**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**

**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 2020**

**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 2020**

|  |  |
| --- | --- |
| 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** |

**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:

..........................................................................................................................................................................................................................................................

3. Nội dung bản thuyết minh:

..........................................................................................................................................................................................................................................................

4. Số lượng, nội dung các bản vẽ (ghi rõ loại, kích thước và cách thực hiện các bản vẽ) và các sản phẩm cụ thể (nếu có):

................................................................................................................................................................................................

5. Cán bộ hướng dẫn (ghi rõ họ tên, cấp bậc, chức vụ, đơn vị, hướng dẫn toàn bộ hay từng phần): ……..................................................................................

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)

**MỤC LỤC**

[**Chương 1:** **Giới thiệu** 1](#_Toc61260172)

[**1.1. Giới thiệu đề tài** 1](#_Toc61260173)

[**1.2. Mục tiêu của đề tài** 2](#_Toc61260174)

[**1.3. Cấu trúc đồ án** 2](#_Toc61260175)

[**Chương 2:** **Tổng quan** 3](#_Toc61260176)

[**2.1. Bài toán tách từ (Word Segmentation)** 3](#_Toc61260177)

[**2.1.1. Mô hình tách từ Maximum Matching** 4](#_Toc61260178)

[**2.1.2. Tách từ kết hợp từ điển và thống kê (Hybrid Approach)** 4](#_Toc61260179)

[**2.1.3. Mô hình tách từ tự động sửa sai sử dụng cây nhị phân SCRDR (Single Classification Ripple Down Rules)** 7](#_Toc61260180)

[**2.2. Bài toán biến đổi biểu diễn từ (Word Embedding)** 11](#_Toc61260181)

[**2.2.1. Biến đổi biểu diễn từ dựa vào tần suất (frequency-based embedding)** 11](#_Toc61260182)

[**2.2.2. Biến đổi biểu diễn từ dựa vào dự đoán (prediction-based embedding)** 15](#_Toc61260183)

[**2.3. Bài toán biến đổi biểu diễn văn bản (Document Embedding)** 28](#_Toc61260184)

[**2.3.1. Mô hình bag-of-words (BOW) cổ điển** 28](#_Toc61260185)

[**2.3.1. Biểu diễn văn bản sử dụng vector TF-IDF** 29](#_Toc61260186)

[**2.3.2. Biểu diễn văn bản sử dụng mô hình Doc2Vec (paragraph vector)** 29](#_Toc61260187)

[**2.3.3. Biểu diễn văn bản sử dụng mô hình biểu diễn từ trong không gian vector** 31](#_Toc61260188)

[**2.4. Bài toán so khớp ngữ nghĩa (Semantic Similarity)** 34](#_Toc61260189)

[**2.5. Kết luận về các mô hình biểu diễn từ và biểu diễn văn bản** 35](#_Toc61260190)

[**Chương 3:** **Cơ sở lý thuyết** 37](#_Toc61260191)

[**3.1. Mạng neuron nhân tạo** 37](#_Toc61260192)

[**3.2. Cơ sở lý thuyết về văn bản và biểu diễn văn bản** 40](#_Toc61260193)

[**3.2. Cơ sở lý thuyết về so khớp ngữ nghĩa** 41](#_Toc61260194)

[**3.3. Lý thuyết về Singular Value Decomposition (SVD) và Principal Component Analysis (PCA)** 41](#_Toc61260195)

[**3.3. Nhận xét** 43](#_Toc61260196)

[**Chương 4:** **Mô hình – Thiết kế - Cài đặt** 44](#_Toc61260197)

[**4.1. Chuẩn bị ngữ liệu** 44](#_Toc61260198)

[**4.2. Thiết kế tổng quan của hệ thống** 44](#_Toc61260199)

[**4.2.1. Thiết kế tổng quan** 44](#_Toc61260200)

[**4.2.2. Module dữ liệu** 45](#_Toc61260201)

[**4.2.3. Module API truy vấn** 48](#_Toc61260202)

[**4.2.4. Giao diện người dùng** 50](#_Toc61260203)

[**4.3. Cài đặt hệ thống** 51](#_Toc61260204)

[**Chương 5:** **Kết quả thí nghiệm và nhận xét** 61](#_Toc61260205)

[**5.1. Thí nghiệm** 61](#_Toc61260206)

[**5.1.1. Các độ đo sử dụng để đánh giá mô hình** 62](#_Toc61260207)

[**5.1.2. Đánh giá kết quả trên bộ dữ liệu cặp câu tương đồng vnPara** 64](#_Toc61260208)

[**5.2. Đánh giá hiệu năng của các phương pháp trên bộ dữ liệu vnPara** 66](#_Toc61260209)

[**5.3. Nhận xét** 67](#_Toc61260210)

[**Kết luận** 68](#_Toc61260211)

[**Tài liệu tham khảo** 69](#_Toc61260212)

**DANH MỤC HÌNH ẢNH**

|  |  |
| --- | --- |
| **Hình 2.1: Đồ thị của một câu…………………………………………….** | **4** |
| **Hình 2.2: Cây khởi tạo SCRDR………………………………………….** | **9** |
| **Hình 2.3: Mô hình CBOW và Skip-gram……………………………….** | **15** |
| **Hình 2.4: Mô hình Transformer…………………………………………** | **21** |
| **Hình 2.5: Mô hình Masked LM………………………………………….** | **26** |
| **Hình 2.6: Chuẩn bị đầu vào cho NPS……………………………………** | **26** |
| **Hình 2.7: Mô hình PV-DM……………………………………………….** | **29** |
| **Hình 2.8: PV-DBOW……………………………………………………..** | **29** |
| **Hình 3.1: Cấu tạo sinh học của một neuron…………………………….** | **37** |
| **Hình 3.2: Mô hình của một perceptron………………………………….** | **38** |
| **Hình 3.3: Giá trị đầu ra của một perceptron……………………………** | **38** |
| **Hình 3.4: Mô hình ví dụ về một mạng neuron nhân tạo cơ bản……….** | **39** |
| **Hình 4.1: Thiết kế tổng quan của hệ thống………………………………** | **45** |
| **Hình 4.2: Thiết kế cơ sở dữ liệu SQL mô phỏng…………………………** | **46** |
| **Hình 4.3: Giao diện màn hình chính của hệ thống………………………** | **50** |
| **Hình 4.4: Giao diện màn hình kết quả tìm kiếm………………………..** | **50** |
| **Hình 4.5: Giao diện chi tiết của kết quả tìm kiếm………………………** | **51** |
| **Hình 4.6: Giao diện tiếp tục tìm kiếm theo một kết quả……………….** | **51** |

**DANH MỤC BẢNG BIỂU**

|  |  |
| --- | --- |
| **Bảng 2.1: Mô tả về các kho ngữ liệu…………………………………….** | **8** |
| **Bảng 2.2: Ví dụ về từ điển ngữ cảnh 5 âm tiết được xây dựng………..** | **8** |
| **Bảng 2.3: Mẫu qui tắc……………………………………………………** | **9** |
| **Bảng 2.4: Ví dụ về ma trận count vector………………………………..** | **12** |
| **Bảng 2.5: Bảng số lượng văn bản thuộc D chứa từ/cụm từ…………....** | **13** |
| **Bảng 2.6: Bảng trình bày biểu diễn văn bản sử dụng TF-IDF cho tập D……………………………………………………………………………** | **14** |
| **Bảng 2.7: Bảng tần suất xuất hiện của các từ trong từng câu thuộc D..** | **28** |
| **Bảng 4.1: Bảng NewsCategory…………………………………………..** | **46** |
| **Bảng 4.2: Bảng News……………………………………………………..** | **47** |
| **Bảng 5.1: Kết quả đánh giá các mô hình với d=100, epochs=10, threshold=0.7……………………………………………………………...** | **65** |
| **Bảng 5.2: Kết quả đánh giá các mô hình với d=100, epochs=10, threshold=0.6……………………………………………………………...** | **65** |
| **Bảng 5.3: Kết quả đánh giá các mô hình với d=300, epochs=10, threshold=0.7……………………………………………………………...** | **65** |
| **Bảng 5.4: Kết quả đánh giá các mô hình với d=300, epochs=10, threshold=0.6……………………………………………………………...** | **65** |
| **Bảng 5.5: Kết quả đánh giá các mô hình PhoBERT với threshold=0.7** | **66** |
| **Bảng 5.6: Kết quả đánh giá các mô hình PhoBERT với threshold=0.6** | **66** |
| **Bảng 5.7: Thời gian huấn luyện và biến đổi…………………………….** | **66** |

# **Giới thiệu**

## **1.1. Giới thiệu đề tài**

Máy tính hỗ trợ con người thực hiện nhiều công việc liên quan đến tính toán, thống kê và so sánh. Trong đó, so sánh được sử dụng nhiều nhất trong tác vụ tìm kiếm thông tin cho con người. Đối với máy tính, mọi biểu diễn đều được quy đổi về dạng số bao gồm các ký tự trong bảng chữ cái 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. Do đó, máy tính sẽ gặp khó khăn trong việc so sánh ngữ nghĩa của câu.

Ngày nay, con người thực hiện nhiều tác vụ mà không thể thiếu được máy tính để lưu trữ và tìm kiếm dữ liệu. Việc biến đổi biểu diễn văn bản được ứng dụng nhiều nhất trong so sánh độ tương đồng ngữ nghĩa (semantic similarity) và dịch máy (machine translation).

Hầu hết các công cụ tìm kiếm sẽ tìm kiếm theo từ khóa và xếp hạng kết quả theo tần suất xuất hiện của từ khóa trong các tài liệu, chính là so sánh về mặt hình thức. Khi tìm kiếm, ta mong muốn thu được kết quả bao gồm một hoặc nhiều văn bản trong đó có những văn bản đã thông qua biến đổi câu từ (paraphrasing) nhưng biểu đạt chung ý nghĩa.

Điều này cũng có ý nghĩa rất lớn trong dịch máy giữa các ngôn ngữ mà vẫn đảm bảo ý nghĩa của câu văn ban đầu. Bằng cách biến đổi biểu diễn hình thức của văn bản về một dạng biểu diễn chung, máy tính sẽ có khả năng thực hiện so sánh và thay thế.

Vì vậy việc biến đổi văn bản kèm theo ý nghĩa/ngữ cảnh về dạng số là cần thiết để máy tính có thể thực hiện phép toán với biểu diễn bằng số đó để so sánh được ý nghĩa của văn bản. Cụ thể hơn, đề tài sẽ giúp ích cho công việc tìm kiếm các văn bản có ngữ nghĩa tương đồng với văn bản đầu vào.

## **1.2. Mục tiêu của đề tài**

Đề tài tập trung vào việc nghiên cứu, nắm bắt các kiến thức về các kỹ thuật/phương pháp 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. Để đạt được mục tiêu của đề tài cần xác định được các công việc phải giải quyết như sau:

* Tìm hiểu các phương pháp/mô hình đã được sử dụng.
* Lựa chọn thí nghiệm 2-3 phương pháp/mô hình.
* Thực hiện huấn luyện, kiểm thử và đánh giá với các mô hình đã chọn.
* Xây dựng hệ thống hỗ trợ tìm kiếm văn bản tương đồng ngữ nghĩa với câu truy vấn.

## **1.3. Cấu trúc đồ án**

Các công việc được thực hiện sẽ được trình bày trong báo cáo theo cấu trúc như sau:

* **Chương 1. Giới thiệu**: khái quát về đề tài, ý nghĩa và ứng dụng thực tế của biến đổi biểu diễn văn bản, cũng là chương hiện hành.
* **Chương 2. Tổng quan**: tình hình trong và ngoài nước của đề tài và các bài toán liên quan.
* **Chương 3.** **Cơ sở lý thuyết**:
* **Chương 4.** **Mô hình – Thiết kế – Cài đặt**: chi tiết về ứng dụng các phương pháp biểu diễn văn bản; thiết kế, cài đặt hệ thống biến đổi biểu diễn văn bản và ứng dụng vào bài toán so sánh ngữ nghĩa
* **Chương 5**. **Kết quả và kết luận**

# **Tổng quan**

Chương này sẽ giới thiệu tổng quan về bài toán biến đổi biểu diễn văn bản và ứng dụng trong việc so khớp ngữ nghĩa bao gồm về phân tích về các bài toán liên quan như: bài toán tách từ, biến đổi biểu diễn từ, biến đổi biểu diễn câu/văn bản, so sánh mức độ tương đồng ngữ nghĩa. Trong mục này chủ yếu đề cập tới các phương pháp tiếp cận của một số công trình nghiên cứu có ý nghĩa lớn với xử lý văn bản tiếng Việt và một vài trong số đó được được coi là SOTA hay state-of-the-art (tân tiến nhất).

## **2.1. Bài toán tách từ (Word Segmentation)**

Bài toán tách từ là một bài toán quan trọng trong xử lý ngôn ngữ tiếng Việt. Mỗi một từ tiếng Việt được cấu thành bởi một hoặc nhiều âm tiết, vì vậy không thể đơn giản tách các âm tiết dựa vào khoảng trắng như các ngôn ngữ hệ Latin khác. Các vấn đề cần giải quyết trong tách từ tiếng Việt chủ yếu gồm:

* Nhập nhằng (ambiguity)
* Từ mới (tên viết tắt, tên riêng, các từ Việt hóa)
* Đầu vào không được chuẩn hóa (lỗi chính tả, teencode, tên riêng viết thường)

Trong đó, nhập nhằng khi tách từ đã được xử lý bằng nhiều phương pháp mà chủ yếu là thông qua liệt kê toàn bộ các cách tách từ có thể xảy ra bằng Maximum Matching dựa vào từ điển tiếng Việt (Vietnamese lexicon) và thường có độ chính xác cao. Từ mới và các thuật ngữ tiếng Việt mới xuất hiện ngày yêu cầu từ điển sử dụng phải được cập nhật liên tục. Khó khăn về chuẩn hóa đa phần đến từ phương tiện truyền thông internet mà điển hình là mạng xã hội. Các phương pháp tiếp cận sắp được đề cập chủ yếu đưa ra phương pháp giải quyết nhập nhằng (vấn đề chính trong tách từ tiếng Việt).

### **2.1.1. Mô hình tách từ Maximum Matching**

Phương pháp tách Maximum Matching là một phương pháp đơn giản, là một cách tiếp cận sử dụng từ điển để tách từ. Tại vị trí bắt đầu tách từ, nếu từ có một âm tiết tồn tại trong từ điển tiếng Việt (Vietnamese lexicon) thì ta sẽ ghép âm tiết tiếp theo vào âm tiết trước đó, nếu vẫn tồn tại, ta xét tiếp với số lượng âm tiết tăng dần. Nếu từ có *N* (*N* ≥ 2*)* âm tiết không tồn tại thì ta sẽ lấy *N-1* âm tiết và tách ra khỏi văn bản. Tuy nhiên, phương pháp này có thể dẫn đến việc tách từ sai do tiếng Việt có nhiều từ gây “nhập nhằng chồng” (Overlapping Ambiguity) và “nhập nhằng hợp” (Combination Ambiguity).

Ví dụ:

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

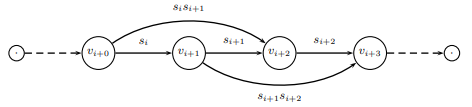
Phương pháp này bao gồm “so khớp cực đại tiến” (Forward Maximum Matching – FMM) và “so khớp cực đại lùi” (Backward Maximum Matching – BMM). Dựa vào sự khác biệt giữa hai kết quả ta có thể biết được vị trí nhập nhằng.

Ví dụ: Học sinh học sinh vật học.

**FMM**: Học\_sinh học\_sinh vật học .

**BMM**: Học sinh\_học sinh\_vật\_học .

### **2.1.2. Tách từ kết hợp từ điển và thống kê (Hybrid Approach)**

Mô hình được đề xuất [1] 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.

**Hình 2.1: Đồ thị của một câu**

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 .

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. Với chuỗi được cấu thành bởi từ (âm tiết trong tiếng Việt) ():

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

Đối với mô hình bigram () đã chọn trong bài báo: xác suất của từ đứng sau chỉ phụ thuộc vào từ đứng trước đó và là . 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). 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, ta cần ước lượng hai giá trị và . Với một bộ dữ liệu kiểm thử (test set) là với câu , ta có xác suất của bộ dữ liệu là . Khi đó, entropy được tính bằng công thức:

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

Trong đó:

* : số lượng từ của bộ dữ liệu .

Từ bộ dữ liệu huấn luyện ta tách ra một phần thành một bộ dữ liệu kiểm định (validation data), cần ước lượng để tìm cực đại của hàm:

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

Trong đó:

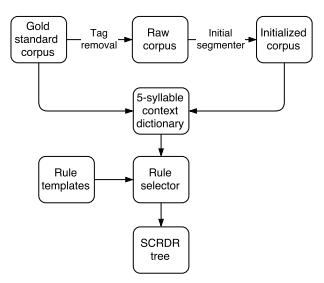
* : số lần xuất hiện của bigram trong tập kiểm định;
* và .

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%.

Mô hình đã được các tác giả xây dựng thành một phần mềm mã nguồn mở với phiên bản đầu tiên là “vnTokenizer” [15] trên Java. Sau đó phần mềm được cập nhật và mang tên “vitk – A Vietnamese Text Processing Tookit” [16] 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” [17]. Các mã nguồn này đều có sẵn trên Github kèm theo hướng dẫn tích hợp qua API.

### **2.1.3. Mô hình tách từ tự động sửa sai sử dụng cây nhị phân SCRDR (Single Classification Ripple Down Rules)**

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). 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. Bộ ngữ liệu thô (raw corpus) – thay thế các dấu “\_” tách từ bằng khoảng trắng (whitespace) được xây dựng từ “gold standard corpus” trên. 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 mục trước (Le et al. 2008) và module này được gọi là “initial segmenter” (bộ tách từ khởi đầu).



**Hình 2.1: Sơ đồ phương pháp tách từ kết hợp cây nhị phân SCRDR**

Ví dụ:

|  |  |
| --- | --- |
| 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 2.1: Mô tả về các kho ngữ liệu**

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. 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.

Đối với mỗi âm tiết, ta xây dựng một tuple có chứa 5 âm tiết với 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 **“”** (hai dấu ngoặc kép). Value sẽ tương ứng với BI tag của âm tiết đó trong BI-formed gold standard corpus. 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.

Ví dụ:

**Bảng 2.2: 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 | **🗶** |

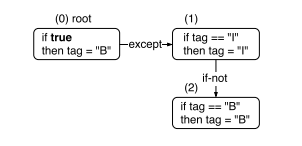
Một cây SCRDR là một cây nhị phân với hai loại nhánh: “ngoại trừ” (except) và “ngược lại” (if-not). Mỗi một nút (node) trên cây tương ứng với một “quy tắc” (rule). Quy tắc có dạng: “*nếu thì ”* với là “điều kiện” (condition) và là “kết luận” (conclusion). Để đảm bảo cây luôn có kết luận, nút gốc (root) 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 2.3: Mẫu quy 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 2.2 và sinh ra các quy tắc (rules) để chỉnh sửa những vị trí có BI tag không đúng với gold standard corpus. Những quy tắc cụ thể này được sinh tự động theo mẫu quy tắc (rules templates).

**Hình 2.2: 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), ta xây dựng một nút cây với điều kiện:

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

THEN tag = **gold-standard-tag**

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. Ta sẽ thu được một tập hợp các quy tắc cụ thể sau khi huấn luyện. Áp dụng mẫu (s-1, s0) với hàng thứ 4 của Bảng 2.2 ta có:

IF s-1 == **“học”**&& s0 == “**sinh”**

THEN tag = “**B”**

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 2.2 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.
* Quy tắc được chọn phải cho giá trị là cao nhất. 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

Sau khi thu được cây SCRDR, ta sẽ thực hiện tách từ như sau: đầu vào là một câu chưa được tách từ, sau khi sử dụng bộ tách từ bằng phương pháp “longest matching” ta có thể lấy được câu gán nhãn BI. Với độ rộng cửa sổ bằng 5, ta lấy các tuples của câu/văn bản cần tách từ để sinh ra bộ từ điển ngữ cảnh 5 âm tiết. Lúc này 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 và đã trở thành một phần trong bộ thư viện hỗ trợ xử lý tiếng Việt mang tên VNLPCore. Mã nguồn này có sẵn trên Github kèm theo hướng dẫn tích hợp qua API.

## **2.2. Bài toán biến đổi biểu diễn từ (Word Embedding)**

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. Các phương pháp chủ yếu bao gồm: frequency-based embedding (dựa vào tần suất), prediction-based embedding (dựa vào dự đoán).

### **2.2.1. Biến đổi biểu diễn từ dựa vào tần suất (frequency-based embedding)**

Các phương pháp sau đây đều có một nhược điểm chung là khi biến đổi biểu diễn từ, các biểu diễn không mô tả được ngữ nghĩa (semantic meaning) của từ và loại bỏ hoàn toàn thứ tự của từ trong câu.

**2.2.1.1. 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. Coi rằng ta 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: V = (học\_sinh, học, sinh\_vật\_học)

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

**2.2.1.2. 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 , ta có thể 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: D = {“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.”}

Vậy ta sẽ có một ma trận ghi lại số lần xuất hiện của từng từ trong với số chiều tương ứng với số lượng văn bản thuộc .

**Bảng 2.4: 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 |

Dựa vào ma trận trên, ta có vector từ tương ứng của “thuốc lá” là (2,0) và “sức khỏe” là (1,1).

**2.2.1.3. TF-IDF (Term Frequency – Inversed Document Frequency)**

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 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

**Bảng 2.5: 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.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

**Bảng 2.6: Bảng trình bày biểu diễn văn bản sử dụng TF-IDF cho tập D**

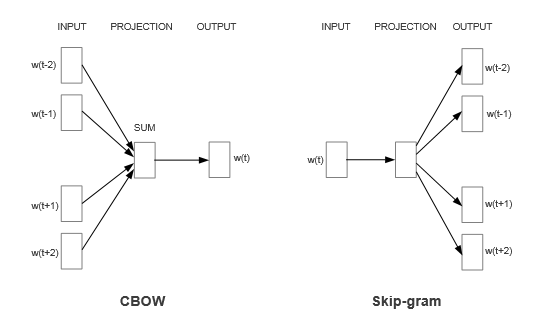
Dựa vào công thức trên, giá trị TF của từng từ sẽ được tính toán dựa trên tần suất xuất hiện của từng từ cho mỗi văn bản. Giá trị IDF sẽ được sử dụng chung với toàn bộ từ vựng. Bảng 6 là ví dụ cho cách tính TF-IDF từng từ của mỗi văn bản trong tập *D*.

### **2.2.2. Biến đổi biểu diễn từ dựa vào dự đoán (prediction-based embedding)**

Mục này sẽ đề cập đến các mô hình học tự giám sát (self-supervised learning) để xây dựng không gian từ (word space).

**2.2.2.1. Mô hình xây dựng biểu diễn từ trong không gian vector Word2Vec [3]**

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. Hai mô hình được giới thiệu lần lượt là CBOW (continuous bag-of-words) và skip-gram. 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 từ. Những từ được sử dụng với cùng ngữ cảnh thường sẽ có mức độ ý nghĩa tương quan nhất định.

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 .

**Hình 2.3: Mô hình CBOW và Skip-gram**

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, 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. 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. Trong báo cáo này, ta sẽ tập trung tìm hiểu mô hình CBOW và ứng dụng của nó trong việc biến đổi biểu diễn văn bản.

**Mô hình huấn luyện CBOW [3]**

Mục đích của mô hình này là học về ngữ cảnh/ý nghĩa của từ được sử dụng có khả năng đi kèm với những từ nào, những từ có khả năng thay thế cho nó sẽ có ý nghĩa tương quan với từ đó.

Với cấu trúc mạng được mô tả trong hình 2.3, 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 ta sẽ chỉ lấy từ phía sau hoặc phía trước nó.

Gọi số lượng từ vựng trong bộ dữ liệu là *V*, số lượng neuron trong lớp hidden (tương ứng lớp projection trong hình 2.3) là *N* (số chiều của không gian vector cần xây dựng), ta có đầu vào của mạng neuron là các vector one-hot encoding xT (). 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.

Ta tính được:

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

Trong đó:

* : vector output trước khi được xử lý softmax.

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

|  |  |  |
| --- | --- | --- |
|  | với *j =0, 1, … N* | **(2.10)** |

Trong đó:

* : giá trị của neuron output (lớp softmax).

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, ta có hàm mất mát (log loss likelihood):

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

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. Ta sử dụng thuật toán lan truyền ngược (backpropagation) và thuật toán giảm gradient (gradient descent) 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 (giá trị 1) 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.

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. 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, mối một hàng ma trận trọng số tương ứng với vector của từ trong bộ từ vựng. Do đó, ta chỉ cần tìm vị trí của từ cần biểu diễn và truy xuất giá trị hàng tương ứng của ma trận .

**2.2.2.2. Mô hình FastText [4]**

Mô hình này được phát triển bởi Facebook với mục đích cải tiến mô hình skip-gram từ Word2Vec. Như đã biết, để có thể huấn luyện mô hình Word2Vec ta cần xây dựng một bộ từ điển từ vựng (vocabulary – vocab) 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 vocab. Những từ không có trong vocab 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, ví dụ như “apple” sẽ được tách thành (app, ppl, ple). Lúc này, biểu diễn vector của từ “apple” sẽ là tổng vector biểu diễn của (app, ppl, ple). Phương pháp này giúp xử lý tốt trong các trường hợp mô hình nhận các từ đầu vào chưa được huấn luyện.

**2.2.2.3. Mô hình Transformer [5]**

Transformer là một mô hình xử lý ngôn ngữ tự nhiên phức tạp thông qua việc sử dụng cơ chế “attention” (attention mechanism). Nhờ cơ chế này, Transformer có khả năng tạo ra các biểu diễn riêng cho cùng một từ với các ngữ cảnh khác nhau. 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 tác giả sử dụng các hyperparameter như sau: vector biểu diễn từ 512 chiều, 6 encoders, 6 decoders, 8 attention-heads). Transformer được coi là một bước tiến lớn trong lĩnh vực dịch máy (machine translation).

Mô hình này gồm 2 thành phần chính: *N* encoder (mã hóa) và *N* decoder (giải mã). Encoder phụ trách biến đổi biểu diễn của từ thành các vector và tạo thành đầu vào cho decoder. Cấu trúc của một encoder và decoder lần lượt được biểu diễn bởi nửa trái và phải trong Hình 2.4. Các từ đầu vào được xử lý tuần tự từ trái sang phải thông qua các lớp như diễn giải dưới đây.

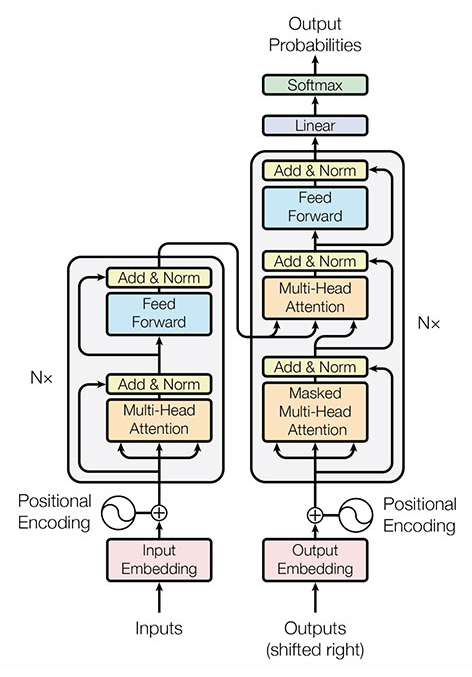
**Inputs**

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 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. Các PE được chuẩn hóa trong khoảng (0, 1). PE được tính khác nhau dựa theo vị trí chẵn lẻ của các phần tử trong vector như sau:

**Hình 2.4: Mô hình Transformer**

|  |  |  |
| --- | --- | --- |
|  |  | **(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. Từ Hình 2.4, khi đi vào encoder, ta thấy word embedding được chia làm ba nhánh. Ba nhánh này tương ứng với 3 ma trận trọng số lần lượt là . 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). Ta có điểm số của từng từ (lấy nhân với từng một):

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

Tiếp theo ta tính giá trị (giúp gradient ổn định hơn) 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 khả năng cao là hai từ có liên quan mật thiết tới nhau.

Với bước cuối cùng, ta lấy tổng các giá trị 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.

**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.

**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.

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

Trong đó:

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

**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”. Lớp này giúp decoder che đi (không tính toán giá trị) “attention” của những từ chưa được xử lý.

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.

**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” bằng cách đạo hàm như Word2Vec, ta sử dụng Adam Optimizer để cập nhật trọng số cho mô hình với công thức như sau:

|  |  |
| --- | --- |
|  | **(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.

**2.2.2.4. Mô hình BERT (Bidirectional Encoder Representations from Transformers) [6]**

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) – nghĩa là tập trung chủ yếu vào việc xây dựng không gian vector biểu diễn của từ.

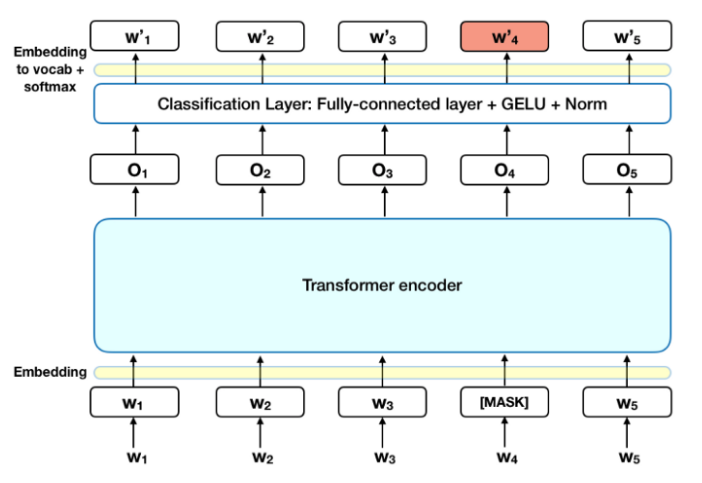
Với mô hình Transformer, chuỗi đầu vào được xử lý tuần tự từ trái sang phải (mô hình directional – một chiều). Về cơ bản BERT sử dụng chính Transformer, tuy nhiên, BERT lấy toàn bộ đầu vào cùng lúc, vì vậy BERT được coi là mô hình hai chiều (bidirectional) để có thể lấy ngữ cảnh của một từ từ những từ xung quanh (context words). 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). 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). 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.

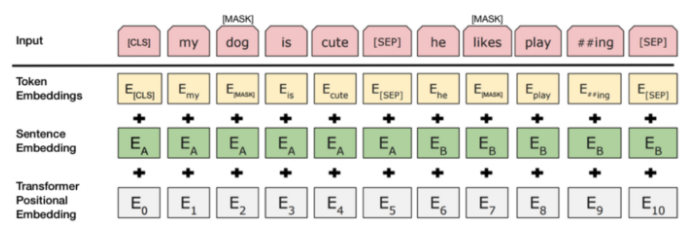
Cấu trúc của encoder Transformer được giữ nguyên. Nối tiếp với encoder là một lớp classification (phân lớp). Lớp này là một mạng neuron hoàn chỉnh (fully connected) sử dụng hàm kích hoạt GELU và chuẩn hóa đầu ra.

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.

**Hình 2.5: Mô hình Masked LM**

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.

**Next Sentence Prediction**

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.

**Hình 2.6: Chuẩn bị đầu vào cho NPS**

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.

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”. Xét cả hai câu ta tìm Sinusoidal Positional Embedding giống Transformer.

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.

**2.2.2.5. Mô hình ngôn ngữ tiếng Việt SOTA 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). Vì vậy, một phiên bản cải tiến của nó là RoBERTa (Robustly optimized BERT approach) đã được xây dựng [7]. RoBERTa được huấn luyện với 161GB dữ liệu từ nhiều nguồn khác nhau [7].

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). 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 [8].

Mô hình này hiện được coi là mô hình xử lý ngôn ngữ state-of-the-art 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. 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 [8].

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

## **2.3. Bài toán biến đổi biểu diễn văn bản (Document Embedding)**

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.

### **2.3.1. Mô hình bag-of-words (BOW) cổ điển**

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 2.7: 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.

### **2.3.1. Biểu diễn văn bản sử dụng vector TF-IDF**

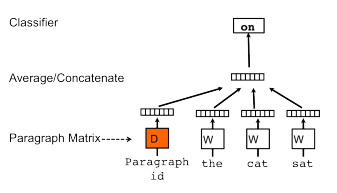
Phương pháp này dựa trên mô hình BOW để biểu diễn văn bản dưới dạng một vector. Ta thay thế các giá trị tần suất xuất hiện bằng các giá trị TF-IDF tương ứng.

Ví dụ: Dựa vào Bảng 2.6, ta có vector TF-IDF của câu (1) trong tập như sau:

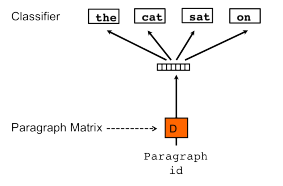
(0.042, 0.085, 0.042, 0.042, 0.042, 0.015, 0.042, 0.042, 0.042, 0.042, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

### **2.3.2. Biểu diễn văn bản sử dụng mô hình 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). Hai mô hình này lần lượt được xây dựng dựa trên mô hình CBOW và Skip-gram từ bài báo trước đó.



**Hình 2.7: PV-DM**



**Hình 2.8: 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 huấn luyện PV-DM [9]**

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. Với mỗi văn bản ta có một “paragraph id” 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.

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.

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 chiếm dụng nhiều tài nguyên hơn mô hình PV-DBOW, tuy nhiên kết quả của mô hình này có độ chính xác cao hơn. 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.

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.

### **2.3.3. Biểu diễn văn bản sử dụng mô hình biểu diễn từ trong không gian vector**

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.

**2.3.3.1. Mô hình Unweighted Averaged Word Vectors (trung bình cộng không có trọng số của các vector từ) dựa trên Word2Vec**

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.

**2.3.3.2. Mô hình SIF (Smooth Inversed Frequency) [10]**

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).

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 [10].

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” [10] 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. Phương pháp này có điểm khá giống với subsampling của Word2Vec. 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).

## **2.4. Bài toán so khớp ngữ nghĩa (Semantic Similarity)**

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.

Các phép đo TS-SS và Euclidean đều 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. Văn bản dài sẽ ảnh hưởng tới độ lớn vector. Độ đo tương đồng cosine bỏ qua chiều dài của văn bản mà chỉ đo góc giữa hai vector biểu diễn.

**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).

Kết quả của phép đo giữa hai vector A và B là thương của tích vô hướng và tích độ dài của hai vector.

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

## **2.5. Kết luận về các mô hình biểu diễn từ và biểu diễn văn bản**

Các mô hình trên đều là các mô hình ngôn ngữ và đều huấn luyện thông qua tối ưu xác suất của từ tiếp theo hoặc dựa vào các từ ngữ cảnh xung quanh nó. Toàn bộ các mô hình đều phải xây dựng một bộ từ vựng và bộ từ vựng là cơ sở để mô hình có khả năng biến đổi ngôn ngữ sang thành các biểu diễn dạng vector.

Các biểu diễn dạng vector thông qua các mô hình học không giám sát có khả năng biến đổi ý nghĩa từ dạng trừu tượng về dạng số có thể so sánh được. Tuy nhiên, nhược điểm của các mô hình này thường là gặp khó khăn khi xử lý từ đầu vào không có trong bộ từ vựng và tốn nhiều tài nguyên để huấn luyện.

Dù vậy, mô hình Doc2Vec phù hợp để huấn luyện một số lượng vừa phải các văn bản có độ dài lớn hơn một câu, tuy nhiên việc cập nhật bộ trọng số của biểu diễn Doc2Vec là khó khăn do ban đầu cả vector từ và vector văn bản đều được khởi tạo ngẫu nhiên.

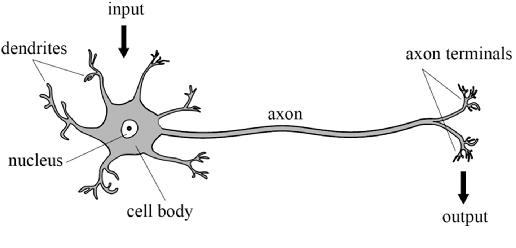
Ngược lại, các mô hình biến đổi biểu diễn từ có thể dễ dàng cập nhật bộ trọng số. Ngoài ra, biểu diễn vector từ hoàn toàn có thể sử dụng để tính toán biểu diễn cho các câu ngắn. Tuy nhiên, ta cần phải chú ý đến vấn đề mạng neuron “quên” – nghĩa là bộ trọng số bị điều chỉnh theo các ví dụ huấn luyện mới và làm giảm ảnh hưởng của các ví dụ ban đầu lên kết quả của các vector thu được trước đó.

# **Cơ sở lý thuyết**

Chương này sẽ trình bày và giải thích một số khái niệm trong xử lý ngôn ngữ đồng thời trong chương cũng sẽ trình bày về một số lý thuyết như mạng neuron,

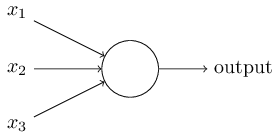
* Mạng neuron nhân tạo
* Cơ sở lý thuyết về văn bản và biểu diễn văn bản
* Cơ sở lý thuyết về so khớp ngữ nghĩa
* Lý thuyết về Singular Value Decomposition (SVD) và Principal Component Analysis (PCA)

## **3.1. Mạng neuron nhân tạo**

Mạng neuron nhân tạo (artificial neural network) là một mô hình xử lý thông tin được mô phỏng lại dựa trên cấu tạo sinh học của não bộ. 

**Hình 3.1: Cấu tạo sinh học của một neuron**

Trong Hình 3.1, neuron (tế bào não) nhận tín hiệu điện từ một hoặc nhiều neuron khác thông qua các dendrites (sợi nhánh) để “kích hoạt” neuron này và truyền tín hiệu đi thông qua axon (sợi trục) đến các neuron tiếp theo và cuối cùng là truyền tín hiệu đến các cơ bắp. Hành động hay phản xạ, phản ứng của động vật nói chung là kết quả của việc truyền tín hiệu điện não từ trung khu thần kinh đến các tế bào cơ bắp.

Với mô hình toán học, một neuron được mô tả lại tương đương với một “perceptron”. Một perceptron nhận một hoặc nhiều đầu vào nhị phân và cho ra một giá trị nhị phân duy nhất.

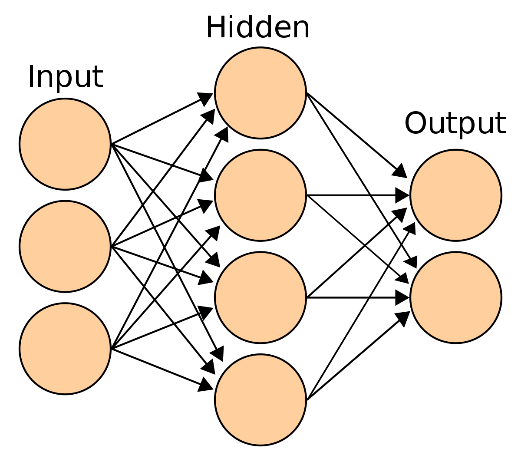
**Hình 3.2: Mô hình của một perceptron**

Trong mô hình được đề xuất, các trọng số (weights) được sử dụng để quyết định sự ảnh hưởng của các đầu vào lên đầu ra. Tổng tích của các đầu vào với trọng số tương ứng sẽ được so sánh với một ngưỡng (threshold), nếu kết quả lớn hơn ngưỡng thì đầu ra sẽ có kết quả bằng 1 và ngược lại trong trường hợp nhỏ hơn hoặc bằng ngưỡng.

**Hình 3.3: Giá trị đầu ra của một perceptron**

|  |  |
| --- | --- |
|  | **(3.1)** |

Qua đó, ta có thể mô phỏng lại được một mạng neuron nhân tạo có khả năng: nhận, tính toán và truyền thông tin tới các neuron khác và thu được kết quả neuron cuối cùng. Hình 3.4 là mô hình mô phỏng lại một mạng neuron nhân tạo dựa trên cấu tạo của neuron sinh học trong hình 3.1. Ngoài ra, còn nhiều cấu trúc mạng neuron phức tạp hơn phục vụ cho các bài toán khác nhau. Điển hình là các mạng RNN (Recurrent Neural Network), LSTM (Long Short-Term Memory), CNN (Convolutional Neural Network) …

Đầu vào của mô hình chính – input là các tín hiệu điện ban đầu và thông qua các kết nối đến neuron khác, chúng được lan truyền thông qua một hoặc nhiều lớp ẩn – lớp hidden và cuối cùng truyền thông tin đến đầu ra – output thì dừng lại (thuật toán lan truyền xuôi – feed forward). Đầu ra của lớp trước chính là đầu vào của lớp phía sau. Giá trị của các neuron không nhất thiết phải là giá trị nhị phân như mô hình perceptron. Các giá trị của neuron có thể được tính thông qua các hàm kích hoạt (activation functions) thay vì chỉ sử dụng tổng tích đầu vào – trọng số như trên. Mỗi một lớp hidden thông thường đều chứa một neuron bias và những neuron này chỉ kết nối tới lớp sau và không bị ảnh hưởng bởi các neuron lớp trước. Các neuron bias là các mang giá trị là hằng số (thông thường là 1) và được sử dụng để điều chỉnh lại các kết quả (phụ thuộc và kết của mong muốn của bài toán) của hàm kích hoạt. Trong mô hình này cũng như xuyên suốt báo cáo, các neuron bias sẽ không được đề cập hoặc sử dụng và có thể coi giá trị của neuron bias bằng 0.

**Hình 3.4: Mô hình ví dụ về một mạng neuron nhân tạo cơ bản**

Mạng neuron có thể được sử dụng để xây dựng biểu diễn mối liên hệ giữa đầu vào và đầu ra thông qua một đa thức. Các trọng số giữa các lớp hình thành các ma trận trọng số, đại diện cho mức độ ảnh hưởng của từng neuron lớp trước tới các neuron lớp sau. Chúng ta cần tìm đầu ra thích hợp từ đầu vào cho trước, vì đó ta cần phải thay đổi các ma trận trọng số để có thể có được đầu ra mong muốn hay sai số của đầu ra là nhỏ nhất.

Thuật toán lan truyền ngược (backpropagation) và thuật toán giảm gradient (gradient descending) được lựa chọn để điều chỉnh giá trị của các trọng số và giá trị của các neuron trong các lớp hidden để giảm dần sai số của đầu ra. Công thức cụ thể để tính toán phụ thuộc vào thiết kế của mạng và hàm kích hoạt.

Tuy nhiên, quá trình huấn luyện mạng neuron là hoàn toàn tự động, vì vậy, mạng neuron rất nhạy cảm với nhiễu (noises – dữ liệu đầu vào sai), điển hình là các giá trị bất hợp lý (outliers). Ví dụ khi chúng ta tìm mối liên hệ giữa chiều cao và độ tuổi, chúng ta không thể lấy các giá trị chiều cao là âm hoặc từ 4 mét trở lên hay tuổi không thể âm và vượt quá một ngưỡng nhất định. Chúng ta có thể sử dụng các công cụ như WEKA để loại thực hiện tiền xử lý dữ liệu. Ngoài ra, mạng neuron có khả năng quên hoàn toàn các biểu diễn/quan hệ đã học khi thực hiện huấn luyện với dữ liệu mới do các trọng số được cập nhật liên tục với mỗi ví dụ huấn luyện. Xu hướng này được gọi là Catastrophic Interference hay Catastrophic Forgetting.

## **3.2. Cơ sở lý thuyết về văn bản và biểu diễn văn bản**

**Khái niệm 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.

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

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

**Khái niệm về biến đổi biểu diễn văn bản (Document Embedding)**

Biểu diễn từ (word embedding) là phương pháp ánh xạ các từ sang không gian vector số. Với từ là đơn vị nhỏ nhất của văn bản, ta có thể nói, biểu diễn văn bản là ánh xạ văn bản sang không gian vector số. Cụ thể hơn, biến đổi biểu diễn văn bản là mã hóa ngữ nghĩa của toàn bộ văn bản sang dạng vector biểu diễn số thực (real-valued representation vector).

## **3.2. Cơ sở lý thuyết về 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.

Một thuật toán dựa trên TF-IDF – OkapiBM25 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 phép đo độ tương đồng đã đề cập trong mục 2.4. Chương trước đã đề cập về các mô hình có khả năng chuyển đồi ngữ nghĩa trừu tượng của ngôn ngữ sang biểu diễn dạng số có thể tính toán được. Do vậy, bài toán biến đổi biểu diễn văn bản là rất quan trọng đối với bài toán so khớp ngữ nghĩa.

## **3.3. Lý thuyết về Singular Value Decomposition (SVD) và Principal Component Analysis (PCA)**

**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:

|  |  |  |
| --- | --- | --- |
|  |  | **(3.2)** |

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:

|  |  |  |
| --- | --- | --- |
|  |  | **(3.3)** |

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 :

|  |  |  |
| --- | --- | --- |
|  |  | **(3.4)** |

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 .

**Principal Component Analysis (PCA)**

PCA là một phương pháp phân tích dữ liệu thông qua biến đổi trực giao để giảm số chiều của ma trận mẫu – đặc trưng và cho phép biểu diễn dữ liệu bằng hai/ba chiều thay cho số chiều tương ứng số lượng đặc trưng. Khi số chiều được giảm xuống còn hai hoặc ba, ta hoàn toàn có thể biểu diễn dữ liệu dưới dạng đồ thị hai/ba chiều.

Mục đích của PCA là tạo ra các đặc trưng/thuộc tính mới kết hợp từ các đặc trưng/thuộc tính cũ và các đặc trưng mới này là độc lập với nhau, không có mối tương quan dữ liệu. Các đặc trưng mới này được gọi là Principal Component và được sắp xếp theo mức độ quan trọng/ảnh hưởng đối với dữ liệu. Các đặc trưng này thể hiện mối tương quan/ảnh hưởng của chúng lên tập dữ liệu và phân cụm. PCA bằng cách phân tích các Principal Component giúp loại bỏ các thành phần không gây nhiều ảnh hưởng tới dữ liệu.

## **3.3. Nhận xét**

Thông qua chương này, ta đã làm rõ các cơ sở lý thuyết liên quan tới biến đổi biểu diễn văn bản sử dụng mạng neuron để xây dựng các không gian vector biểu diễn. Ngoài ra, chương này còn làm rõ tầm quan trọng của các phương pháp biến đổi văn bản với bài toán so khớp ngữ nghĩa.

# **Mô hình – Thiết kế - Cài đặt**

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 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.

## **4.1. 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. 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**

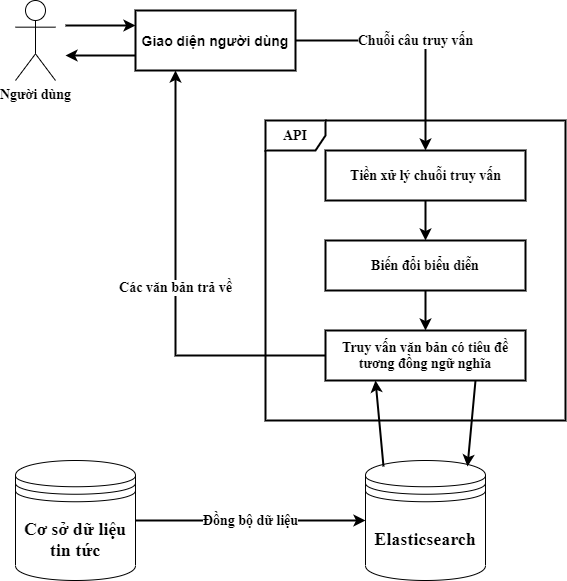
Thông thường, tiền xử lý sẽ gồm các bước sau:

* Loại bỏ nhiễu (các chuỗi không phải ngôn ngữ tự nhiên)
* Tách từ trong câu
* Chuẩn hóa các từ (chuyển về cùng font, viết thường…)
* Loại bỏ từ dừng/từ vô nghĩa (stop words)

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, 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.

## **4.2. Thiết kế tổng quan của hệ thống**

### **4.2.1. Thiết kế tổng quan**

Trong hình 4.1, 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 tin tức là cơ sở dữ liệu đã và đang được sử dụng cho các phần mềm/ứng dụng quản lý. Hệ thống sẽ hỗ trợ chuyển đổi văn bản từ cơ sở dữ liệu SQL hiện hành sang search engine Elasticsearch để phục vụ tìm kiếm văn bản theo ngữ nghĩa.

**Hình 4.1: Thiết kế tổng quan của hệ thống**

### **4.2.2. Module dữ liệu**

Cơ sở dữ liệu truyền thống SQL luôn được sử dụng để lưu trữ các thông tin lâu dài, tuy nhiên các cơ sở dữ liệu 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 cơ sở dữ liệu SQL, việc truy vấn toàn bộ văn bản để thực hiện so sánh biểu diễn là không khả thi do:

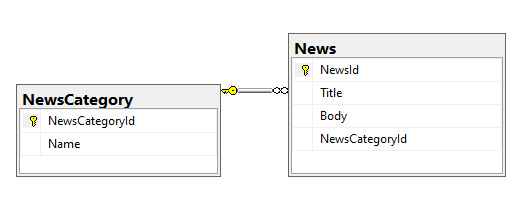
* Tốn nhiều tài nguyên xử lý
* Tốc độ đọc chậm do sử dụng I/O trên ổ đĩa, không thể đáp ứng trả về kết quả thời gian thực

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 cơ sở dữ liệu NoSQL Elasticsearch.

**Module cơ sở dữ liệu tin tức**

Cơ sở dữ liệu tin tức là một cơ sở dữ liệu 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 cơ sở dữ liệu 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). Cơ sở dữ liệu được sử dụng để mô phỏng là SQL Server 2017 Express.



**Hình 4.2: Thiết kế cơ sở dữ liệu SQL mô phỏng**

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

**Bảng 4.1: 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 4.2: 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 cơ sở lưu trữ dữ liệu phi cấu trúc NoSQL.

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 cơ sở dữ liệu SQL và Elasticsearch

### **4.2.3. Module API truy vấn**

**Tiền xử lý văn bản**

Phương pháp được sử dụng là phương pháp tách từ maximum matching và sửa lỗi nhập nhằng bằng cây nhị phân RDR. RDRSegmenter là một mô hình tách SCRDR từ với cây RDR được huấn luyện sẵn từ bộ dữ liệu Gold Standard đã được con người thực hiện tách từ.

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

Để sử dụng thư viện **vncorenlp**, ta cần tải file jar 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**

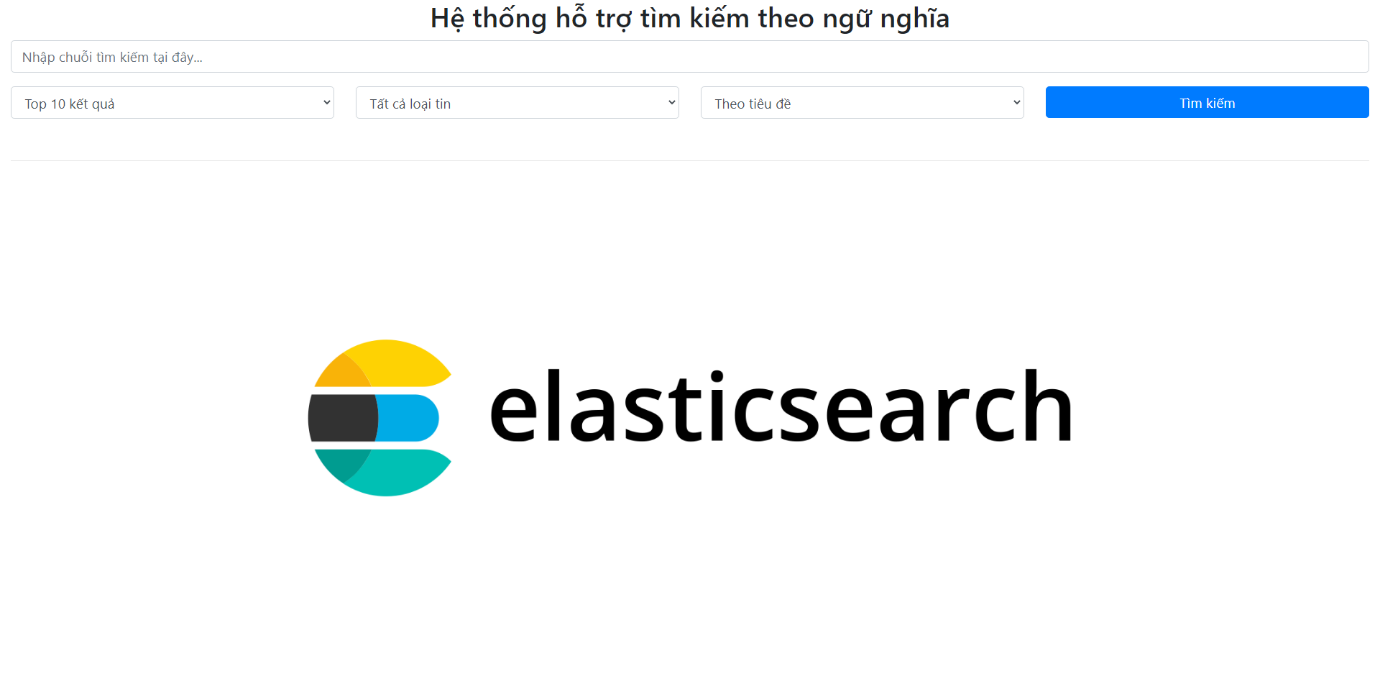
Dưới sự hỗ trợ của search engine Elasticsearch, các văn bản được đánh chỉ mục kèm theo các trường “dense vector” để lưu trữ biểu diễn với số chiều được định trước.

Elasticsearch hỗ trợ lưu trữ biểu diễn lên tới 2048 chiều. Trong hình 4.4, biểu diễn văn bản được lưu dưới trường [‘embedding’] và có số chiều là 100.

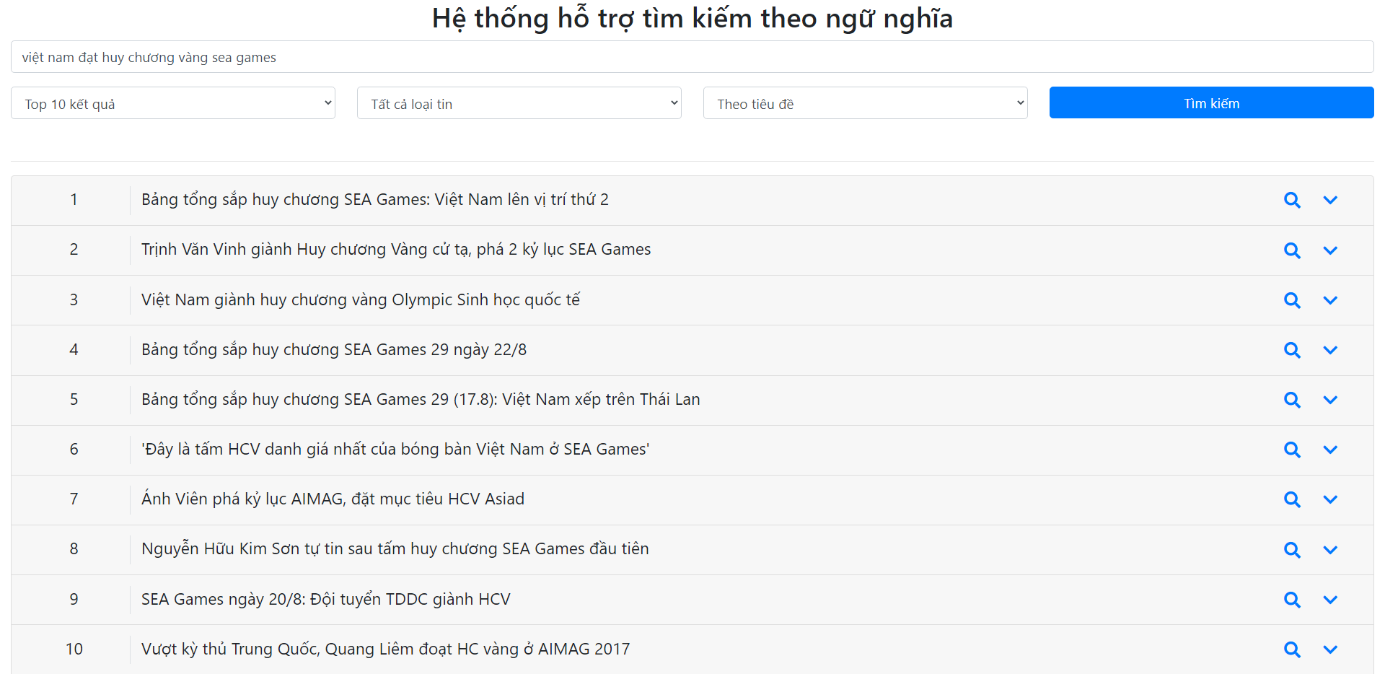
Để truy vấn các văn bản có tiêu đề tương đồng, trước tiên ta cần thực hiện tiền xử lý và truyền mảng các từ vào module biến đổi biểu diễn văn bản. Sau khi thu được biểu diễn vector của tiêu đề, ta thực hiện thao tác tìm kiếm và xếp hạng dựa trên độ đo tương đồng cosine thay vì thuật toán mặc định Okapi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

Đối với search engine, 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.

### **4.2.4. Giao diện người dùng**

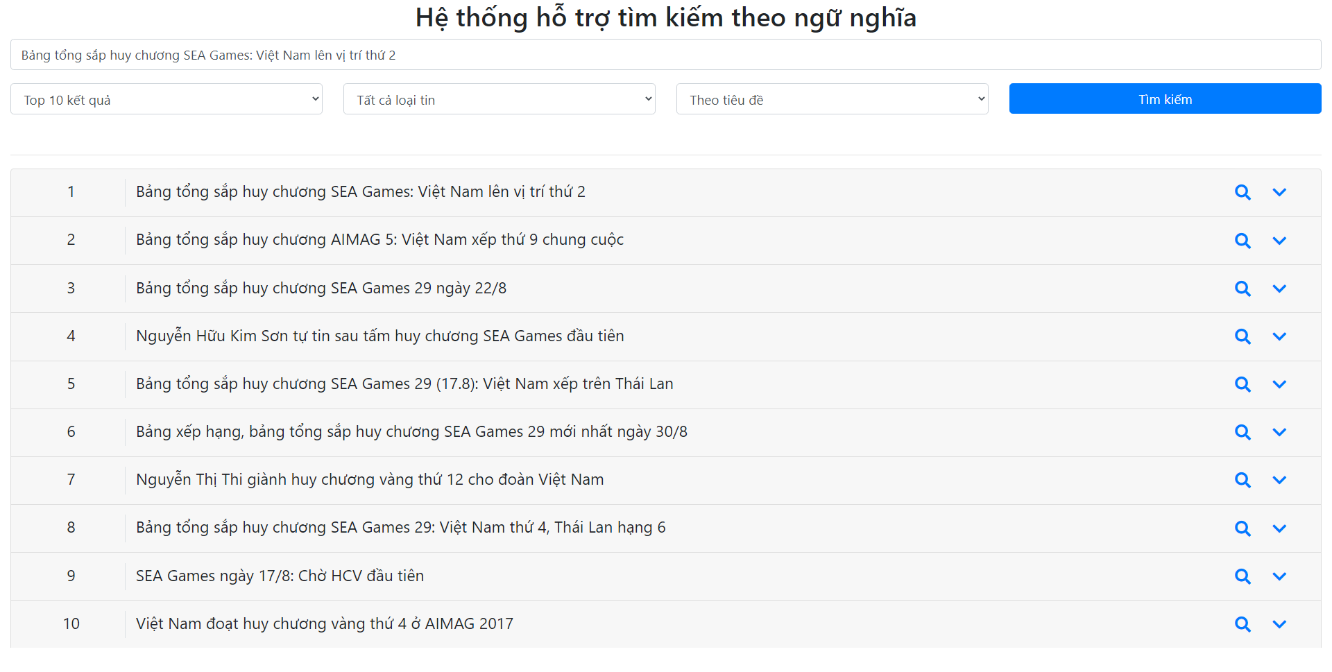


**Hình 4.3: Giao diện màn hình chính của hệ thống**

****

**Hình 4.4: Giao diện màn hình kết quả tìm kiếm**

**Hình 4.5: Giao diện chi tiết của kết quả tìm kiếm**

****

**Hình 4.6: Giao diện tiếp tục tìm kiếm theo một kết quả**

## **4.3. 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 cơ sở dữ liệu SQL và tham số mô hình.



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

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

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

Cụ thể phương thức này sẽ thực hiện câu lệnh sau để lấy tổng số lượng bản ghi trong bảng News:



Sử dụng câu lệnh trên có sẽ có thời gian 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.

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

Phương thức này được sử dụng để chuẩn bị dữ liệu đồng bộ từ cơ sở dữ liệu 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





Elasticsearch có hỗ trợ lưu trữ vector với số chiều biểu diễn 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**. 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.

Cấu hình kết nối tới Elasticsearch hoàn toàn phụ thuộc vào cấu hình Elasticsearch được sử dụng.



* Các phương thức thao tác với chỉ mục









* Kiểm tra số lượng bản ghi trong cơ sở dữ liệu SQL và Elasticsearch



File **app.py**:là file chứa ứng dụng chính, là nơi định nghĩa đồng thời cung cấp các điểm truy cập và sử dụng API và một số phương thức.

Ta cài đặt chức năng tiền xử lý dữ liệu như sau:





Để thực hiện biến đổi biểu diễn văn bản, ta cần tải mô hình ngôn ngữ thu được sau huấn luyện. Cụ thể là mô hình SIF:



Thành phần chính của file này là cung cấp một 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. Kết quả trả về của API là dữ liệu JSON có các trường dữ liệu giống cấu trúc mappings của tin tức khi cài đặt Elasticsearch. Tuy nhiên, các trường dữ liệu lưu trữ biểu diễn sẽ bị loại bỏ do không mang ý nghĩa với người dùng và giảm lượng dữ liệu cần truyền tải.

Ngoài ra, file này còn định nghĩa API đồng bộ dữ liệu. Hiện tại, dữ liệu sẽ được đồng bộ sử dụng toàn bộ dữ liệu trong SQL. Trong thực tế, sau khi thêm/cập nhật các bản ghi thành công vào cơ sở dữ liệu SQL, mỗi bản ghi sẽ được đánh chỉ mục ngay vào Elasticsearch. API này chỉ nên dùng trong trường hợp tích hợp với hệ thống có sẵn và cần đánh chỉ mục toàn bộ dữ liệu từ cơ sở dữ liệu SQL sang Elasticsearch. Đây là một tác vụ tốn nhiều tài nguyên, vì vậy, ta cần khóa tác vụ này trong khi nó đang được thực hiện để tránh các yêu cầu đồng bộ liên tiếp. Trong khi đồng bộ có thể cần thiết khóa thao tác nhập liệu trên hệ thống quản lý và chọn thời gian thích hợp.





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 sẽ không có chức năng đồng bộ dữ liệu.

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.

# **Kết quả 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.

## **5.1. Thí nghiệm**

**Dữ liệu thí nghiệm**

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

“vnPara” bao gồm 3083 cặp câu tiếng Việt kèm theo nhãn đánh dấu hai câu là tương đồng ngữ 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, 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 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**

Doc2Vec 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 PCA/SVD. 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.

### **5.1.1. Các độ đo sử dụng để đánh giá 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 mức độ chính xác khi gán nhãn/phân loại 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.

### **5.1.2. Đánh giá kết quả trên bộ dữ liệu cặp câu tương đồng vnPara**

**5.1.2.1. 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;
* Mô hình PV-DM sử dụng 20 vòng lặp để biến đổi biểu diễn văn bản.

**Bảng 5.1: 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 5.2: 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 5.3: 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 5.4: 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 |

**5.1.2.2. 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 |

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

**Bảng 5.7: 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 |

(\*) 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.

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

## **5.3. Nhận xét**

Ta 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, ta thấy được các kết quả khá khả quan. Thông qua hiệu năng và độ chính xác tương đối, ta có thể đánh giá mô hình SIF là một mô hình khá tốt cho biểu diễn câu ngắn.

Thực tế, để đánh giá mô hình ngôn ngữ chúng 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.

# **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.

Với các mô hình ngôn ngữ học máy ngày càng phát triển, ta cần phải tìm hiểu thêm các công nghệ mới hỗ trợ nhiều hơn cho bài toán biểu diễn văn bản và so khớp ngữ nghĩa.

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.

# **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] | 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. |
| [3] | G. Carrado, K. Chen, J. Dean and T. Mikolov, "Efficient Estimation of Word Representations in Vector Space," *CoRR,* 2013. |
| [4] | 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. |
| [5] | 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. |
| [6] | 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. |
| [7] | 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. |
| [8] | 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. |
| [9] | 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. |
| [10] | S. Arora, Y. Liang and T. Ma, "A Simple but Tough-to-Beat Baseline for Sentence Embeddings," in *ICRL*, 2017. |
| [11] | V. Q. Bình, "News Corpus," GitHub. |
| [12] | 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. |
| [13] | 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. |
| [14] | O. Borchers, "Fast sentence embeddings," GitHub, 2019. |
| [15] | L. H. Phuong, "vnTokenizer," GitHub, 2009. |
| [16] | L. H. Phuong, "Vitk - A Vietnamese Text Processing Toolkit," GitHub, 2015. |
| [17] | L. H. Phuong, "VLP - Vietnamese Language Processing toolkit," 2020. |
| [18] | 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. |