**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**HỌ VÀ TÊN: ĐẶNG ĐỨC TRUNG**

**KHÓA: KHÓA 14**

**HỆ ĐÀO TẠO:** **KỸ SƯ DÂN SỰ**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**CHUYÊN NGÀNH: CÔNG NGHỆ DỮ LIỆU**

**ĐỀ TÀI: NGHIÊN CỨU CÁC PHƯƠNG PHÁP BIỂU DIỄN VĂN BẢN TIẾNG VIỆT VÀ ỨNG DỤNG TRONG BÀI TOÁN SO KHỚP NGỮ NGHĨA**

**NĂM 2021**

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**HỌ VÀ TÊN: ĐẶNG ĐỨC TRUNG**

**KHÓA: KHÓA 14**

**HỆ ĐÀO TẠO: DÂN SỰ**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH: CÔNG NGHỆ DỮ LIỆU**

**MÃ SỐ: 52480201**

**ĐỀ TÀI: NGHIÊN CỨU CÁC PHƯƠNG PHÁP BIỂU DIỄN VĂN BẢN TIẾNG VIỆT VÀ ỨNG DỤNG TRONG BÀI TOÁN SO KHỚP NGỮ NGHĨA**

***Cán bộ hướng dẫn khoa học: Thiếu tá, GV, ThS. Vi Bảo Ngọc***

**NĂM 2021**

|  |  |
| --- | --- |
| **HỌC VIỆN KỸ THUẬT QUÂN SỰ** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA CÔNG NGHỆ THÔNG TIN** | **Độc lập - Tự do - Hạnh phúc** |
| **BỘ MÔN AN TOÀN THÔNG TIN** |  |

**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Họ và tên: Đặng Đức Trung Lớp: Công nghệ dữ liệu Khóa: 14

Ngành: Công nghệ thông tin Chuyên ngành: Công nghệ dữ liệu

1. Tên đề tài: Nghiên cứu các phương pháp biểu diễn văn bản tiếng Việt và ứng dụng trong bài toán so khớp ngữ nghĩa.

2. Các số liệu ban đầu:

* Nghiên cứu tổng quan về mô hình Word2Vec, Paragraph Vector, SIF, Transformer, BERT, PhoBERT.
* Nghiên cứu tổng quan về bài toán tách từ và các công cụ hỗ trợ.
* Nghiên cứu tổng quan về bài toán so khớp ngữ nghĩa.
* Bộ ngữ liệu tin tức của Vương Quốc Bình.
* Nghiên cứu về cách sử dụng Elasticsearch để lưu trữ và tìm kiếm văn bản theo biểu diễn vector.

3. Nội dung bản thuyết minh:

* Mục lục
* Danh mục từ viết tắt
* Danh mục hình ảnh
* Danh mục bảng biểu
* Lời nói đầu
* Chương 1: Tổng quan về biểu diễn văn bản
* Chương 2: Các phương pháp biểu diễn văn bản tiếng Việt và ứng dụng trong bài toán so khớp ngữ nghĩa
* Chương 3: Thiết kế và cài đặt hê thống hỗ trợ tìm kiếm văn bản theo ngữ nghĩa
* Chương 4: Thí nghiệm và nhận xét
* Kết luận
* Tài liệu tham khảo

4. Số lượng, nội dung các bản vẽ:

5. Cán bộ hướng dẫn: Thiếu tá, GV, ThS. Vi Bảo Ngọc

Ngày giao: .........../........../............. Ngày hoàn thành: .........../........../.............

*Hà Nội, ngày ........ tháng ........ năm ........*

|  |  |
| --- | --- |
| **Chủ nhiệm bộ môn** | **Cán bộ hướng dẫn** |
|  | (Ký, ghi rõ họ tên, học hàm, học vị) |

                                                                **Học viên thực hiện**

                                   Đã hoàn thành và nộp đồ án ngày … tháng …. năm ……

                                                               (Ký và ghi rõ họ tên)

**DANH MỤC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| **BERT** | Bidirectional Encoder Representations from Transformers | Bộ mã hóa biểu diễn hai chiều từ mô hình Transformers |
| **BOW** | Bag-of-words |  |
| **CBOW** | Continuous Bag-of-words |  |
| **CSDL** | Database | Cơ sở dữ liệu |
| **Doc2Vec** | Document to vector | Biến đổi văn bản sang vector |
| **NoSQL** | Not SQL / Not only SQL | Cơ sở dữ liệu phi cấu trúc |
| **PV-DM** | Paragraph vector – distributed memory |  |
| **SCRDR** | Single Classification Ripple Down Rules |  |
| **SIF** | Smooth Inversed Frequency |  |
| **SOTA** | State-of-the-art | Tân tiến nhất |
| **SVD** | Singular Value Decomposition |  |
| **TF-IDF** | Term Frequency – Inversed Document Frequency | Tần suất của từ - Nghịch đảo tần suất từ trong văn bản |
| **Word2Vec** | Word to vector | Biến đổi từ sang vector |

**DANH MỤC HÌNH ẢNH**

|  |  |
| --- | --- |
| [Hình 1.1: Đồ thị của một câu 3](#_Toc61393103)  [Hình 1.2: Sơ đồ phương pháp tách từ kết hợp cây nhị phân SCRDR 6](#_Toc61393104)  [Hình 1.3: Cây khởi tạo SCRDR 8](#_Toc61393105)  [Hình 2.1: Mô hình CBOW và Skip-gram 16](#_Toc61393106)  [Hình 2.2: Mô hình Transformer 21](#_Toc61393107)  [Hình 2.3: Mô hình Masked LM 26](#_Toc61393108)  [Hình 2.4: Chuẩn bị đầu vào cho NPS 27](#_Toc61393109)  [Hình 2.5: Mô hình PV-DM 30](#_Toc61393110)  [Hình 2.6: Mô hình PV-DBOW 30](#_Toc61393111)  [Hình 3.1: Thiết kế tổng quan của hệ thống 39](#_Toc61393112)  [Hình 3.2: Thiết kế cơ sở dữ liệu SQL mô phỏng 40](#_Toc61393113)  [Hình 3.3: Giao diện màn hình chính của hệ thống 44](#_Toc61393114)  [Hình 3.4: Giao diện màn hình kết quả tìm kiếm 44](#_Toc61393115)  [Hình 3.5: Giao diện chi tiết của kết quả tìm kiếm 45](file:///W:\WORK\Python\sent_emb_api_git\sentence_embedding_api\Đặng%20Đức%20Trung%20-%20Đồ%20án%20tốt%20nghiệp%20đại%20học%20-%20DocEmbSim.docx#_Toc61393116)  [Hình 3.6: Giao diện tiếp tục tìm kiếm theo một kết quả 45](#_Toc61393117) |  |

**DANH MỤC BẢNG BIỂU**

|  |  |
| --- | --- |
| [Bảng 1.1: Mô tả về các kho ngữ liệu 6](#_Toc61393062)  [Bảng 1.2: Ví dụ về từ điển ngữ cảnh 5 âm tiết được xây dựng 7](#_Toc61393063)  [Bảng 1.3: Mẫu qui tắc 7](#_Toc61393064)  [Bảng 2.1: Ví dụ về ma trận count vector 13](#_Toc61393065)  [Bảng 2.2: Bảng số lượng văn bản thuộc D chứa từ/cụm từ 14](#_Toc61393066)  [Bảng 2.3: Bảng trình bày biểu diễn văn bản sử dụng TF-IDF cho tập D 15](#_Toc61393067)  [Bảng 2.4: Bảng tần suất xuất hiện của các từ trong từng câu thuộc D 29](#_Toc61393068)  [Bảng 3.1: Bảng NewsCategory 40](#_Toc61393069)  [Bảng 3.2: Bảng News 41](#_Toc61393070)  [Bảng 3.3: Ý nghĩa các trường trong mappings 47](#_Toc61393071)  [Bảng 4.1: Tham số fine-tune mô hình PhoBERT 56](#_Toc61393072)  [Bảng 4.2: Kết quả đánh giá các mô hình với d=100, epochs=10, threshold=0.7 59](#_Toc61393073)  [Bảng 4.3: Kết quả đánh giá các mô hình với d=100, epochs=10, threshold=0.6 59](#_Toc61393074)  [Bảng 4.4: Kết quả đánh giá các mô hình với d=300, epochs=10, threshold=0.7 59](#_Toc61393075)  [Bảng 4.5: Kết quả đánh giá các mô hình với d=300, epochs=10, threshold=0.6 60](#_Toc61393076)  [Bảng 4.6: Thời gian huấn luyện và biến đổi 61](#_Toc61393077) |  |

MỤC LỤC

[**LỜI NÓI ĐẦU 1**](#_Toc61438756)

[**1. Ý nghĩa thực tiễn của đề tài 1**](#_Toc61438757)

[**2. Mục đích của đề tài 1**](#_Toc61438758)

[**3. Đối tượng nghiên cứu của đồ án 2**](#_Toc61438759)

[**4. Nhiệm vụ nghiên cứu 2**](#_Toc61438760)

[**5. Phương pháp nghiên cứu 2**](#_Toc61438761)

[**CHƯƠNG 1: TỔNG QUAN VỀ BIỂU DIỄN VĂN BẢN 4**](#_Toc61438762)

[**1.1. Tổng quan bài toán biến đổi biểu diễn văn bản 4**](#_Toc61438763)

[**1.1.1. Khái niệm về văn bản 4**](#_Toc61438764)

[**1.1.2. Khái niệm văn bản trong xử lý ngôn ngữ tự nhiên 4**](#_Toc61438765)

[**1.1.3. Khái niệm về biến đổi biểu diễn từ 4**](#_Toc61438766)

[**1.1.4. Khái niệm về biến đổi biểu diễn văn bản 4**](#_Toc61438767)

[**1.2. Các phương pháp tiếp cận bài toán biến đổi biểu diễn văn bản 5**](#_Toc61438768)

[**1.2.1. Tiền xử lý văn bản 5**](#_Toc61438769)

[**1.2.2. Phương pháp tiếp cận dựa vào tần suất (frequency-based) 13**](#_Toc61438770)

[**1.2.3. Phương pháp tiếp cận dựa vào dự đoán (prediction-based) 13**](#_Toc61438771)

[**1.3. Kết luận chương 1 14**](#_Toc61438772)

[**CHƯƠNG 2: CÁC PHƯƠNG PHÁP BIỂU DIỄN VĂN BẢN TIẾNG VIỆT VÀ ỨNG DỤNG TRONG BÀI TOÁN SO KHỚP NGỮ NGHĨA 15**](#_Toc61438773)

[**2.1. Các phương pháp biểu diễn từ 15**](#_Toc61438774)

[**2.1.1. One-hot encoding (Binary Encoding – mã hóa nhị phân) 15**](#_Toc61438775)

[**2.1.2. Count vector 15**](#_Toc61438776)

[**2.1.3. Mô hình Term Frequency – Inversed Document Frequency (TF-IDF) 16**](#_Toc61438777)

[**2.1.4. Mô hình Word-To-Vector (Word2Vec) 18**](#_Toc61438778)

[**2.1.5. Mô hình FastText 23**](#_Toc61438779)

[**2.1.6. Mô hình Transformer 24**](#_Toc61438780)

[**2.1.7. Mô hình Bidirectional Encoder Representations from Transformers (BERT) 29**](#_Toc61438781)

[**2.1.8. Mô hình ngôn ngữ tiếng Việt PhoBERT (Phở-BERT) 31**](#_Toc61438782)

[**2.2. Các phương pháp biểu diễn văn bản 31**](#_Toc61438783)

[**2.2.1. Mô hình Bag-Of-Words (BOW) 32**](#_Toc61438784)

[**2.2.2. Biểu diễn văn bản sử dụng vector TF-IDF 33**](#_Toc61438785)

[**2.2.3. Mô hình Document-To-Vector (Doc2Vec – paragraph vector) 33**](#_Toc61438786)

[**2.2.4. Biểu diễn văn bản dựa trên các vector biểu diễn từ 35**](#_Toc61438787)

[**2.2.5. Mô hình Smooth Inversed Frequency (SIF) 36**](#_Toc61438788)

[**2.3. Ứng dụng trong bài toán so khớp ngữ nghĩa 39**](#_Toc61438789)

[**2.4. Kết luận chương 2 41**](#_Toc61438790)

[**CHƯƠNG 3: THIẾT KẾ VÀ CÀI ĐẶT HỆ THỐNG HỖ TRỢ TÌM KIẾM VĂN BẢN THEO NGỮ NGHĨA 42**](#_Toc61438791)

[**3.1. Chuẩn bị ngữ liệu 42**](#_Toc61438792)

[**3.2. Huấn luyện mô hình 42**](#_Toc61438793)

[**3.3. Thiết kế tổng quan của hệ thống 44**](#_Toc61438794)

[**3.3.1. Thiết kế tổng quan 44**](#_Toc61438795)

[**3.3.2. Module dữ liệu 45**](#_Toc61438796)

[**3.3.3. Module REST API truy vấn 48**](#_Toc61438797)

[**3.3.4. Giao diện người dùng 50**](#_Toc61438798)

[**3.4. Cài đặt hệ thống 51**](#_Toc61438799)

[**3.5. Kết luận chương 3 63**](#_Toc61438800)

[**CHƯƠNG 4: THÍ NGHIỆM VÀ NHẬN XÉT 64**](#_Toc61438801)

[**4.1. Thí nghiệm 64**](#_Toc61438802)

[**4.1.1. Các độ đo sử dụng để đánh giá các mô hình 66**](#_Toc61438803)

[**4.1.2. Đánh giá kết quả trên bộ dữ liệu cặp câu tương đồng vnPara 68**](#_Toc61438804)

[**4.2. Đánh giá hiệu năng của các phương pháp trên bộ dữ liệu vnPara 69**](#_Toc61438805)

[**4.3. Nhận xét 70**](#_Toc61438806)

[**4.4. Kết luận chương 4 70**](#_Toc61438807)

[**KẾT LUẬN 72**](#_Toc61438808)

[**TÀI LIỆU THAM KHẢO 73**](#_Toc61438809)

# LỜI NÓI ĐẦU

## **1. Ý nghĩa thực tiễn của đề tài**

Ngày nay, máy tính hỗ trợ con người thực hiện nhiều công việc liên quan đến tính toán, thống kê và so sánh. Đối với máy tính, mọi biểu diễn dưới bất kỳ ký tự nào đều được quy đổi về dạng số để thực hiện phép toán so sánh. Con người có thể sử dụng các loại ngôn ngữ khác nhau, các phương thức biểu đạt khác nhau, v.v... để mô tả về cùng một sự vật, hiện tượng. Hai câu phát biểu có thể khác nhau về hình thức nhưng về mặt ngữ nghĩa là tương đương nhau. Đối với tác vụ tìm kiếm, máy tính không thể hiểu được các ý nghĩa trừu tượng của văn bản, do đó, máy tính sẽ coi hai câu có ý nghĩa giống nhau nhưng sử dụng ngôn từ, ngôn ngữ khác nhau là khác nhau.

Hầu hết các công cụ tìm kiếm trong các phần mềm, ứng dụng sẽ tìm kiếm theo từ khóa hoặc sự xuất hiện của từ khóa trong các tài liệu, và phần lớn sẽ thực hiện lọc tài liệu theo tiêu chí. Tuy nhiên, vào một số thời điểm tìm kiếm, người dùng mong muốn thu được kết quả bao gồm các văn bản có thể đã thực hiện trích dẫn không chứa từ khóa – biến đổi câu từ (paraphrasing) nhưng biểu đạt chung ý nghĩa. Ví dụ như người dùng muốn tìm kiếm “tổng thống Nga gặp mặt tổng thống Mỹ” sẽ có thể tìm được một tiêu đề như “Putin đến thăm Donald Trump”.

Vì vậy việc biến đổi biểu diễn văn bản từ dạng ký tự mang ý nghĩa trừu tượng về dạng số là cần thiết để máy tính có thể thực hiện so sánh. Khi đó, để có thể tìm kiếm văn bản theo ngữ nghĩa, máy tính sẽ thực hiện so sánh giữa biểu diễn của chuỗi tìm kiếm với các biểu diễn đã lưu lại của các văn bản trong kho lưu trữ và trả về kết quả có mức độ tương đồng cao nhất cho người dùng.

Do đó, em đã chọn đề tài: **Nghiên cứu các phương pháp biểu diễn văn bản tiếng Việt và ứng dụng trong bài toán so khớp ngữ nghĩa**.

## **2. Mục đích của đề tài**

Đề tài tập trung vào việc nghiên cứu, nắm bắt kiến thức về các phương pháp được sử dụng để xây dựng mô hình biến đổi biểu diễn văn bản từ dạng chuỗi ký tự về dạng số hay cụ thể là các vector. Sau đó, kết quả thu được sau huấn luyện sẽ được sử dụng để biến đổi biểu diễn văn bản và thực hiện so sánh độ tương đồng thông qua biểu diễn vector.

* Nghiên cứu làm rõ các phương pháp biểu diễn văn bản.
* Ứng dụng biểu diễn văn bản vào bài toán so khớp ngữ nghĩa.

## **3. Đối tượng nghiên cứu của đồ án**

Nghiên cứu các phương pháp biểu diễn văn bản nổi bật bao gồm các phương pháp cổ điển và tập trung vào một số phương pháp học máy.

Nghiên cứu cách áp dụng các phương pháp biểu diễn văn bản vào bài toán so khớp ngữ nghĩa.

Nghiên cứu các công cụ, công nghệ hỗ trợ lưu trữ và tìm kiếm văn bản theo ngữ nghĩa.

## **4. Nhiệm vụ nghiên cứu**

Nghiên cứu làm chủ và sử dụng được các phương pháp biểu diễn văn bản.

Áp dụng các phương pháp biểu diễn văn bản để lập trình hệ thống tìm kiếm văn bản theo ngữ nghĩa.

## **5. Phương pháp nghiên cứu**

* Phương pháp nghiên cứu tài liệu: tìm hiểu và đọc các tài liệu, bài báo, công trình lớn liên quan tới biểu diễn văn bản nói riêng và các bài toán xử lý ngôn ngữ tự nhiên liên quan nói chung.
* Phương pháp thực hành: thí nghiệm và đánh giá các mô hình biểu diễn trên bộ dữ liệu kiểm thử; lập trình thử nghiệm hệ thống hỗ trợ tìm kiếm theo ngữ nghĩa.

Em xin chân thành gửi lời cảm ơn tới cô Vi Bảo Ngọc, người đã trực tiếp hướng dẫn em hoàn thành đồ án trong thời gian qua. Nếu không có cô định hướng, hướng dẫn các phương pháp, cũng như cung cấp các bài báo, tài liệu tham khảo cùng bộ ngữ liệu kiểm thử thì đồ án khó lòng mà hoàn thiện được.

Xin cảm ơn các thầy cô trong khoa Công Nghệ Thông Tin đã dày công dạy dỗ, truyền đạt những tri thức quý báu trong suốt 5 năm theo học. Em xin cảm ơn thầy Cao Tuấn Anh, chị Nguyễn Hải Anh và anh Nguyễn Huy Liêm đã tận tình giúp đỡ trong công việc và truyền đạt các kinh nghiệm quý giá.

Xin cảm ơn trường Học viện Kỹ thuật Quân sự đã đào tạo, rèn luyện những kiến thức, những kinh nghiệm trong suốt thời gian qua.

Em xin gửi lời cảm ơn tới gia đình và bạn bè, những người đã đồng hành, động viên, hỗ trợ và giúp đỡ rất nhiều trong quá trình học tập cũng như trong thời gian thực hiện đồ án này.

Xin cảm ơn tới các nhà nghiên cứu khoa học, đặc biệt là những nhà nghiên cứu người Việt Nam đã có những công trình quyết định đóng góp không nhỏ tới đồ án này.

Một lần nữa, em xin chân thành cảm ơn!

**Đặng Đức Trung**

# TỔNG QUAN VỀ BIỂU DIỄN VĂN BẢN

* 1. Tổng quan bài toán biến đổi biểu diễn văn bản
     1. Khái niệm về văn bản

Theo nghĩa rộng, văn bản được hiểu là một phương tiện để ghi và truyền đạt thông tin từ chủ thể này đến chủ thể khác bằng một ký hiệu hoặc ngôn ngữ nhất định.

Theo nghĩa hẹp, văn bản được hiểu là các tài liệu, giấy tờ được sử dụng trong hoạt động của các cơ quan đoàn thể, tổ chức xã hội.

Một văn bản có thể được cấu thành từ một hoặc nhiều đoạn văn. Mỗi đoạn văn có thể được cấu thành từ một hoặc nhiều câu. Mỗi một câu được cấu thành bởi một hoặc nhiều từ. Mỗi một từ được cấu thành bởi một hoặc nhiều âm tiết. Tuy nhiên, một số âm tiết đứng một mình không mang ý nghĩa, vì vậy, đơn vị nhỏ nhất của văn bản là từ. Thông thường, ngôn ngữ tự nhiên được sử dụng để biểu diễn (represent) văn bản.

* + 1. Khái niệm văn bản trong xử lý ngôn ngữ tự nhiên

Trong xử lý ngôn ngữ tự nhiên, một văn bản (document) là một chuỗi ký tự hoặc ký hiệu độc lập (distinctive text). Một quyển sách hoặc một bài báo có thể được coi là một văn bản. Với định nghĩa trên, có thể coi rằng một đoạn văn hoặc một câu là một văn bản.

* + 1. Khái niệm về biến đổi biểu diễn từ

Biến đổi biểu diễn từ (word embedding) là phương pháp xây dựng mô hình ngôn ngữ hay kỹ thuật học các đặc trưng và thông thường sử dụng các phép biến đổi toán học với mục đích ánh xạ các từ trong một bộ từ vựng (vocabulary) sang dạng biểu diễn (representation) vector số thực.

* + 1. Khái niệm về biến đổi biểu diễn văn bản

Tương tự với khái niệm biến đổi biểu diễn từ, biến đổi biểu diễn văn bản (document embedding) là phương pháp biến đổi một văn bản từ dạng biểu diễn ngôn ngữ tự nhiên sang dạng biểu diễn vector số thực.

* 1. Các phương pháp tiếp cận bài toán biến đổi biểu diễn văn bản
     1. Tiền xử lý văn bản

Trước khi thực hiện biến đổi biểu diễn từ ngôn ngữ tự nhiên sang dạng vector, các văn bản cần thông qua bước tiền xử lý dữ liệu. Các ngôn ngữ khác nhau sẽ có các bước tiền xử lý khác nhau. Cụ thể, các văn bản tiếng Việt cần được xử lý lần lượt theo các mục sau đây.

* + - 1. Làm sạch dữ liệu

Đầu tiên dữ liệu cần được làm sạch và biến đổi về dạng ngôn ngữ tự nhiên. Các văn bản thu thập trên các Internet thường đi kèm với các thẻ HTML. Đối với ngôn ngữ tự nhiên, các thẻ HTML không có ý nghĩa đối với khả năng đọc hiểu thông thường của con người. Ngoài ra, các ký hiệu đặc biệt như các hình vẽ biểu cảm hay những chuỗi ký tự vô nghĩa cũng cần được loại bỏ để tránh tạo nhiễu (noise) trong dữ liệu gây ảnh hưởng tới kết quả.

* + - 1. Tách từ trong câu (word segmentation)

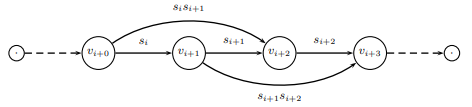
Trong một văn bản tiếng Anh, các từ cách nhau bằng một khoảng trắng, do đó chỉ cần thực hiện tách văn bản theo khoảng trắng để thu được các từ. Đối với tiếng Việt, mỗi một từ được tạo bởi một hoặc nhiều âm tiết, mỗi âm tiết cách nhau bằng một khoảng trắng. Vì vậy, các phương pháp tách từ dành riêng cho tiếng Việt là cần thiết. Bài toán tách từ là một bài toán khó do tiếng Việt có nhiều trường hợp gây nhập nhằng (ambiguity).

Ví dụ:

* + Nhập nhằng chồng (Overlapping Ambiguity): “học” và “sinh vật học” trong “Học sinh học sinh vật học”.
  + Nhập nhằng hợp (Combination Ambiguity): “chanh chua” trong “Cô gái chanh chua” và “Quả chanh chua”.

Tách từ là một bước tiền xử lý quan trọng cần độ chính xác cao và có ảnh hưởng rất lớn tới kết quả cuối cùng của bài toán biến đổi biểu diễn văn bản. Cụ thể, đã có các công trình nghiên cứu tách từ tiếng Việt có kết quả tách từ có độ chính xác cao. Điển hình là phương pháp áp dụng từ điển và thống kê. Phương pháp thứ hai được cải thiện và sửa lỗi từ phương pháp trước ứng dụng cây nhị phân SCRDR.

**Tách từ kết hợp từ điển và thống kê (Hybrid Approach)** [1]



Hình .: Đồ thị của một câu

Mô hình được đề xuất coi các âm tiết trong câu là một đỉnh của một đồ thị có hướng tuyến tính và với và lần lượt là đỉnh bắt đầu và kết thúc. Từ tới tương ứng với âm tiết trong câu. Các cung thể hiện khả năng nối các âm tiết liền nhau để tạo thành một từ có nhiều âm tiết [1].

Các tác giả đề xuất các cách tách từ khả thi tương ứng với các đường đi ngắn nhất từ điểm bắt đầu đến cuối đồ thị. Tuy nhiên, khả năng nhập nhằng chồng có thể xảy ra trong trường hợp ba âm tiết liền nhau có thể tách thành hoặc [1].

Để giải quyết nhập nhằng, các tác giả sử dụng mô hình ngôn ngữ n-gram: xác suất của một chuỗi được tính bằng tích của tất cả các từ cấu thành chuỗi với mỗi xác suất được tính bằng xác xuất có điều kiện với ( phụ thuộc vào mô hình n-gram được chọn) từ đã xuất hiện trước nó [1]. Với chuỗi được cấu thành bởi từ (âm tiết trong tiếng Việt) ():

|  |  |  |
| --- | --- | --- |
|  |  | **(2.1)** |

Đối với mô hình bigram (): xác suất của từ đứng sau chỉ phụ thuộc vào từ đứng trước đó và là [1]. Áp dụng phương pháp “ước lượng hợp lý cực đại” (MLE – Maximum Likelihood Estimation) để tìm xác suất trên từ bộ dữ liệu huấn luyện:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.2)** |

Trong đó:

* : số lần chuỗi xuất hiện trong bộ dữ liệu huấn luyện;
* : tổng số từ trong bộ dữ liệu huấn luyện.

Các tác giả sử dụng phương pháp “nội suy tuyến tính” (linear interpolation – Jelinek-Mercer smoothing) và điều chỉnh ước lượng sử dụng mô hình unigram (xác suất xuất hiện của từ trong bộ dữ liệu huấn luyện) [1]. Phương pháp này được áp dụng để tránh trường hợp nếu chuỗi không xuất hiện trong dữ liệu huấn luyện. Do vậy, xác suất xuất hiện ước lượng của từ dựa vào từ trước nó là:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.3)** |

Trong đó:

* : giá trị cần ước lượng;
* và .

Để có thể tính ra xác suất của các cách tách từ khả thi đã chọn, cần phải ước lượng hai giá trị và . Với một bộ dữ liệu kiểm thử (test set) là với câu , có xác suất của bộ dữ liệu là [1]. Khi đó, entropy được tính bằng công thức:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.4)** |

Trong đó:

* : số lượng từ của bộ dữ liệu .

Một phần bộ dữ liệu huấn luyện sẽ được tách ra để hình thành một bộ dữ liệu kiểm định (validation data). Với bộ dữ liệu kiểm định, cần ước lượng để tìm cực đại của hàm [1]:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.5)** |

Trong đó:

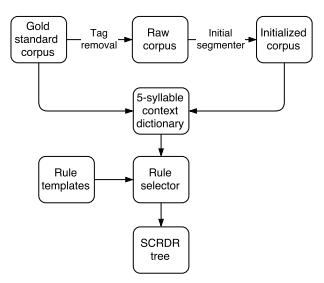
* : số lần xuất hiện của bigram trong tập kiểm định;
* và .

Với các giá trị tìm được, hoàn toàn có thể ước lượng được xác suất của từng cách tách từ. Cách tách từ có xác suất cao nhất sẽ được chọn. Thông qua thí nghiệm, các tác giả đã chứng minh phương pháp có kết quả chính xác cao, khoảng 96% [1].

Mô hình đã được các tác giả xây dựng thành một phần mềm mã nguồn mở với phiên bản đầu tiên là “vnTokenizer” [2] trên Java. Sau đó phần mềm được cập nhật và mang tên “vitk – A Vietnamese Text Processing Tookit” [3] để chạy trên Apache Spark để tăng khả năng xử lý dữ liệu lớn. Phiên bản mới nhất hiện tại được viết bằng Scalar và mang tên “vlp” [4]. Các mã nguồn này đều có sẵn trên Github kèm theo hướng dẫn tích hợp qua API.

**Mô hình tách từ cải tiến tự động sửa sai sử dụng cây nhị phân Single Classification Ripple Down Rules (SCRDR)** [5]

Phương pháp này kết hợp giữa một phương pháp tách từ ban đầu và sử dụng một cây nhị phân SCRDR được xây dựng tự động khi huấn luyện để sửa các lỗi sai được phát hiện khi huấn luyện tách từ. Các tác giả đã áp dụng phương pháp đánh dấu/dán nhãn từ loại (POS tagging – Part Of Speech tagging) đối với âm tiết gồm: B (âm tiết bắt đầu của một từ) và I (âm tiết nằm trong từ) [5]. Một corpus (kho ngữ liệu) đã được tách từ với dấu “\_” (underscore-based) có “tiêu chuẩn vàng” (gold standard) được sử dụng làm dữ liệu huấn luyện xây dựng cây SCRDR [5]. Bộ ngữ liệu thô (raw corpus) – thay thế các dấu “\_” tách từ bằng khoảng trắng (whitespace) được xây dựng từ “gold standard corpus” trên [5]. Sau đó, tách từ theo phương pháp “longest matching” (maximum matching) sử dụng từ điển tiếng Việt (Vietnamese lexicon) được xây dựng trong nghiên cứu đã đề cập trước đó [1] và module này được gọi là “initial segmenter” (bộ tách từ khởi đầu) [5].



Hình .: Sơ đồ phương pháp tách từ kết hợp cây nhị phân SCRDR

Sau đó, một bộ từ điển ngữ cảnh gồm 5 âm tiết (5-syllable context dictionary) sẽ được xây dựng dựa trên raw corpus và gold standard corpus [5]. được xây dựng dưới dạng key-value với key tương ứng một tuple gồm 5 âm tiết theo thứ tự, value sẽ tương ứng với tag BI của các âm tiết từ gold standard corpus [5].

Đối với mỗi âm tiết, một tuple/key chứa 5 âm tiết sẽ được xây dựng gồm hai âm tiết trước đó, chính nó và hai âm tiết sau kèm theo BI tag sau mỗi âm tiết. Các âm tiết/BI tag trống sẽ được thay bằng dấu **“”** [5]. Value sẽ tương ứng với BI tag của âm tiết đó trong BI-formed gold standard corpus [5]. Các chuỗi bị tách sai sẽ có BI tag khác với giá trị BI tag của âm tiết tương ứng trong BI-formed gold standard corpus [5].

Ví dụ:

Bảng .: Mô tả về các kho ngữ liệu

|  |  |
| --- | --- |
| Gold standard corpus | Học\_sinh học sinh\_vật\_học |
| Raw corpus | Học sinh học sinh vật học |
| BI-formed gold standard corpus | Học/B sinh/I học/B sinh/B vật/I học/I |
| Initial segmenter on raw corpus | Học\_sinh học\_sinh vật học |
| BI-formed raw corpus | Học/B sinh/I học/B sinh/I vật/B học/B |

Bảng .: Ví dụ về từ điển ngữ cảnh 5 âm tiết được xây dựng

|  |  |  |
| --- | --- | --- |
| **Tuple/Key** | **Value** |  |
| (“”, “”, “”, “”, **học, B**, sinh, I, học, B) | B | **✓** |
| (“”, “”, học, B, **sinh, I**, học, B, sinh, I) | I | **✓** |
| (học, B, sinh, I, **học, B**, sinh, I, vật, B) | B | **✓** |
| (sinh, I, học, B, **sinh, I**, vật, B, học, B) | B | **🗶** |
| (học, B, sinh, I, **vật, B**, học, B, “”, “”) | I | **🗶** |
| (sinh, I, vật, B, **học, B**, “”, “”, “”, “”) | I | **🗶** |

Một cây SCRDR là một cây nhị phân với hai loại nhánh: “ngoại trừ” (except) và “ngược lại” (if-not). Mỗi một nút (node) trên cây tương ứng với một “quy tắc” (rule). Quy tắc có dạng: “*nếu thì ”* với là “điều kiện” (condition) và là “kết luận” (conclusion). Để đảm bảo cây luôn có kết luận, nút gốc (root) có điều kiện luôn thỏa mãn ( và còn được gọi là nút mặc định.

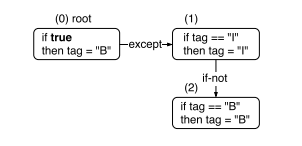
Với mỗi một trường hợp dữ liệu đi vào cây SCRDR sẽ được xử lý theo trình tự sau:

* B1**:** Bắt đầu tại nút gốc, nút hiện tại là nút gốc, gọi là kết luận của cây.
* B2**:** Với nút hiện tại: nếu , nút hiện tại là nút “if-not”. Nếu , . Xét điều kiện nút “except”: nếu , kết thúc duyệt cây. Nếu , gán nút hiện tại bằng nút “except” rồi thực hiện lại B2.

Bảng .: Mẫu qui tắc

|  |  |
| --- | --- |
| Âm tiết / Syllable | s-2, s-1,s0,s+1,s+2 |
| (s-2, s0), (s-1, s0), (s-1, s+1), (s0, s+1), (s0, s+2) |
| (s-2, s-1, s0), (s-1, s0, s+1), (s0, s+1, s+2) |
| Nhãn / BI Tag | t-2, t-1,t0,t+1,t+2 |
| (t-2, t-1), (t-1, t+1), (t+1, t+2) |
| Âm tiết và nhãn | (t-2, s0), (s0, t+1), (t-1, s0, t+1), (t-2, t-1, s0), (s0, t+1, t+2) |

Dựa vào các vị trí âm tiết có BI tag sai, cây SCRDR sẽ được xây dựng dựa trên một cây SCRDR khởi tạo như Hình 1.3 và sinh ra các quy tắc (rules) để chỉnh sửa những vị trí có BI tag không đúng với gold standard corpus [5]. Những quy tắc cụ thể này được sinh tự động theo mẫu quy tắc (rules templates).



Hình .: Cây khởi tạo SCRDR

Các mẫu quy tắc được hiểu như sau: các số (-2, -1, 0, 1, 2) đánh dấu các vị trí của âm tiết và nhãn B/I tương ứng của các tuples trong . Giả sử với quy tắc mẫu (s-1, s0), một nút cây có thể được xây dựng với điều kiện:

IF s-1 == tuple.s-1 && s0 == tuple.s0

THEN tag = gold-standard-tag

Tương tự, áp dụng mẫu (s-1, s0) với hàng thứ 4 của Bảng 2.2 sẽ xây dựng được nút cây có điều kiện:

IF s-1 == “học”&& s0 == “sinh”

THEN tag = “B”

Trong đó, các giá trị in đậm sẽ được thay cụ thể bằng các giá trị trong các tuples và nhãn B/I chuẩn của gold standard corpus. Các nút cây sẽ được xây dựng tương ứng và tạo ra một tập hợp các quy tắc cụ thể sau khi huấn luyện.

Khi huấn luyện, mỗi tuple sẽ được đưa vào cây hiện tại, khi đi hết cây và thỏa mãn điều kiện ở nút cuối cùng – N và giá trị nhãn trả về là sai thì một nút “except” mới sẽ được thêm vào ngay sau nút kết luận đó, ngược lại nếu không thỏa mãn điều kiện của nút cuối thì nút mới sẽ được thêm vào nhánh “if-not”. Tuy nhiên, quy tắc của nút mới sẽ phải được chọn từ tập hợp các quy tắc cụ thể đã được sinh ra khi huấn luyện. Ngoài ra, quy tắc được chọn cần thỏa mãn các ràng buộc sau:

* Nếu N không phải là 3 nút đầu tiên trong Hình 1.3 thì không được chọn các quy tắc mà có điều kiện thỏa mãn cho các tuples đã được trả về nhãn B/I đúng tại N [5].
* Quy tắc được chọn phải cho giá trị là cao nhất [5]. Trong đó:
  + a: Số lượng tuple thỏa mãn điều kiện và gán nhãn đúng;
  + b: Số lượng tuple thỏa mãn điều điện nhưng gán nhãn sai.
* Giá trị không được nhỏ hơn một giá trị chọn trước [5].

Sau khi thu được cây SCRDR, quy trình tách từ sẽ được thực hiện như sau:

* Đầu vào là một câu chưa được tách từ
* Sau khi sử dụng bộ tách từ ban đầu bằng phương pháp đầu tiên sẽ thu được câu gán nhãn BI.
* Với độ rộng cửa sổ bằng 5, tạo các tuples của câu/văn bản cần tách từ để sinh ra bộ từ điển ngữ cảnh 5 âm tiết.
* Sau đó các tuples được đưa vào cây SCRDR để sửa những nhãn B/I sai.
* Sau khi hoàn thành sửa các nhãn, câu từ dạng gán nhãn B/I sẽ được gán dấu “\_” để làm liên kết giữa các âm tiết trong một từ.

Nghiên cứu này có mã nguồn mở mang tên RDRsegmenter [5] và đã trở thành một phần trong bộ thư viện hỗ trợ xử lý tiếng Việt mang tên VNCoreNLP [6]. Mã nguồn này có sẵn trên Github kèm theo hướng dẫn tích hợp qua API.

* + - 1. Loại bỏ từ dừng (stopwords)

Các từ dừng là các từ xuất hiện nhiều trong ngôn ngữ tự nhiên nhưng không mang nhiều ý nghĩa.

Ví dụ : à, ơi, này, kia…

Có hai phương pháp loại bỏ từ dừng :

* Sử dụng từ điển : đây là phương pháp đơn giản nhất, tuy nhiên, việc này sẽ loại bỏ cả những từ có tần suất xuất hiện thấp, với một số mô hình biểu diễn, tần suất xuất hiện của từ thấp sẽ được coi là mang nhiều ý nghĩa hơn.
* Sử dụng tần suất xuất hiện : những từ có tần suất xuất hiện lớn sẽ được loại bỏ và thông thường những từ đó sẽ là từ dừng. Phương pháp này thường được kết hợp cùng các phương pháp xác suất thống kê để xác định khả năng giữ lại hay loại bỏ từ.
  + - 1. Xây dựng bộ từ vựng

Sau khi toàn bộ văn bản đã được tách từ, các từ sẽ được sử dụng để cấu thành một bộ từ vựng (vocabulary). Các phương pháp xử lý ngôn ngữ tự nhiên hầu hết đều dựa trên bộ từ vựng được xây dựng từ kho ngữ liệu được chọn. Do từ là đơn vị nhỏ nhất của một văn bản, có thể nói bộ từ vựng là cơ sở để các phương pháp biến đổi biểu diễn có thể được thực hiện.

* + 1. Phương pháp tiếp cận dựa vào tần suất (frequency-based)

Phương pháp này dựa vào tần suất xuất hiện của các từ trong các văn bản cần được biến đổi biểu diễn. Thông thường, có hai phương pháp được sử dụng để tính toán tần suất xuất hiện của từ.

* Tần suất xuất hiện trên toàn bộ ngữ liệu : đếm số lần xuất hiện của từ trên toàn bộ dữ liệu.
* Tần suất xuất hiện trên mẫu ngữ liệu : đếm số lần xuất hiện của từ trên một số lượng từ nhất định. Thông thường, các từ sẽ được tính toán tần suất trên 1 triệu từ.

Các mô hình sử dụng tần suất sẽ biến đổi các văn bản từ dạng ngôn ngữ tự nhiên về dạng vector số với số chiều tương ứng với độ lớn của bộ từ vựng và giá trị trên từng chiều sẽ được tính toán thông qua tần suất xuất hiên của từ tương ứng. Đây là các phương pháp đơn giản, dễ thực hiện và có kết quả khá tốt khi ứng dụng trong bài toán tìm kiếm.

Tuy nhiên, phương pháp này tồn tại những nhược điểm sau :

* Không có khả năng so sánh, mô tả ngữ nghĩa.
* Số chiều biểu diễn của vector tăng tuyến tính theo độ lớn của bộ từ vựng.
* Loại bỏ thứ tự trước sau của từ.
  + 1. Phương pháp tiếp cận dựa vào dự đoán (prediction-based)

Phương pháp này dựa vào việc tính toán và tối ưu xác suất xuất hiện của từ dựa vào các từ xuất hiện xung quanh – hay các từ ngữ cảnh (context words) khi duyệt văn bản. Hầu hết các phương pháp hiện hành đều là các mô hình học máy sử dụng mạng neuron để tiến hành huấn luyện và xây dựng mô hình ngôn ngữ thông qua dự đoán. Trọng số của các lớp ẩn (hidden layers) mà mô hình học được sẽ quyết định kết quả của ánh xạ hoặc biến đổi từ ngôn ngữ tự nhiên sang dạng vector. Phương pháp này có khả năng nhúng (embed) ngữ nghĩa trìu tượng vào biểu diễn để có thể áp dụng trong bài toán tìm kiếm theo ngữ nghĩa.

Số chiều của biểu diễn vector không phụ thuộc vào độ lớn của bộ từ vựng, do đó, các biểu diễn sẽ có số chiều được lựa chọn trước và thường nằm trong khoảng 300-1000 chiều. Đương nhiên, số chiều tăng lên đồng nghĩa với yêu cầu về tính toán và tài nguyên sẽ tăng lên. Do đó, cần có các phương pháp đặc biệt để tính toán và cập nhật trọng số cho mô hình để tốc độ huấn luyện. Đối với đa số các tác vụ yêu cầu, biểu diễn vector chỉ cần nằm trong khoảng 100-300 chiều.

Ngoại trừ một số mô hình sử dụng các phương pháp đặc biệt để mã hóa thông tin về vị trí của từ trong câu giúp tạo ra các biểu diễn khác nhau dựa trên ngữ cảnh của cùng một từ, các mô hình biểu diễn khác cũng thường loại bỏ thứ tự trước sau của các từ trong câu.

Ví dụ: “thuốc lá” mang hai nghĩa khác nhau trong “sắc thuốc lá” và “hút thuốc lá”.

Các mô hình dựa vào dự đoán phức tạp hơn so với các mô hình dựa vào tần suất. Tuy nhiên, các mô hình khác nhau thường có cấu trúc mạng neuron được xây dựng khác nhau, những mạng neuron có cấu trúc phức tạp thường có kết quả chính xác hơn. Những mô hình phức tạp thường tốn nhiều thời gian để huấn luyện mô hình cũng như sử dụng mô hình. Vì vậy, khi lựa chọn mô hình sử dụng, người dùng cần cân nhắc và cân đối mức độ chính xác và thời gian đáp ứng của dữ liệu.

* 1. Kết luận chương 1

Chương 1 đã giải quyết các kiến thức tổng quan về bài toán biểu diễn văn bản. Làm rõ các định nghĩa, khái niệm, đơn vị cấu thành văn bản; làm rõ một số phương pháp tách từ tiếng Việt – một bước quan trọng trong tiền xử lý ngôn ngữ tiếng Việt; khái quát các phương pháp tiếp cận bài toán biểu diễn văn bản.

# CÁC PHƯƠNG PHÁP BIỂU DIỄN VĂN BẢN TIẾNG VIỆT VÀ ỨNG DỤNG TRONG BÀI TOÁN SO KHỚP NGỮ NGHĨA

* 1. Các phương pháp biểu diễn từ

Từ là đơn vị nhỏ nhất trong một văn bản. Vì vậy, bài toán biến đổi biểu diễn từ là một trong những bước quan trọng của biến đổi biểu diễn văn bản.

* + 1. One-hot encoding (Binary Encoding – mã hóa nhị phân)

Với một tập từ điển từ vựng (vocabulary) với từ đã biết là , một từ có thể được biểu diễn dưới dạng one-hot encoding. Giả sử, có một vector với số chiều là , mỗi một chiều tương ứng với một từ trong theo thứ tự sắp xếp. Biểu diễn vector của một từ sẽ bao gồm giá trị tại chiều tương ứng vị trí từ trong từ điển là 1 và các chiều còn lại có giá trị 0. Ưu điểm của phép biến đổi này là đơn giản, tuy nhiên số chiều của vector biểu diễn sẽ tăng lên tuyến tính với số lượng từ vựng của và không nắm bắt được liên quan ngữ nghĩa giữa các từ.

Ví dụ: = (học\_sinh, học, sinh\_vật\_học)

Biểu diễn của từ “học\_sinh” là: v = (1,0,0)

* + 1. Count vector

Phương pháp này tính toán số lần xuất hiện của một từ trong một văn bản trên toàn bộ tập văn bản. Đối với một tập các văn bản và bộ từ vựng tương ứng là sẽ xây dựng một ma trận gọi là “count vector matrix”. Vector từ xây dựng được sẽ có số chiều bằng số lượng văn bản trong tập .

Ví dụ: = {“Hút thuốc\_lá có hại cho sức\_khỏe. Hãy hạn\_chế tối\_đa sử\_dụng thuốc\_lá.”, “Các bác\_sĩ luôn quan\_tâm sức\_khỏe bệnh\_nhân.”}

Một ma trận ghi lại số lần xuất hiện của từng từ trong với số chiều tương ứng với số lượng văn bản thuộc sẽ được sinh ra. Ma trận count vector sẽ có dạng là .

Dựa vào kết quả trong bảng 2.1, vector từ tương ứng của “thuốc lá” là (2,0) và “sức khỏe” là (1,1). Tương tự, các từ còn lại sẽ được biến đổi tương đương với các tần suất xuất hiện trong bảng 2.1.

Bảng .: Ví dụ về ma trận count vector

|  |  |  |
| --- | --- | --- |
|  |  |  |
| hút | 1 | 0 |
| thuốc lá | 2 | 0 |
| có | 1 | 0 |
| hại | 1 | 0 |
| cho | 1 | 0 |
| sức khỏe | 1 | 1 |
| hãy | 1 | 0 |
| hạn chế | 1 | 0 |
| tối đa | 1 | 0 |
| sử dụng | 1 | 0 |
| các | 0 | 1 |
| bác sĩ | 0 | 1 |
| luôn | 0 | 1 |
| quan tâm | 0 | 1 |
| bệnh nhân | 0 | 1 |

* + 1. Mô hình Term Frequency – Inversed Document Frequency (TF-IDF)

Phương pháp này tính toán trọng số của từ trên toàn bộ kho ngữ liệu (mức độ đặc trưng) thay vì đếm số lần xuất hiện và xây dựng vector cho từ. Trọng số TF-IDF được tính bằng tích giữa TF (tần suất xuất hiện của từ) và IDF (nghịch đảo tần suất xuất hiện của từ trong bộ ngữ liệu). Có nhiều phương pháp tính riêng cho TF và IDF tuy nhiên trong mục này chỉ đề cập đến công thức nguyên bản.

Ví dụ: Giả sử có bộ dữ liệu D gồm các câu sau:

* (1): Hôm nay trời rất quang mây và là một ngày đẹp trời
* (2): Mặt trời cung cấp nguồn nhiệt và ánh sáng cho Trái Đất
* (3): Giếng trời cung cấp ánh sáng tự nhiên cho không gian nội thất

Bảng .: Bảng số lượng văn bản thuộc D chứa từ/cụm từ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Từ / cụm từ | Tần suất | Từ / cụm từ | Tần suất | Từ / cụm từ | Tần suất |
| hôm nay | 1 | đẹp | 1 | tự nhiên | 1 |
| trời | 1 | mặt trời | 1 | không gian | 1 |
| rất | 1 | cung cấp | 2 | nội thất | 1 |
| quang | 1 | nguồn | 1 |  | |
| mây | 1 | nhiệt | 1 |
| và | 2 | ánh sáng | 2 |
| là | 1 | cho | 2 |
| một | 1 | trái đất | 1 |
| ngày | 1 | giếng trời | 1 |

Công thức trọng số TF-IDF được tính bằng tích:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.6)** |
|  |  | **(2.7)** |

Trong đó:

* *wt,d*: trọng số TF-IDF;
* tf*t,d*: tần suất xuất hiện của từ *t* trong văn bản *d*;
* idft,D: nghịch đảo tần suất xuất hiện của từ *t* trong toàn bộ tập *D*;
* *f(t,d)*: số lần xuất hiện của *t* trong *d*;
* *N*: tổng số văn bản trong *D*;
* *nt*: tổng số văn bản xuất hiện *t* trong *D.*

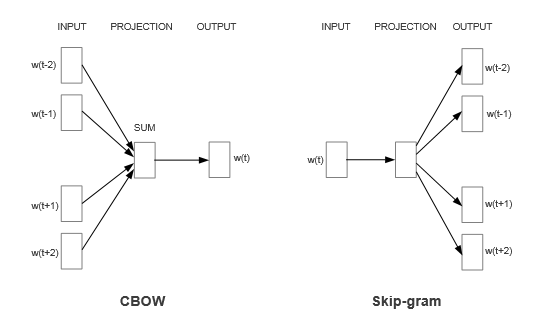
Dựa vào công thức trên, giá trị TF của từng từ sẽ được tính toán dựa trên tần suất xuất hiện của từng từ cho mỗi văn bản. Giá trị IDF sẽ được sử dụng chung với toàn bộ từ vựng. Bảng 2.3 là ví dụ cho cách tính TF-IDF từng từ của mỗi văn bản trong tập *D*.

Bảng .: Bảng trình bày biểu diễn văn bản sử dụng TF-IDF cho tập D

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TF | | | IDF | TF\*IDF | | |
| Câu  Từ | [1] | [2] | [3] | [1] | [2] | [3] |
| hôm nay | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| trời | 2/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.085 | 0 | 0 |
| rất | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| quang | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| mây | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| và | 1/11 | 1/11 | 0/11 | log(3/2) = 0.17 | 0.015 | 0.015 | 0 |
| là | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| một | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| ngày | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| đẹp | 1/11 | 0/11 | 0/11 | log(3/1) = 0.47 | 0.042 | 0 | 0 |
| mặt trời | 0/11 | 1/11 | 0/11 | log(3/1) = 0.47 | 0 | 0.042 | 0 |
| cung cấp | 0/11 | 1/11 | 1/11 | log(3/2) = 0.17 | 0 | 0.015 | 0.015 |
| nguồn | 0/11 | 1/11 | 0/11 | log(3/1) = 0.47 | 0 | 0.042 | 0 |
| nhiệt | 0/11 | 1/11 | 0/11 | log(3/1) = 0.47 | 0 | 0.042 | 0 |
| ánh sáng | 0/11 | 1/11 | 1/11 | log(3/2) = 0.17 | 0 | 0.015 | 0.015 |
| cho | 0/11 | 1/11 | 1/11 | log(3/2) = 0.17 | 0 | 0.015 | 0.015 |
| trái đất | 0/11 | 1/11 | 0/11 | log(3/1) = 0.47 | 0 | 0.042 | 0 |
| giếng trời | 0/11 | 0/11 | 1/11 | log(3/1) = 0.47 | 0 | 0 | 0.042 |
| tự nhiên | 0/11 | 0/11 | 1/11 | log(3/1) = 0.47 | 0 | 0 | 0.042 |
| không gian | 0/11 | 0/11 | 1/11 | log(3/1) = 0.47 | 0 | 0 | 0.042 |
| nội thất | 0/11 | 0/11 | 1/11 | log(3/1) = 0.47 | 0 | 0 | 0.042 |

* + 1. Mô hình Word-To-Vector (Word2Vec)

Một nhóm nghiên cứu của Google đã đề xuất về hai mô hình mạng neuron nhằm biến đổi biểu diễn của từ trong câu sang các vector trong không gian [7]. Hai mô hình được giới thiệu lần lượt là CBOW (continuous bag-of-words) và skip-gram [7]. Các mô hình này được gọi chung là mô hình Word2Vec. Mô hình Word2Vec sử dụng không gian vector để thể hiện mối quan hệ giữa các vector biểu diễn từ. Những từ được sử dụng với cùng ngữ cảnh thường sẽ có mức độ ý nghĩa tương quan nhất định.



Hình .: Mô hình CBOW và Skip-gram

Không gian vector được xây dựng và có khả năng thực hiện các phép toán với vector với kết quả trả về là tương đối chính xác về mối quan hệ giữa các từ. Một ví dụ điển hình là cho kết quả vector có biểu diễn sát với [7].

Ngoài ra, mô hình còn có nhược điểm là toàn bộ các ngữ nghĩa của từ trong nhiều ngữ cảnh khác nhau sẽ được biến đổi về một biểu diễn vector duy nhất. Vì vậy, khả năng xảy ra “catastrophic interference” (catastrophic forgetting) là không thể tránh khỏi. “Catastrophic interference” là trường hợp khi mạng neuron có thể “quên” khi được huấn luyện thêm với dữ liệu mới.

Đầu vào của mô hình Word2Vec là các vector one-hot encoding [7], tức là các vector thỏa mãn điều kiện: tọa độ trên một chiều là 1, các chiều còn lại là 0 (Ví dụ: [0,1,0,0,0] hoặc [1,0,0,0,0]) với tổng số chiều tương đương với tổng số lượng từ vựng trong từ điển được xây dựng từ bộ dữ liệu huấn luyện. Các vector này đại diện cho một từ theo thứ tự tương ứng trong từ điển từ vựng.

Mô hình Word2Vec là một mạng neuron đơn giản chỉ gồm 3 lớp (hình 3.3): input, hidden (projection) và output. Đầu vào của Word2Vec sẽ là một bộ dữ liệu gồm nhiều văn bản đã được tách từ trước. Mô hình này không nhất thiết phải sử dụng phương pháp loại bỏ từ dừng do công đoạn “frequent word subsampling”. “Frequent word subsampling” sử dụng một công thức để loại bỏ các từ có tần suất xuất hiện quá cao gây mất cân bằng với các từ hiếm ngoài ra nó còn giúp cải thiện tốc độ huấn luyện cũng như kết quả của các vector biểu diễn [7]. Xác suất giữ lại từ tính bằng công thức:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.8)** |

Trong đó:

* : xác suất giữ lại từ ;
* : một ngưỡng được chọn trước, thường quanh ngưỡng 10-5;
* : tần suất xuất hiện của từ .

Quá trình huấn luyện sẽ bắt đầu sau khi hoàn thành xây dựng bộ từ điển từ vựng từ dữ liệu đầu vào. Báo cáo này sẽ chỉ tập trung tìm hiểu mô hình CBOW và ứng dụng của nó trong việc biến đổi biểu diễn văn bản.

**Mô hình Continuos Bag-Of-Words (CBOW)** [7]

Với cấu trúc mạng được mô tả trong hình 2.1, đầu ra của CBOW là một từ cần được “dự đoán” (target word) - *wt* và đầu vào là các từ “ngữ cảnh” bao quanh không bao gồm từ cần dự đoán (context words) với độ rộng cửa sổ (window) bằng 5 (hai từ đứng trước *wt* và hai từ đứng sau). Trong trường hợp từ cần dự đoán nằm ở đầu hoặc cuối văn bản, các từ phía sau hoặc phía trước nó sẽ được chọn.

Gọi số lượng từ vựng trong bộ dữ liệu là , số lượng neuron trong lớp hidden (tương ứng lớp projection trong hình 2.1) là *N* (số chiều của không gian vector cần xây dựng), đầu vào của mạng neuron là các vector one-hot encoding (). Gọi là ma trận trọng số của lớp hidden và là ma trận trọng số lớp softmax. Ma trận () và () được khởi tạo ngẫu nhiên.

Giá trị của lớp output được tính bằng công thức:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.9)** |

Trong đó:

* : vector đầu ra của lớp hidden;
* : ma trận chuyển vị của ma trận ;
* : ma trận chuyển vị của ma trận ;
* : vector đầu vào one-hot encoding.

Đầu ra sử dụng hàm softmax để tính xác xuất phân bố trên bộ từ vựng:

|  |  |  |
| --- | --- | --- |
|  | với | **(2.10)** |

Trong đó:

* : giá trị của neuron output (lớp softmax);
* : giá trị của neuron output trước khi được kích hoạt tại vị trí
* : giá trị của neuron output trước khi được kích hoạt tại vị trí

Nếu đầu ra không khớp với từ cần được học thì cần phải tăng xác suất phân phối của từ đó. Do hàm softmax là hàm tính phân phối xác suất – tổng các suất tính được sẽ bằng 1. Với *j\** là từ cần dự đoán, hàm mất mát (log loss likelihood) có công thức:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.11)** |

Trong đó:

* : giá trị của hàm mất mát;
* : giá trị của neuron output tại vị trí *j\**;
* : giá trị của neuron output trước khi được kích hoạt tại vị trí ;
* : giá trị của neuron output trước khi được kích hoạt tại vị trí

Mục đích của việc huấn luyện mô hình là thu được ma trận trọng số tối ưu của lớp hidden – bảng tra cứu vector của các từ trong bộ từ vựng. Vì vậy, hàm mất mát cần được tối ưu – đạt cực tiểu. Lúc này, thuật toán lan truyền ngược (backpropagation) và thuật toán giảm gradient (gradient descent) sẽ được sử dụng để tìm ra các ma trận trọng số tối ưu.

Đạo hàm của hàm mất mát là:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.12)** |

Áp dụng quy tắc chuỗi (chain rule):

|  |  |  |
| --- | --- | --- |
|  |  | **(2.13)** |

Và

|  |  |  |
| --- | --- | --- |
|  |  | **(2.14)** |

Trong đó:

* : là các trọng số kết nối neuron của input đến lớp hidden;
* : là các trọng số kết nối neuron của lớp hidden đến của ma trận .

Sau khi đạo hàm, ta có:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.15)** |

Và

|  |  |  |
| --- | --- | --- |
|  |  | **(2.16)** |

Trong đó:

* , : Kronecker delta (, khi và , ngược lại = 0);
* *x*: trung bình cộng vector của các từ ngữ cảnh.

Áp dụng vào thuật toán gradient descent, trọng số sẽ được cập nhật như sau:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.17)** |

và

|  |  |  |
| --- | --- | --- |
|  |  | **(2.18)** |

Ngoài ra, thay vì phải cập nhật toàn bộ trọng số của hai ma trận và , phương pháp “negative sampling” được áp dụng để giảm mức độ phức tạp tính toán của mô hình huấn luyện mà cụ thể là giảm số lượng neuron cần cập nhật tại đầu ra [7].

Các từ được cập nhật neuron được chọn theo công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.19)** |

Trong đó:

* : xác suất chọn từ để cập nhật;
* : tần suất xuất hiện của từ ;
* : tổng số lần xuất hiện của toàn bộ các từ trong .

Các tác giả chỉ ra rằng: xác suất trên được nâng lên mũ 3/4 sẽ cải thiện tốc độ huấn luyện [7]. Cách này thực tế sẽ giúp tăng xác suất cho các từ hiếm (less frequent words) và giảm xác suất được chọn của những từ có tần suất xuất hiện lớn (more frequent words).

Sau khi hoàn thành huấn luyện, mỗi một hàng ma trận trọng số tương ứng với vector của từ trong bộ từ vựng [7].

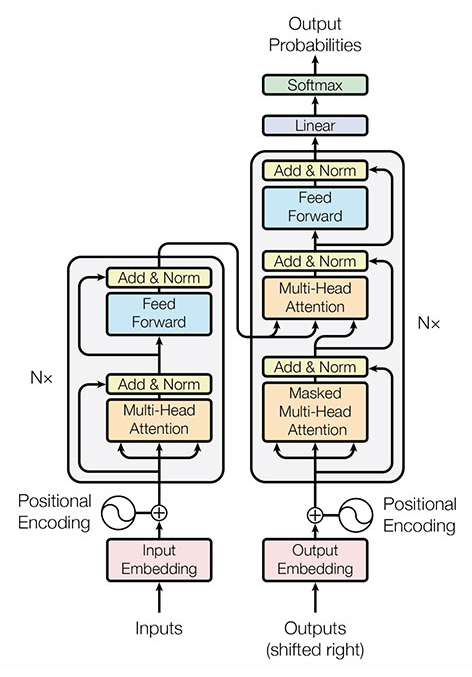
* + 1. Mô hình FastText

Mô hình này được phát triển bởi Facebook với mục đích cải tiến mô hình skip-gram từ Word2Vec [8]. Như đã đề cập trong mục trước, Word2Vec cần một bộ từ điển từ vựng (vocabulary) được xây dựng từ bộ dữ liệu huấn luyện. Do vậy, khi biến đổi biểu diễn của từ, ta chỉ có thể thu được vector của những từ tồn tại trong bộ từ vựng. Những từ không có trong bộ từ vựng sẽ gây ra lỗi OOV (out of vocabulary – không nằm trong bộ từ vựng).

FastText đã áp dụng mô hình n-gram lên các từ, tách các từ thành các đơn vị nhỏ hơn [8], ví dụ như “ngượng” sẽ được tách thành (ngư, gượ, ượn, ợng). Lúc này, biểu diễn vector của từ “apple” sẽ là tổng vector biểu diễn của các biểu diễn của các âm tiết (ngư, gượ, ượn, ợng). Phương pháp này giúp xử lý tốt trong các trường hợp mô hình nhận các từ đầu vào chưa được huấn luyện [8].

* + 1. Mô hình Transformer

Transformer là một mô hình xử lý ngôn ngữ tự nhiên phức tạp thông qua việc sử dụng cơ chế “attention” (attention mechanism) [9]. Nhờ cơ chế này, Transformer có khả năng tạo ra các biểu diễn riêng cho cùng một từ với các ngữ cảnh khác nhau [9]. Mặc dù Transformer tạo ra các vector từ với độ chính xác cao nhưng lại yêu cầu một lượng tài nguyên rất lớn để có thể huấn luyện mô hình (các hyperparameter được cài đặt như sau: vector biểu diễn từ 512 chiều, 6 encoders, 6 decoders, 8 attention-heads) [9].



Hình .: Mô hình Transformer

Mô hình này gồm 2 thành phần chính: *N* encoder (mã hóa) và *N* decoder (giải mã) [9]. Encoder phụ trách biến đổi biểu diễn của từ thành các vector và tạo thành đầu vào cho decoder [9]. Cấu trúc của một encoder và decoder lần lượt được biểu diễn bởi nửa trái và phải trong Hình 2.2. Các từ đầu vào được xử lý tuần tự từ trái sang phải thông qua các lớp như diễn giải dưới đây.

**Inputs**

Đầu vào của encoder là một câu/văn bản là một câu đã được tách từ sẵn. Mỗi từ đầu vào được mã hóa bằng one-hot encoding. Như đã đề cập trong các mục trước, các vector one-hot encoding có thể được ánh xạ sang từ trong từ điển từ vựng.

**Input Embedding**

Bước này chính là bước biến đổi các từ sang dạng vector thông qua nhân ma trận trọng số với vector one-hot encoding (tương tự Word2Vec). Sau bước này ta đã lấy được biểu diễn ban đầu của từ.

**Postional Encoding / Sinusoidal Positional Embedding**

Thay vì sử dụng số tự nhiên đánh dấu thứ tự từ trong văn bản huấn luyện, Transformer sử dụng “positional encoding” (PE – mã hóa vị trí) để gắn/nhúng (embed) ngữ cảnh của những từ xung quanh vào biểu diễn của từ [9]. Các PE được chuẩn hóa trong khoảng (0, 1) [9]. PE được tính khác nhau dựa theo vị trí chẵn lẻ của các phần tử trong vector như sau:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.20)** |
|  |  | **(2.21)** |

Trong đó:

* : chiều của không gian biểu diễn từ;
* : vị trí của từ trong câu;
* : vị trí của giá trị PE;
* Giá trị của hai hàm trên tính theo radian.

Ví dụ: “Học\_sinh học sinh\_vật\_học”

Giả sử với từ “học” ở vị trí và (chiều của vector biểu diễn từ), ta có vector là:

Cuối cùng, – biểu diễn của từ sẽ được cộng trở thành:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.22)** |

**Self-Attention**

Self-attention là một cơ chế giúp Transformer xây dựng quan hệ về ngữ cảnh giữa các từ trong một câu. Trong hình 2.4, khi đi vào encoder, ta thấy input embedding được chia làm ba nhánh. Ba nhánh này tương ứng với 3 ma trận trọng số lần lượt là [9]. Vì vậy, với mỗi từ ta thu được ba vector tương ứng là với số chiều của là và là với điều kiện (giúp tính toán của multi-head attention phần lớn trở nên ổn định) [9]. Điểm số của từng từ (lấy nhân với từng một) được tính như sau:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.23)** |
|  |  | **(2.24)** |

Tiếp theo ta tính giá trị từ các giá trị thu được. Cuối cùng ta áp dụng softmax lên của từ đối với các từ còn lại để lấy xác suất phân bố của từng từ. Giá trị softmax thể hiện mức độ liên quan của các từ khác với từ được tính toán. Hàm softmax cho giá trị càng lớn thì hai từ càng có khả năng tương đồng ngữ nghĩa với nhau.

Với bước cuối cùng, tổng các giá trị được tính cho từng từ. Cuối cùng, với input là từ ta thu được một ma trận có số chiều là .

Thông qua self-attention này, Transformer có thể tạo ra được các biểu diễn khác nhau cho cùng một từ thông qua ngữ cảnh tạo ra bởi những từ xung quanh [9]. Do đó, cơ chế attention có thể tạo ra các biểu diễn khác nhau cho từ “thuốc lá” trong ví dụ tại mục 1.2.3.

**Multi-Head Attention**

Với số lượng attention-head là *h*, ta sẽ có *h* bộ ma trận trọng số khác nhau, tương đương với *h* vector khác nhau. Do vậy ta thu được *h* ma trận *Z*. Ta sử dụng phép nối (concatenate) để nối *h* ma trận *Z* với nhau và nhân với một ma trận trọng số với số chiều là . Từ đó một ma trận *Z* với số chiều nhỏ hơn mang toàn bộ thông tin về các attention heads sẽ được đưa vào mạng FF (Feed Forward).

**Add & Norm / Residuals**

Ta có thể thấy được các lớp “Add & Norm” nhân đầu vào là ma trận biểu diễn vector từ và ma trận *Z*. Tại lớp này ta sẽ thực hiện cộng ma trận biểu diễn và ma trận *Z* để làm input cho sub-layer (lớp con) tiếp theo. Giống với các mô hình mạng RNN (recurrent neural network), lớp này được dùng để lưu trữ thông tin trạng thái (state) mang ngữ cảnh tại thời điểm hiện tại và chuyển tiếp tới từ tiếp theo.

**Feed Forward**

Lớp này là một lớp mạng neuron hoàn chỉnh (fully connected network), các vector *Z* sẽ được đưa vào tính toán bằng hàm kích hoạt ReLU (Rectified Linear activation Unit – ReLU(y) = ) tại lớp hidden trước khi chuyển sang lớp decoder. Các tác giả sử dụng một lớp hidden layer tương đương một vector có số chiều là 2048; đầu vào và đầu ra đều là vector 512 chiều [9].

|  |  |  |
| --- | --- | --- |
|  |  | **(2.25)** |

Trong đó:

* : giá trị vector output;
* : vector input;
* : trọng số lớp input;
* : bias lớp input;
* : trọng số lớp hidden;
* : bias lớp hidden.

**Output embedding**

Tương tự như mô hình Word2Vec, đầu ra của decoder là một từ tiếp theo được dự đoán có xác suất cao nhất. Vector biểu diễn từ được dự đoán sẽ chính là output embedding. Vector này sẽ được đưa vào tính toán cùng với từ tiếp theo.

**Masked Multi-head Attention**

Lớp này có cấu trúc tương tự lớp Multi-head Attention nhưng được cộng thêm một ma trận gọi là Mask (mặt nạ) và bước này được gọi là “masking” [9]. Lớp này giúp decoder che đi (không tính toán giá trị) “attention” của những từ chưa được xử lý [9].

Với ma trận (scaled score) thu được ta thực hiện cộng nửa ma trận nằm trên đường chéo chính với . Khi tính softmax, các giá trị sẽ cho giá trị softmax là 0 [9].

**Linear (fully connected network)**

Lớp này là một mạng neuron hoàn chỉnh được sử dụng để nhận vector từ lớp trước và biến đổi nó về một vector có số chiều bằng với số từ trong vocab. Lớp này có tác dụng tạo ra các điểm số cho từng từ trong vocab.

**Softmax**

Tương tự Word2Vec tính xác suất phân bố trên vocab. Từ có xác suất cao nhất là từ cần dự đoán.

**Hàm mất mát và cập nhật trọng số mô hình**

Với hàm mất mát ta sử dụng hàm mất mát tương tự mô hình Word2Vec là “log loss likelihood”.

Thay vì sử dụng “stochastic gradient descent” như Word2Vec, ta sử dụng Adam Optimizer để cập nhật trọng số cho mô hình với công thức như sau [9]:

|  |  |
| --- | --- |
|  | **(2.26)** |

Trong đó:

* : learning rate;
* : số chiều của vector biểu diễn;
* : số bước chạy mô hình hiện tại;
* : số bước chạy khởi động mô hình với gradient lớn.
  + 1. Mô hình Bidirectional Encoder Representations from Transformers (BERT)

Transformer được tạo ra với mục đích cải thiện độ chính xác trong lĩnh vực dịch máy. BERT là một mô hình sử dụng lớp encoder của Transformer với mục đích mô hình hóa ngôn ngữ (language modeling) [10] – nghĩa là tập trung chủ yếu vào việc xây dựng không gian vector biểu diễn của từ.

Với mô hình Transformer, chuỗi đầu vào được xử lý tuần tự từ trái sang phải (mô hình directional – một chiều). Về cơ bản BERT sử dụng chính Transformer, tuy nhiên, BERT lấy toàn bộ đầu vào cùng lúc, vì vậy BERT được coi là mô hình hai chiều (bidirectional) để có thể lấy ngữ cảnh của một từ từ những từ xung quanh (context words) [10]. Có hai mô hình BERT là BERT­BASE (L=12, H=768, A=12 với tổng cộng 110 triệu tham số) và BERTLARGE (L=24, H=1024, A=16 với tổng cộng 110 triệu tham số) [10]. Trong đó: L tương ứng số lớp encoder, H tương ứng với số chiều của biểu diễn, A là số lượng self-attention head.

BERT sử dụng hai loại hình huấn luyện là: MLM (Masked LM – Masked Language Modeling) và NSP (Next Sentence Prediction) [10]. Cả hai mô hình sẽ cùng được huấn luyện để tối ưu mất mát của cả hai loại hình.

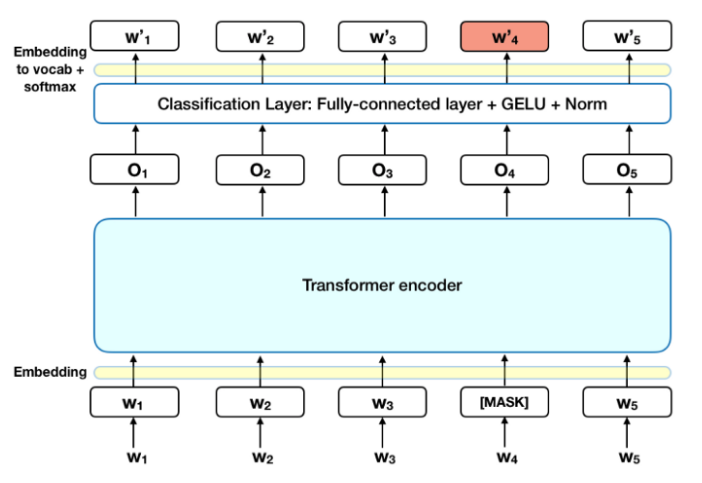
**Masked LM**

Trước khi huấn luyện, 15% số từ trong mỗi văn bản/chuỗi/câu được thay bằng một dấu [MASK]. Thực tế, những từ được chọn sẽ có 80% được thay bằng [MASK], 10% thay bằng một từ ngẫu nhiên, 10% giữ nguyên [10].

Cấu trúc của encoder Transformer được giữ nguyên. Nối tiếp với encoder là một lớp classification (phân lớp). Lớp này là một mạng neuron hoàn chỉnh (fully connected) sử dụng hàm kích hoạt GELU và chuẩn hóa đầu ra.

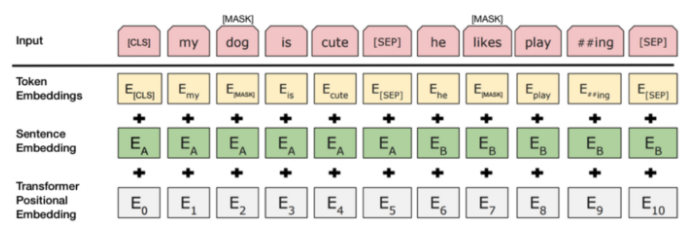
Output đó sẽ được nhân với một ma trận biểu diễn từ để chuyển về dạng vector biểu diễn. Sau đó, ta sẽ dùng softmax để tính xác suất phân bố trên bộ từ vựng.

Hàm mất mát của softmax là tương tự những mô hình trước, tuy nhiên, ta chỉ cần tính hàm mất mát đối với những từ bị thay thế bởi [MASK]. Nghiên cứu chỉ ra rằng, BERT thay vì tối ưu xác suất của từ cần dự đoán, các xác suất sẽ được giữ ổn định hơn cho những từ còn lại để tránh xảy ra overfitting (quá khớp). Tuy nhiên, mô hình sẽ hội tụ chậm hơn [10].



Hình .: Mô hình Masked LM

**Next Sentence Prediction (NPS)**



Hình .: Chuẩn bị đầu vào cho NPS

Với cách huấn luyện này, đầu vào của mô hình là hai câu nối tiếp nhau. Dấu [CLS] sẽ được thêm vào đầu câu thứ nhất và dấu [SEP] sẽ được thêm vào cuối mỗi câu [10].

50% lượng dữ liệu đầu vào là hai câu thực sự nối tiếp nhau. Nửa còn lại có các câu thứ 2 được chọn ngẫu nhiên từ corpus và được coi rằng không liên kết ý nghĩa với câu đầu tiên [10].

Theo Hình 2.4, trước khi huấn luyện, mỗi một từ sẽ được biến đổi sang biểu diễn vector tương ứng gọi là “token embedding”. Từng từ được đánh dấu để phân biệt những từ thuộc câu thứ nhất hoặc câu thứ hai, cách đánh dấu này là “sentence embedding”. Cuối cùng ta thực hiện cộng Sinusoidal Positional Embedding như Transformer theo văn bản được tạo thành bởi hai câu.

Cả ba embedding này được cộng với nhau và đưa vào mô hình huấn luyện. Cuối cùng, hàm softmax sẽ được sử dụng để dự đoán xem câu thứ hai có thật sự là câu nối tiếp.

* + 1. Mô hình ngôn ngữ tiếng Việt PhoBERT (Phở-BERT)

Mô hình BERT là một mô hình cồng kềnh, rất khó khăn để có thể huấn luyện từ đầu, ngoài ra, các nhà nghiên cứu chỉ ra rằng dữ liệu đã được sử dụng để huấn luyện BERT là quá ít (16GB) [11]. Vì vậy, một phiên bản cải tiến của nó là RoBERTa (Robustly optimized BERT approach) đã được xây dựng. RoBERTa được huấn luyện với 161GB dữ liệu từ nhiều nguồn khác nhau [11].

Dựa trên mô hình cải tiến của RoBERTa, nhóm nghiên cứu từ VinAI đã huấn luyện và công bố mô hình huấn luyện sẵn dành riêng cho tiếng Việt (đơn ngữ - monolingual) và được gọi là PhoBERT [12]. Mô hình này được huấn luyện sẵn với 20GB dữ liệu bao gồm dữ liệu từ Wikipedia tiếng Việt (~1GB) và một kho ngữ liệu về tin tức (~19GB) sau khi xử lý các bài báo trùng lặp [12].

Mô hình này hiện được coi là mô hình xử lý ngôn ngữ state-of-the-art (SOTA) cho tiếng Việt trên các tác vụ như: POS tagging (Part-Of-Speech tagging), NER (Named Entity Recognition), NLI (Natural Language Inference) và dependency parsing [12]. Để sử dụng mô hình, chúng ta có thể thực hiện huấn luyện mô hình bằng dữ liệu trên miền/lĩnh vực (domain) theo nhu cầu.

Mô hình PhoBERT hiện có sẵn trên API của HuggingFace và có thể dễ dàng sử dụng với PyTorch.

* 1. Các phương pháp biểu diễn văn bản

Trong mục này, các nghiên cứu được đề cập sẽ tập trung chủ yếu vào biến đổi các câu từ dạng chuỗi ký tự sang dạng số. Ngoài các phương pháp cổ điển, một số công trình nghiên cứu liên quan cũng sẽ được đề cập. Một số nghiên cứu áp dụng thành quả của biến đổi biểu diễn từ để tạo ra các mô hình có khả năng biến đổi biểu diễn văn bản.

* + 1. Mô hình Bag-Of-Words (BOW)

Mô hình BOW là một cách biểu diễn đơn giản trong xử lý ngôn ngữ tự nhiên. Ta coi rằng mỗi một câu hoặc một văn bản được biểu diễn bằng một tập hợp với phần tử là các từ chứa trong đó. Mô hình này rất đơn giản nhưng nhiều nhược điểm như: làm mất ngữ nghĩa hay thứ tự của các từ và chỉ thực hiện quan sát tần suất xuất hiện của chúng.

Giả sử ta có bộ dữ liệu D gồm các câu sau:

* [1]: Hôm nay trời rất quang mây và là một ngày đẹp trời
* [2]: Mặt trời cung cấp nguồn nhiệt và ánh sáng cho Trái Đất
* [3]: Giếng trời cung cấp ánh sáng tự nhiên cho không gian nội thất

Ta tách các câu trên thành tập hợp của các từ và thu được Bảng 3.1. Chúng ta có thể gọi bảng này là từ điển từ vựng (vocabulary) của bộ dữ liệu.

Như vậy ta có thể biểu diễn các câu trên như sau: ta biểu diễn dưới dạng vector với số chiều tương ứng với tổng số từ trong từ điển và tọa độ phụ thuộc vào tần suất xuất hiện của từ đó trong câu.

Bảng .: Bảng tần suất xuất hiện của các từ trong từng câu thuộc D

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Từ  Câu | hôm nay | trời | rất | quang | mây | và | là | một | ngày | đẹp | mặt trời | … |
| [1] | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | … |
| [2] | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | … |
| [3] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … |

Một nhược điểm của BOW là số lượng chiều của biểu diễn vector bị phụ thuộc vào số lượng từ vựng của bộ dữ liệu. Điểm yếu này có thể cải thiện phần nào nhờ phương pháp loại bỏ các từ dừng (stop words) – các từ có tần suất xuất hiện lớn và không có nhiều ý nghĩa như: và, là, … Bước này sẽ được thực hiện ngay sau khi tách từ và rất hiệu quả đối với những văn bản dài, đa dạng. Ngoài ra, thứ tự các từ sẽ không được chú ý và sẽ làm mất đi ngữ cảnh được tạo ra bởi thứ tự của các từ.

* + 1. Biểu diễn văn bản sử dụng vector TF-IDF

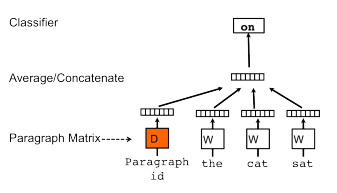
Phương pháp này dựa trên mô hình BOW để biểu diễn văn bản dưới dạng một vector. Ta thay thế các giá trị tần suất xuất hiện bằng các giá trị TF-IDF tương ứng.

Ví dụ: Dựa vào Bảng 2.3, ta có vector TF-IDF của câu (1) trong tập như sau:

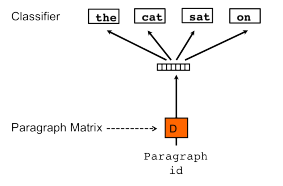
(0.042, 0.085, 0.042, 0.042, 0.042, 0.015, 0.042, 0.042, 0.042, 0.042, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

* + 1. Mô hình Document-To-Vector (Doc2Vec – paragraph vector)

Mô hình “paragraph vector” là mô hình được cải tiến từ Word2Vec. Các mô hình được đề xuất gồm PV-DM (distributed memory) và PV-DBOW (distributed bag-of-words) [13]. Hai mô hình này lần lượt được xây dựng dựa trên mô hình CBOW và Skip-gram từ bài báo trước đó [7].



Hình .: Mô hình PV-DM



Hình .: Mô hình PV-DBOW

Mục tiêu của các mô hình này là biến đổi biểu diễn của các văn bản đầu vào, huấn luyện để tìm ra biểu diễn tối ưu của văn bản.

**Mô hình Paragraph Vector – Distributed Memory (PV-DM)**

Cách huấn luyện của mô hình PV-DM không khác biệt so với CBOW, tuy nhiên input gồm có các biểu diễn văn bản được khởi tạo ngẫu nhiên với số chiều được chọn trước – một ma trận (paragraph matrix) với là số lượng văn bản và là số chiều của các văn bản [13]. Với mỗi văn bản ta có một “paragraph id” [13] để định danh văn bản đó (có thể là tên hoặc mã định danh GUID/UUID hoặc là đơn giản là một số nguyên để thực hiện truy xuất biểu diễn).

Trước khi huấn luyện, một bộ từ vựng cũng sẽ được xây dựng dựa trên bộ dữ liệu huấn luyện. Các văn bản có biểu diễn được khởi tạo ngẫu nhiên. Biểu diễn của các từ thu được không nhất thiết có cùng số chiều với các vector biểu diễn văn bản. Biểu diễn của các văn bản thêm vào được hiểu như các ngữ cảnh (context) của các văn bản đó do ngữ cảnh bị mất đi khi huấn luyện mô hình Word2Vec [13].

Đầu vào vẫn là các vector one-hot encoding của các “context words” và kèm thêm mã định danh của văn bản được chọn làm đầu vào, đầu ra vẫn sẽ là “target word”. Có thể biểu diễn mã định danh của văn bản dưới dạng one-hot encoding và “paragraph matrix” là một ma trận trọng số. Lớp hidden (projection) sẽ thực hiện lấy trung bình (average) giữa kết quả của lớp hidden CBOW và “paragraph vector” hoặc ghép vector (concatenate) để làm đầu ra của lớp hidden [13].

Ta tiếp tục gọi ma trận trọng số của lớp hidden là  và ma trận của lớp softmax là , gọi là ma trận trọng số biểu diễn văn bản. Ta thực hiện cập nhật trọng số của và giống Word2Vec. sẽ được cập nhật cuối cùng phụ thuộc vào lớp hidden được tính toán bằng trung bình cộng hay ghép.

Nếu ta lấy trung bình, đảo chiều công thức tính trung bình vector để thu được giá trị cập nhật của vector văn bản được huấn luyện. Trong trường hợp sử dụng ghép hai vector, giá trị của các neuron lớp hidden nối với chính là các giá trị cần cập nhật.

Sau khi hoàn thành huấn luyện, ta thu được là ma trận biểu diễn từ và . Để tìm được biểu diễn của một văn bản mới, ta khởi tạo ngẫu nhiên một vector biểu diễn, sau đó thực hiện các bước huấn luyện và giữ nguyên các giá trị của và . Sau một số lần cập nhật biểu diễn tùy chọn, ta thu được biểu diễn vector của văn bản mới. Tuy nhiên, nếu từ không tồn tại trong bộ từ vựng, từ đó sẽ bị bỏ qua (one-hot encoding là một “vector không”).

Mô hình PV-DM có tốc độ chậm hơn và sử dụng nhiều tài nguyên hơn mô hình PV-DBOW, tuy nhiên kết quả của PV-DM được cho rằng có độ chính xác ổn định hơn [13]. Các tác giả chỉ ra rằng kết hợp hai mô hình huấn luyện cho các vector có độ chính xác cao hơn nhưng với hầu hết các trường hợp, ta chỉ cần sử dụng một trong hai mô hình là đủ [13].

Nhược điểm khi huấn luyện mô hình Doc2Vec là tốn bộ nhớ máy tính. Giả sử ta có văn bản và số chiều đặc trưng của biểu diễn là . Mỗi một chiều được biểu diễn bằng kiểu *float* (4 bytes). Mô hình PV-DM không chỉ học biểu diễn của các văn bản mà còn đồng thời học biểu diễn của các từ trong từ điển được xây dựng từ bộ dữ liệu huấn luyện. Chỉ tính riêng bộ nhớ cho ta cần:

|  |  |  |
| --- | --- | --- |
|  |  | (2.27) |

Với 1 triệu biểu diễn văn bản huấn luyện với 100 chiều đặc trưng, dung lượng bộ nhớ ta cần riêng cho là:

Trong các mô hình tốt, hầu hết số chiều của biểu diễn nằm trong khoảng 300-1000.

* + 1. Biểu diễn văn bản dựa trên các vector biểu diễn từ

Đây là một trong những phương pháp tiếp cận đơn giản, lợi dụng các mô hình Word2Vec đã được huấn luyện sẵn (pretrained Word2Vec model). Do mô hình Word2Vec tập hợp toàn bộ các ngữ cảnh của từ vào trong cùng một biểu diễn, vì vậy độ chính xác của phương pháp này sẽ không cao. Trên thực tế, một số nghiên cứu chỉ ra rằng: việc sử dụng các mô hình Word2Vec huấn luyện sẵn cho kết quả khả quan với độ chính xác chấp nhận được. Các phương pháp tiếp cận sau đây dù không phải là SOTA nhưng được coi là các cơ sở tốt (strong/fast baseline) trong biến đổi biểu diễn văn bản.

Một văn bản (document) được cấu thành bởi các từ (words), có hoặc không có các dấu câu (punctuation). Vì vậy, ta có thể thực hiện phép toán tính trung bình cộng của các vector biểu diễn từ (word vectors) để tìm ra biểu diễn của văn bản.

Để tìm biểu diễn văn bản của một tập văn bản *D*, ta thực hiện huấn luyện mô hình Word2Vec trên tập *D*. Sau khi hoàn thành huấn luyện, với từng văn bản trong *D*, ta tính trung bình cộng các vector từ trong văn bản đó. Vector trung bình cộng (averaged vector) thu được chính là vector biểu diễn văn bản.

Phương pháp này là một phương pháp đơn giản, có khả năng sử dụng lại các mô hình Word2Vec huấn luyện sẵn, giảm thời gian và chi phí huấn luyện. Tuy nhiên, nếu các mô hình không được huấn luyện với dữ liệu thuộc phạm vi sử dụng (domain) thì các vector sẽ không có kết quả tốt do các vector từ không bao hàm nghĩa cần tìm hoặc xảy ra trường hợp từ không có trong từ điển từ vựng (out of vocabulary). Ngoài ra, các văn bản có độ dài lớn sẽ không có độ chính xác cao, do phép cộng các vector từ sẽ có kết quả là một vector từ có độ tương đồng lớn nhất so với vector tổng hợp. Vì vậy, phương pháp này thường được áp dụng trên các văn bản hoặc câu ngắn.

* + 1. Mô hình Smooth Inversed Frequency (SIF)

Mô hình này được đề xuất nhằm huấn luyện một tập văn bản *D* với một mô hình Word2Vec/FastText đã được huấn luyện trước. Đây là một mô hình đơn giản, tuy không phải là SOTA nhưng được coi là một cơ sở chắc chắn trong biến đổi biểu diễn câu (sentence embedding) [14].

Phương pháp này tính trọng số của mỗi từ bằng công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.28)** |

Trong đó:

* : trọng số của từ;
* : smoothing hyper-parameter, thông thường là 0.001;
* : tần suất xuất hiện của từ, thông thường trên 10­6 từ.

Thông qua thực nghiệm, các tác giả chỉ ra rằng cách đánh giá trọng số (weighting scheme) này khá giống với TF-IDF.

Để tìm được các vector từ có trọng số, ta nhân trọng số với vector từ tương ứng. Vậy ta sẽ thu được vector biểu diễn câu có trọng số (weighted average vector) là trung bình cộng của các vector từ có trọng số (weighted word vector). Do đó vector biểu diễn câu được tính như sau:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.29)** |

Trong đó:

* : vector biểu diễn của câu;
* : tổng số từ trong câu;
* : trọng số của từ tương ứng;
* : vector biểu diễn của từ.

Sau khi thu được tập hợp các “weighted average vector” ta xây dựng một ma trận X với với là số lượng câu và là số chiều của các vector từ. Tiếp theo, ta thực hiện “common component removal” [14] hay loại bỏ FSV – first singular vector. Để thực hiện được bước này, ta coi như một ma trận và áp dụng SVD (Singular Value Decomposition) lên X. Ta có:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.30)** |

Trong đó E là một ma trận có đường chéo được tạo bởi các trị riêng (eigenvalue) sắp xếp theo giá trị giảm dần từ trên xuống.

Ví dụ:

; ; ; E(4,4) = 0 …

E không bắt buộc là một ma trận vuông. “Đường chéo” bắt đầu từ phần tử (1,1) và kết thúc khi nó chạm biên của ma trận E.

Gọi trị riêng lớn nhất trong E (hàng đầu tiên) là – tương ứng phần tử (1,1). FSV chính là cột đầu tiên của ma trận E. Ta có:

Sau cùng, để thu được vector biểu diễn câu, ta thực hiện:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.31)** |

Mục đích của “common component removal” là để loại bỏ những từ thường gặp (common words) mà có entropy cao ảnh hưởng đến kết quả vector biểu diễn. Phương pháp này khá giống với subsampling của Word2Vec [7]. Các câu được chọn làm dữ liệu huấn luyện chính là các mẫu được sử dụng để tìm ra vector “principal component”. Vector này sẽ được lưu lại để sử dụng cho những câu mà mô hình chưa được huấn luyện.

Mô hình này được coi là phù hợp cho “domain adaptation” (thích ứng miền): các vector từ được huấn luyện bằng nhiều bộ dữ liệu khác nhau sẽ được sử dụng để thử nghiệm với nhiều môi trường kiểm thử khác nhau (ngữ nghĩa trải rộng trên nhiều miền) [14].

**Singular Value Decomposition (SVD)**

Trong đại số tuyến tính, một ma trận vuông được gọi là chéo hóa được nếu tồn tại ma trận đường chéo D và ma trận khả nghịch P sao cho:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.32)** |

SVD là phương pháp sử dụng để tách bất cứ ma trận A nào thành ba ma trận đặc biệt trên thay vì phải là một ma trận vuông chéo hóa được.

Ma trận có thể được phân tích thành dạng:

|  |  |  |
| --- | --- | --- |
|  |  | **(2.33)** |

Trong đó:

* : là các ma trận trực giao;
* : là một ma trận đường chéo không vuông.

Mỗi cột của chính là một vector riêng của (mỗi cột này được gọi là left singular vector), tương tự mỗi cột của (right singular vector) là một vector riêng của (với mỗi trị riêng ta có thể tìm được một vector riêng). Cụ thể, sau khi tìm các trị riêng của và , ta tìm các vector riêng tương ứng với độ lớn bằng vector đơn vị. Các vector riêng được sắp xếp theo trị riêng tương ứng với độ lớn giảm dần.

Ví dụ :

Ta có . Trị riêng lần lượt là : . Tương ứng với các trị riêng ta có các nghiệm tổng quát và . Để tìm được các vector riêng ứng với các trị riêng, ta sử dụng công thức :

|  |  |  |
| --- | --- | --- |
|  |  | **(2.34)** |

Trong đó:

* : ma trận cần tìm vector riêng ;
* : vector riêng ;
* : trị riêng.

Với ta có : . Suy ra vector riêng tương ứng với trị riêng là .

Với ta có : . Suy ra vector riêng tương ứng với trị riêng là .

Ma trận đường chéo không vuông được tạo bởi căn bậc hai các trị riêng dương của và .

* 1. Ứng dụng trong bài toán so khớp ngữ nghĩa

So khớp ngữ nghĩa (semantic matching) là một kỹ thuật sử dụng trong khoa học máy tính để tìm kiếm các thông tin liên quan về mặt ngữ nghĩa. Để thực hiện được so khớp ngữ nghĩa, ta cần có một đại lượng (metric) để so sánh ngữ nghĩa giữa truy vấn và dữ liệu. Đại lượng này được gọi là *semantic similarity* (mức độ tương đồng ngữ nghĩa).

Có hai cách chính để đo lường (measure) đại lượng này: so khớp mặt chữ (phrase matching) và so sánh sự tương đồng giữa hai vector văn bản với ngữ nghĩa đã được biến đổi sang dạng số.

OkapiBM25 – một thuật toán dựa trên TF-IDF là một trong những phương pháp tính điểm và xếp hạng kết quả tìm kiếm dựa vào các từ xuất hiện trong văn bản.

Khi văn bản đã được biến đổi về dạng vector, ta có thể sử dụng các phép đo độ tương đồng. Các mục trước đã đề cập về các mô hình có khả năng chuyển đồi ngữ nghĩa trừu tượng của ngôn ngữ sang biểu diễn dạng vector có thể tính toán được. Do vậy, bài toán biến đổi biểu diễn văn bản sẽ được ứng dụng vào bài toán so khớp ngữ nghĩa.

Các mục trên đã cung cấp kiến thức cơ bản về việc biểu diễn văn bản từ dạng chuỗi ký tự sang vector trong một không gian đa chiều. Với hai mô hình trên, biểu diễn vector của các từ/văn bản được tính toán thông qua các từ ngữ cảnh. Do đó, những từ/văn bản có chung ngữ cảnh thường sẽ mang ý nghĩa tương tự hoặc liên quan tới nhau ở một mức độ nhất định. Ánh xạ sang hình học tọa độ, ta có thể coi rằng hai từ/văn bản có nghĩa giống nhau nếu vector của chúng trùng nhau. Qua đó ta có thể coi rằng mức độ đồng nghĩa giữa hai từ/văn bản phụ thuộc vào góc hoặc khoảng cách giữa hai biểu diễn vector. Góc hoặc khoảng cách giữa chúng càng hẹp/nhỏ tương đương với khả năng tương đồng về mặt ngữ nghĩa càng lớn.

Để so sánh mức độ tương đồng ngữ nghĩa giữa hai văn bản ta có thể sử dụng một trong các phép đo sau: độ tương đồng cosine, TS-SS hoặc khoảng cách Euclidean.

Độ tương đồng cosine bỏ qua chiều dài của văn bản mà chỉ đo góc giữa hai vector biểu diễn. Ngược lại, khoảng cách Euclidean được tính toán sử dụng đến độ lớn của vector biểu diễn văn bản do sử dụng tần suất xuất hiện của từ trong văn bản. Độ dài của văn bản sẽ quyết định độ lớn của vector, do đó sẽ ảnh hưởng tới độ lớn của phép đo và ảnh hưởng đến kết quả so sánh. TS-SS là một phương pháp sử dụng phép tính diện tích – nghĩa là sử dụng cả góc và độ lớn của vector.

Độ tương đồng cosine là một phép đo đơn giản và có khả năng so sánh mức độ tương đồng giữa hai văn bản. Kết quả của phép đo lớn thể hiện mức độ tương đồng ngữ nghĩa giữa hai văn bản.

Phép đo TS-SS và khoảng cách Euclidean phù hợp để tìm và xếp hạng một tập hợp các vector hơn là được sử dụng để so sánh. Do kết quả của các phép đo không nằm cố định trong khoảng nào nên ta chỉ có thể xác định các văn bản gần nghĩa hơn với văn bản tìm kiếm.

**Độ tương đồng cosine (Cosine Similarity)**

Phép đo độ tương đồng cosine là một phép đo đơn giản. Phép đo này bỏ qua độ lớn của hai vector và tìm giá trị cosine của góc giữa hai biểu diễn vector. Giá trị của cosine nằm trong đoạn [-1, 1] tương ứng với góc (1800, 00).

Phép đo được tính bằng thương của tích vô hướng và tích độ dài của hai vector.

|  |  |
| --- | --- |
|  | **(2.35)** |

* 1. Kết luận chương 2

Trong chương 2 đã đề cập tới các phương pháp biểu diễn văn bản và ứng dụng của chúng trong giải quyết bài toán so khớp ngữ nghĩa. Những phương pháp sử dụng tần suất cổ điển có thể biểu diễn văn bản dưới dạng vector nhưng không mang ý nghĩa trừu tượng. Ngược lại, các phương pháp học máy phức tạp hơn nhưng các biểu diễn vector thu được có khả năng biểu diễn ngữ nghĩa trong không gian đa chiều. Vì vậy, các phương pháp biểu diễn sử dụng mô hình học máy sẽ là tiền đề để xây dựng các biểu diễn có khả năng áp dụng vào trong bài toán so khớp ngữ nghĩa.

|  |
| --- |
|  |

# THIẾT KẾ VÀ CÀI ĐẶT HỆ THỐNG HỖ TRỢ TÌM KIẾM VĂN BẢN THEO NGỮ NGHĨA

Chương này sẽ đề cập đến cách huấn luyện và các siêu tham số (hyper-parameters) được lựa chọn cho các mô hình. Đồng thời, chương này đề xuất thiết kế một hệ thống với kiến trúc micro-service có khả năng tích hợp vào một hệ thống đang hoạt động thông qua cung cấp các REST API. Dịch vụ này có khả năng hỗ trợ truy xuất các tin tức có tiêu đề hoặc tóm tắt có ngữ nghĩa tương đồng với chuỗi tìm kiếm do người dùng nhập vào.

* 1. Chuẩn bị ngữ liệu

Bộ ngữ liệu được sử dụng là bộ ngữ liệu tin tức tiếng Việt được xây dựng bởi Vũ Quốc Bình [15]. Tiêu đề các bài báo (hơn 9 triệu tiêu đề) sẽ được sử dụng làm dữ liệu huấn luyện mô hình Word2Vec.

**Tiền xử lý dữ liệu**

Tiền xử lý sẽ gồm các bước sau:

* Tách từ trong câu
* Chuyển các từ về viết thường

Trong bộ dữ liệu được chọn, các tiêu đề đã được thực hiện loại bỏ các thẻ HTML. Ngoài ra, đối với mô hình Word2Vec và Paragraph Vector, phương pháp subsampling được sử dụng để làm giảm số lần dự đoán đối với các từ có tần suất xuất hiện lớn (thông thường là các từ dừng) mà không cần thực hiện loại bỏ từ dừng.

* 1. Huấn luyện mô hình

Mô hình được chọn để sử dụng cho hệ thống là mô hình SIF. Để thực hiện huấn luyện mô hình SIF. Ta cần có một mô hình biểu diễn từ.

Đầu tiên, ta thực hiện cài đặt thư viện “gensim” [16] và thư viện “fse” [17] trên Python thông qua pip bằng các lệnh trên terminal:



Để tăng tốc độ huấn luyện, ta có thể sử dụng thư viện Cython để sử dụng các hàm tính toán được viết trên ngôn ngữ C – có tốc độ thực thi nhanh hơn Python.

Ta sẽ sử dụng mô hình Word2Vec mà cụ thể là CBOW trong thư viện “gensim” và thực hiện cài đặt các tham số và đường dẫn tới file ngữ liệu đã được tiền xử lý.



Sau đó, ta xây dựng một lớp để tạo ra một thực thể có khả năng mở một stream để đọc dữ liệu tuần tự thay vì tải toàn bộ ngữ liệu lên bộ nhớ máy tính.

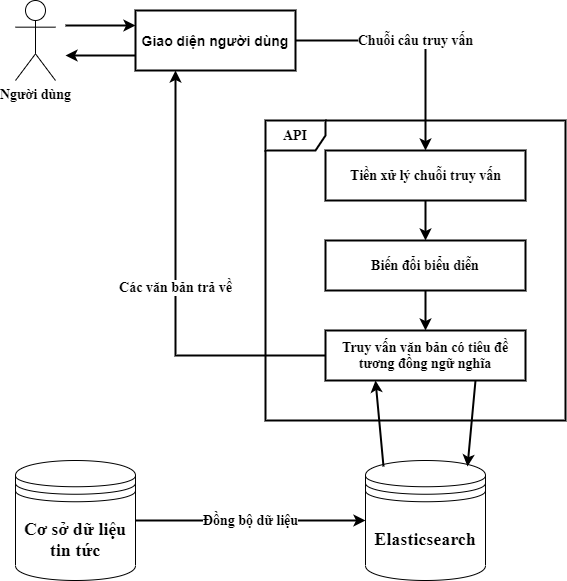


Cuối cùng, mô hình SIF trong thư viện “fse” sẽ được gọi để bắt đầu huấn luyện biểu diễn văn bản. Mục đích chính của mô hình SIF là tìm ra PC1 của các vector biểu diễn từ. Vì vậy, không cần thiết phải sử dụng toàn bộ kho ngữ liệu (không được quá ít) để tìm ra PC1. Mô hình thu được chứa các vector biểu diễn cho các câu đã được huấn luyện và giá trị của PC1.



* 1. Thiết kế tổng quan của hệ thống
     1. Thiết kế tổng quan

Trong hình 2.7, hệ thống được thiết kế dưới dạng một micro-service để có thể dễ dàng tích hợp với hệ thống đang hoạt động. Cơ sở dữ liệu (CSDL) tin tức là CSDL đã và đang được sử dụng cho các phần mềm/ứng dụng quản lý tin tức. Hệ thống sẽ hỗ trợ chuyển đổi văn bản từ CSDL SQL hiện hành sang search engine Elasticsearch để phục vụ tìm kiếm văn bản theo ngữ nghĩa.



Hình 3.1: Thiết kế tổng quan của hệ thống

* + 1. Module dữ liệu

CSDL truyền thống SQL luôn được sử dụng để lưu trữ các thông tin lâu dài, tuy nhiên các CSDL truyền thống hoạt động tốt hơn với các tác vụ thống kê và lọc chính xác thay vì tìm kiếm mờ (fuzzy search).

Đối với CSDL SQL, việc truy vấn toàn bộ văn bản để thực hiện so sánh biểu diễn là không khả thi do:

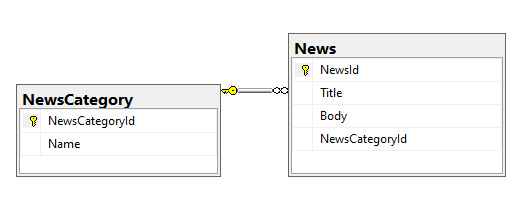
* Tốn nhiều tài nguyên xử lý
* Tốc độ đọc chậm do sử dụng I/O trên ổ đĩa, không thể đáp ứng trả về kết quả thời gian thực

Do đó, để có khả năng tìm kiếm văn bản thông qua tính toán độ tương đồng vector trực tiếp, ta sẽ sử dụng search engine - đồng thời cũng là CSDL phi cấu trúc (NoSQL) Elasticsearch.

**Module CSDL tin tức**

CSDL tin tức là một CSDL SQL truyền thống, phục vụ việc lưu trữ tin tức lâu dài, là nơi lưu trữ dữ liệu chính cho ứng dụng quản lý và thống kê.

Để mô phỏng tương tác giữa hệ thống và một CSDL đã tồn tại, một cơ sở dự liệu tạm thời sẽ được xây dựng để lưu trữ văn bản (tiêu đề, nội dung, loại tin). CSDL được sử dụng để mô phỏng là SQL Server 2017 Express.



Hình 3.2: Thiết kế CSDL SQL mô phỏng

Trong đó, các trường dữ liệu được thiết kế trong hai bảng trên như sau:

Bảng 3.1: Bảng NewsCategory

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Khóa chính** | **Khóa ngoại** | **Tên trường** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | x |  | NewsCategoryId | uniqueidentifier | Mã loại tin tức |
| 2 |  |  | Name | nvarchar(MAX) | Tên loại tin tức |

Bảng 3.2: Bảng News

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Khóa chính** | **Khóa ngoại** | **Tên trường** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | x |  | NewsId | uniqueidentifier | Mã tin tức |
| 2 |  |  | Title | nvarchar(MAX) | Tiêu đề tin |
| 3 |  |  | Abstract | nvarchar(MAX) | Tóm tắt tin |
| 4 |  |  | Body | nvarchar(MAX) | Nội dung tin |
| 5 |  | x | NewsCategoryId | uniqueidentifier | Mã loại tin tức |

**Module Elasticsearch**

Elasticsearch là một search engine dựa trên nền Lucene phục vụ tìm kiếm văn bản toàn văn dựa trên xếp hạng về điểm số của các văn bản sử dụng thuật toán OkapiBM25. Đồng thời, Elasticsearch còn là một CSDL NoSQL, do đó, tốc độ truy vấn sẽ nhanh hơn CSDL SQL nhiều lần. Tuy nhiên, các CSDL NoSQL thường tốn nhiều tài nguyên để vận hành.

Elasticsearch cung cấp khả năng tìm kiếm văn bản dựa theo tự động hoàn thiện chuỗi tìm kiếm (autocomplete). Đồng thời, với sự phát triển trong lĩnh vực xử lý ngôn ngữ tự nhiên, Elasticsearch hỗ trợ tìm kiếm văn bản dựa vào xếp hạng điểm số được tính bằng độ tương đồng cosine.

Tuy nhiên, để tăng tốc độ xử lý cũng như giảm lượng tài nguyên mà Elasticsearch sử dụng, chỉ những trường dữ liệu cần thiết phục vụ truy vấn dữ liệu sẽ được đánh chỉ mục.

Module Elasticsearch sẽ được cài đặt thông qua lớp **ES\_Repository**. Lớp này không chỉ giúp thiết lập kết nối mà còn cung cấp các phương thức thao tác với các chỉ mục lưu trữ dữ liệu trong Elasticsearch.

Các chức năng chính của module gồm:

* Khởi tạo các mappings – cấu trúc dữ liệu lưu trữ cho kho dữ liệu Elasticsearch
* Tạo kết nối tới Elasticsearch
* Các phương thức thao tác với chỉ mục
* Kiểm tra số lượng bản ghi giữa CSDL SQL và Elasticsearch
* Thực hiện tìm kiếm văn bản dựa vào độ tương đồng cosine và biểu diễn văn bản
  + 1. Module REST API truy vấn

**Tiền xử lý văn bản**

Phương pháp tách từ được sử dụng là phương pháp tách từ sửa sai sử dụng cây nhị phân SCRDR.

Tuy nhiên, với sự trợ giúp của thư viện VnCoreNLP, ta không cần phải thực hiện huấn luyện cây nhị phân SCRDR.

Để sử dụng thư viện VnCoreNLP, ta cần tải file jar của công cụ VnCoreNLP – chứa API xử lý ngôn ngữ tiếng Việt mà cụ thể là tách từ (word segmentation – wseg). Thư mục “models” chứa các mô hình xử lý ngôn ngữ tiếng Việt cần nằm trong cùng thư mục với file jar.

Sau khi tách từ, các từ đã được tách ra sẽ được chuyển đổi từ viết hoa sang viết thường.

**Biến đổi biểu diễn văn bản**

Trong các mô hình PV-DM, SIF và PhoBERT, ta cần đánh giá và chọn ra một mô hình phù hợp.

Như đã đề cập trong cuối mục 2.3.2, mô hình PV-DM sử dụng rất nhiều bộ nhớ để lưu trữ các vector biểu diễn với lượng văn bản lớn. Do đó, PV-DM phù hợp hơn với huấn luyện các văn bản dài với tổng số lượng văn bản thấp. Ngoài ra, các biểu diễn văn bản và các biểu diễn từ được huấn luyện cùng nhau. Vì vậy, để cập nhật các biểu diễn từ mới là khó khăn vì các biểu diễn từ mới sẽ thuộc những câu mới và ban đầu các biểu diễn đều được khởi tạo ngẫu nhiên.

Mô hình SIF lấy trung bình của các biểu diễn sẽ tốn nhiều thời gian để thực hiện các phép tính toán nếu đó là một văn bản dài. Tuy nhiên, việc sử dụng các câu huấn luyện để tính toán và lưu lại Principal Component đầu tiên (PC1) giúp tăng tốc độ biến đổi biểu diễn câu. Ngoài ra, mô hình SIF sử dụng mô hình Word2Vec được huấn luyện sẵn và khả năng cập nhật từ mới của Word2Vec dễ dàng hơn rất nhiều. Tốc độ huấn luyện của mô hình SIF là khá nhanh, vì vậy, có thể coi SIF là một mô hình khả thi hoàn toàn có thể triển khai.

Đối với PhoBERT, mô hình có sử dụng kiến trúc Transformer nên rất cồng kềnh và phức tạp. Thời gian để tải mô hình, huấn luyện và biến đổi biểu diễn đều rất chậm. PhoBERT sinh ra các biểu diễn từ với số chiều là 768 và biểu diễn văn bản thông qua tổng/trung bình các vector của 4 lớp hidden cuối cùng. Do đó, chi phí để tính toán được độ đo tương đồng sẽ rất lớn và đòi hỏi các GPU có tốc độ cao và nhiều nhân xử lý song song. Tốc độ huấn luyện và biến đổi biểu diễn của mô hình có thể được cải thiện nhờ vào sử dụng các GPU tân tiến với CUDA capability ≥ 3.5.

**Truy vấn văn bản theo tương đồng ngữ nghĩa của tiêu đề**

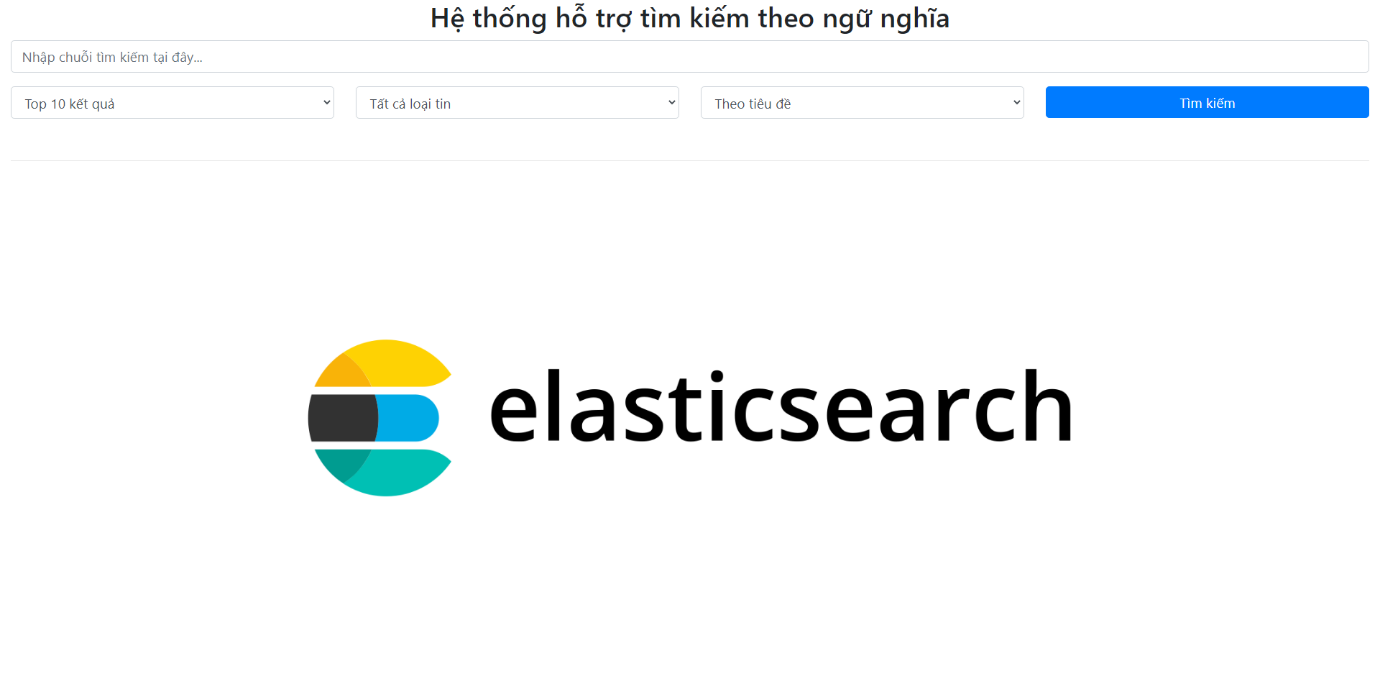
Dưới sự hỗ trợ của search engine Elasticsearch, các văn bản được đánh chỉ mục kèm theo các trường “dense vector” để lưu trữ biểu diễn với số chiều được định trước.

Elasticsearch hỗ trợ lưu trữ biểu diễn lên tới 2048 chiều. Trong hệ thống, biểu diễn của tiêu đề được lưu dưới trường [“title\_embedding”] và có số chiều là 100. Tương tự, ta có biểu diễn của đoạn tóm tắt tin được lưu trong trường [“abstract\_embedding”] với số chiều là 100.

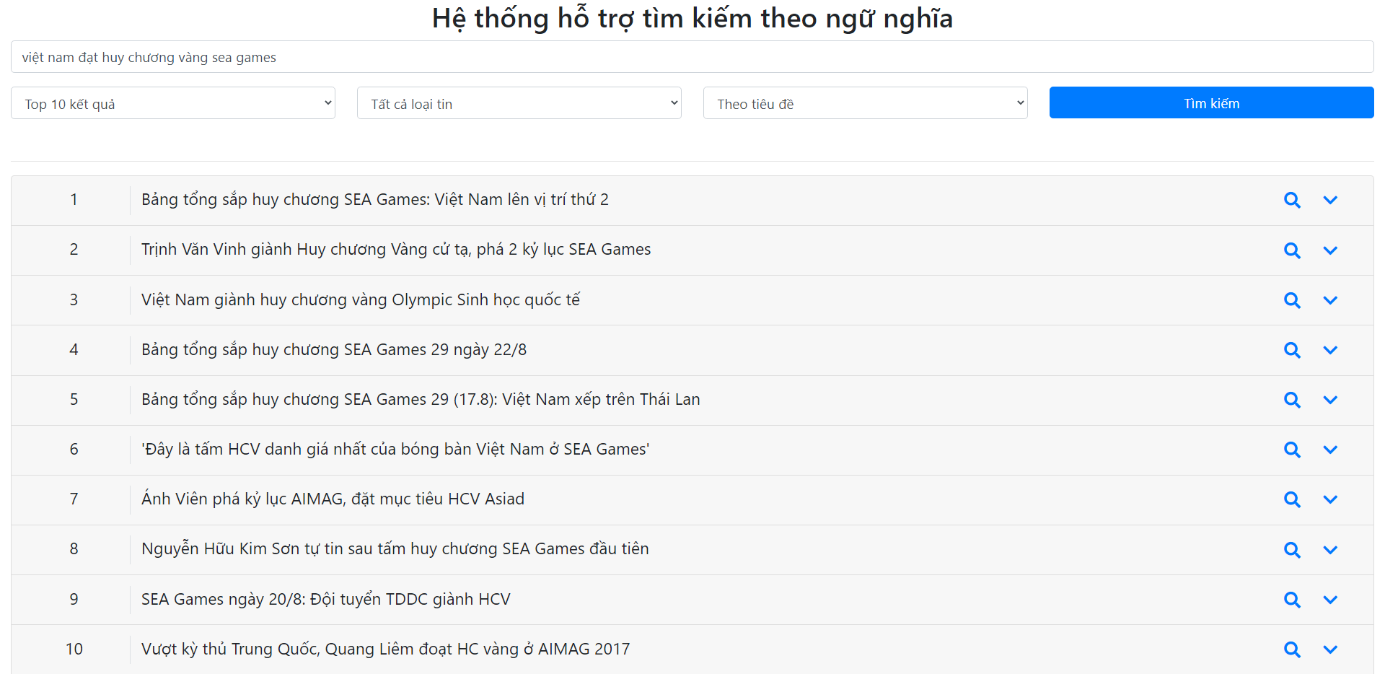
Để truy vấn các văn bản có tiêu đề tương đồng, trước tiên ta cần thực hiện tiền xử lý và truyền mảng các từ vào module biến đổi biểu diễn văn bản. Sau khi thu được biểu diễn vector của tiêu đề, ta thực hiện thao tác tìm kiếm và xếp hạng dựa trên độ đo tương đồng cosine thay vì thuật toán mặc định Okapi BM25.

Đối với Elasticsearch, điểm số đánh giá để xếp hạng kết quả không thể âm (Okapi BM25 dựa trên TF-IDF), vì vậy, khi tính toán độ tương đồng cosine, ta cần cộng thêm 1 để tránh kết quả bị âm.

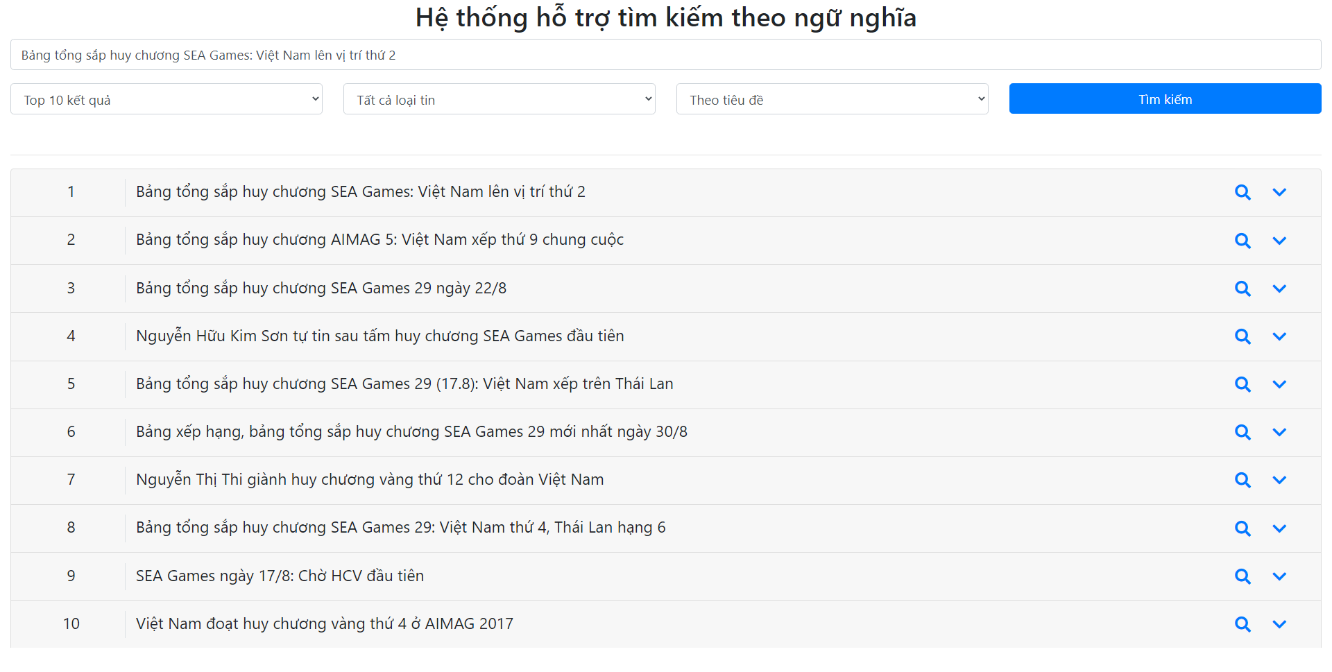
* + 1. Giao diện người dùng



Hình .: Giao diện màn hình chính của hệ thống

****

Hình .: Giao diện màn hình kết quả tìm kiếm

****

Hình .: Giao diện chi tiết của kết quả tìm kiếm

Hình .: Giao diện tiếp tục tìm kiếm theo một kết quả

* 1. Cài đặt hệ thống

File **config.ini**: chứa cấu hình của ứng dụng. File này chứa các cấu hình có khả năng thay đổi theo thời gian thay vì cố định trong mã nguồn. Các cấu hình được lưu là các cấu hình sử dụng để thiết lập kết nối tới Elasticsearch, các tham số kết nối CSDL SQL và tham số mô hình.



File **sql.py**: chứa kết nối tới CSDL SQL Server và các phương thức thao tác với dữ liệu trong CSDL SQL.

Kết nối giữa hệ thống và ứng dụng được quản lý thông qua thư viện **pyodbc** trên ngôn ngữ Python.



Phương thức khởi tạo kết nối giữa hệ thống tìm kiếm và CSDL SQL Server được thiết lập thông qua các tham số được lưu trữ trong file cấu hình gồm tài khoản, mật khẩu người dùng SQL và tên server/địa chỉ IP và tên CSDL.

Module này cung cấp các phương thức truy xuất dữ liệu sau:

* **SQLRecordsCount**: là phương thức lấy số lượng bản ghi tin tức trong CSDL SQL Server.

Câu lệnh truy vấn thông tin trong “sys.dm\_db\_partition\_stats” sẽ nhanh hơn so với sử dụng câu lệnh COUNT để tìm tổng số lượng bản ghi trong trường hợp số lượng bản ghi là rất lớn. Cụ thể phương thức này sẽ thực hiện câu lệnh sau để lấy tổng số lượng bản ghi trong bảng News:



* **GetAllNews**: là phương thức lấy toàn bộ dữ liệu từ bảng News trong CSDL SQL Server.

Phương thức này được sử dụng để chuẩn bị dữ liệu đồng bộ từ CSDL SQL sang Elasticsearch bằng câu truy vấn sau:



* **GetAllCategories**: là phương thức lấy toàn bộ dữ liệu từ bảng NewsCategory sử dụng truy vấn sau:

****

Lớp **ES\_Repository**: lớp thực hiện các phương thức thao tác tới search engine Elasticsearch.

* Khởi tạo các mappings – cấu trúc dữ liệu lưu trữ cho kho dữ liệu Elasticsearch. Cấu trúc dữ liệu được định nghĩa bởi mappings sẽ không thể chỉnh sửa. Các mappings nên được thiết kế kỹ lưỡng và xác định kiểu dữ liệu phù hợp trước khi thực hiện tạo chỉ mục lưu trữ.

Bảng 3.3: Ý nghĩa các trường trong mappings

|  |  |
| --- | --- |
| **Tên trường** | **Ý nghĩa** |
| title | Tiêu đều tin tức |
| abstract | Tóm tắt tin tức |
| body | Nội dung tin tức |
| newscategoryid | Mã loại tin tức |
| title\_embedding | Biểu diễn của tiêu đề |
| abstract\_embedding | Biểu diễn của tóm tắt |



Để có thể nhanh chóng trả về kết quả cho người dùng, Elasticsearch sẽ được sử dụng lưu trữ không chỉ biểu diễn mà còn lưu trữ trực tiếp các trường thông tin trong CSDL SQL. Có thể coi đây là một dạng dư thừa dữ liệu. Tuy nhiên, để đáp ứng được bài toán tìm kiếm, thời gian trả về cần phải là thời gian thực (real time) hoặc nằm trong một khoảng chấp nhận được.

Đối với các biểu diễn, Elasticsearch có hỗ trợ lưu trữ vector với số chiều tối đa lên tới 2048 chiều. Kiểu dữ liệu này trong Elasticsearch được gọi là “dense\_vector”. Để có thể lưu trữ biểu diễn văn bản, cấu trúc dữ liệu cần có các trường dữ liệu “dense\_vector” với số chiều được định sẵn. Đối với cấu trúc dữ liệu trên, số chiều của biểu diễn phụ thuộc vào số chiều được cài đặt của mô hình biến đổi biểu diễn văn bản sử dụng được lưu trong file cấu hình.

Các chỉ mục nên được cấu thành các lớp riêng của chúng. Các lớp đó cần chứa thuộc tính mang giá trị mappings. Khi đó, các chỉ mục lưu trữ sẽ được khởi tạo một cách thuận tiện nếu không tồn tại thông qua hàm **\_\_bootstrap\_index** sau:



* Tạo kết nối tới Elasticsearch

Kết nối tới Elasticsearch sẽ được thiết lập thông qua hàm khởi tạo của lớp **ES\_Repository**. Lớp này sẽ được gọi và sử dụng trong API để duy trì kết nối tới Elasticsearch. Khi thiết lập kết nối thành công, ta sẽ gọi đến các hàm khởi tạo khác, trong đó có hàm khởi tạo chỉ mục lưu trữ **\_\_bootstrap\_index** trên.



* Các phương thức thao tác với chỉ mục









* Kiểm tra số lượng bản ghi trong CSDL SQL và Elasticsearch



* Thực hiện tìm kiếm văn bản dựa vào độ tương đồng cosine và biểu diễn văn bản





File **app.py**:là file chứa ứng dụng chính, cung cấp các REST API. Phần này sẽ tập trung làm rõ các chức năng đặc thù của hệ thống mà không đề cập tới các API cơ bản như thêm/sửa/xóa.

Chức năng tiền xử lý dữ liệu được cài đặt bằng cách sử dụng lớp VnCoreNLP trong thư viện wrapper “vncorenlp”. Ta sẽ sử dụng thư viện này để tải file jar chứa server của VnCoreNLP và thực hiện thao tác tách từ thông qua phương thức tokenize() của lớp VnCoreNLP. Để tránh tốn nhiều thời gian tải file. Khi hệ thống khởi động, file jar VnCoreNLP sẽ được nạp vào trước và cung cấp API tách từ trong suốt quá trình hệ thống chạy. Tương tự, mô hình biểu diễn văn bản sẽ được nạp vào khi hệ thống khởi động và duy trì trong quá trình hoạt động thay vì phải tải lên khi có yêu cầu truy vấn.



Để thực hiện biến đổi biểu diễn văn bản, ta cần tải mô hình ngôn ngữ thu được sau huấn luyện lên bộ nhớ với sự hỗ trợ của thư viện ‘fse’. Sau khi gọi thư viện ‘fse’, một biến chứa thông tin mô hình sẽ được khai báo sử dụng đường dẫn:



Đoạn mã sau thực hiện cung cấp một REST API có khả năng giao tiếp thông qua giao thức HTTP GET để phục vụ cho mục đích tìm kiếm.



Các tham số “text”, “size”, “newscategoryid” và “isTitle” lần lượt tương ứng với chuỗi tìm kiếm, số lượng kết quả trả về, mã định danh loại tin tức và giá trị nhị phân xác định điều kiện tìm kiếm theo tiêu đề hoặc tóm tắt. Các trường dữ liệu lưu trữ biểu diễn nên được loại bỏ do không mang ý nghĩa với người dùng và giảm lượng dữ liệu cần truyền tải.

Đối với chức năng đồng bộ dữ liệu, API đồng bộ sẽ được khóa cho tới khi đồng bộ xong. Điều này nhằm ngăn chặn thực hiện nhiều yêu cầu đồng bộ cùng lúc gây chiếm tài nguyên quá tải phần cứng hệ thống. Đây là một tác vụ chiếm nhiều tài nguyên, vì vậy cần chọn thời điểm thích hợp để thực hiện đồng bộ dữ liệu. Trong khi thực hiện đồng bộ dữ liệu, hệ thống được tích hợp nên được khóa chức năng nhập liệu để đảm bảo dữ liệu được đồng bộ đúng. Chức năng này sẽ không nằm trên giao diện của hệ thống. Hệ thống được tích hợp nên có mục quản lý và cấu hình hệ thống hỗ trợ tìm kiếm, trong đó bao gồm chức năng đồng bộ dữ liệu. Chức năng này cần được phân quyền chỉ cho phép người quản trị được sử dụng để tránh các tình huống lỗi gây ra bởi người dùng.





File **interface.html**: giao diện chính của ứng dụng tìm kiếm.

Giao diện được thiết kế dựa trên nền trình duyệt sử dụng ngôn ngữ HTML kết hợp thư viện Bootstrap để tạo giao diện người dùng tương tác với hệ thống. Trên giao diện người dùng sẽ không có chức năng đồng bộ dữ liệu. Chức năng đồng bộ dữ liệu nên được gọi thông qua hệ thống được tích hợp và thực hiện phân quyền cho chức năng.

Giao diện bao gồm các thành phần chính sau:

* Tiêu đề hệ thống
* 1 input text để nhập dữ liệu truy vấn
* 3 input select để chọn: số lượng kết quả trả về; loại tin tức; sử dụng tìm kiếm theo biểu diễn của tiêu đề hoặc theo biểu diễn của tóm tắt.
* 1 button để thực hiện tìm kiếm
* 1 khung hiển thị kết quả

File **interface.js**: chứa các phương thức gọi và xử lý kết quả của API tìm kiếm và hỗ trợ trình bày kết quả.

Thư viện JQuery được sử dụng để hỗ trợ sử dụng nhanh các phương thức truy cập phần tử HTML và Ajax để gọi và xử lý API.

File này gồm các hàm:

* GetNewsCategory(): gọi tới REST API lấy thông tin về các loại tin tức, thông thường sẽ gọi đến API của hệ thống được tích hợp.
* STS\_Search(): gọi tới REST API truy vấn dữ liệu đồng nghĩa sử dụng chuỗi văn bản truy vấn trong input text và các giá trị được chọn trong các input select.
  1. Kết luận chương 3

Trong chương này, ta đã thực hiện thiết kế khái quát và tiếp cận hệ thống theo hướng microservice. Với kiến trúc này, ta có thể dễ dàng tích hợp và triển khai hệ thống hỗ trợ tìm kiếm theo ngữ nghĩa với các hệ thống quản lý/quản trị hiện hành. Ta chỉ cần thay đổi mô hình biểu diễn sang một mô hình được huấn luyện tương ứng với dữ liệu của lĩnh vực cần thiết. Chương còn đề cập tới những công nghệ như search engine Elasticsearch đã phát triển khả năng lưu trữ các vector biểu diễn và thực hiện tìm kiếm, xếp hạng các văn bản theo mức độ tương đồng ngữ nghĩa.

# THÍ NGHIỆM VÀ NHẬN XÉT

Chương này sẽ trình bày các kết quả thực nghiệm so sánh mức độ tương đồng ngữ nghĩa đạt được thông qua so sánh các biểu diễn văn bản đạt được sử dụng các mô hình biến đổi biểu diễn văn bản.

* 1. Thí nghiệm

**Dữ liệu thí nghiệm**

Bộ dữ liệu “vnPara” [18] được sử dụng để làm cơ sở đánh giá khả năng của các mô hình biến đổi biểu diễn câu.

vnPara [18] bao gồm 3083 cặp câu tiếng Việt kèm theo nhãn đánh dấu hai câu là có đồng nghĩa (1) hay không (0).

Cụ thể hơn, bộ dữ liệu trên gồm 1577 cặp được đánh giá là đồng nghĩa và 1506 cặp được đánh giá là không đồng nghĩa.

**Huấn luyện**

Tùy thuộc vào các phương pháp biến đổi biểu diễn văn bản, các cách tiến hành cài đặt sẽ khác nhau. Sau khi thực hiện loại bỏ các câu trùng lặp trong bộ dữ liệu vnPara [18], bộ dữ liệu huấn luyện thực sự sẽ có 5440 câu.

Đối với riêng Doc2Vec và SIF, các mô hình biểu diễn sẽ được huấn luyện với số chiều đặc trưng lần lượt là 100 và 300.

Toàn bộ mô hình trừ mô hình PhoBERT được huấn luyện sử dụng máy tính cá nhân với vi xử lý CPU có tốc độ 3.4Ghz và sử dụng 12 luồng song song.

Các mô hình sử dụng cấu trúc Transformer thường cồng kềnh, thời gian xử lý sử dụng CPU là rất lớn. Vì vậy, mô hình PhoBERT được fine-tuned bằng Google Colab sử dụng GPU và thời gian huấn luyện sử dụng CPU ược ước lượng thông qua thời gian của một số vòng lặp.

**Kiểm nghiệm**

Các mô hình được thí nghiệm sẽ lần lượt thực hiện biến đổi hai câu trong từng cặp câu và thực hiện so sánh ngữ nghĩa thông qua độ đo cosine. Thông qua một ngưỡng được chọn ta sẽ thực hiện phân loại câu đồng nghĩa (1) và không đồng nghĩa (0) nếu giá trị độ đo lớn hơn hoặc bằng ngưỡng đã chọn. Sau đó, kết quả dự đoán sẽ được so sánh với nhãn của câu để xây dựng một ma trận nhầm lẫn nhằm tính toán các độ đo đánh giá mô hình sử dụng.

**Mô hình PV-DM**

PV-DM là một mô hình học máy không giám sát, vì vậy, mô hình này được huấn luyện trực tiếp trên bộ dữ liệu 5440 câu trên. Cụ thể, mô hình PV-DM sẽ được huấn luyện trên 5440 câu thu được.

**Mô hình SIF**

Mô hình SIF sử dụng một mô hình Word2Vec đã được huấn luyện sẵn. Trong thí nghiệm này, mô hình SIF sẽ sử dụng bộ dữ liệu huấn luyện 5440 câu để thực hiện SVD để tìm “first singular value” hay PC1. Các mô hình SIF được thí nghiệm gồm:

* **SIF base**: sử dụng mô hình Word2Vec được huấn luyện trực tiếp trên 5440 câu của vnPara.
* **SIF pretrain**: sử dụng mô hình Word2Vec được huấn luyện không sử dụng vnPara.
* **SIF fine-tune**: sử dụng mô hình Word2Vec trên nhưng được cập nhật sử dụng 5440 câu của vnPara.

**Mô hình PhoBERT**

Đối với PhoBERT, có hai phương pháp được thí nghiệm:

* PhoBERTBASE: trực tiếp sử dụng mô hình PhoBERTBASE và lấy vector tổng của bốn lớp hidden cuối cùng.
* PhoBERTBASE fine-tune: thực hiện huấn luyện fine-tuning mô hình PhoBERTBASE sử dụng 5440 câu của vnPara và lấy vector tổng của bốn lớp hidden cuối cùng.

Bảng .: Tham số fine-tune mô hình PhoBERT

|  |  |  |  |
| --- | --- | --- | --- |
| **Tham số huấn luyện** | epochs | learning\_rate | batch\_size |
| **Giá trị** | 10 | 1e-5 | 32 |

Trong đó, các tham số mang ý nghĩa:

* epochs: số vòng lặp/số lần mô hình thực hiện huấn luyện toàn bộ dữ liệu;
* learning\_rate: hệ số cập nhật trọng số sử dụng Adam Optimizer;
* batch\_size: số câu được đưa vào mô hình trong một lần cập nhật.
  + 1. Các độ đo sử dụng để đánh giá các mô hình

**Precision**

Precision thể hiện độ chính xác khi dự đoán của mô hình. Độ đo này được tính bằng tỉ lệ các cặp câu có nhãn đồng nghĩa được phân loại đúng trên toàn bộ các câu được mô hình phân loại là đồng nghĩa.

|  |  |  |
| --- | --- | --- |
|  |  | **(5.1)** |

Trong đó:

* : precision;
* : true positive – số lượng các cặp câu được phân loại là đồng nghĩa và thực sự đồng nghĩa;
* : false positive – số lượng các cặp câu được phân loại là đồng nghĩa nhưng không thực sự đồng nghĩa.

**Recall**

Recall hay độ nhạy (sensitivity) đánh giá khả năng tìm kiếm các cặp câu thực sự đồng nghĩa của mô hình – tức là tìm ra được bao nhiêu cặp đồng nghĩa trên tổng số cặp có nhãn là đồng nghĩa.

|  |  |  |
| --- | --- | --- |
|  |  | **(5.2)** |

Trong đó:

* : recall;
* : true positive – số lượng các cặp câu được phân loại là đồng nghĩa và thực sự đồng nghĩa;
* : false negative – số lượng các cặp câu được phân loại là không đồng nghĩa nhưng thực sự đồng nghĩa.

**F-score**

F-score đánh giá độ chính xác của mô hình trên tập dữ liệu kiểm thử, giá trị cao nhất của F-score là 1. Tuy nhiên, F-score cao chỉ chứng minh mức độ chính xác của mô hình trên một tập dữ liệu kiểm thử cụ thể. F-score có công thức chung là:

|  |  |  |
| --- | --- | --- |
|  |  | **(5.3)** |

Trong đó:

* : F-score;
* : hệ số thể hiện mức độ quan trọng của so với ;
* : precision;
* : recall.

Thông thường sẽ nhận các giá trị 0.5, 1 và 2 tương ứng với mức độ quan trọng tăng dần của so với . thể hiện và có mức độ quan trọng tương đương nhau.

**Accuracy**

Accuracy đánh giá độ chính xác khi phân loại của mô hình. Tuy nhiên, accuracy không hẳn là một thước đo tốt để đánh giá mô hình. Accuracy được tính bằng tỉ lệ giữa các cặp được phân loại/gán nhãn đúng trên tổng số cặp cần được gán nhãn hoặc bằng công thức:

|  |  |  |
| --- | --- | --- |
|  |  | **(5.4)** |

Trong đó:

* : true positive – số lượng các cặp câu được phân loại là đồng nghĩa và thực sự đồng nghĩa;
* : false positive – số lượng các cặp câu được phân loại là đồng nghĩa và không thực sự đồng nghĩa;
* : true negative – số lượng các cặp câu được phân loại là không đồng nghĩa và thực sự không đồng nghĩa;
* : false negative – số lượng các cặp câu được phân loại là không đồng nghĩa nhưng thực sự đồng nghĩa.
  + 1. Đánh giá kết quả trên bộ dữ liệu cặp câu tương đồng vnPara
       1. Kết quả đánh giá vnPara bằng các mô hình PV-DM, SIF base, SIF pretrain và SIF fine-tune

Các mô hình được huấn luyện theo các tham số sau:

* d: số chiều của biểu diễn;
* epochs: số vòng lặp huấn luyện của mô hình PV-DM và Word2Vec;
* threshold: ngưỡng được chọn của độ đo cosine;

Bảng .: Kết quả đánh giá các mô hình với d=100, epochs=10, threshold=0.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 score | Accuracy |
| PV-DM | 0.66 | 0.94 | 0.77 | 0.73 |
| SIF base | 0.69 | 0.91 | 0.79 | 0.66 |
| SIF pretrain | 0.53 | 0.93 | 0.68 | 0.90 |
| SIF fine-tune | 0.53 | 0.94 | 0.68 | 0.90 |

Bảng .: Kết quả đánh giá các mô hình với d=100, epochs=10, threshold=0.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 score | Accuracy |
| PV-DM | 0.72 | 0.97 | 0.83 | 0.68 |
| SIF base | 0.75 | 0.93 | 0.83 | 0.63 |
| SIF pretrain | 0.59 | 0.96 | 0.73 | 0.83 |
| SIF fine-tune | 0.59 | 0.97 | 0.73 | 0.84 |

Bảng .: Kết quả đánh giá các mô hình với d=300, epochs=10, threshold=0.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 score | Accuracy |
| PV-DM | 0.68 | 0.95 | 0.79 | 0.70 |
| SIF base | 0.71 | 0.91 | 0.80 | 0.64 |
| SIF pretrain | 0.51 | 0.91 | 0.65 | 0.91 |
| SIF fine-tune | 0.51 | 0.92 | 0.65 | 0.92 |

Bảng .: Kết quả đánh giá các mô hình với d=300, epochs=10, threshold=0.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 score | Accuracy |
| PV-DM | 0.75 | 0.97 | 0.85 | 0.65 |
| SIF base | 0.76 | 0.93 | 0.84 | 0.63 |
| SIF pretrain | 0.55 | 0.96 | 0.70 | 0.88 |
| SIF fine-tune | 0.55 | 0.96 | 0.70 | 0.88 |

* + - 1. Kết quả đánh giá vnPara bằng các mô hình PhoBERTBASE và PhoBERTBASE fine-tune

**Bảng 5.5: Kết quả đánh giá các mô hình PhoBERT với threshold=0.7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 score | Accuracy |
| PhoBERTBASE | 0.55 | 0.86 | 0.67 | 0.79 |
| PhoBERTBASE fine-tune | 0.55 | 0.86 | 0.67 | 0.79 |

**Bảng 5.6: Kết quả đánh giá các mô hình PhoBERT với threshold=0.6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 score | Accuracy |
| PhoBERTBASE | 0.66 | 0.93 | 0.77 | 0.71 |
| PhoBERTBASE fine-tune | 0.66 | 0.93 | 0.77 | 0.71 |

* 1. Đánh giá hiệu năng của các phương pháp trên bộ dữ liệu vnPara

Số lượng câu/văn bản đầu vào huấn luyện là 5440 câu. Các mô hình thực hiện so sánh 3083 cặp câu.

* **D100**: thời gian huấn luyện các mô hình có số chiều biểu diễn là 100
* **D300**: thời gian huấn luyện các mô hình có số chiều biểu diễn là 300
* **FT**: thời gian fine-tune PhoBERT
* **Infer100**: thời gian biến đổi và so sánh cặp câu với biểu diễn 100 chiều
* **Infer300**: thời gian biến đổi và so sánh cặp câu với biểu diễn 300 chiều
* **PB**: thời gian biến đổi và so sánh cặp câu sử dụng 4 lớp hidden cuối của PhoBERT

(\*) Thời gian cộng gộp với thời gian huấn luyện mô hình Word2Vec

(\*\*) Thời gian huấn luyện thông qua sử dụng GPU trên Google Colab. Đối với CPU được ước lượng ~22 tiếng.

Bảng .: Thời gian huấn luyện và biến đổi

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **D100** | **D300** | **FT** | **Infer100** | **Infer300** | **PB** |
| PV-DM | 8.41 | 9.14 | - | 10.23 | 11.83 | - |
| SIF base | 4.19\* | 4.65\* | - | 1.75 | 1.89 | - |
| SIF pretrain | 8.27 | 9.52 | - | 3.00 | 3.39 | - |
| SIF fine-tune | 10.73\* | 21.24\* | - | 2.96 | 3.29 | - |
| PhoBERTBASE | - | - | - | - | - | 537.31 |
| PhoBERTBASE fine-tune | - | - | ~2550\*\* | - | - | 579.75 |

* 1. Nhận xét

Có thể dễ dàng thấy được khi số chiều của biểu diễn của PV-DM và SIF tăng lên, khả năng dự đoán (precision) của mô hình tăng lên tuy với một số mô hình, độ phân loại chính xác (accuracy) giảm đi không đáng kể.

So với một bộ phân loại hai lớp ngẫu nhiên, các kết quả thu được của các mô hình khá khả quan. Thông qua hiệu năng và độ chính xác tương đối, có thể nhận xét rằng mô hình SIF là một mô hình khá tốt cho biểu diễn câu ngắn.

Thực tế, để có thể đánh giá khả năng của một mô hình ngôn ngữ, ta cần nhiều dữ liệu dành cho kiểm thử hơn nữa. Nhận xét trên chỉ có thể mang tính tương đối do tiếng Việt còn thiếu thốn nhiều bộ dữ liệu dán nhãn so khớp ngữ nghĩa.

* 1. Kết luận chương 4

Trong chương 4, ta đã thực hiện so sánh khả năng của các mô hình biểu diễn văn bản. Thông qua đó, có thể kết luận rằng, các mô hình phức tạp chưa hẳn phù hợp và cho kết quả tốt hơn. Để có thể áp dụng một mô hình biểu diễn vào bài toán so khớp ngữ nghĩa, cần phải chú ý lựa chọn các mô hình phù hợp với điều kiện/tài nguyên và dữ liệu. Hầu hết các hệ thống tìm kiếm yêu cầu phải đáp ứng trả kết quả thời gian thực cho người dùng hoặc trong một khoảng thời gian chấp nhận được. Vì vậy, ta cần chọn mô hình dựa trên sự cân đối về mức độ chính xác và hiệu năng.

# KẾT LUẬN

Bài toán biến đổi biểu diễn văn bản và so khớp ngữ nghĩa không phải là những bài toán mới, tuy nhiên bài toán so khớp ngữ nghĩa luôn là một bài toán khó. Kết quả đạt được với bộ dữ liệu trên vẫn chưa đủ để có thể đưa ra kết luận về khả năng thực sự của các mô hình ngôn ngữ đã được nghiên cứu và thí nghiệm trong báo cáo này.

Mục đích của báo cáo này là thực hiện thí nghiệm tính khả thi khi triển khai một mô hình ngôn ngữ phục vụ cho bài toán tìm kiếm so khớp ngữ nghĩa thông qua hiệu năng cũng như khả năng gán nhãn đồng nghĩa đúng. Đồng thời, việc tìm hiểu và cập nhật các phương pháp, công nghệ mới hỗ trợ nhiều hơn cho bài toán biểu diễn văn bản và so khớp ngữ nghĩa là rất cần thiết.

Để bài toán so khớp ngữ nghĩa có thể được cải thiện cần có những bộ dữ liệu tốt được gán nhãn đồng nghĩa theo mức độ thay vì gán nhãn nhị phân. Tiếng Việt còn thiếu nhiều bộ dữ liệu tương đồng ngữ nghĩa để kiểm nghiệm cũng như huấn luyện cho các mô hình ngôn ngữ phục vụ cho bài toán so khớp ngữ nghĩa.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | N. T. M. Huyen, L. H. Phuong, H. T. Vinh and A. Roussanaly, "A Hybrid Approach to Word Segmentation of Vietnamese Texts," in *LATA 2008: Language and Automata Theory and Applications*, 2008. |
| [2] | L. H. Phuong, "vnTokenizer," GitHub, 2009. |
| [3] | L. H. Phuong, "Vitk - A Vietnamese Text Processing Toolkit," GitHub, 2015. |
| [4] | L. H. Phuong, "VLP - Vietnamese Language Processing toolkit," 2020. |
| [5] | N. Q. Dai, N. Q. Dat, V. Thanh, M. Dras and M. Johnson, "A Fast and Accurate Vietnamese Word Segmenter," in *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, 2018. |
| [6] | N. Q. Dai, N. Q. Dat, V. Thanh, M. Dras and M. Johnson, "VnCoreNLP: A Vietnamese Natural Language Processing Toolkit," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations,* pp. 56-60, 2018. |
| [7] | G. Carrado, K. Chen, J. Dean and T. Mikolov, "Efficient Estimation of Word Representations in Vector Space," *CoRR,* 2013. |
| [8] | P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics,* vol. 5, pp. 135-146, 2017. |
| [9] | A. N. Gomez, L. Jones, Ł. Kaiser, N. Parmar, I. Polosukhin, J. Uszkoreit, A. Vaswani and N. Shazeer, "Attention is All you Need," *Advances in Neural Information Processing Systems,* vol. 30, pp. 5998-6008, 2017. |
| [10] | M.-W. Chang, J. Devlin, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* vol. 1, pp. 4171-4186, 2019. |
| [11] | D. Chen, J. Du, N. Goyal, M. Joshi, O. Levy, M. Lewis, Y. Liu, M. Ott, V. Stoyanov and L. Zettlemoyer, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692,* 2019. |
| [12] | N. Q. Dat and N. A. Tuan, "PhoBERT: Pre-trained language models for Vietnamese," *Findings of the Association for Computational Linguistics: EMNLP 2020,* pp. 1037-1042, 2020. |
| [13] | L. V. Quoc and T. Mikolov, "Distributed Representations of Sentences and Documents," *Proceedings of the 31st International Conference on International Conference on Machine Learning,* vol. 32, pp. II-1188–II–1196, 2014. |
| [14] | S. Arora, Y. Liang and T. Ma, "A Simple but Tough-to-Beat Baseline for Sentence Embeddings," in *ICRL*, 2017. |
| [15] | V. Q. Bình, "News Corpus," GitHub. |
| [16] | R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," *Proceedings of the LREC 2010 Workshop on New,* pp. 45-50, 2010. |
| [17] | O. Borchers, "Fast sentence embeddings," GitHub, 2019. |
| [18] | N. X. Bach, N. T. Hai, T. T. Oanh and T. M. Phuong, "Paraphrase Identification in Vietnamese Documents," *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE),* pp. 174-179, 2015. |
| [19] | Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692,* 2019. |
| [20] | J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* vol. 1, pp. 4171-4186, 2019. |
| [21] | A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is All you Need," *Advances in Neural Information Processing Systems,* vol. 30, pp. 5998-6008, 2017. |