bộ dữ liệu gồm các bài đăng tiếng Việt từ nguồn Twitter.
  - Khảo sát và đánh giá các phương pháp phát hiện tin nóng dựa trên gom cụm: k-Nearest Neighbor, Boost Named Entity, Locality Sensitive Hashing.
  - Xây dựng được hệ thống có khả năng nhận biết tin nóng, đang được triển khai tại công ty VCCorp.

Hướng phát triển

Tuy hệ thống đạt được kết quả tương đối khá tốt trong việc phát hiện các sự kiện, cần cải thiện độ chính xác của thuật toán thông qua việc cân chỉnh kĩ hơn các thông

số, đồng thời xem xét tìm hiểu thêm và so sánh với các phương pháp khác.

Trong tương lai hệ thống cần lấy thêm dữ liệu từ các nguồn khác như Facebook và các trang báo mạng để làm giàu nguồn tin. Thêm các chức năng tiện ích cho biên tập viên như thống kê dữ liệu tổng quát lẫn chi tiết, cung cấp cho biên tập viên nhiều góc nhìn về sự kiện hơn.

TÀI LIỆU THAM KHẢO

[1] Jonathan G. Fiscus and George R. Doddington. Topic detection and tracking. chapter Topic Detection and Tracking Evaluation Overview, pages 17–31. Kluwer Academic Publishers, Norwell, MA, USA, 2002. 6, 7

[2] James Allan. Topic detection and tracking. chapter Introduction to Topic Detection and Tracking, pages 1–16. Kluwer Academic Publishers, Norwell, MA, USA, 2002. 7, 8

[3] George Dimitoglou, James A Adams, and Carol M Jim. Comparison of the c4. 5 and a naïve bayes classifier for the prediction of lung cancer survivability. *arXiv preprint arXiv:1206.1121*, 2012. 8

[4] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: News in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM. 16

[5] S. Phuvipadawat and T. Murata. Breaking news detection and tracking in twitter. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 120–123, Aug 2010. 16, 21, 33

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 16, 17

[7] James Allan, Victor Lavrenko, Daniella Malin, and Russell Swan. Detections,

bounds, and timelines: Umass and tdt-3. In *In Proceedings of Topic Detection and Tracking Workshop (TDT-3)*, 2000. 17

[8] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM. 24

[9] Saša Petrović, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 181–189, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 28

[10] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson Education, Limited, 2014. 29

[11] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973. 31

Phụ lục. Giới thiệu về thư viện Apache Lucene

Giới thiệu

Truy hồi thông tin là một lĩnh vực nghiên cứu lớn nhắm giải quyết vấn đề tìm kiếm, lọc đường thông tin cần thiết, hữu ích từ lượng dữ liệu khổng lồ. Apache Lucene là một thư viện mã nguồn mở, được viết bằng Java và dùng để hỗ trợ giải bài toán này. Thư viện cung cấp khả năng đánh chỉ mục (index) trên dữ liệu và thực hiện tìm kiếm trên chỉ mục đó, cùng với các chức năng hỗ trợ khác.

Lucene lưu trữ dữ liệu ở dạng chỉ mục ngược (inverted index), cho phép tìm kiếm các đối tượng văn bản dựa trên từ khóa một cách nhanh chóng. Một đối tượng văn bản trong Lucene được gọi là Document. Mỗi Document có một hoặc nhiều Field, ứng với các thuộc tính của Document đó. Một bài viết hay một trang web có thể là một Document, với các Field như: tiêu đề, nội dung, tác giả, ngày đăng,...

Một số class chính của thư viện:

- Analyzer và các class con: có nhiệm vụ phân tích dữ liệu văn bản thành những token/term trước khi ghi vào index. Một số class con như StandardAnalyzer, WhitespaceAnalyzer, SimpleAnalyzer, KeywordAnalyzer.
- IndexWriter: nhận luồng dữ liệu đã tokenize bằng Analyzer và ghi vào index.
- IndexReader, IndexSearcher: đọc và tìm kiếm trên index đã tạo từ trước, ngoài ra cung cấp các thông tin khác từ index như danh sách term, tần số của term trong một Document, trong toàn bộ dữ liệu.

- Query và các class con: dùng để xây dựng câu truy vấn và truyền vào IndexSearcher để thực hiện tìm kiếm. Một số class con như TermQuery, RangeQuery, Boolean Query.

Hệ thống chủ yếu sử dụng Lucene hỗ trợ trong bước tiền xử lý dữ liệu, nhằm tính và biểu diễn các bài viết ở dạng vector tf-idf, thông qua các thông tin về tần số term trong index.

Sử dụng Lucene trong Java

Tạo IndexWriter

Để ghi dữ liệu vào Lucene index, ta cần tạo IndexWriter như sau:

```
indexDirectory = FSDirectory.open(new File(indexDir));
analyzer = new StandardAnalyzer(Version.LUCENE_36,
    Collections.emptySet());
IndexWriterConfig config = new
    IndexWriterConfig(Version.LUCENE_36, analyzer);
config.setOpenMode(OpenMode.CREATE);
IndexWriter writer = new IndexWriter(indexDirectory, config);
```

Thêm một document vào index

Giả sử ta có một tweet với ID là "001", nội dung là "Tai nan kinh hoang khi xe tai mat phanh", dưới đây là cách tạo và thêm vào index document với 2 field tương ứng là "tweetID" và "tweetContent".

```
Field tweetID = new Field("tweetID", "001", Field.Store.YES,
    Field.Index.NO);
Field tweetContent = new Field("tweetContent", "Tai nan kinh hoang
    khi xe tai mat phanh", Field.Store.YES, Field.Index.ANALYZED,
    Field.TermVector.YES);
```

```
Document lucenceDocument = new Document();
lucenceDocument.add(tweetID);
lucenceDocument.add(tweetContent);
writer.addDocument(luceneDocument);
```

Đọc dữ liệu từ index

Sau khi tạo index, ta có thể đọc thông tin trong index thông qua IndexReader hoặc IndexSearcher.

```
IndexReader reader = IndexReader.open(indexDir);
IndexSearcher searcher = new IndexSearcher(reader);
```

Cách tính giá trị **idf** cho tất cả term trong field "tweetContent" trong bộ dữ liệu:

```
int docCount = reader.numDocs();
TermEnum listOfTerms = reader.terms();
TreeMap<String, Double> idfVector = new TreeMap<String, Double>();
while (listOfTerms.next()) {
   String currentTerm = listOfTerms.term().text();
   int docFreq = searcher.docFreq(new Term("tweetContent",
      currentTerm));
   double idf = 1 + Math.log((double) docCount / docFreq);
   idfVector.put(currentTerm, idf);
}
```

Cách tính vector tf-idf cho document thứ **i** trong index:

```
TermFreqVector tfv = reader.getTermFreqVector(i, "tweetContent");
String[] termList = tfv.getTerms(); //list of terms in this
   document
int[] termFreqList = tfv.getTermFrequencies();
```

```
int totalTermCount = 0;
LinkedHashMap<String, Double> tfidfVector = new
    LinkedHashMap<String, Double>();

// calculate (total) term count in document i
for (int temp : termFreqList) {
   totalTermCount += temp;
}
// loop through all term in doc i and calculate tfidf vector
int uniqueTermCount = termList.length;
for (int j = 0; j < uniqueTermCount; j++) {
   if (termFreqList[j] != 0 && idfVector.containsKey(termList[j])){
      double tfidf = ((double)termFreqList[j] / totalTermCount)
      * idfVector.get(termList[j]);
      tfidfVector.put(termList[j], tfidf);
   }
}
```