TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 2**

**XÂY DỰNG HỆ THỐNG HỎI VÀ TRẢ LỜI TỰ ĐỘNG**

*Người hướng dẫn*: **TS NGUYỄN HỒNG VŨ**

*Người thực hiện*: **NGUYỄN DUY THANH - 51403318**

Lớp **: 14050302**

Khoá  **: 18**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2017**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 2**

**XÂY DỰNG HỆ THỐNG HỎI VÀ TRẢ LỜI TỰ ĐỘNG**

*Người hướng dẫn*: **TS NGUYỄN HỒNG VŨ**

*Người thực hiện*: **NGUYỄN DUY THANH - 51403318**

Lớp **: 14050302**

Khoá  **: 18**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2017**

LỜI CẢM ƠN

Đầu tiên em xin gửi lời cảm ơn đến Thầy Nguyễn Hồng Vũ – giảng viên hướng dẫn Đồ Án 2 của em, thầy đã tận tình hướng dẫn, giới thiệu những kiến thức, giải đáp thắc mắc giúp em có thể hoàn thành đồ án này. Chúc thầy sức khỏe để tiếp tục thực hiện sứ mệnh cao đẹp của mình.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của TS Nguyễn Hồng Vũ;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Duy Thanh*

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trong đồ án này em tìm hiểu về mô hình bài toán hỏi và trả lời tự động, áp dụng mô hình RNN, LSTM và xây dựng ứng dụng minh họa cho Tiếng Việt.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc502232297)

[TÓM TẮT iv](#_Toc502232298)

[MỤC LỤC 1](#_Toc502232299)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 3](#_Toc502232300)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc502232301)

[CHƯƠNG 1 – HỆ THỐNG HỎI VÀ TRẢ LỜI TỰ ĐỘNG 5](#_Toc502232302)

[1.1 Information Retrieval based Question Answering 5](#_Toc502232303)

[1.1.1 Question processing 6](#_Toc502232304)

[1.1.2 Passage Retrieval 6](#_Toc502232305)

[1.1.3 Answer Processing 8](#_Toc502232306)

[1.2 Knowledge-based Question Answering 11](#_Toc502232307)

[CHƯƠNG 2 – WORD EMBEDDING 12](#_Toc502232308)

[2.1 Nhiệm vụ giả 12](#_Toc502232309)

[2.2 Chi tiết mô hình 13](#_Toc502232310)

[2.3 Lớp ẩn 14](#_Toc502232311)

[CHƯƠNG 3 – RNN, LSTM 15](#_Toc502232312)

[3.1 Mạng nơ-ron hồi quy RNN là gì? 16](#_Toc502232313)

[3.2 Khả năng của RNN 17](#_Toc502232314)

[3.3 Huấn luyện RNN 17](#_Toc502232315)

[3.4 Mạng LSTM 18](#_Toc502232316)

[CHƯƠNG 4 – VẤN ĐỀ TÁCH TỪ TRONG VĂN BẢN TIẾNG VIỆT 18](#_Toc502232317)

[4.1 Tách từ là gì? 18](#_Toc502232318)

[4.2 Tại sao tách từ trong Tiếng Việt lại khó? 19](#_Toc502232319)

[4.3 Biểu thức chính quy (Regular Expressions) 20](#_Toc502232320)

[4.4 Biểu đồ cụm từ 22](#_Toc502232321)

[4.5 Sự mơ hồ chồng chéo 24](#_Toc502232322)

[4.6 Mô hình Ngôn ngữ 24](#_Toc502232323)

[4.7 Phân loại khoảng trắng 25](#_Toc502232324)

[4.8 Phần mềm 26](#_Toc502232325)

[CHƯƠNG 5 – HIỆN THỰC 26](#_Toc502232326)

[5.1 Tách từ tiếng Việt 26](#_Toc502232327)

[5.1.1 Chương trình trên Java 26](#_Toc502232328)

[5.1.2 Chương trình trên Python 27](#_Toc502232329)

[5.2 Mô hình 28](#_Toc502232330)

[5.2.1 Mô hình lựa chọn tài liệu 28](#_Toc502232331)

[5.2.2 Mô hình trả lời 29](#_Toc502232332)

[5.3 Ứng dụng demo 31](#_Toc502232333)

[5.3.1 Nguyên lý hoạt động 31](#_Toc502232334)

[5.3.2 Một số hình ảnh minh họa 31](#_Toc502232335)

[TÀI LIỆU THAM KHẢO 33](#_Toc502232336)

[PHỤ LỤC 34](#_Toc502232337)

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

**CÁC KÝ HIỆU**

**CÁC CHỮ VIẾT TẮT**

NLP Xử lý ngôn ngữ tự nhiên (Natural language processing)

QA Hệ thống hỏi và trả lời ( Question answering)

IR Hệ thống truy xuất thông tin(information retrieval)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1: IR based question answering có ba giai đoạn: question processing, passgage retrieval và answer processing 6](#_Toc502232276)

[Hình 1.2: Năm đoạn trích Google trả về khi tìm kiếm “When was plane invented?” 9](#_Toc502232277)

[Hình 2.1: Ví dụ input cho word2vec 14](#_Toc502232278)

[Hình 2.2: Kiến trục mạng nơ-ron 15](#_Toc502232279)

[Hình 2.3: Ma trận trọng số 16](#_Toc502232280)

[Hình 3.1: Mạng nơ-ron hồi quy RNN 17](#_Toc502232281)

[Hình 4.1: Biểu độ cụm từ “sinh viên vi phạm quy chế thi cử” 24](#_Toc502232282)

[Hình 4.2: Biểu đồ câu văn “bảo lãnh bằng chứng khoán” 25](#_Toc502232283)

[Hình 5.1: Code py4j trên Java 28](#_Toc502232284)

[Hình 5.2: Code py4j trên Python 28](#_Toc502232285)

[Hình 5.3: Mô hình lựa chọn tài liệu 30](#_Toc502232286)

[Hình 5.4: Mô hình chọn câu trả lời 32](#_Toc502232287)

[Hình 5.5: Minh họa demo 1 32](#_Toc502232288)

[Hình 5.6: Minh họa demo 2 32](#_Toc502232289)

**DANH MỤC BẢNG**

[Bảng 1.1: Một vài ví dụ về IR based Question Answering 6](#_Toc502232290)

[Bảng 1.2: Một vài biểu thức trích xuất câu trả lời cho câu hỏi định nghĩa 11](#_Toc502232291)

[Bảng 1.3: Ví dụ biểu thức logic được tao ra bởi một sematic parser cho QA 12](#_Toc502232292)

[Bảng 1.4: Ví dụ RDF 13](#_Toc502232293)

[Bảng 4.1: Các ví dụ Regex 22](#_Toc502232294)

[Bảng 4.2: Các định lượng Regex phổ biến 22](#_Toc502232295)

[Bảng 4.3: Ví dụ kết hợp Regex 23](#_Toc502232296)

# HỆ THỐNG HỎI VÀ TRẢ LỜI TỰ ĐỘNG

Việc học hỏi làm cho con người trở nên sâu xắc hơn và máy tính cũng như vậy ngay cả trong việc xử lý ngôn ngữ tự nhiên(NLP), con người đã cố gắng sử dụng máy tính cho việc trả lời các câu hỏi từ thời kỳ đầu của máy tính. Đầu như năm thập niên 60, có hai mô hình cho hệ thống hỏi và trả lời(QA) Information Retrieval based question answering (Hệ thống hỏi và trả lời dựa trên truy xuất thông tin) và knowledge based question answering (Hệ thống hỏi và trả lời dựa trên kiến thức).

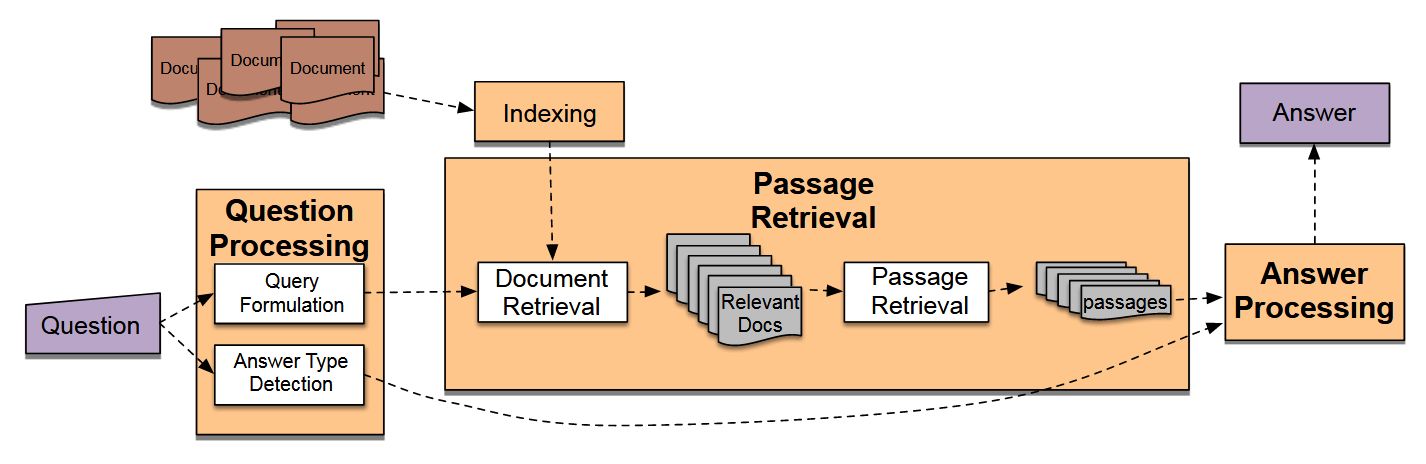
* 1. Information Retrieval based Question Answering

Để có được câu trả lời cho câu hỏi của người dùng IR-based question answering tìm kiếm một đoạn văn ngắn trên web hoặc một bộ tài liệu nào đó. Ví dụ.

|  |  |
| --- | --- |
| Câu hỏi | Trả lời |
| Where is the Louvre Museum located? | In Paris, France |
| What currency is used in China? | The yuan |
| What's the official language of Algeria? | Arabic |

Bảng 1.1: Một vài ví dụ về IR based Question Answering

Để có được câu trả lời cho ví dụ trên cần trả qua ba giai đoạn của hệ thống IR based question answering system: question processing (xử lý câu hỏi), passage retrieval and ranking (truy xuất và xếp hạng) và answer processing (xử lý câu trả lời).



Hình 1.1: IR based question answering có ba giai đoạn: question processing, passgage retrieval và answer processing

* + 1. Question processing

Mục đích của quá trình này là trích xuất một số thông tin từ câu hỏi. **Answer type** (loại câu trả lời) xác định loại thực thể trong câu trả lời (người, nơi chốn, thời gian, etc.). **Query** xác định các từ khóa dùng cho việc tìm kiếm trong các tài liệu.

Một vài hệ thống còn trích xuất **focus**, đó là chuỗi các từ trong câu hỏi mà có thể sẽ được thay thế bởi câu trả lời trong bất kỳ chuỗi câu trả lời nào được tìm thấy. Một vài hệ thống cũng phân loại câu hỏi **question type**: nó có phải là một câu hỏi định nghĩa, câu hỏi toán học, hay một danh sách câu hỏi? Ví dụ.

Which US state capital has the largest population?

Question processing sẽ cho ra kết quả tương tự như bên dưới:

**Answer type:** city

**Query:** US state capital, largest, population

**Focus:** state capital

* + 1. Passage Retrieval

Query được tạo ra sau khi câu hỏi gốc trả qua giai đoạn question processing được đưa vào hệ thống information retrieval (hệ thống truy xuất thông tin), một bộ tài liệu được đánh chỉ mục hoặc một công cụ tìm kiếm trên Web. Kết quả của quá trình này là một bộ tài liệu.

Mặc dù bộ tài liệu đó thường đã được xếp hạng dựa vào độ liên quan, nhưng tài liệu được xếp hàng nhất không phải lúc nào cũng là câu trả lời. Điều này là do độ liên quan không phải đơn vị thích hợp để tìm ra câu trả lời. Một tài liệu có độ liên quan cao và lớn không hứa hẹn có chứa câu trả lời cho giai đoạn xử lý tiếp theo.

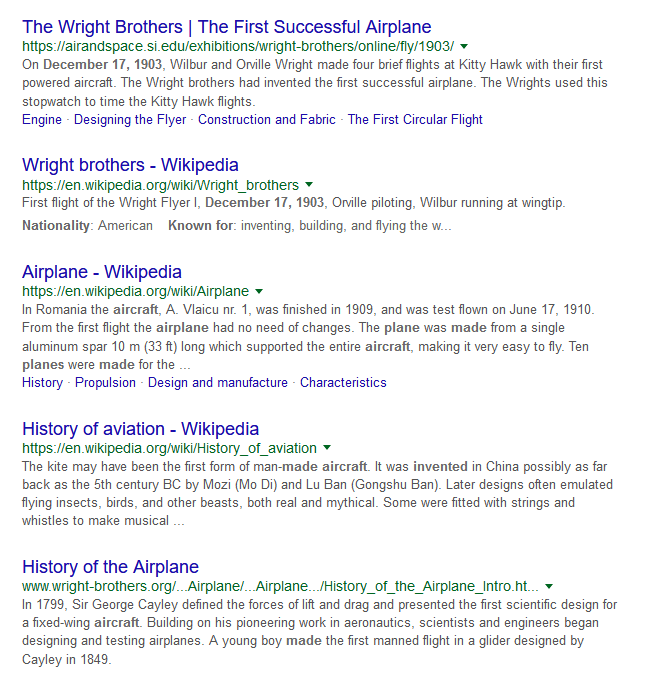
Do đó, công việc tiếp theo cần làm là trích xuất được những đoạn văn trả lời từ tập tài liệu truy xuất được. Đoạn trả lời bao gồm chương, đoạn văn, câu…

Tiếp theo chúng ta tiến hành passage retrieval. Trong giai đoạn này, đầu tiên chúng ta loại bỏ những đoạn văn từ những tài liệu được trả về mà không chứa các câu trả lời tiền năng và xếp hạng các tài liệu con lại dựa trên khả năng chứa câu trả lời cho câu hỏi. bước đầu tiên là trong quá trình này là chạy một thực thể được đặt tên hoặc một answer type classification trên các đoạn văn truy xuất được. Nhờ answer type được xác định từ câu hỏi chúng ta có thể lọc ra những tài liệu không chứa câu trả lời đúng loại.

Đoạn văn còn lại được xếp hạng, thường được làm bởi supervised machine learning(học máy có giám sát), dự trên một bộ nhỏ các thuộc tính có thể dễ dàng trích xuất được từ những đoạn văn có khả năng chứa câu trả lời, như là:

* Số lượng của những thực thể có kiểu phù hợp trong đoạn văn.
* Số lượng của những từ khóa câu hỏi trong đoạn văn.
* Câu dài nhất được trích xuất của những từ khóa trong câu hỏi xuất hiện trong đoạn văn.
* Hạng của tài liệu mà đoạn văn được trích xuất từ đó.
* Sự tương đồng của từ khóa từ query gốc tới đoạn văn.
* N-gram giữa đoạn văn và câu hỏi.

Cho QA từ web, thay vì trích xuất đoạn văn từ mọi tài liệu được trả về, chúng ta có thể dựa trên kết quả tìm kiếm từ web. Chúng ta làm điều này bằng cách dùng đoạn trích được tao ra bởi Web search engine(công cụ tìm kiếm). Ví dụ.



Hình 1.2: Năm đoạn trích Google trả về khi tìm kiếm “When was plane invented?”

* + 1. Answer Processing

Giai đoạn cuối cùng của QA là trích xuất câu trả lời từ đoạn văn, nhờ đó có thể chuyển đến người dùng với một câu trả lời như 29,029 feet cho câu hỏi “how tall is Mt. Everest”.

Hai loại giải thuật được áp dụng để trích xuất câu trả lời, một là dựa trên **answer-type pattern extraction** và một là dựa trên **N-gram tiling.**

Trong phương pháp patter-extraction, chúng ta sử dụng thông tin về answer type với regular expression patterns (biểu thức chính quy). Ví dụ, cho câu hỏi với answer type là HUMAN (con người), chúng ta kiểm tra các đoạn văn và trả vế bất cứ thực thể nào được gán nhãn là HUMAN. Như vậy, trong ví dụ sau đây, những thực thể được gạch chân có nghĩa là được trích xuất từ các đoạn văn có khả năng chứa câu trả lời cho câu hỏi HUMAN và DISTANCE-QUANTITY (khoảng cách):

“Who is the prime minister of India?”

Manmohan Singh, Prime Minister of India, had told left leaders that he deal would not be renegotiated.

“How tall is Mt. Everest?”

The official height of Mount Everest is 29029 feet

Không may, những câu trả lời cho một vài câu hỏi, như là DEFINITION (định nghĩa), thì có xu hướng không thuộc về một loại thực thể. Đối với các loại câu hỏi như vậy chúng ta sửa dụng biểu thức chính quy viết tay để giúp trích xuất câu trả lời. Biểu thức cũng hiệu quả trong trường hợp đoạn văn chứa nhiều mẫu của cùng một kiểu thực thể. Hình bên dưới cho thấy một vài biểu thức từ Pasca (2003) cho question phrase(QP) và answer phrase(AP) cho câu hỏi định nghĩa.

|  |  |  |
| --- | --- | --- |
| Biểu thức | Câu hỏi | Trả lơi |
| <AP> such as <QP> | What is autism? | “, developmental disorders such as autism” |
| <QP>, a <AP> | What is a caldera? | “the Long Valley caldera, a volcanic crater 19 miles long” |

Bảng 1.2: Một vài biểu thức trích xuất câu trả lời cho câu hỏi định nghĩa

Nguồn: Pasca 2003

Biểu thức được đặc tả cho mỗi kiểu câu hổi và có thể được viết bằng tay hoặc học tự động sử dụng phương pháp trích xuất quan hệ. biểu thức có thể được sử dụng cùng với những thông tin khác như một tham số để xếp hạng câu trả lời. Chúng ta có thể trích xuất những câu trả lời tìm năng bằng cách sử dụng những thực thể đã được đặt tên hoặc chỉ bằng cách xem xét tất cả các câu trả về từ đoạn văn và xếp hạng chúng sử dụng một trong các phương pháp phân lớp dưới đây.

**Answer type match** (khớp với kiểu câu trả lời): đúng nếu câu trả lời ứng viên chứa một đoạn đúng với kiểu câu trả lời.

**Number of matched question keywords** (số lượng từ khóa hỏi khớp)**:** số lượng từ khóa hỏi chứa trong câu trả lời tìm năng.

**Keyword distance** (khoảng cách từ khóa)**:** khoảng cách giữa câu trả lời tìm năng và từ khóa hỏi truy vấn.

**Novelty factor** (tính mới): đúng nếu có ít nhất một từ trong câu trả lời tìm năng là mới, có nghĩa là nó không xuất hiện trong truy vấn.

**Punctuation location** (vị trí dấu chấm câu): đúng nếu câu trả lời tìm năng theo xâu bởi dấu phẩy, gạch ngang, dấu nháy, dấu chấm phẩy, dấu chấm than.

Một phương pháp thay thế để trích xuất câu trả lời, trong trường hợp sử dụng công cụ tìm kiếm là dựa trên **N-gram tiling**, đôi khi còn được gọi là **redundancy based approach** (cách tiếp cận dự phòng). Đây là phương pháp đơn giản nhất bắt đầu với trích dẫn được từ về từ công cụ tìm kiếm. Bước đầu tiên, **N-gram mining**, mọi unigram, bigram và trigram xuất hiện trong trích dẫn được trích xuất ra và đo lường. **N-gram filtering** (lọc), trong bước này N-gram được chấm điển bằng cách xem nó có phải là answer type hay không. Cuối cùng, một thuật toán **N-gram tiling** được sử dụng để nối các phần lại với nhau cho ta được một câu trả lời dài. Thông thường một giải thuật tham lam được áp dụng, bắt đầu với cái có điểm cao nhất và thử nối nó với các ứng cử viên khác. Cho tới khi nào một câu trả lời duy nhất còn lại.

Với bất kỳ phương pháp trích xuất câu trả lời nào, thì chúng phải có khả năng trình bày câu trả lời tới người dùng.

* 1. Knowledge-based Question Answering

Mặc dù một lượng lớn thông tin được mã hóa dưới dạng chữ viết trên web, thông tin rõ ràng cũng tồn tại dưới nhiều dạng cấu trúc khác nhau. Người ta sử dụng thuật ngữ **knowledge-based question answering** (trả lời dựa trên kiến thức) cho ý tưởng trả lời câu hỏi ngôn ngữ tự nhiên bằng cách biến đổi chúng thành một truy vấn tới cơ sở dữ liệu có cấu trúc. Cách tiếp cận này là cách mà những hệ thống QA thời kỳ đầu sử dụng, BASEBALL (Green et al., 1961) là một ví dụ.

Hệ thống biến đổi từ văn bản sang thành biểu thức logic được gọi là **sematic parsers** (???). Sematic parsers cho QA thường được sử dụng để biến đổi thành định lý từ vừng hoặc một ngôn ngữ truy vấn như SQL hoặc SPARQL, như trong ví dụ bên dưới.

|  |  |
| --- | --- |
| Câu hỏi | Dạng logic |
| When was Ada Lovelace born? | birth-year (Ada Lovelace, ?x) |
| What states border Texas? |  |
| What is largest state? | argmax() |
| How many people survived the sinking of the Titanic? | (count(!fb:event.disaster.survivors fb:en.sinking.of.the.titanic)))) |

Bảng 1.3: Ví dụ biểu thức logic được tao ra bởi một sematic parser cho QA

Cơ sở dữ liệu được sử dụng có thể là cơ sở dữ liệu quan hệ, hoặc đơn giảng là cơ sở dữ liệu có cấu trúc như **RDF triples**. Đó là một bộ ba, một biểu thức cho biết quan hệ hoặc luật giữa hai đối số. Các công nghệ phổ biến như Freebase (Bollacker et al., 2008) hoặc DBpedia (Bizer et al., 2009) có một số lượng lớn dữ liệu từ Wikipedia **infoboxes**, một bản cấu trúc tốt từ những bài báo trên Wikipedia.

Cách đơn giảng nhất để hình thành một hệ thống knowledge-based QA là trả lời câu hỏi dựa trên tham số bị mất trong bộ ba. Xem ví dụ bên dưới.

|  |  |  |
| --- | --- | --- |
| **Subject** | **Predicate** | **Object** |
| Ada Lovelace | birth-year | 1815 |

Bảng 1.4: Ví dụ RDF

Bộ ba này có thể được dùng để trả lời các câu hỏi dạng “When was Ada Lovelace born?” hoặc “Who was born in 1815?”. QA trong trườn hợp này sẽ biến đổi “When was … born?” thành một quan hệ trong cơ sở tri thức như birth-yeah. Chúng ta có thể phát họa kết quả như sau:

“When was Ada Lovelace born?” 🡪 birth-year (Ada Lovelace, ?x)

“What is the capital of England?” 🡪 capital-city (?x, England)

# WORD EMBEDDING

Có nhiều cách để biểu thị nghĩa của một từ trong học máy, trong số đó có một cách được sử dụng rộng rải nhất, đơn giản và hiệu quả cao đó là word2vec.

Word2Vec sử dụng một mẹo đơn giản bạn có thể đã thấy đâu đó trong machine learning. Chúng ta sẽ huấn luyện một mạng nơ-ron đơn giản với một lớp duy nhất để thực hiện một nhiệm vụ nào đó, nhưng sau đó chúng ta sẽ không sử dụng mạng nơ-ron cho nhiệm vụ mà chúng ta đã huấn luyện! Thay vào đó, mục tiêu thực sự chỉ để học các trọng số của lớp ẩn, chúng ta sẽ thấy ràng các trọng số này thực sự là “véc-tơ của từ” mà chúng ta đang cố gắng để tìm.

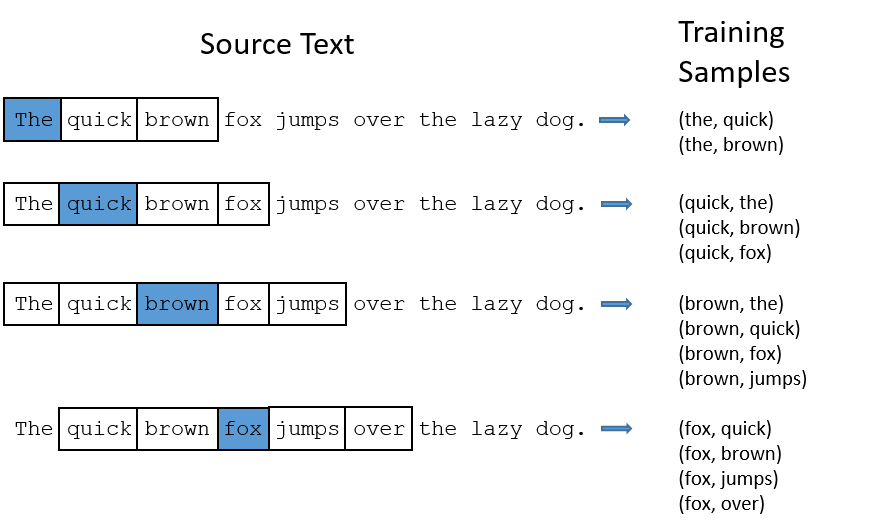
2.1 Nhiệm vụ giả

Bây giờ chúng ta cần nói về nhiệm vụ giả thứ mà chúng ta sẽ dùng để xây dựng mạng nơ-ron và thu lấy nó để có word – véc-tơ.

Chúng ta sẽ huấn luyện mạng nơ-ron bằng cách đưa vào một từ và các từ đứng gần nó. Sau khi kết thúc mạng nơ-ron sẽ cho chúng ta biết xác xuất của mỗi từ trong danh sách từ vựng của chúng ta có xuất hiện gần từ đầu vào.

Ví dụ, nếu bạn huấn luyện với từ đầu vào là “soviet” thì xác xuất thu được sẽ cao hơn với từ “union” và “Russia” hơn là “watermelon” và “kangaroo”.

Chúng ta sẽ huấn luyện mạng nơ-ron để làm điều này bằng cách cho vào từng cặp từ trong tài liệu. Bên dưới là ví dụ.



Hình 2.1: Ví dụ input cho word2vec

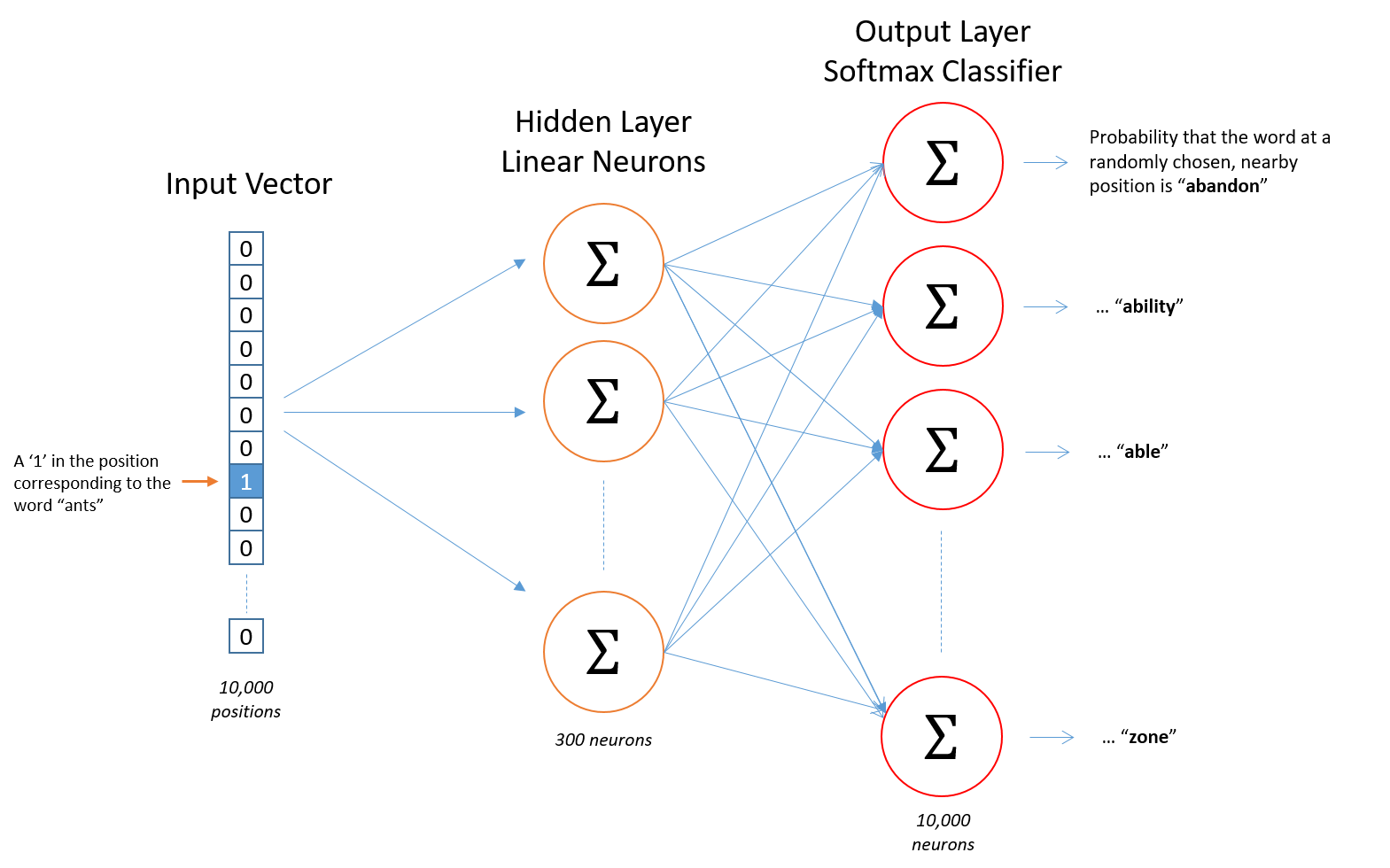
Mạng nơ-ron sẽ thống kê số lần mỗi cặp từ xuất hiện cạnh nhau.

2.2 Chi tiết mô hình

Trước hết, bạn không thể cho chuỗi vào mạng nơ-ron, vì vậy chúng ta cần biểu diễn từ theo cách nào đó để mạng nơ-ron hiểu được. Đề làm được điều này, đầu tiên chúng ta xây dựng một bộ từ vựng của tài liệu huấn luyện, giả sử chúng ta có 10,000 từ không trùng nhau.

Chúng ta trình bày một từ như là một on-hot-vector. Véc-tơ này sẽ có 10,000 chiều(mỗi chiều cho một từ của chúng ta) và chúng ta sẽ gán giá trị 1 tại ví trí của từ đó và 9,999 giá trị 0 và phần còn lại.

Đầu ra của mạng là một véc-tơ đơn chứa từ xuất hiện cạnh từ input.



Hình 2.2: Kiến trục mạng nơ-ron

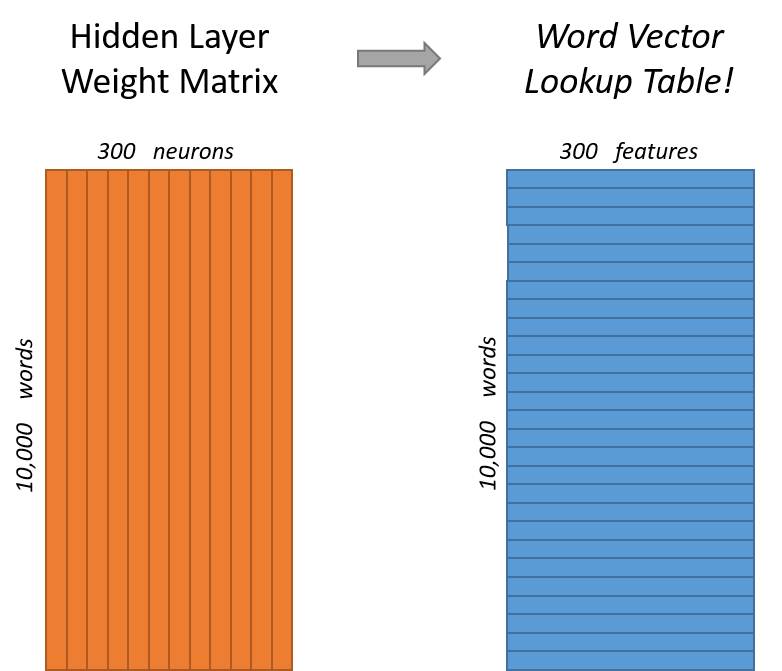
Mạng nơ-ron này có input là một one-hot-vector và đầu ra cũng là một one-hot-vector. Nhưng trong quá trình huấn liệu lớp ẩn trong mạng nơ-ron chứa xác xuất phân phối.

2.3 Lớp ẩn

Ví dụ chúng ta sẽ học một word-vector với 300 đặc trưng. Vì vậy lớp ẩn sẽ trình bày ma trận trọng số 10,000 dòng và 300 cột.

300 đặc trưng là số lượng mà Google đã sử dụng trên mô hình công bố của họ được huấn luyện trên bộ dữ liệu Google news.

Nếu bạn nhìn vào dòng của ma trận trọng số, nó thật ra là những word-vector của chúng ta.



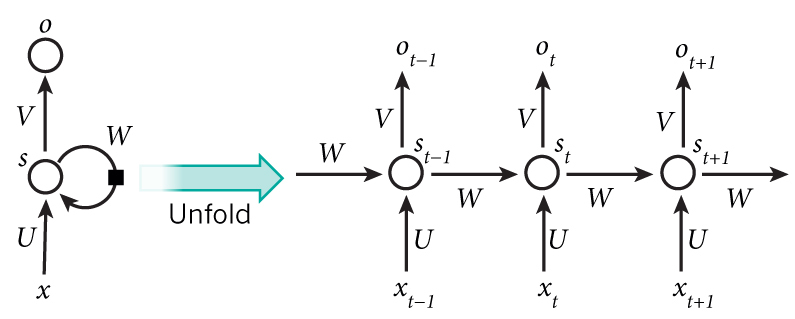
Hình 2.3: Ma trận trọng số

# RNN, LSTM

Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) là một thuật toán được chú ý rất nhiều trong thời gian gần đây bởi các kết quả tốt thu được trong lĩnh vực xử lý ngôn ngữ tự nhiên.

3.1 Mạng nơ-ron hồi quy RNN là gì?

Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán. Ví dụ, nếu muốn đoán từ tiếp theo có thể xuất hiện trong một câu thì ta cũng cần biết các từ trước đó xuất hiện lần lượt thế nào chứ nhỉ? RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó mà thôi. Về cơ bản một mạng RNN có dạng như sau:



Hình 3.1: Mạng nơ-ron hồi quy RNN

Mô hình trên mô tả phép triển khai nội dung của một RNN. Triển khai ở đây có thể hiểu đơn giản là ta vẽ ra một mạng nơ-ron chuỗi tuần tự. Ví dụ ta có một câu gồm 5 chữ “*Đẹp trai lắm gái theo*”, thì mạng nơ-ron được triển khai sẽ gồm 5 tầng nơ-ron tương ứng với mỗi chữ một tầng. Lúc đó việc tính toán bên trong RNN được thực hiện như sau:

* là đầu vào tại bước . Ví dụ, là một véc-tơ one-hot tương ứng với từ thứ 2 của câu (trai).
* là trạng thái ẩn tại bước . Nó chính là ***bộ nhớ*** của mạng. được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó:. Hàm thường là một hàm phi tuyến tính như tang hyperbolic (tanh) hay ReLu. Để làm phép toán cho phần tử ẩn đầu tiên ta cần khởi tạo thêm , thường giá trị khởi tạo được gán bằng 0.

3.2 Khả năng của RNN

Trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP - Natural Language Processing), đã ghi nhận được nhiều thành công của RNN cho nhiều vấn đề khác nhau. Tại thời điểm này, tôi muốn đề cập tới một mô hình phổ biến nhất được sử dụng của RNN là **LSTM**. LSTM (Long Short-Term Memory) thể hiện được sự ưu việt ở điểm có thể nhớ được nhiều bước hơn mô hình RNN truyền thống. Nhưng bạn không cần phải quá lo lắng vì LSTM về cơ bản giống với cấu trúc của RNN truyền thống, chúng chỉ khác nhau ở cách tính toán của các nút ẩn.

3.3 Huấn luyện RNN

Huấn luyện mạng RNN cũng tương tự như các mạng nơ-ron truyền thống, tuy nhiên giải thuật lan truyền ngược (backpropagation) phải thay đổi một chút. Đạo hàm tại mỗi đầu ra phụ thuộc không chỉ vào các tính toán tại bước đó, mà còn phụ thuộc vào các bước trước đó nữa, vì các tham số trong mạng RNN được sử dụng chung cho tất cả các bước trong mạng. Ví dụ, để tính đạo hàm tại a phải lan truyền ngược cả 3 bước phía trước rồi cộng tổng đạo hàm của chúng lại với nhau. Việc tính đạo hàm kiểu này được gọi là lan truyền ngược liên hồi (BPTT - Backpropagation Through Time). Nếu giờ bạn chưa thể hiểu được BPTT thế nào thì cũng đừng lo sợ vì trong bài sau ta sẽ xem xét cụ thể nó là gì sau. Còn giờ, chỉ cần nhớ rằng với các bước phụ thuộc càng xa thì việc học sẽ càng khó khăn hơn vì sẽ xuất hiện vấn đề hao hụt/bùng nổ (vanishing/exploding) của đạo hàm. Có một vài phương pháp được đề xuất để giải quyết vấn đề này và các kiểu mạng RNN hiện nay đã được thiết kế để triệt tiêu bớt chúng như LSTM chẳng hạn.

3.4 Mạng LSTM

Gần đây, mạng LSTM mà ta có đề cập một chút phía trên được chú ý và sử dụng khá phổ biến. Về cơ bản mô hình của LSTM không khác mô hình truyền thống của RNN, nhưng chúng sử dụng hàm tính toán khác ở các trạng thái ẩn. Bộ nhớ của LSTM được gọi là tế bào (Cell) và bạn có thể tưởng tượng rằng chúng là các hộp đen nhận đầu vào là trạng thái phía trước và đầu vào hiện tại . Bên trong hộp đen này sẽ tự quyết định cái gì cần phải nhớ và cái gì sẽ xoá đi. Sau đó, chúng sẽ kết hợp với trạng thái phía trước, nhớ hiện tại và đầu vào hiện tại. Vì vậy mà ta có thể truy xuất được quan hệ của các từ phụ thuộc xa nhau rất hiệu quả.

# VẤN ĐỀ TÁCH TỪ TRONG VĂN BẢN TIẾNG VIỆT

4.1 Tách từ là gì?

Tách từ là một vấn đề cơ bản trong xử lý ngôn ngữ tự nhiên. Đó là vấn đề phân đoạn văn bản thành từng từ hoặc token (ta dùng từ “token” để chỉ đơn vị sau khi tách từ). Về cơ bản, đơn vị từ vựng là đơn vị ngữ nghĩa nhỏ nhất có ý nghĩa dễ hiểu của con người. Khi đọc văn bản, con người dùng các quy trình thần kinh để tách từ câu văn thành các token. Nhưng làm thế nào chúng ta có thể thực hiện những quy trình này trong một chương trình máy tính như là bước đầu tiên để giúp máy tính hiểu văn bản?

Một số ngôn ngữ viết có điểm đánh dấu ranh giới rõ ràng như ký tự không gian như văn bản tiếng Anh hoặc các ngôn ngữ phương Tây khác. Đối với những ngôn ngữ này, phân đoạn chữ là một vấn đề dễ dàng. Ví dụ, cho một câu tiếng Anh:

* On the evening of March 31st, Elon Musk unveiled Tesla’s sinuous Model 3, the company’s first affordable electric-car model.

Câu này có thể dễ dàng đánh dấu vào các thẻ bằng cách sử dụng ký tự khoảng cách, dấu chấm câu và các ký tự dấu phân cách khác.

* [On] [the] [evening] [of] [March] [31st] [,] [Elon] [Musk] [unveiled] [Tesla]… [electric-car] [model] [.]

Tuy nhiên, đối với nhiều ngôn ngữ như tiếng Việt, khoảng cách không phải luôn luôn là dấu tách từ và do đó tách từ là một vấn đề không nhỏ. Ví dụ, câu sau đây:

* Pep Guardiola sẽ có cơ hội phục hận cho đội bóng ông từng gắn bó nhiều năm.

nên được tách như sau

* [Pep Guardiola] [sẽ] [có] [cơ hội] [phục hận] [cho] [đội] [bóng] [ông] [từng] [gắn bó] [nhiều] [năm] [.]

Như đã thấy, trong các văn bản tiếng Việt, khoảng cách được sử dụng để phân biệt âm tiết, chứ không phải từ, và một từ ghép có thể có nhiều âm tiết. Cần lưu ý rằng mặc dù hầu hết các âm tiết đều là những từ, có nhiều âm tiết không thể được sử dụng như một từ. Ví dụ, "thạc sĩ" là một từ, nhưng "thạc" không phải là một từ trong từ điển tiếng Việt.

4.2 Tại sao tách từ trong Tiếng Việt lại khó?

Như đã trình bày ở phần trên, tách từ chữ Việt không phải là một vấn đề tầm thường. Một số nhà khoa học máy tính Việt Nam đã giải quyết vấn đề này và phát triển một số thuật toán máy tính hiệu quả để giải quyết nó với độ chính xác cao.

Trở ngại lớn nhất trong tách từ Tiếng Việt là sự mơ hồ của dấu khoảng trắng. Câu hỏi chính là khoảng trắng nào dùng để tách từ khoảng trắng nào dùng để tách âm tiết? Trong một cụm từ đơn giản chứa ba âm tiết liên tiếp “a b c”, có bốn kiểu tách từ có thể được tạo ra: “[a] [b] [c]”, “[a b] [c]”, “[a] [b c]”, and “[a b c]”. Đối với một cụm từ dài hơn, số lượng các khả năng phân chia tăng theo cấp số nhân.

Việc phân tích cẩn thận các trường hợp không rõ ràng cho thấy có hai loại nhầm lẫn chính khi phân cụm từ:

1. **Chồng chèo mơ hồ**: đưa ra một cụm từ có ba âm tiết "a b c", hoặc "[a] [b c]" hoặc "[a b] [c]" là cách tách hợp lệ. Ví dụ, “thuộc địa bàn” có hai cách tách hợp lý “[thuộc địa] [bàn]” hoặc“[thuộc] [địa [bàn]”; tương tự “tổ hợp âm” có thể được chia thành “[tổ hợp] [âm]” hoặc “[tổ] [hợp âm]”. Xác định cách tách nào là tốt nhất so với các cách tách còn lại không phải là vấn đề dễ giàn.
2. **Kết hợp mô hồ**: đưa ra một cụm từ có hai âm tiết “a b”, “[a b]” hoặc “[a] [b]” là hợp lệ và không thể đưa ra lựa chọn tốt mà không biết ý nghĩa của cụm từ trong ngữ cảnh. Ví dụ, hai âm tiết “chanh chua” nên được coi như một từ “[chanh chua]” trong câu “Cô gái chanh chua”. Tuy nhiên, chúng nên được tách thành hai từ “[chanh] [chua]” trong câu “Cô gái ăn quả chanh chua”.

Ngoài ra, việc nhận biết các thực thể được đặt tên bằng văn bản là mơ hồ trong nhiều trường hợp. Vì một thực thể được đặt tên được biểu thị bằng chữ viết hoa ở đầu những âm tiết có liên quan, chẳng hạn như “thủ tướng Nguyễn Tấn Dũng”, chúng ta có thể rút ra được một quy tắc để nắm bắt được sự thường xuyên này và trích xuất “Nguyễn Tấn Dũng” như là một token. Tuy nhiên, nếu chúng ta thấy một câu mở với “Ông Nguyễn Tấn Dũng”, hoặc “Theo Đài Hà Nội”, hoặc “Theo Walcott đã ghi bàn thắng”?

4.3 Biểu thức chính quy (Regular Expressions)

Biểu thức chính quy là một ngôn ngữ mẫu tìm kiếm mạnh mẽ có thể giúp xác định và kết hợp nhiều mẫu. Biểu thức chính quy chủ yếu được sử dụng với chuỗi. Học cách xây dựng và sử dụng các biểu thức thông thường là một chủ đề trong lý thuyết ngôn ngữ và lý thuyết. khoa học máy tính.

Nhiều ngôn ngữ lập trình cung cấp khả năng biểu thức chính quy (regexp), một số được xây dựng sẵn (ví dụ: Perl hoặc JavaScript), một số thông qua thư viện chuẩn (ví dụ: Java, Python hoặc C / C ++). Trong phần sau, tôi mô tả một số cấu trúc cơ bản của regexp và các ví dụ của chúng trong ngôn ngữ lập trình Java.

|  |  |
| --- | --- |
| [abc] | a, b hoặc c |
| [^abc] | mọi ký tự không phải a, b, hoặc c |
| [a-z] | a tới z |
| [A-Z] | A tới Z |
| . | mọi ký tự |
| \d | một chữ số: [0-9] |
| \D | một ký tự không phải chữ số:[^\d] |
| \s | một khoảng trắng, có thể cả ký tự xuống dòng |
| \S | không phải một ký tự trống, ký tự xuống dòng |
| \w | một ký tự chữ |
| \W | không phải một ký tự chữ |

Bảng 4.1: Các ví dụ Regex

Các định lượng tham lam phổ biến là:

|  |  |
| --- | --- |
| X? | X, một lần hoặc không |
| X\* | X, không hoặc nhiều lần |
| X+ | X, một hoặc nhiều lần |
| X{n} | X, chính xác n lần |
| X{n,} | X, ít nhất n lần |
| X{n,m} | X, ít nhất n nhưng không nhiều hơn m lần |

Bảng 4.2: Các định lượng Regex phổ biến

Lưu ý rằng các ký tự ‘.’, ‘-’, ‘?’, ‘\*’, ‘+’, ‘[’, ‘]’, ... được gọi là các siêu-kí tự vì chúng được sử dụng để chỉ các mẫu. Để đại diện cho một siêu ký tự, chúng ta cần một "escape" bằng dấu gạch chéo ngược trước, ví dụ \. cho ký tự dấu chấm, \ \* cho ký tự sao, v.v ...

Bằng cách kết hợp các cấu trúc regexp cơ bản ở trên, bây giờ chúng ta có thể xác định các mẫu văn bản không tầm thường. Ví dụ:

|  |  |  |
| --- | --- | --- |
| **Regexp** | **Miêu tả** | **Ví dụ** |
| \d{4} | một năm với 4 chữ số | 2016, 1890 |
| \d{4}\ – \d{4} | một khoảng thời gian với một năm bắt đầu và một năm kết thúc | 1890-1969 |
| (0\*[1-9]|1[012])[\-/\.]\d{4} | một ngày định dạng mm-yyyy, hoặc mm / yyyy, hoặc mm.yyy | 10-1980, 07/2012, 9/2015 |
| [\+\-]?([0-9]\*)?[0-9]+([\.,]\d+)\* | số bất kỳ, số dương hoặc âm, số nguyên hoặc số thực bằng tiếng Anh hoặc tiếng Việt | 22.30, -22,30, +34,567,89 |

Bảng 4.3: Ví dụ kết hợp Regex

Chúng ta cũng có thể trích xuất các thực thể được đặt tên như Elon Musk hoặc Pep Guardiola trong các câu ví dụ bằng cách thiết kế một biểu thức chính quy phù hợp với chúng. Một thực thể được đặt tên có chứa một hoặc nhiều âm tiết có chữ cái đầu tiên được viết hoa và các âm tiết được cách nhau bởi các ký tự khoảng cách, vì vậy chúng ta có thể chỉ định regexp sau để khớp với các thực thể có tên tiếng Anh: ([A-Z][a-z]\*)([\s+\-][A-Z][a-z]+)\*. Đối với các thực thể được đặt tên ở Việt Nam, thì phức tạp hơn một chút vì chúng ta phải giải quyết các ký tự Unicode. Để kết hợp một thực thể có tên như Nguyễn Tấn Dũng, chúng ta có thể chỉ định regexp sau đây:

([\p{Lu}][\p{L}&&[^\p{Lu}]]\*)([\s+\-][\p{Lu}][\p{L}&&[^\p{Lu}]]+)\*.

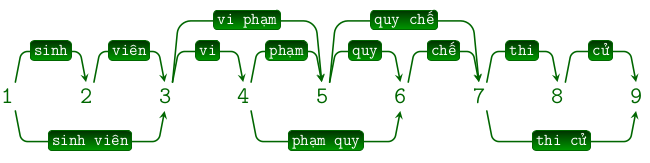
Trong một cách tương tự, chúng ta có thể xây dựng các biểu thức thông thường để kết hợp và trích xuất nhiều loại mã thông báo bằng văn bản tiếng Việt hoặc tiếng Anh, chẳng hạn như viết hoa hết hoặc chữ viết tắt (ví dụ., TP HCM, ĐHQG, TDT ...), email, liên kết web, ngày và giờ, v.v. Đây là bước thiết yếu đầu tiên của tách từ chữ Việt.

4.4 Biểu đồ cụm từ

Trong phần trên, chúng ta đã trình bày cách để khớp với những mẫu trong Tiếng Việt sử dụng biểu thức chính quy và sự mơ hồ khi tách từ Tiếng Việt. Phần này và phần dưới sẽ trình bày cách tách từ có nhiều âm tiết và giải quyết sự mơ hồ đó.

Đầu tiên chúng ta trình đoạn văn từ dưới dạng đồ thị đơn giản *G=(V,E)* trong đó tập nút V có n + 1 nút được biểu hiện bởi các số nguyên liên tiếp từ 1 đến n + 1 và tại đó tập cạnh E chứa các đường dẫn trực tiếp (u, v) từ một nút nguồn tới một nút đích v Trong đồ thị này, đường cung luôn luôn hướng từ trái sang phải, nghĩa là u <v cho tất cả u và v. Có một vòng cung (u, v) nếu dãy âm tiết tạo thành một từ trong từ điển tiếng Việt. Để dễ trình bày, mỗi vòng cung được gắn nhãn với từ tương ứng.

Ví dụ, biểu đồ của đoạn văn *[sinh viên vi phạm quy chế thi cử]* được miêu tả như sau:



Hình 4.1: Biểu độ cụm từ “sinh viên vi phạm quy chế thi cử”

Để kiểm tra xem một chuỗi ký tự có phải là một từ trong bộ từ vựng hay không, chúng ta sử dụng bảng băm được xây dựng trước, hoặc áp dụng một cấu trúc dữ liệu hiệu quả hơn như cây tiền tố, hoặc một automat trạng thái hữu hạn (FSA), ý tưởng là tối thiểu FSA để tra cứu rất nhanh.

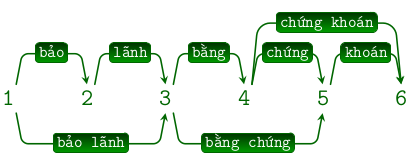
Bây giờ, chúng ta thấy rằng một phân đoạn từ tương ứng với một đường dẫn trong biểu đồ này. Mỗi đường đi bắt đầu từ nút đầu tiên 1 và kết thúc tại nút cuối cùng n + 1. Thật thú vị, một con đường ngắn nhất, đi qua một số lượng tối thiểu các nút, thường tương ứng với phân đoạn từ đúng nhất có thể. Trong ví dụ trên, đường đi ngắn nhất là 1-> 3-> 5-> 7-> 9, tương ứng với phân đoạn từ sau:

* *[sinh viên] [vi phạm] [quy chế] [thi cử]*

Cách tiếp cận này được gọi là phương pháp kết hợp tối đa vì chúng ta luôn xem xét các chuỗi con dài nhất tương ứng với các đường dẫn ngắn nhất trên một biểu đồ cụm từ.

4.5 Sự mơ hồ chồng chéo

Xem xét cụm từ sau: *bảo lãnh bằng chứng khoán*. Biểu đồ tương ứng của nó như sau:



Hình 4.2: Biểu đồ câu văn “bảo lãnh bằng chứng khoán”

Chúng tôi thấy rằng có hai đường dẫn ngắn nhất trên biểu đồ này. Con đường ngắn nhất đầu tiên là 1-> 3-> 4-> 6. Thứ hai là 1-> 3-> 5 -> 6. Những đường dẫn tương ứng với hai phân đoạn có thể:

* *[bảo lãnh] [bằng] [chứng khoán]*
* *[bảo lãnh] [bằng chứng] [khoán]*

Điều này là do sự tồn tại của một nhóm âm tiết chồng chéo *[bằng chứng khoán]*. Lưu ý rằng trong biểu đồ đầu tiên, chúng ta cũng có một nhóm âm tiết chồng chéo *[phạm quy chế]*. Tuy nhiên, từ *phạm* đầu tiên được bao gồm trong từ dài hơn *vi phạm*, dẫn đến phân khúc chính xác dẫn đến cách tách chính xác *[vi phạm] [quy chế]*, tương ứng với đường đi ngắn nhất 3-> 5-> 7.

4.6 Mô hình Ngôn ngữ

Cách tiếp cận để lựa chọn cách tách từ tốt nhất trong nhiều giải pháp thay thế là sử dụng một mô hình ngôn ngữ để tính xác suất của các chuỗi từ. Đầu tiên chúng ta tính toán xác suất của các chuỗi từ khác nhau tương ứng với các cách tách từ khác nhau và sau đó chọn một với xác suất cao nhất, nghĩa là cách tách từ có nhiều khả năng nhất theo mô hình ngôn ngữ.

Về mặt kỹ thuật, một mô hình ngôn ngữ thống kê là sự phân bố xác suất đối với các chuỗi từ. Khả năng ước lượng tương đối xác suất của các chuỗi từ khác nhau làm cho các mô hình ngôn ngữ rất hữu ích trong nhiều ứng dụng, đặc biệt là trong nhận dạng giọng nói, dịch máy, và các vấn đề ghi nhãn câu.

Giả sử chúng ta có một mô hình ngôn ngữ cho tiếng Việt, chúng ta có thể sử dụng nó để tính toán xác suất của mỗi chuỗi từ tiếng Việt. Cụ thể, trong ví dụ trên, nhiều khả năng là xác suất của chuỗi từ thứ nhất *P([bảo lãnh], [bằng], [chứng khoán])* lớn hơn xác suất của chuỗi từ thứ hai *P([bảo lãnh], [bằng chứng], [khoán])*, và do đó, cách tách từ đầu tiên được chọn bởi vì nó có khả năng chính xác hơn cách tách từ khác.

4.7 Phân loại khoảng trắng

Trong phần trước, chúng ta đã thấy sự mơ hồ chồng chéo có thể được giải quyết hiệu quả bằng cách sử dụng một thuật toán đồ thị và một mô hình ngôn ngữ thống kê. Một câu hỏi vẫn còn tồn tại: những sự mơ hồ kết hợp được giải quyết như thế nào?

Nhớ lại từ phần đầu rằng sự mơ hồ kết hợp thường liên quan đến một cụm từ có hai âm tiết a b trong đó [a b] hoặc [a] [b] là sự tách từ chính xác trong ngữ cảnh. Trong phương pháp tiếp cận tối đa, phân đoạn đầu tiên luôn được chọn; do đó phương pháp này không thể giải quyết sự mơ hồ kết hợp. Một phương pháp đơn giản để đối phó với sự mơ hồ kết hợp là sử dụng một trình phân loại khoảng trắng có khả năng phân biệt bản chất của ký tự khoảng trắng giữa hai âm tiết a và b. Đây là một phân loại nhị phân xác định một ký tự khoảng trắng là một dấu phân cách âm tiết hoặc một dấu phân tách chữ, tương ứng với phân đoạn [a b] hoặc phân đoạn [a] [b].

Chúng ta có thể sử dụng bất kỳ phương pháp phân loại nhị phân, dựa trên quy tắc hoặc thống kê để giải quyết sự mơ hồ kết hợp. Một số mô hình thống kê phổ biến và được sử dụng rộng rãi bao gồm cây quyết định, hồi quy hậu cần, hoặc các mô hình trình tự tiên tiến hơn như mô hình Markov tối đa-entropy hoặc các trường ngẫu nhiên có điều kiện. Nó nằm ngoài phạm vi của đồ án này.

4.8 Phần mềm

Các phương pháp phân đoạn từ được mô tả trong chương này đã được thực hiện trong bộ công cụ Vitk và phần mềm vnTokenizer. Chúng ta sẽ sử dụng nó thay vì hiện thực lại.

# HIỆN THỰC

5.1 Tách từ tiếng Việt

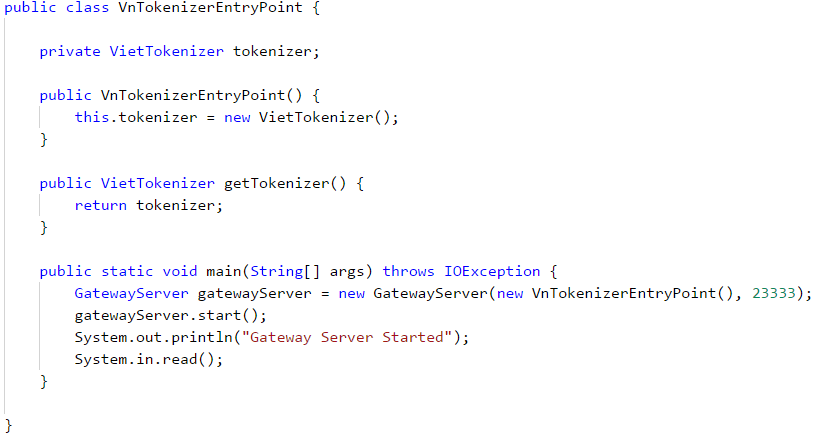
Để có thể sử dụng vnTokenizer được viết bằng Java trên chương trình Python chúng ta cần py4j thứ cho phép:

1. Truy cập JVM đang chạy chương trình của bạn.
2. Truy cập các đối tượng mà bạn đã tạo ra trong JVM.

5.1.1 Chương trình trên Java

Các tính năng trên được cung cấp bởi hai đối tượng. Đối tượng đầu tiên là một cá thể GatewayServer: nó cho phép các chương trình Python giao tiếp với JVM thông qua một local socket. Đối tượng thứ hai được gọi là entry point và nó có thể là bất kỳ đối tượng nào (ví dụ: Facade, một singleton, một danh sách, v.v.).

GatewayServer cung cấp bởi Py4J có thể được sử dụng nhưng bạn cũng có thể cấu hình nó và chỉ định một địa chỉ mạng và cổng, nếu mặc định (localhost, 25333) không làm việc. Constructor của GatewayServer có thể nhận tham số đầu vào khác.



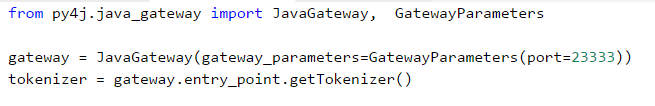
Hình 5.1: Code py4j trên Java

Có một vài dòng quan trọng trong đoạn mã này:

1. Trước tiên, bạn khai báo một lớp sẽ cung cấp quyền truy cập vào một toeknizer được cấu hình sẵn.
2. Sau đó, bạn tạo ra một hàm main. Hàm main này có thể được đặt trong một lớp khác. Điều đầu tiên bạn làm trong hàm main là khởi tạo một GatewayServer và liên kết nó với một entrypoint.
3. Cuối cùng, bạn cần phải start gateway để nó có thể chấp nhận yêu cầu đến từ Python.

5.1.2 Chương trình trên Python

Bây giờ bạn sẽ viết chương trình python sẽ truy cập chương trình Java của bạn.



Hình 5.2: Code py4j trên Python

1. Bước đầu tiên là nhập class Py4J cần thiết.
2. Tiếp theo, khởi tạo một JavaGateway. Thông số mặc định thường là đủ cho các trường hợp thông thường. Khi bạn tạo một JavaGateway, Python cố gắng kết nối với một JVM với một cổng (localhost trên cổng 23333).
3. Từ đối tượng gateway chúng ta có thể truy cập entrypoint bằng cách đề cập đến thuộc tính entry\_point của nó:

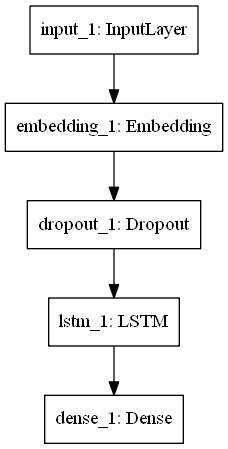
Biến tokenizer bây giờ đã chứa tokenizer chúng ta có thể sử dụng nó như trên Java.

5.2 Mô hình

Trong bài làm này tôi xây dựng QA dựa trên IR-based QA nó không hoàn toàn là một IR-based QA nhưng các bước thực hiện đều nằm trong IR-based QA. Như đã đề cập ở chương đầu, trước hết chúng ta cần trích xuất những thông tin cần thiết từ câu hỏi, sau đó dùng những thông tin đó để tìm tài liệu và đoạn văn phù hợp, cuối cùng là trả ra đáp án dựa trên câu hỏi, đoạn văn và tài liệu tìm được.

5.2.1 Mô hình lựa chọn tài liệu

Đây là mô hình trả ra khả năng mà một tài liệu có thể dùng để trả lời câu hỏi.

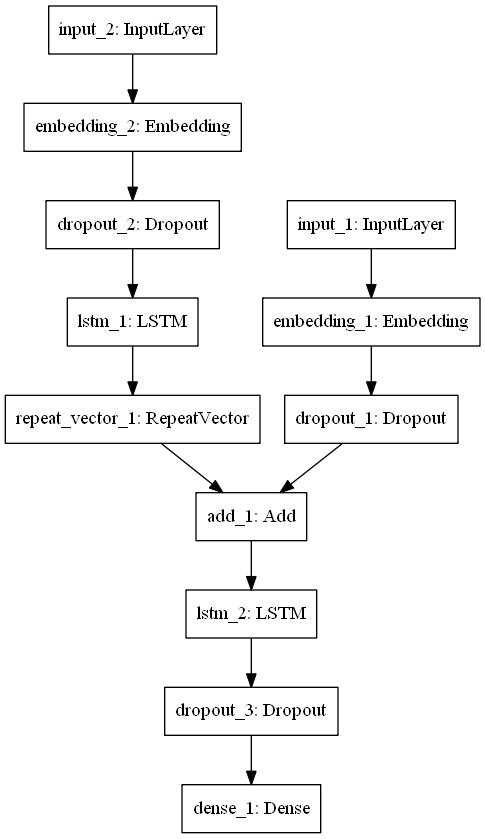


Hình 5.3: Mô hình lựa chọn tài liệu

Chúng ta sẽ truyền vào các token đã được tách ra từ câu hỏi dưới dạng véc-tơ và sẽ thu được xác suất phân phối ở cuối.

5.2.2 Mô hình trả lời

Đây là mô hình dùng sinh câu trả lời từ một đoạn văn đã được lựa chọn và câu hỏi truyền vào.



Hình 5.4: Mô hình chọn câu trả lời

Đây là mô hình với 2 input nên chúng ta cho đoạn văn được lựa chon và câu hỏi vào.

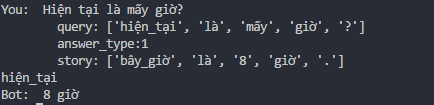
5.3 Ứng dụng demo

5.3.1 Nguyên lý hoạt động

Để trả lời một câu hỏi người dùng nhập vào:

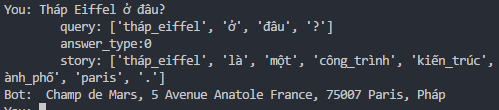
1. Ứng dụng trước tiên chọn tài liệu phù hợp với câu hỏi nhờ vào mô hình lựa chọn tài liệu.
2. Sau đó tiến thành lựa chọn đoạn văn phù hợp với câu hỏi dựa trên câu hỏi bằng một thuật toán xếp hạng phù hợp với loại tài liệu được chọn ở bước trước đó.
3. Chọn ra mô hình trả lời phù hợp với loại tài liệu và đưa các thông tin vào để lấy ra câu trả lời.

5.3.2 Một số hình ảnh minh họa



Hình 5.5: Minh họa demo 1

* Tài liệu 1: Thời gian
* Xếp hạng theo thời gian
* Mô hình trả lời 1



Hình 5.6: Minh họa demo 2

* Tài liệu 0: Địa điểm
* Xếp hạng theo số lượng token của câu hỏi xuất hiện trong đoạn văn
* Mô hình trả lời 0

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. Đỗ Minh Hải (2017), Bài giới thiệu RNN.
2. viet.jnlp.org (2017), Vietnamese Natural Language Processing

**Tiếng Anh**

1. Dan Jurafsky and James H. Martin (2017), Speech and Language Processing.
2. Chris McCormick (2016), Word2Vec Tutorial.
3. Le Hong Phuong (2016), Vietnamese Word Segmentation

PHỤ LỤC