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

Cùng với sự phát triển nhanh chóng của internet là sự tăng trưởng mạnh mẽ của ngành truyền thông báo chí. Cụ thể, tính đến ngày 31/12/2015, cả nước Việt Nam có 105 cơ quan báo điện tử, 207 trang thông tin điện tử tổng hợp của các cơ quan báo chí [1], và cứ mỗi giờ có khoảng 2,118 bài được đăng trên các trang báo mạng Việt Nam [2]. Cùng với sự tăng trưởng lớn về thông tin là nhu cầu phân tích và xử lý các thông tin đó để phục vụ cho nhiều mục tích khác nhau, và ngày nay nhiều doanh nghiệp đã và đang xây dựng các hệ thống phân tích dữ liệu tin tức để phục vụ tốt hơn cho người dùng và biên tập viên.

Việc phân tích dữ liệu tin tức đòi hỏi hệ thống phải thu thập dữ liệu từ nhiều website tin tức khác nhau để có được một nguồn dữ liệu đa dạng, phong phú và đủ lớn. Tuy nhiên, dữ liệu thu thập từ các website có bản chất không phù hợp với mục đích của hệ thống phân tích có thể gây ra nhiễu và khiến cho kết quả phân tích, xử lý không chính xác. Do đó, việc sàn lọc dữ liệu thu thập được trước khi đưa vào phân tích và xử lý là cần thiết.

Hiểu được nhu cầu trên, chúng em thực hiện khóa luận này với mục tiêu xây dựng một hệ thống lọc "rác" cho dữ liệu tin tức từ các nguồn báo chính thống Việt Nam, cụ thể là lọc "rác" cho hệ thống "Phát hiện tin nóng".

Trong phạm vi của khóa luận, chúng em đã nghiên cứu một số thuật toán phân lớp: SVM, Naive Bayes, J48.

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 trên các website tin tức Việt Nam.
- Đánh giá và so sánh các thuật toán: SVM, Naive Bayes, J48.

[1] http://mic.gov.vn/Pages/TinTuc/116095/Tinh-hinh-phat-trien-linh-vuc-bao-chi-va-phat-thanh-truyen-hinh-nam-2015.html

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

- Xây dựng được hệ thống lọc rác cho hệ thống phát hiện tin nóng.

Chương 1

MỞ ĐẦU

1.1 Dẫn nhập

Trong thời đại bùng bổ về thông tin, báo chí điện tử là một trong những nguồn thông tin lớn và quan trọng cần được khai thác và phân tích một cách hiệu quả. Hệ thống phát hiện tin nóng hiện đang được triển khai và hoạt động tại VCCorp là một trong các hệ thống phân tích và xử lý dữ liệu tin tức, được phát triển nhằm mục đích xác định tin nóng từ các trang báo điện tử Việt Nam. Tuy nhiên, sự đa dạng và phong phú của nguồn tin dẫn đến việc bản chất của nhiều bài thu thập được không phù hợp với mục tiêu của hệ thống. Những bài đó, đối với hệ thống, được hiểu là "rác". Số lượng "rác" lớn sẽ gây nhiễu và làm cho việc phân tích và xử lý trở nên kém chính xác. Một cơ chế lọc rác hiệu quả sẽ giúp cải thiện độ chính xác cũng như giảm tải cho hệ thống từ việc phân tích và xử lý các dữ liệu không có giá trị sử dụng.

Bài toán đặt ra là làm thể nào để có thể lọc "rác" trong quá trình thu thập tin tức một các chính xác. 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 "rác" để tăng hiệu quả cho hệ thống phát hiện tin nóng và giúp đánh giá độ đáng tin cậy của các nguồn tin.

1.2 Mục tiêu đề tài

- Tìm hiểu và đánh giá các thuật toán phân lớp cho việc lọc rác dữ liệu tin tức
- Xây dựng hệ thống lọc rác áp dụng các thuật toán phân lớp.

1.3 Nội dung thực hiện

- Tìm hiểu bài toán phân loại tin tức, tìm hiểu các phương pháp và các hướng tiếp cận
- Thử nghiệm đánh giá các phương pháp đã tìm hiểu:
  - Thu thập dữ liệu tin tức từ cơ sở dữ liệu của công ty VCCorp.
  - Tiến hành một số thống kê trên dữ liệu thu thập được.
  - Huấn luyện và so sánh các thuật toán phân lớp: Naive Bayes, SVM, J48.
- Xây dựng hệ thống:
  - Tìm hiểu về MongoDB, framework Struts 2, thư viện Apache Lucene, Weka, LibSVM.
  - 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 báo chính thống Việt Nam.
- Ngôn ngữ: tiếng Việt.
- Các thuật toán tìm hiểu: Naive Bayes, SVM, J48

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 lọc rác 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 lọc rác.

- **Chương 3**: Hiện thực hệ thống lọc rác: 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 LỌC RÁC TIN TỨC VÀ CÁC PHƯƠNG PHÁP TIẾP CẬN PHỔ BIẾN

2.1 Mở đầu

Chương này sẽ giới thiệu bài toán lọc rác tin tức. Trình bày cơ sở lý thuyết và phát biểu bài toán lọc rác tin tức. Cuối cùng trình bày một số phương pháp tiếp cận bài toán lọc rác tin tức 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 $D = (d_1, d_2, d_3, \dots d_n)$: 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 $C = (c_1, c_2, \dots c_m)$, với mỗi phần tử $c_i = (d_{i1}, d_{i2}, \dots d_{ik})$ 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 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 [3] 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 [4] 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.

2.4 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 [5]:

$$D_{Euclidean}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

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 [5]:

$$Similarity_{Cosine}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \cdot ||\vec{y}||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.2}$$

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

2.5 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*.

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

2.5.1.1 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 [6]. 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.

2.5.1.2 Minh họa thuật toán

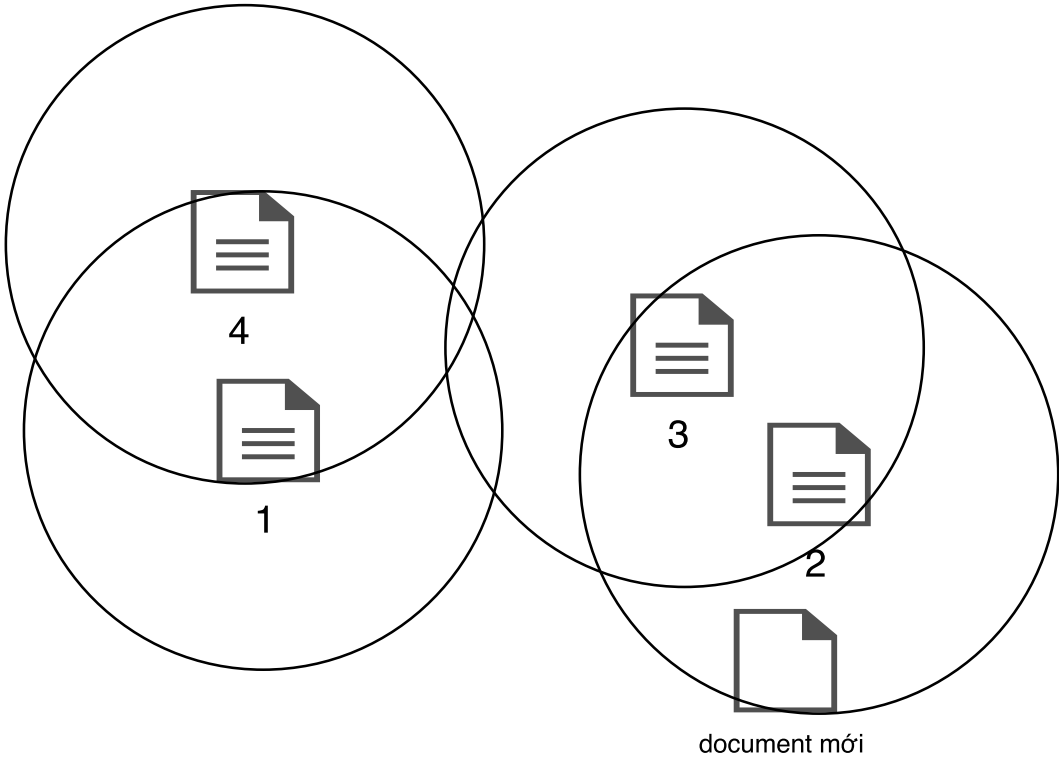
Giả sử ta có dữ liệu như hình vẽ. Đường tròn thể hiện ngưỡng khoảng cách để một document được xem là first story hay không, chọn ngưỡng giá trị 0.8. Ta có sẵn 4 document và 1 document mới vào hệ thống như sau:

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

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

2.5.1.3 Mã giả

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

luồng bài viết từ Twitter, giá trị ngưỡng MergeThreshold 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 $similarity(d, d')$ $S(d) \leftarrow S(d) \cup d'$ $index[t] \leftarrow index[t] \cup d$ (thêm document d vào index của term t) similarity score của d: $similarity_{max}(d) \leftarrow 0$ **foreach** document d' trong S(d) **do** $similarity(d, d') > similarity_{max}(d)$ $similarity_{max}(d) \leftarrow similarity(d, d')$ $similarity_{max}(d) < MergeTheshold$ d là một first story

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 2.4
- $index(t)$: danh sách cách document có chứa term t
- $dis_{min}(d)$: khoảng cách giữa document d với document gần nhất

2.5.1.4 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.

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

2.5.2.1 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 [4]. 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.

2.5.2.2 Minh họa thuật toán

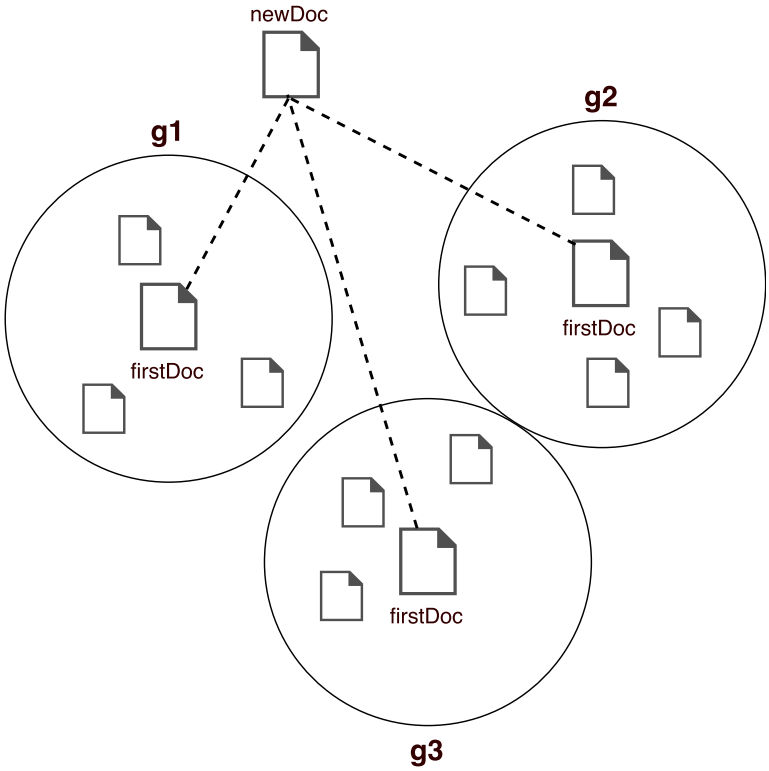
Giả sử ta có một số document đã được gom nhóm sẵn thành 3 cụm như hình, ngưỡng MergeThreshold = 0.7, mỗi cụm có một firstDoc làm đại diện. Đường tròn thể hiện ngưỡng khoảng cách để một document được xem là first story hay không.

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

2.5.2.3 Mã giả

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

luồng bài viết từ Twitter, giá trị ngưỡng MergeThreshold 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 $Score_d[g] \leftarrow similarity(d, g.firstDoc)$ $MaxScore_d < Score_d[g]$ $MaxScore_d \leftarrow Score_d[g]$ $IDMaxScore_d \leftarrow g.groupID$ $MaxScore_d >$ MergeThreshold gán d vào cụm $IDMaxScore_d$ tạo cụm g_{new} $g_{new}.firstDoc \leftarrow d$

Chú thích thuật toán:

1. $MaxScore_d$: 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:

$$similarity(d_1, d_2) = \sum_{t \in d_1,\ t \in g.topTerms} [tf(t, d_2) \times idf(t) \times boost(t)] \tag{2.3}$$

$$tf(t, d) = \frac{count(t \in d)}{size(d)} \tag{2.4}$$

$$idf(t) = 1 + \log \frac{N}{count(m\ has\ t)} \tag{2.5}$$

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

2.5.2.4 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.

2.5.3 Thuật toán Locality Sensitive Hashing

2.5.3.1 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 [7].

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.

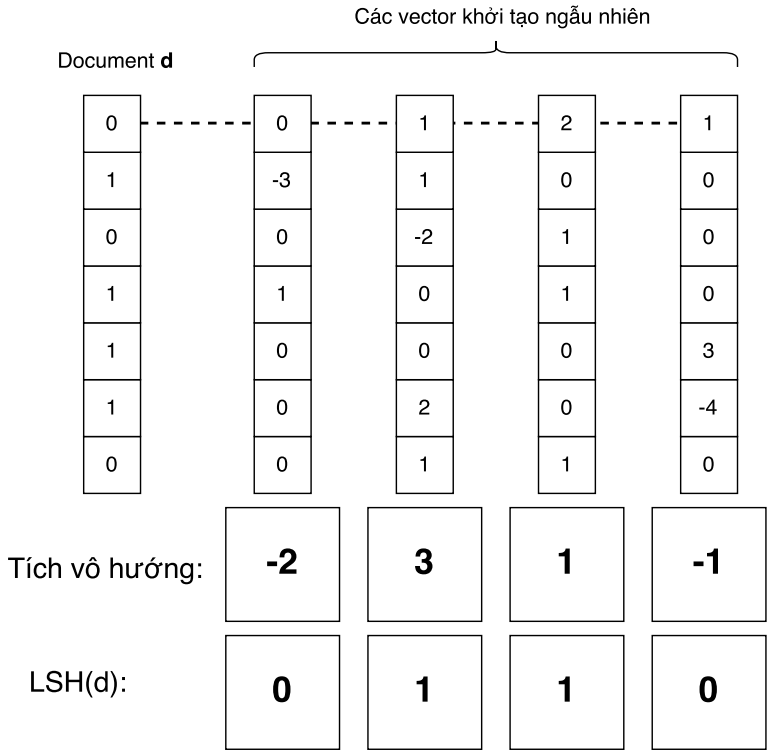
Vì các siêu phẳng được chọn một cách ngẫu nhiên, nên đôi khi các document dù gần nhau vẫn có thể bị phân vào vùng khác nhau. Do đó ta thường dùng cùng lúc nhiều hash table để tăng thêm khả năng tìm được láng giềng gần nhất.

2.5.3.2 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 2.3: Cách tính hash code cho một document trong một hash table

2.5.3.3 Mã giả

Algorithm 3 Phát hiện tin nóng sử dụng Locality Sensitive Hashing

luồng bài viết từ Twitter, giá trị ngưỡng t những cụm bài viếtKhởi 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 $dis_{min}(d) = 1$ **foreach** document d' trong S **do** $distance(d, d') < dis_{min}(d)$ $dis_{min}(d) = distance(d, d')$ $dis_{min}(d) >= t$ tạo cụm mới chứa d thêm d vào cụm chứa hàng xóm gần nhất của d

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.

2.5.3.4 LSH cải tiến

Dựa vào dữ liệu thực nghiệm của mình, Sasa Petrovic và cộng sự [8] 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

luồng bài viết từ Twitter, giá trị ngưỡng t những cum bài viếtKhởi 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 $dis_{min}(d) = 1$ **foreach** document d' trong S **do** $distance(d, d') < dis_{min}(d)$ $dis_{min}(d) = distance(d, d')$ $dis_{min}(d) >= t$ 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. $dis_{min}(d) >= t$ tạo cụm mới chứa d thêm d vào cụm chứa hàng xóm gần nhất của d

2.6 Giới thiệu một số độ đo đánh giá hiệu năng gom cụm

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 [9], 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.

2.6.1 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:

$$interclusterDistance = \frac{1}{size(C) * [size(C) - 1]} \sum_{C_i, C_j \in C} distance(C_i, C_j) \tag{2.6}$$

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 2.4
- $size(C_i)$: số lượng điểm dữ liệu được gom vào cụm i

2.6.2 Độ đo Dunn index

Dunn index được đề xuất bởi J. C. Dunn [10], 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:

$$DI = \frac{min_{1 \le i \le j \le n} distance(C_i, C_j)}{max_{1 \le k \le n} intraclusterDistance(k)} \tag{2.7}$$

Với n là tổng số cụm thu được, và intraclusterDistance được tính theo một trong những cách ở mục 2.6.1.

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 (2.7) 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.

2.6.3 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:

$$s(i) = \frac{b_i - a_i}{max\{a_i, b_i\}} \tag{2.8}$$

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.

2.6.4 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:

$$purity(C_i) = max_j(\frac{m_{ij}}{m_i}) \tag{2.9}$$

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:

$$purity = \sum_{i=1}^{K} \frac{m_i}{m} \, purity(C_i) \tag{2.10}$$

Trong đó:

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

2.7 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 [4].

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:

$$Score(C_i) = \frac{1}{Z} \sum_{j=1}^{k} \frac{S_i}{log_2(\Delta_j + 2)} \tag{2.11}$$

Với S_i tính bằng:

$$S_i = w_1 \sum_{u_i \in C_i} FollowerCount(u_i) + w_2\ TotalRetweetCount(C_i) + w_3\ TotalFavoriteCount(C_i) \tag{2.12}$$

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

2.8 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

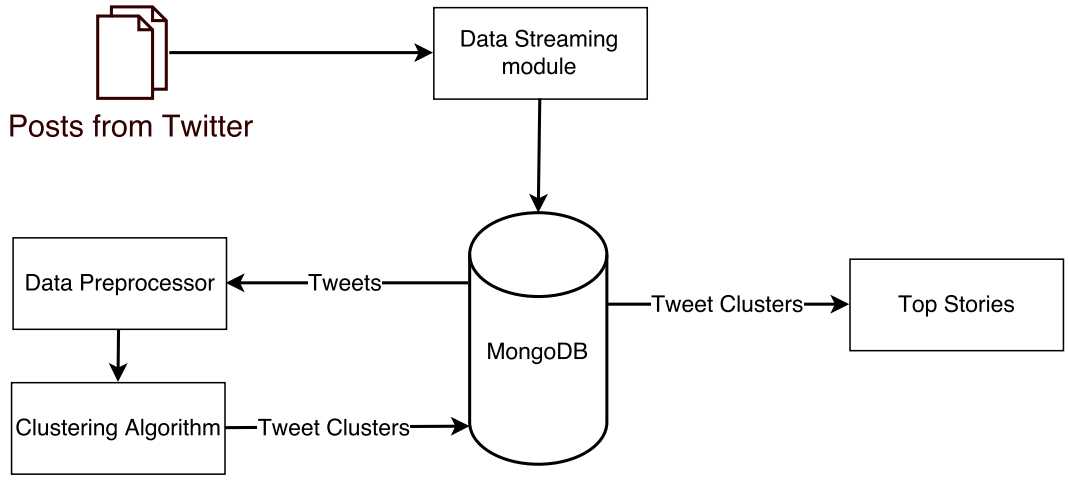
HIỆN THỰC HỆ THỐNG PHÁT HIỆN TIN NÓNG

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

Hệ thống có 2 loại người dùng chính: quản trị viên hệ thống (admin) và biên tập viên. Quản trị viên có nhiệm vụ theo dõi hệ thống, điều khiển bật tắt, thay đổi thông

số của các tác vụ trong hệ thống. Biên tập viên sử dụng hệ thống thông qua giao diện Spam Filtering để xem các sự kiện phát hiện được và các thông tin liên quan.

3.3 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

3.4 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.

3.5 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

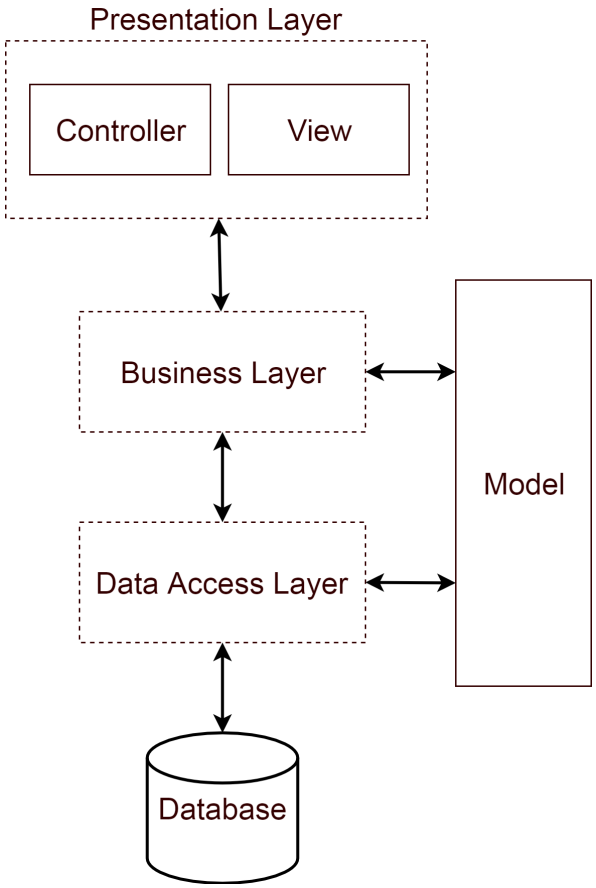
Sau khi đã phân loại, hệ thống sẽ lưu nhãn đã gán cho tin tức xuống cơ sở dữ liệu MongoDB.

3.6 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

3.7 Thiết kế hệ thống



Hình 3.2: Kiến trúc hệ thống 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.
- 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.

3.8 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

3.8.1 Các package

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

- 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

3.8.2 Cơ sở dữ liệu MongoDB

Hệ thống sử dụng MongoDB để lưu trữ dữ liệu tin tức và quản lý kết quả gom cụm. 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

3.8.2.1 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

3.8.2.2 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

3.8.3 Cơ sở dữ liệu MySQL

MySQL được dùng để lưu trữ các dữ liệu khác của hệ thống như dữ liệu người dùng, các từ khóa dùng cho việc streaming dữ liệu Twitter,...

3.8.3.1 Bảng User

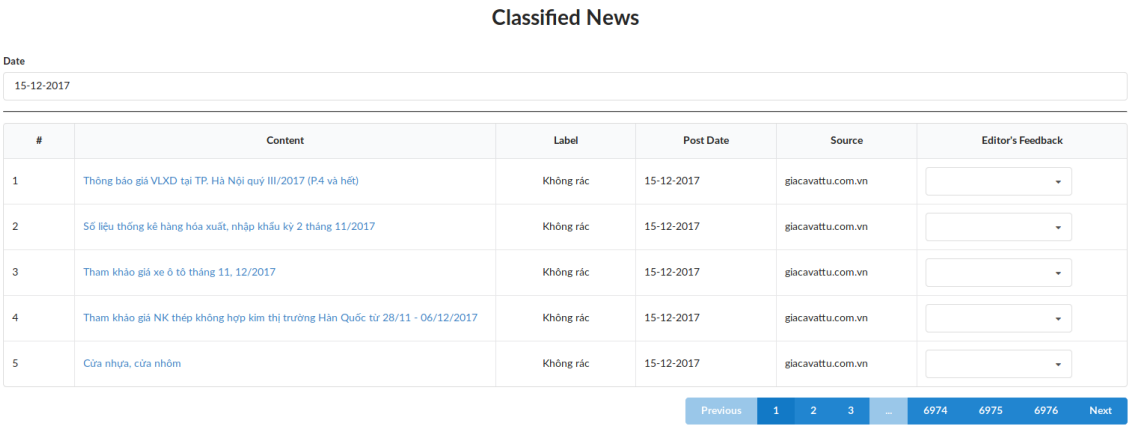
Bảng này dùng để quản lý danh sách người dùng hệ thống.

Thuộc tính	Loại	Ý nghĩa
ID	int(11)	ID của người dùng
Username	varchar(100)	Tên tài khoản
Password	Long	Mật khẩu
Email	varchar(100)	Địa chỉ email người dùng
Fullname	varchar(100)	Họ tên đầy đủ của người dùng
Phone	varchar(20)	Số điện thoại
UserTypeId	int(11)	ID phân loại người dùng

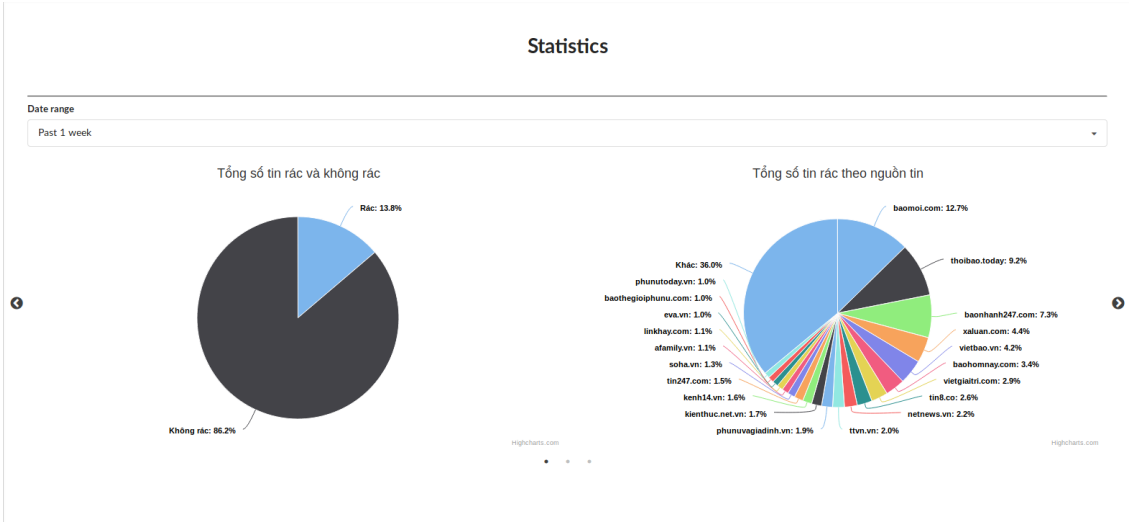
Bảng 3.4: Bảng User

3.9 Kết quả

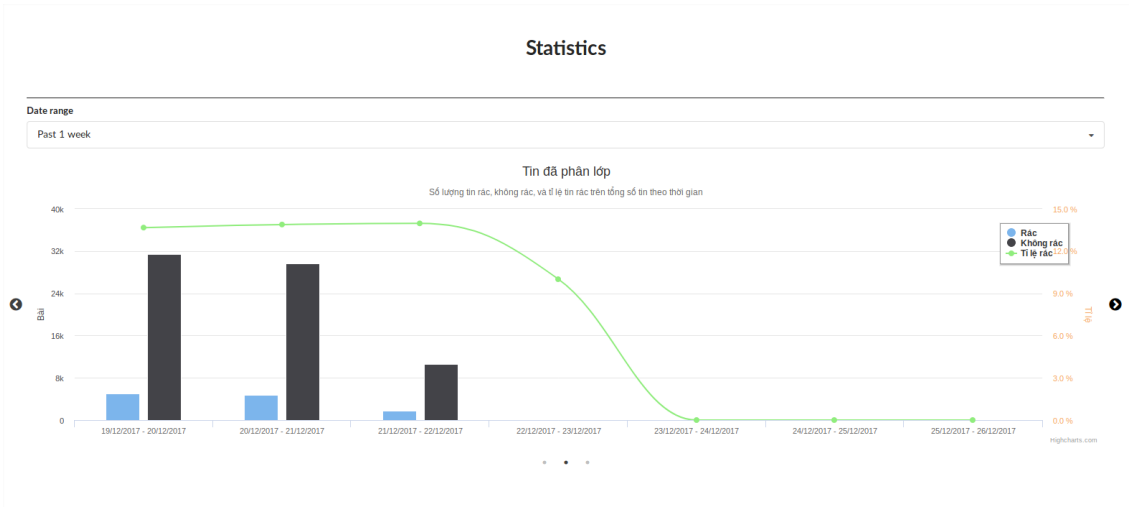
Hệ thống đã hoàn thiện các chức năng: Thu thập dữ liệu từ Twitter, điều khiển chạy luồng của thuật toán phát hiện tin nóng, và hiển thị kết quả cho biên tập viên. Dưới đây là một số hình ảnh của hệ thống:



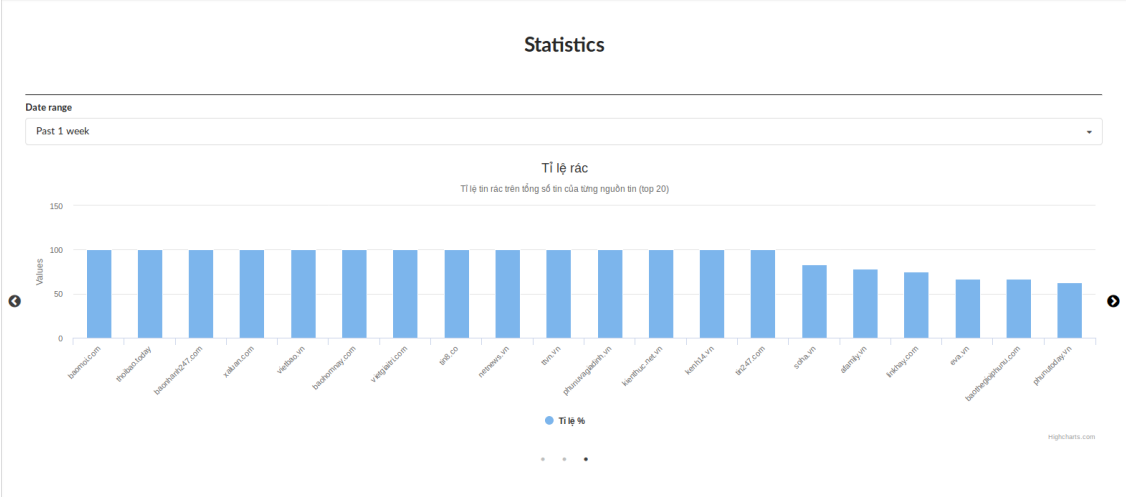
Hình 3.3: Giao diện danh sách tin đã phân lớp



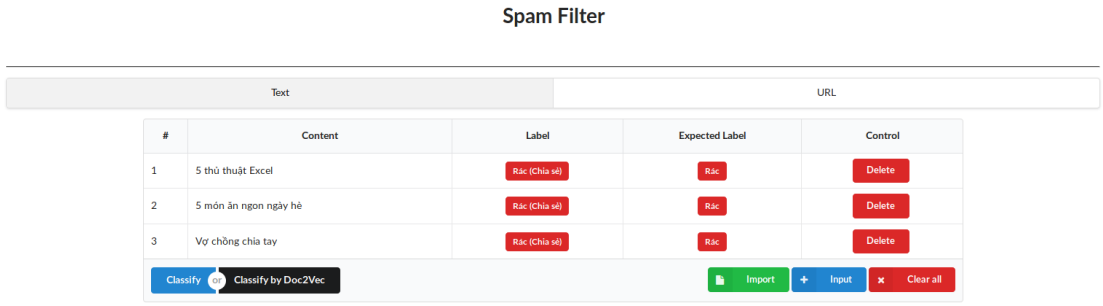
Hình 3.4: Giao diện thống kê biểu đồ thứ 1 và thứ 2



Hình 3.5: Giao diện thống kê biểu đồ thứ 3



Hình 3.6: Giao diện thống kê biểu đồ thứ 4



Hình 3.7: Giao diện bộ phân lớp tin tức

3.10 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 kết quả huấn luyện các thuật toán phân lớp 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 các trang báo điện tử Việt Nam. Dữ liệu được lấy về từ cơ sở dữ liệu tin tức của công ty VCCorp. Dữ liệu thu thập được được sàn lọc thông qua một tập các từ khóa đã được định nghĩa trước. [11].

Mỗi mẫu tin bao gồm các thông tin như sau: tựa đề tin, nội dung tin, mô tả tin, ngày đăng tin, và nguồn tin.

Bộ dữ liệu huấn luyện gồm 15,612 tin tiếng Việt, được thu thập và sàn lọc trong khoảng thời gian từ.

Chia sẻ	thủ thuật, mẹo, tình cảm, chia tay, món ăn, ngon, tâm sự
Tuyển dụng	việc làm, tuyển dụng, cộng tác viên, lao động
Quảng cáo	giảm giá, khuyến mãi, trúng thưởng

Bảng 4.1: 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=0B6FWPU7hs7CIRUZudGl3bG9DbEk

4.3 Thiết lập thực nghiệm, cách đánh giá

Sử dụng 3 thuật toán: Naive Bayes, SVM, J48.

Ta đánh giá và so sánh kết quả về thời gian xử lý và chất lượng cụm thông qua một số độ đo đã trình bày ở mục 2.6: intracluster distance, intercluster distance.

Cả 4 thuật toán đều sử dụng giá trị ngưỡng để xét điểm dữ liệu có thuộc cụm hay không. Thuật toán Nearest Neighbor và Boost Named Entity sử dụng ngưỡng về *độ tương đồng* MergeThreshold, ngược lại thuật toán LSH sử dụng ngưỡng *khoảng cách* NoveltyThreshold. Thực chất $NoveltyThreshold = 1 - MergeThreshold$ nên ta vẫn có thể quy đổi giá trị giữa chúng để so sánh kết quả các thuật toán. Tất cả giá trị ngưỡng trong chương này sẽ được quy đổi về giá trị của **MergeThreshold** tương ứng.

Riêng thuật toán LSH có thêm các thông số như: số hashtable, số siêu phẳng, lượng document tối đa cho mỗi bucket,... ta sẽ cần xét sự ảnh hưởng của chúng đến quá trình và kết quả gom cụm.

4.4 Kết quả thực nghiệm

4.4.1 Thay đổi MergeThreshold

Dưới đây là kết quả các thuật toán khi thay đổi giá trị MergeThreshold từ 0.1 đến 0.9:

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	100	3.453057	0.483357	0.996435	279,439
Boost Named Entity	134	4.507409	0.847914	1	3,462,536
LSH	510	2.609871	0.255654	0.996635	30,704
LSH+	510	2.609871	0.255654	0.996635	29,687

Bảng 4.2: MergeThreshold = 0.1

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	1358	2.640104	0.239420	0.995582	239,357
Boost Named Entity	134	4.507409	0.847914	1	3,157,517
LSH	2545	2.431249	0.150437	0.994624	30,274
LSH+	2510	2.434242	0.151118	0.994679	30,629

Bảng 4.3: MergeThreshold = 0.2

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	3097	2.418943	0.128281	0.993779	242,266
Boost Named Entity	135	4.523543	0.844617	0.999963	4,240,763
LSH	3955	2.340955	0.091359	0.992901	29,495
LSH+	3925	2.342387	0.092103	0.992924	30,394

Bảng 4.4: MergeThreshold = 0.3

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	4184	2.341865	0.079299	0.992050	241,671
Boost Named Entity	143	4.65682	0.845813	0.999771	6,031,529
LSH	4734	2.309055	0.062843	0.991542	32,644
LSH+	4710	2.309963	0.063133	0.991563	32,858

Bảng 4.5: MergeThreshold = 0.4

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	4778	2.308508	0.051345	0.991100	270,376
Boost Named Entity	164	4.330696	0.832134	0.999370	8,562,791
LSH	5171	2.290040	0.043537	0.990804	29,906
LSH+	5155	2.290858	0.043968	0.990812	31,688

Bảng 4.6: MergeThreshold = 0.5

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	5153	2.291737	0.034421	0.990652	269,129
Boost Named Entity	192	4.195825	0.820760	0.999014	10,740,287
LSH	5468	2.279607	0.030737	0.990459	29,418
LSH+	5455	2.280260	0.031068	0.990461	34,689

Bảng 4.7: MergeThreshold = 0.6

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	5499	2.281347	0.022001	0.990179	255,027
Boost Named Entity	232	4.042305	0.775990	0.998580	14,239,743
LSH	5759	2.271498	0.020295	0.990157	30,494
LSH+	5751	2.271919	0.020370	0.990165	32,434

Bảng 4.8: MergeThreshold = 0.7

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	5884	2.269258	0.010429	0.989852	260,766
Boost Named Entity	291	3.844293	0.750010	0.998067	17,408,783
LSH	6096	2.261388	0.009540	0.989862	28,664
LSH+	6090	2.261666	0.009558	0.989866	31,418

Bảng 4.9: MergeThreshold = 0.8

Thuật toán	Số cụm	Entropy	Intracluster distance	Intercluster distance	Thời gian (ms)
KNN	6296	2.263796	0.003356	0.989604	250,952
Boost Named Entity	368	3.640344	0.704845	0.997461	23,017,213
LSH	6459	2.257797	0.003221	0.989643	33,374
LSH+	6454	2.258063	0.003224	0.989642	30,236

Bảng 4.10: MergeThreshold = 0.9

4.4.2 Thay đổi các thông số của thuật toán LSH

Lấy MergeThreshold = 0.5, ta thử nghiệm với số lượng hash table, số lượng siêu phẳng thay đổi, kết quả thử nghiệm được trình bày trong Bảng 4.11.

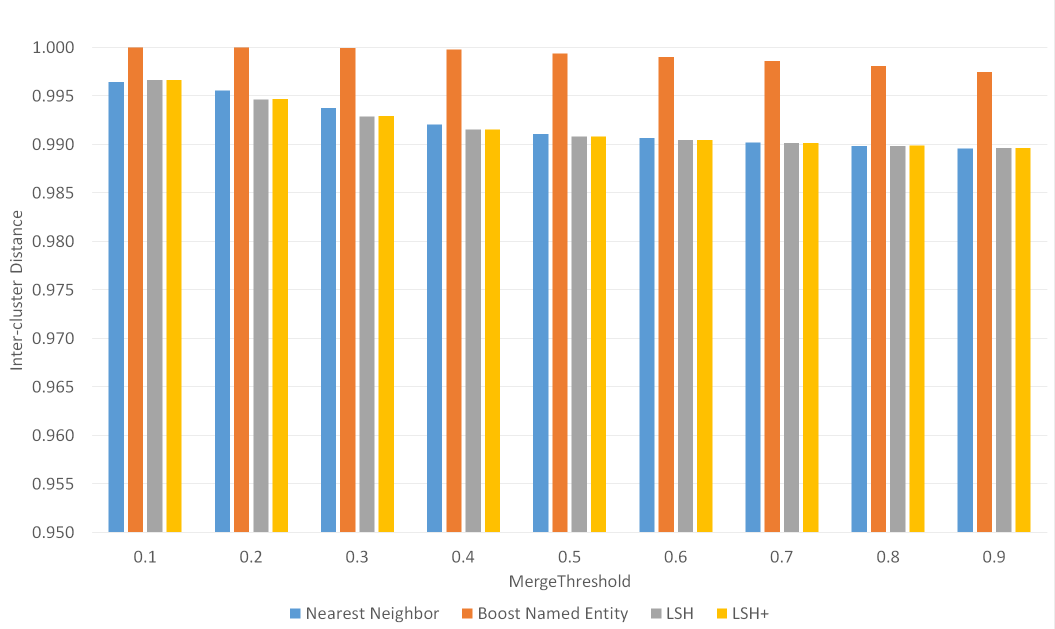
HashTable	Số siêu phẳng	Số cụm	Entropy	Local Distance	Global Distance	Thời gian (ms)
5	50	5,547	2.284131	0.038684	0.990546	9,977
5	100	5,478	2.284564	0.039357	0.990575	12,973
5	200	5,366	2.285870	0.040672	0.990598	11,950
5	300	5,278	2.288058	0.041977	0.990631	9,280
10	50	5,362	2.285823	0.040393	0.990733	19,656
10	100	5,378	2.285774	0.039731	0.990582	20,377
10	200	5,216	2.287042	0.041284	0.990638	21,433
10	300	5,095	2.290325	0.042983	0.990700	22,510
15	50	5,400	2.284835	0.039679	0.990746	24,548
15	100	5,269	2.287836	0.041994	0.990762	25,364
15	200	5,116	2.291248	0.043725	0.990790	23,976
15	300	4,992	2.295168	0.045385	0.990841	23,297
20	50	5,362	2.285823	0.040393	0.990733	31,487
20	100	5,155	2.290858	0.043968	0.990812	30,706
20	200	5,019	2.295531	0.045570	0.990868	33,559
20	300	4,925	2.300117	0.047161	0.990895	34,167

Bảng 4.11: Kết quả thử nghiệm LSH+ khi thay đổi số hash table và số siêu phẳng

4.5 Nhận xét

4.5.1 Nhận định về MergeThreshold

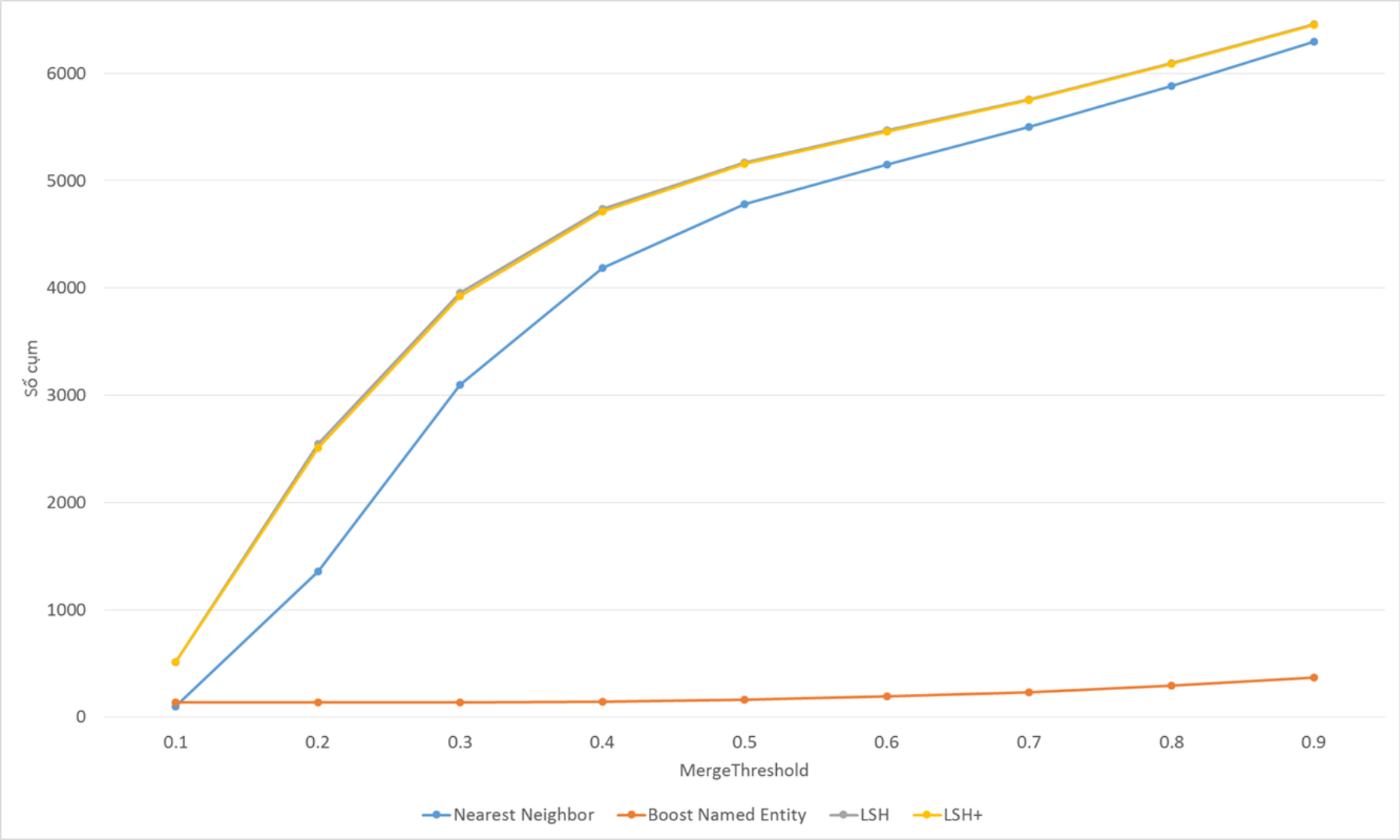
Dựa vào Hình 4.1 và bản chất của giá trị MergeThreshold, ta thấy số cụm tỷ lệ thuận với ngưỡng này, do khi ngưỡng càng cao thì khả năng một document vào chung cụm với document khác càng thấp, dẫn đến sinh ra nhiều cụm nhỏ lẻ hơn. Ngưỡng tăng cũng dẫn đến giá trị intracluster distance và intercluster distance giảm, vì cụm ít phần tử dẫn đến khoảng cách cục bộ nhỏ, và số lượng cụm nhiều lại dẫn đến khoảng cách toàn cục giảm.



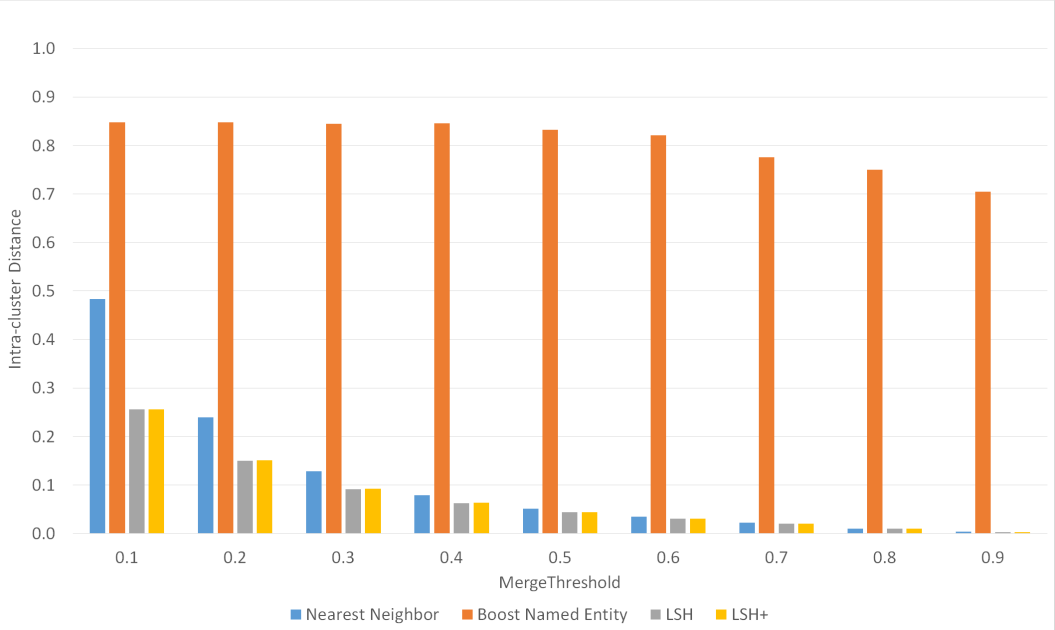
Hình 4.3: Ảnh hưởng của giá trị threshold đến intercluster distance

Hình 4.4 cho thấy đối với cùng số lượng mẫu trong dữ liệu, hai thuật toán LSH xử lý nhanh nhất, kế đến là Nearest Neighbor và cuối cùng là thuật toán Boost Named Entity. Giá trị của MergeThreshold không có ảnh hưởng rõ rệt đến thời gian chạy của thuật toán Nearest Neighbor và LSH, tuy nhiên thuật toán Boost Named Entity thì lại tăng đáng kể.

Lý do chính thuật toán Boost Named Entity có thời gian xử lý lâu như vậy (gấp 10-600 lần các thuật toán còn lại) là vì thư viện nhận diện thực thể tiếng Việt được sử dụng có thời gian xử lý khá chậm, chiếm phần lớn thời gian thực thi của thuật toán này.



Hình 4.1: Ảnh hưởng của giá trị threshold đến số cụm



Hình 4.2: Ảnh hưởng của giá trị threshold đến intracluster distance

4.5.2 Nhận định về các tham số của LSH

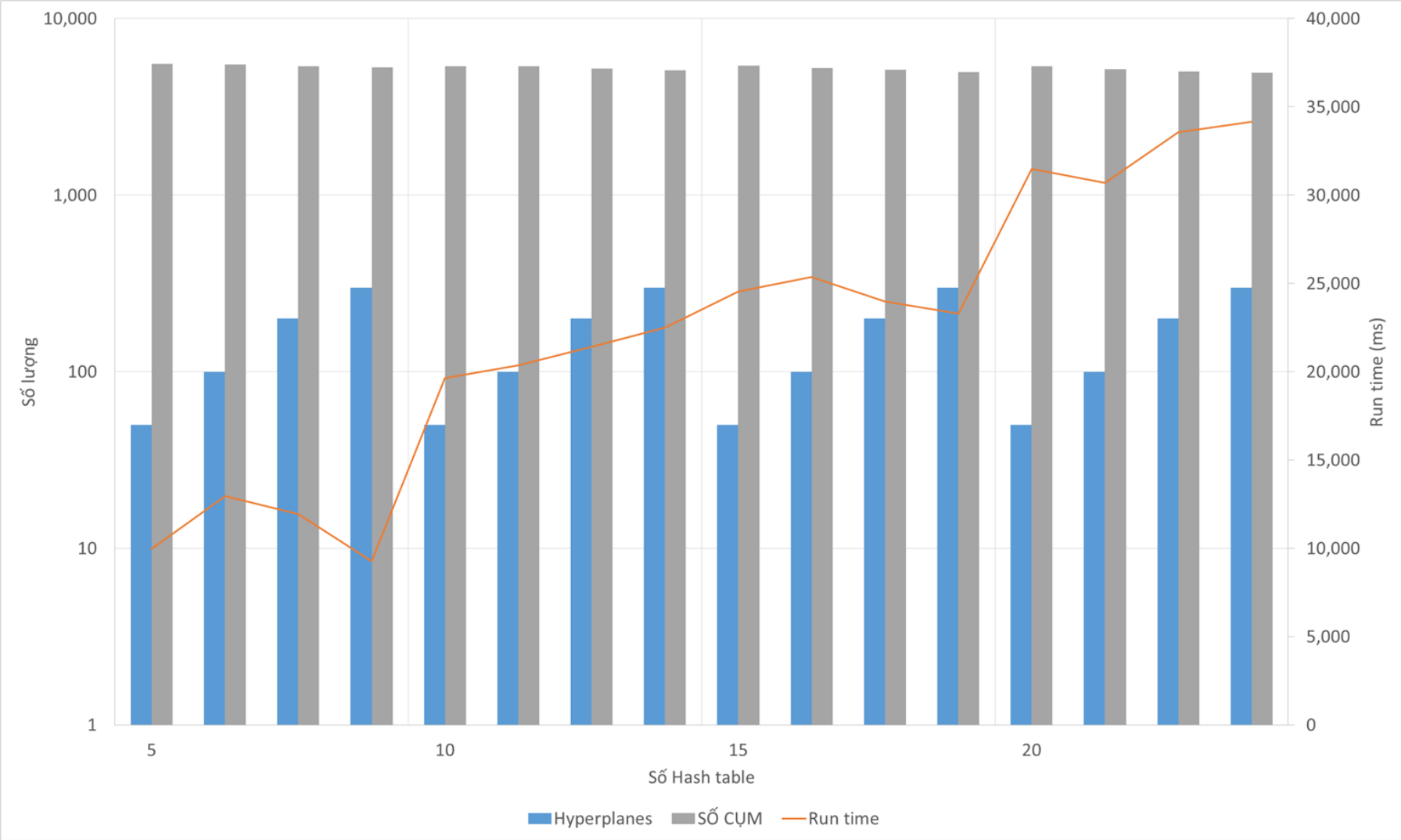
Thử nghiệm với thuật toán LSH cải tiến, theo Bảng 4.11 và Hình 4.5, với giá trị MergeThreshold cố định bằng 0.5, số hash table lẫn số siêu phẳng đều tỷ lệ nghịch với số cụm thu được, tuy nhiên chúng không làm số cụm thay đổi nhiều như giá trị MergeThreshold.

Về mặt thời gian xử lý, số lượng hash table tăng giảm làm thay đổi trực tiếp thời gian thực thi thuật toán. Trong khi đó, số siêu phẳng hầu như không gây ảnh hưởng nhiều.

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 tốc độ, số lượng và chất lượng cụm kết quả. Ta thấy thuật toán LSH có tốc độ xử lý nhanh trong khi vẫn giữ được độ chính xác tốt so với thuật toán Nearest Neighbor. Riêng thuật toán Boost Named Entity do hạn chế bởi thư viện nhận diện thực thể nên thời gian thực thi lẫn chất lượng kết quả gom cụm chưa được tốt.

Hình 4.4: Ảnh hưởng của giá trị threshold đến thời gian thực thi



Hình 4.5: Ảnh hưởng của các tham số đến thời gian thực thi và số cụm của LSH

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] 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. 8

[4] 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. 8, 13, 22

[5] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 9

[6] 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. 10

[7] 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. 16

[8] 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. 18

[9] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson Education, Limited, 2014. 19

[10] 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. 20

[11] Tdt 2004: Annotation manual. https://catalog.ldc.upenn.edu/docs/LDC2006T19/TDT2004V1.2.pdf, 2004. Accessed: 2016-12-6. 34

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);
   }
}
```