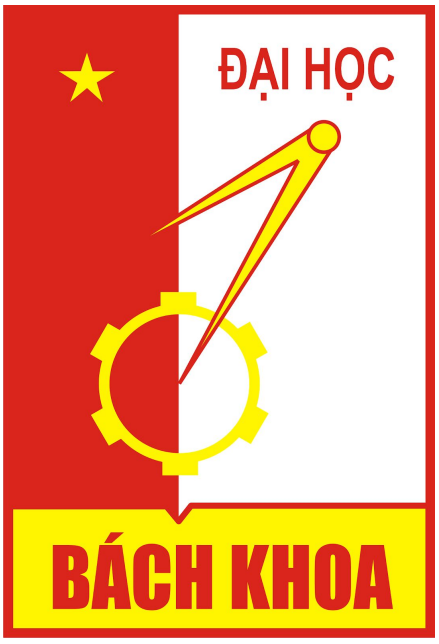
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



Báo cáo bài tập lớn môn học khai phá web
Đề tài: Keyphrase extraction SIFRank

GVHD: TS. Nguyễn Kiêm Hiếu
Nhóm sinh viên thực hiện: Nhóm 9

Họ tên:	MSSV
Phạm Hùng Cường	20160576
Cao Văn Đức	20161056
Lê Duy Hưng	20162009
Nguyễn Tử Toàn Lợi	20162569

HÀ NỘI 06/2020

1. Giới thiệu bài toán

Hiện nay, trong thời đại bùng nổ của công nghệ thông tin, mạng internet đang ngày càng phổ biến. Lượng thông tin con người có thể tiếp cận được là cực kỳ khổng lồ. Mạng xã hội, các hình thức báo điện tử, tuyển dụng trực tuyến ... đang phát triển mạnh và sẽ trở thành xu thế trong tương lai. Trong hoàn cảnh này, nhu cầu tóm tắt thông tin, bóc tách thông tin tự động trong văn bản là rất lớn.

Bóc tách cụm từ quan trọng của văn bản từ lâu đã là bài toán quan trọng trong những bài toán về xử lý ngôn ngữ tự nhiên. Là bài toán cần xử lý khi muốn tự động tóm tắt văn bản, cũng cho phép tìm kiếm nhanh hơn và chính xác hơn những văn bản chuyên sâu, thuộc những chuyên ngành hẹp trong các bộ dữ liệu lớn, hay góp phần tăng độ chính xác cho những lớp bài toán khác như phân loại văn bản. Đặc biệt, trong hoàn cảnh khối lượng thông tin con người có thể tiếp cận được là rất nhiều như hiện tại, bài toán này càng cần được xử lý triệt để, kể cả về hiệu năng lẫn độ chính xác.

Vì lý do này, nhóm em quyết định chọn bài toán **Bóc tách cụm từ quan trọng trong văn bản** làm đề tài bài tập lớn. Vừa để tìm hiểu, nghiên cứu các kiến thức mới trong lĩnh vực xử lý ngôn ngữ tự nhiên, vừa để tiếp cận trước với các bài toán quan trọng cần xử lý trong tương lai, đặc biệt là bài toán bóc tách dữ liệu trong văn bản.

2. Kiến thức liên quan

2.1. Stanford CoreNLP

Stanford CoreNLP cung cấp một bộ công cụ công nghệ ngôn ngữ của con người. Nó có thể đưa ra các dạng cơ bản của từ, phần nói của chúng, cho dù chúng là tên của các công ty, con người, v.v., bình thường hóa ngày, giờ và số lượng, đánh dấu cấu trúc câu theo cụm từ và phụ thuộc cú pháp, chỉ ra cụm từ danh từ nào đề cập đến cùng một thực thể, biểu thị tình cảm, trích xuất quan hệ cụ thể hoặc lớp mở giữa các đề cập thực thể, nhận được các trích dẫn mà mọi người nói, v.v.

Các ưu điểm:

- Một bộ công cụ NLP tích hợp một loạt các công cụ phân tích cú pháp
- Hiện đại, được cập nhật thường xuyên
- Có sẵn các API
- Có khả năng chạy như một dịch vụ web đơn giản

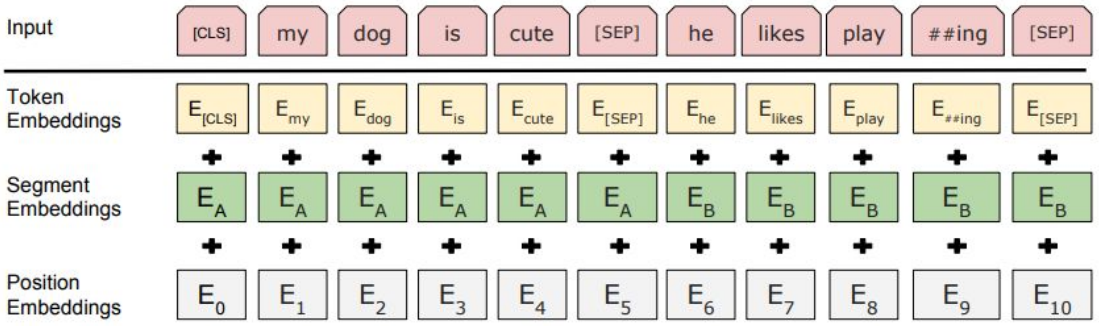
Mục tiêu của Stanford CoreNLP là làm cho việc áp dụng một loạt các công cụ phân tích ngôn ngữ vào một đoạn văn bản rất dễ dàng. Một đường ống công cụ có thể được chạy trên một đoạn văn bản đơn giản chỉ với hai dòng mã. CoreNLP được thiết kế rất linh hoạt và có thể mở rộng. Với một tùy chọn duy nhất, bạn có thể thay đổi công cụ nào sẽ được bật và tắt. Stanford CoreNLP tích hợp nhiều công cụ NLP của Stanford, bao gồm trình gắn thẻ một phần của bài phát biểu (POS) , trình nhận dạng thực thể có tên (NER) , trình phân tích cú pháp , hệ thống phân giải lõi , phân tích tình cảm , học tập mô hình khởi động và công cụ khai thác thông tin mở.

2.2. BERT

BERT (Bidirectional Encoder Representations from Transformers), là một mô hình NLP pre-train phát triển bởi Google. Điều đặc biệt của Bert ở chỗ nó là pre-train model bidrirection thành công đầu tiên trong deep neural network. Ngoài ra còn đặc điểm khác là nó học biểu diễn ngôn ngữ không giám sát và pre-train với chỉ plain-text corpus. Và cuối cùng, Bert được pre-train bằng tập corpus lớn unlabeled, bao gồm: toàn bộ Wikipedia (2,500 million words) và book corpus (800 million words).

Cách BERT hoạt động: So với các mô hình context-free truyền thống (w2v hay GloVe) chỉ sinh ra một biểu diễn với mỗi từ (ví dụ: từ right có cùng biểu diễn trong 2 câu “I’m sure I’m right” và “Take a right turn”), thì BERT lại

biểu diễn dựa vào cả văn cảnh trước và sau của từ (do đó nó được coi là bi-directional). Và **BERT** đạt được điều đó nhờ sử dụng 2 chiến lược.



Chiến lược đầu tiên - **Mask Language Model (MLM)** - bằng cách che đi (masking) một vài từ trong input và để cho các từ còn lại đoán từ được che. Trước khi input được đưa vào model, 15% từ trong chuỗi sẽ được thay thế bằng [MASK] token và để cho model đoán từ nguyên gốc trước khi bị che dựa vào ngữ cảnh mà các từ còn lại cung cấp.

Input: The $[MASK]_1$ is not working. It's unable to $[MASK]_2$

Labels: $[MASK]_1$ = computer; $[MASK]_2$ = start.

Ví dụ chuỗi với 2 từ bị che

Chiến lược thứ 2 là **Next Sentence Prediction (NSP)**, để BERT có thể học được mối liên hệ giữa các câu. Ở quá trình training, model sẽ được input một cặp câu, và phải dự đoán xem cặp câu này có phải là 2 câu liên tiếp nhau không. Ví dụ, có cặp câu A và B, model cần xác định xem B là câu kế tiếp câu A trong corpus hay chỉ là một câu ngẫu nhiên.

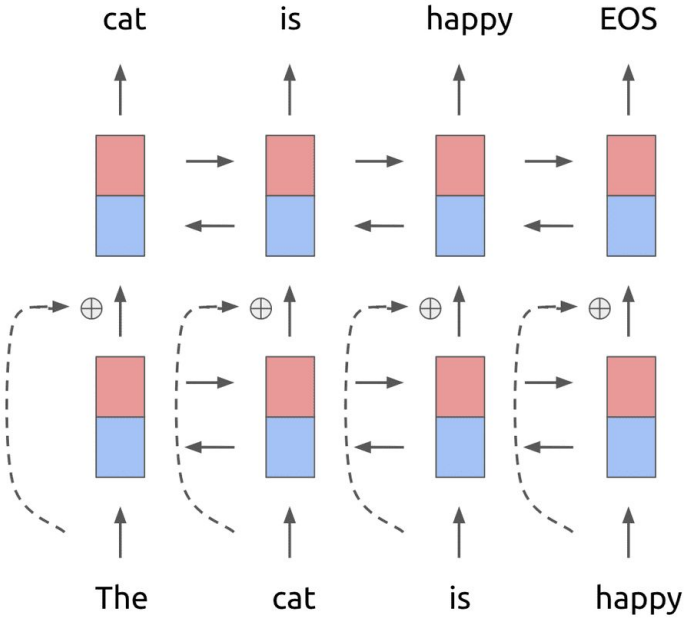
Sentence A = The computer is not working.
Sentence B = It's unable to start.
Label = IsNextSentence

Sentence A = The computer is not working.
Sentence B = Coffee is very tasty.
Label = NotNextSentence

Quá trình training BERT model cần kết hợp cả 2 chiến lược này nhằm tối thiểu hoá hàm lỗi kết hợp giữa 2 chiến lược. Output của BERT là vector mang đặc trưng ngữ nghĩa của từ.

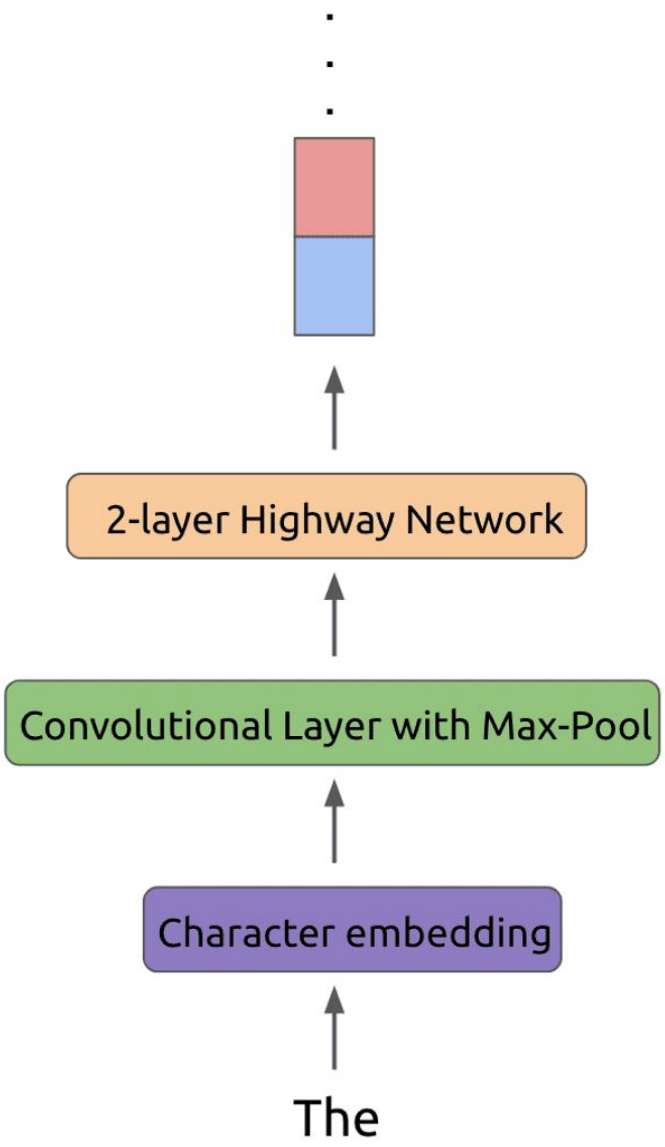
2.3. ELMo

Kiến trúc của ELMo xây dựng dựa trên kiến trúc 2-layer bidirectional LSTM, vì vậy nó không chỉ hiểu từ phía trước nó mà cả từ đằng sau. Với các residual connection được thêm vào ở giữa layer 1 và layer 2 (input của layer 1 thêm vào output của nó rồi được input vào layer 2), giúp gradients không bị truyền qua các hàm non-linear activation mà flow trực tiếp qua network.



Với các mô hình truyền thống, mỗi token từ input layer 1 (The cat is happy) được chuyển thành 1 word-embedding độ dài cố định bằng cách tạo ma trận với size = *(Vocabulary size)* x *(Word embedding dimension),* hoặc sử dụng các pretrained embedding như GLoVe cho mỗi token.

Còn với ELMo, thay vì sử dụng embedding với độ dài cố định cho mỗi word, toàn bộ câu sẽ được sử dụng để sinh ra embedding. Công việc này sẽ phức tạp hơn. Mỗi token sẽ được biểu diễn bằng character embeddings, sau đó đi qua tầng nhân chập (convolutional layer) với 1 vài bộ lọc, và cuối cùng đưa đến max-pool layer. Và cuối cùng, sẽ được truyền vào 2-layer highway network trước khi input vào LSTM layer. (tác dụng của các layer sẽ được giải thích ở dưới).

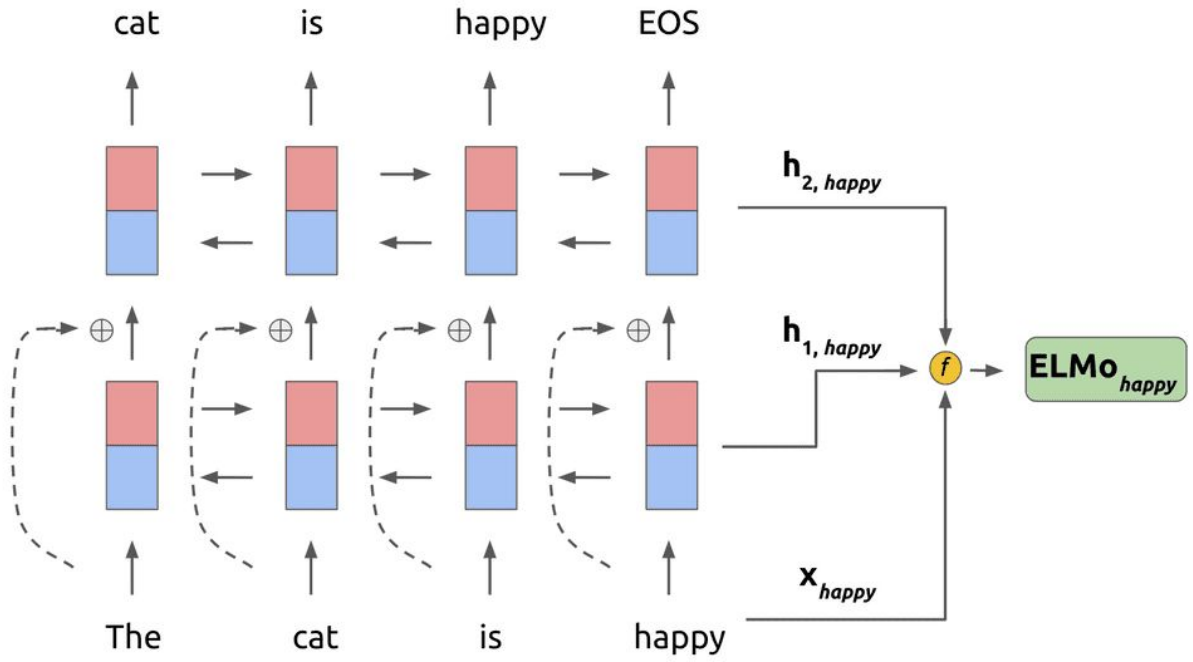


Transformation cho mỗi token trước khi input vào LSTM layer đầu

Cách transform input token này mang lại nhiều lợi ích:

- Thứ nhất, character embedding cho phép lấy được hình thái từ (thứ mà word-level embedding có thể bỏ sót) và đảm bảo vẫn vẫn có biểu diễn từ khi OOV
- Thứ hai, bộ lọc nhân chập cho phép lấy n-gram đặc trưng để tạo ra biểu diễn từ chính xác hơn.
- Cuối cùng, highway network layer cho phép thông tin từ input truyền vào tốt hơn.

Một khi đã được trained, ELMo sẽ được sử dụng như sau:



Giả sử, ta chỉ tập trung vào từ "happy", sử dụng trained 2-layer model, ta lấy biểu diễn từ của "happy" (x_happy) kết hợp với 2 cái ở hidden layer nhằm tạo thành biểu diễn mới cho task riêng biệt. Hàm F sẽ nhân mỗi vector với trọng số từ hidden layer.
Note: biểu diễn ELMo chỉ được sử dụng cho task riêng.

Tóm lại, những điểm nổi bật của ELMo:

- **Character-based**: ELMo có biểu diễn từ dựa vào character, cho phép network sử dụng hình thái từ tạo nên biểu diễn tốt hơn cho những token OOV trong quá trình training.
- **Deep:** biểu diễn từ tổng hợp từ tất cả layer trong deep pretrained neural network.
- **Contextual**: Biểu diễn từ phụ thuộc vào ngữ cảnh của từ

2.4. SIF

SIF (viết tắt của Smooth Inverse Frequency) là một phương pháp tính vector embedding của câu bằng cách tính tổng có trọng số của các vector embedding của từ trong câu, tuy có vẻ đơn giản nhưng thực sự đem lại hiệu quả và ở một số task nhất định còn đem lại hiệu suất cao hơn với những phương pháp có giám sát tinh vi khác, bao gồm cả một số model LSTM và RNN.

Để xây dựng model thì tác giả bài báo đã sử dụng các giả định là:

- Quá trình sinh ra một văn bản là một quá trình động, trong đó từ thứ t là được tạo ra ở bước t. Quá trình này được điều khiển bởi random walk của vector $c_t \in R^d$ được gọi là discourse vector hay có thể gọi là vector chủ đề của câu, vector này đại diện cho những gì đang được diễn đạt trong câu.
- Mỗi từ w trong từ điển cũng có vector biểu diễn $v_w \in R^d$.
- Xác suất mà từ w xuất hiện ở bước t tỉ lệ thuận với độ tương đồng giữa vector của từ v_w và vector chủ đề của câu ở bước t:

$$\Pr[w \text{ emitted at time } t \mid c_t] \propto \exp(\langle c_t, v_w \rangle)$$

- Vector chủ đề của câu c_t thực hiện slow random walk, vì thế trong một câu thì vector chủ đề không thay đổi nhiều và có thể coi như không đổi

Vector c_t chủ đề của câu chính là sentence embedding mà chúng ta cần tính.

Từ những giả định trên thì tác giả cho thấy rằng vector chủ đề của câu có thể được ước tính bằng Maximum A Posteriori (MAP) và bằng trung bình của các vector word embedding trong câu. [Arora et al. 2016, theorem 1]

Nhưng như vậy thì quá đơn giản và không xử lý được thực tế đó là có những từ có thể xuất hiện trong câu mà không hề liên quan đến chủ đề của câu, và có những từ phổ biến xuất hiện trong tất cả các chủ đề (ví dụ: “just” ,”the”, ”but”, etc). Vì thế họ bổ sung thêm tham số α, β và vector chủ đề phổ biến c_0 để biểu diễn cho chủ đề xuất hiện nhiều nhất mà chỉ liên quan đến ngữ pháp (chủ đề của các từ “just” ,”the”,...):

$$\Pr[w \text{ emitted in sentence } s \mid c_s] = \alpha p(w) + (1-\alpha)\frac{\exp(\langle \tilde{c}_s, v_w \rangle)}{Z_{\tilde{c}_s}}$$

trong đó $\overline{c_s} = \beta c_0 + (1-\beta)c_s$ và $c_0 \perp c_s$. Các tham số α, β là các đại lượng vô hướng, $Z_{\overline{c_s}} = \sum_{w \in V} exp(\langle \overline{c_s}, v_w \rangle)$ là một hằng số chuẩn hóa, p(w) là xác suất unigram của một từ xuất hiện trong cả tập corpus.

Khi đó ta thấy với $\alpha > 0$ thì bất kì từ nào cũng có thể xuất hiện nhờ vào p(w), và với $\beta > 0$ thì những từ liên quan đến chủ đề phổ biến cũng có thể xuất hiện.

Sử dụng Maximum Likelihood Estimation thì tác giả đã cho thấy rằng:

$$\tilde{c}_s = \sum_{w \in s} \frac{a}{p(w) + a} v_w$$

Với a = $\frac{1-\alpha}{\alpha Z}$ là siêu tham số để làm mịn. Ở đây ta có thể thấy với từ xuất hiện càng nhiều thì trọng số $\frac{1-\alpha}{\alpha Z}$ càng nhỏ.

Cuối cùng để tính được c_s thì chúng ta cần ước lượng được vector chủ đề phổ biến c_o. Vì c_s và c_0 trực giao, và vector c_0 là chung cho tất cả các câu, tác giả đề xuất có thể ước lượng c_0 bằng thành phần chính thứ nhất (first principal component) của c_s trên một tập các câu, sentence embedding cuối cùng thu được bằng cách trừ phép chiếu của c_s đi thành phần chính thứ nhất.

2.5. Cosine similarity

Trong xử lý ngôn ngữ tự nhiên, đầu vào của các bài toán là các câu, từ, kí tự. Nhưng đối với máy tính nó không thể hiểu được những câu, từ, kí tự trên là gì. Để giải quyết vấn đề trên người ta cần xây dựng mô hình hóa các câu, từ, kí tự thành các con số để máy tính có thể hiểu được nó. Cách thường được sử dụng ngày nay là biến đổi các câu, từ, kí tự thành các vector có số chiều nhất định, mỗi một thành phần trong vector đó sẽ biểu diễn một đặc trưng của từ. Cosine similarity là cơ chế tính độ tương đồng (gần nghĩa) của các câu, từ, kí tự khi đã được biểu diễn thành các vector.

Công thức tính Cosine similarity:

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

2.6. Yake và Tf-idf

2.6.1. YAKE

YAKE: Phương pháp tiếp cận không giám sát để trích xuất từ khóa tự động bằng tính năng văn bản.

YAYE! là một phương pháp trích xuất từ khóa tự động không giám sát có kích thước nhẹ, dựa trên các tính năng thống kê văn bản được trích xuất từ các tài liệu duy nhất để chọn các từ khóa quan trọng nhất của văn bản. Yake không cần phải được đào tạo về một bộ tài liệu cụ thể, nó cũng không phụ thuộc vào từ điển, văn bản bên ngoài, kích thước của văn bản, ngôn ngữ hoặc tên miền..

Các tính năng chính của YAKE:

- Cách tiếp cận không giám sát
- Corpus-Độc lập
- Tên miền và ngôn ngữ độc lập
- Tài liệu đơn

2.6.2. TF-IDF

tf–idf, viết tắt của thuật ngữ tiếng Anh *term frequency – inverse document frequency*, của một từ là một con số thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản.

tf-idf (của 1 từ trong 1 văn bản) **= tf x idf**

trong đó:

tf: thể hiện tần xuất xuất hiện của từ đó trong văn bản.

idf: thể hiện độ hiếm của những văn bản chứa từ đó.

2.7. VncoreNLP

VnCoreNLP là công cụ xử lý ngôn ngữ tự nhiên (NLP) cho tiếng Việt. Nó cung cấp nhiều tác vụ để xử lý các thành phần chính của NLP như: tách từ, gán nhãn từ loại (POS tagging), gán nhãn thực thể có tên(NER) và phân tích phụ thuộc.

Các ưu điểm của VnCoreNLP:

- Chính xác: Là bộ công cụ chính xác nhất cho việc xử lý ngôn ngữ tự nhiên tiếng Việt. Nó đạt được kết quả tốt nhất trên bộ dữ liệu chuẩn.
- Nhanh: VnCoreNLP rất nhanh, do đó ta có thể sử dụng nó cho những bộ dữ liệu lớn cần phân tích.
- Dễ dùng: Người dùng cuối không cần cài thêm bất cứ gói phụ thuộc nào bên ngoài. Người dùng có thể xử lý các tác vụ cần thiết qua giao diện dòng lệnh hoặc gọi đến API của VnCoreNLP

2.8. PhoBERT

Tuy đã đạt được những cải thiện đáng kể trong những công việc về xử lý ngôn ngữ tự nhiên nhưng BERT và những biến thể của nó phần lớn chỉ giới hạn cho tiếng Anh. Đối với các ngôn ngữ khác người ta cần xây dựng những mô hình cho những ngôn ngữ cụ thể dựa trên kiến trúc của mô hình BERT. phoBERT là một pre-train language models cho tiếng Việt.

- PhoBERT có 2 vesion là "base" và "large" là các mô hình pre-train language đơn ngữ đầu tiên được công khai cho tiếng Việt. PhoBERT pre-training đươc xây dựng dựa trên RoBERT nhằm tối ưu hóa BERT pre-training để có được hiệu suất tốt hơn.
- PhoBERT vượt trội so với các cách tiếp cận đơn ngữ và đa ngôn ngữ trước đây, đạt được các kết quả tốt về bốn nhiệm vụ NLP trong tiếng Việt về gắn thẻ gán nhãn loại từ(POS-tagging), phân tích phụ thuộc, nhận dạng thực thể(NER) và suy luận ngôn ngữ tự nhiên.

3. Bộ dữ liệu

3.1. Tiếng Anh

Bộ dữ liệu DUC2001 bao gồm 308 bài báo từ TREC-9. Các bài viết đến từ một số tờ báo và được chia thành 30 chủ đề. Các bài báo đã có sẵn các keyphrase làm nhãn để test. Có thể tải bộ dữ liệu DUC2001 tại đường dẫn : https://www-nlpir.nist.gov/projects/duc/guidelines/2001.html

Bộ dữ liệu SemEval2017 là nhiệm vụ thứ 10 trong cuộc thi SemEval 2017. Nó chứa 493 đoạn được chọn từ tạp chí ScienceDirect, bao gồm khoa học máy tính, khoa học vật liệu và vật lý. Mỗi tài liệu được chú thích với các cụm từ khóa bởi một đại học và một chuyên gia.Các chú thích này được sử dụng làm nhãn khi trong quá trình test. Có thể tải bộ dữ liệu SemEval2017 tại đường dẫn : https://scienceie.github.io/

Bộ dữ liệu Inspec bao gồm 2000 tài liệu ngắn được chọn từ các tóm tắt của tạp chí khoa học. Có 1000 tài liệu để đào tạo, 500 để validation và 500 để kiểm tra. Chọn phần kiểm tra để kiểm thử mô hình được đề xuất trong paper. Có thể tải bộ dữ liệu Inspec tại đường dẫn : https://github.com/boudinfl/hulth-2003-pre

3.2. Tiếng Việt

3.2.1 Crawl dữ liệu

Do không có bộ dữ liệu mẫu cho tiếng việt nên nhóm quyết định crawl những bài báo của trang dantri.com.vn để tự tạo tập corpus và bộ dữ liệu kiểm thử cho dữ liệu.

Hiện nay có một số công nghệ cho phép crawl dữ liệu dựa trên nền tảng Python như: scrapy, BeautifulSoup, selenium, splash ... Do một số lợi thế so với các công cụ khác của scrapy nên nhóm quyết định sử dụng scrapy để crawl nội dung các bài báo của trang báo dantri.com.vn.

Một số lý do để nhóm quyết định chọn scrapy là công cụ để crawl dữ liệu:

- Nó dễ dàng hơn để xây dựng và quy mô các dự án thu thập thông tin lớn.

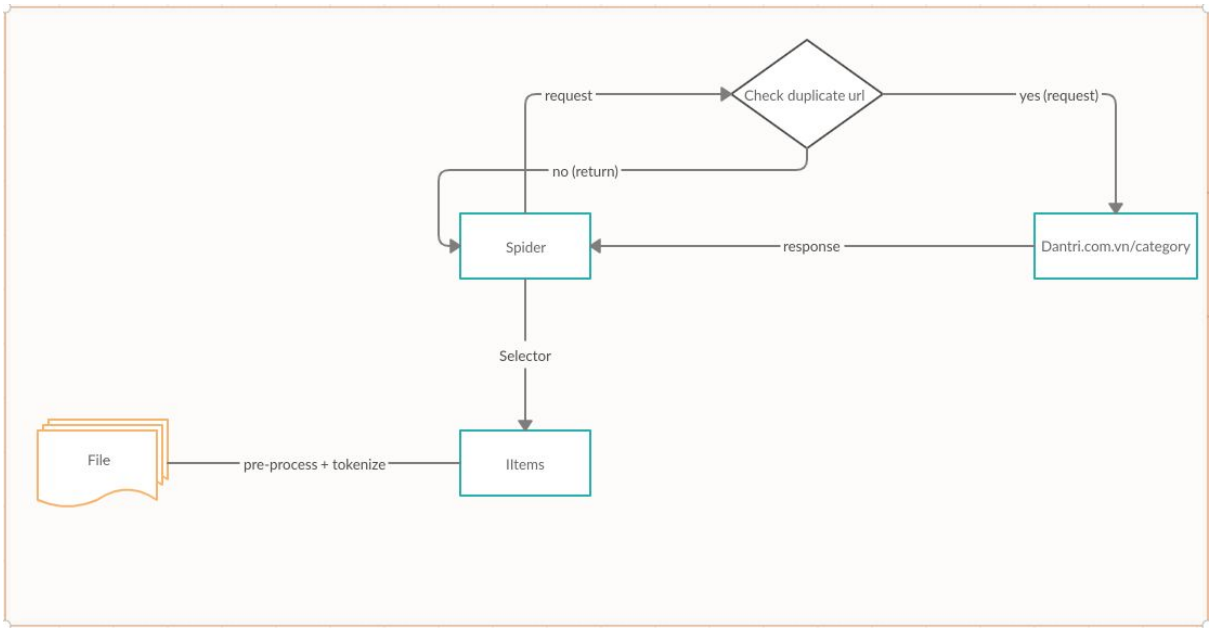
- Nó có một cơ chế tích hợp có tên là Selector, để trích xuất dữ liệu từ các trang web.
- Scrapy có hỗ trợ tích hợp để chọn và trích xuất dữ liệu từ các nguồn bằng các biểu thức XPath hoặc CSS.

Ưu điểm:

- Scrapy dễ dàng mở rộng, nhanh chóng và mạnh mẽ.
- Xử lý bất đồng bộ.
- Scrapy đi kèm với dịch vụ tích hợp có tên Scrapy cho phép tải lên các dự án và kiểm soát các spider bằng dịch vụ web JSON.

Nhược điểm:

- Chỉ dùng được trên python 2.7+
- Cài đặt khác nhau trên các nền tảng khác nhau



Các bước để crawl dữ liệu và xử lý dữ liệu

B1: Spider sẽ gửi yêu cầu download dantri.com.vn để download html của trang web thông qua một bộ kiểm tra trùng lặp url để tránh crawl 1 bài báo nhiều lần.

B2: Khi nhận được response từ dantri.com.vn spider sẻ dùng công cụ Selector để bóc tách những dữ liệu cần thiết.

B3: Dữ liệu sau khi bóc tách chỉ là dữ liệu thô cần xử lý để loại bỏ các ký tự không cần đến như khoảng trắng, dấu xuống dòng, ... và tokenize để có được bộ dữ liệu cần thiết.

3.2.2 Tạo bộ dữ liệu gán nhãn

Sử dụng 160 văn bản trong bộ dữ liệu đã crawl được để tiến hành gán nhãn các keyphrase.

Nhóm sử dụng thêm 2 phương pháp bóc tách keyphrase cho dữ liệu tiếng Việt là Yake và Tf-idf. Đối với Tf-idf, nhóm đơn thuần bóc tách các từ đơn có điểm tf*idf cao nhất. Nhóm tiến hành gán nhãn dữ liệu theo phương pháp sau:

- Đối với mỗi phương pháp, trích xuất 10 key phrase có điểm cao nhất. Ta có 30 ứng cử viên (candidate) là keyphrase đúng của văn bản
- Đọc văn bản và các ứng cử viên, chọn ra các keyphrase đúng trong tập ứng cử viên để làm nhãn đúng cho văn bản
- Tính precision, recall và f1 với dữ liệu đã có nhãn.

4. Mô hình SIFRank

4.1. Pattern NounPhrase

Pattern để trích xuất các cụm danh từ xuất hiện trong câu:

- NounPhrase = Adjective(s)(optional) + Noun
- NounPhrase = V-ing(s)(optional) + Noun
- NounPhrase = V-ed(s)(optional) + Noun
- NounPhrase = Noun(s)(optional) + Noun

=> NounPhrase = (JJ|V-ing|V-ed|NN)(s)(optional) + Noun

4.2. Luồng dữ liệu

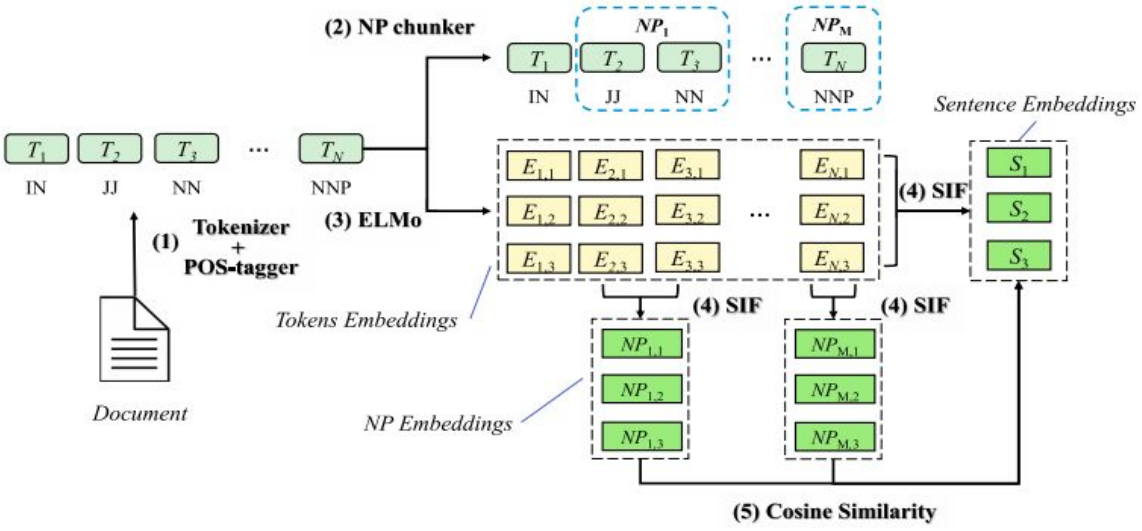


FIGURE 1. The framework of the SIFRank model.

- Bước 1 - Tokenizer + Pos-tagger : Document đi được đi qua 1 bộ tiền xử lý dữ liệu cho tiếng Anh (Stanford coreNLP) để tách từ và gắn các thẻ pos. Ta được các chuỗi tokens và token-tagged đại diện cho document. Mỗi document sẽ tương ứng với một hoặc nhiều chuỗi tokens.
- Bước 2 - NP Chunker : Trích xuất các cụm danh từ (NP) từ document theo các thẻ Pos bằng NP-chunker (Pattern được viết bởi biểu thức chính quy). Các cụm danh từ được trích xuất từ document là các candidate của keyphrase.
- Bước 3 - ELMo : Đưa các chuỗi tokens vào mô hình ngôn ngữ ELMo để trích xuất đặc trưng text (word embedding) cho từng word. Do output của ELMo gồm 3 layer embedding khác nhau, ta có các chiến lược khác nhau để trích xuất đặc trưng cho từng word. Trong thực nghiệm, nhóm dùng layer L1 để làm word embedding.
- Bước 4 - SIF : Các word embedding được đi qua mô hình sentence embedding SIF để tính toán embedding cho cả document và từng cụm danh từ. Các embedding cho document và các cụm danh từ sẽ có cùng số chiều.
- Bước 5 - Cosine Similarity : Tính độ tương đồng cosin giữa embedding của document và các cụm danh từ. Những cụm danh từ có độ tương đồng cosine lớn nhất là những cụm danh từ mang nhiều ý nghĩa của document nhất. Chọn Top-N cụm danh từ có độ tương đồng cao nhất để làm keyphrase của document.

4.3. SIFRank+

SIFRank+ là 1 phần mở rộng cho SIFRank, dùng để trích xuất các cụm từ quan trọng trong các văn bản dài, hỗ trợ chạy song song ELMo và sử dụng các bộ corpus cho từng domain, ngoài bộ common corpus để cải thiện độ chính xác cho các văn bản.

4.3.1. Điều chỉnh domain

Đối với những văn bản trong các domain khác nhau, phân phối xác suất của các từ có thể khác nhau. Trong một số miền cụ thể, các từ hiếm có lại có thể xuất hiện rất phổ biến. Để mô hình hoạt động tốt nhiệm vụ của các miền khác nhau, trọng số SIF cần được hiệu chỉnh. Trọng số mới sẽ là tổng trọng số của tập common corpus và domain corpus.

$$\begin{aligned}\text{Weight}(w) &= \lambda \text{Weight}_{com}(w) + (1-\lambda)\text{Weight}_{dom}(w) \\ &= \lambda \frac{a}{a+f_w} + (1-\lambda)\frac{a'}{a'+f'_w}\end{aligned}$$

Trong đó, Weight_com(w) là trọng số của w trong bộ common corpus và được SIF truyền thống sử dụng, Weight_dom(w) là trọng số của w trong bộ domain corpus, là trọng số hiệu chỉnh.

Những từ không có trong corpus, frequency của từ đó được đặt mặc định là 1.

4.3.2. Phân đoạn tài liệu và hiệu chỉnh embedding

Đối với những văn bản dài, thời gian dùng ELMo để tính word embedding là rất lớn, làm giảm hiệu suất của chương trình. Do đó, khi có input là văn bản dài, SIFRank+ sẽ tự động chia văn bản thành nhiều đoạn nhỏ và thực hiện tính embedding song song.

Tuy nhiên khi tính toán song song các word embedding, word embedding của 1 từ sẽ khác nhau ở mỗi đoạn. Do đó cần hiệu chỉnh embedding về 1 giá trị duy nhất. SIFRank+ lấy giá trị trung bình của tất cả các embedding của 1 từ là embedding đại diện cho từ đó.

4.3.3. Trọng số thiên vị vị trí với document dài

Đối với hầu hết các tài liệu dài, tác giả có xu hướng viết chủ đề chính của tài liệu, điều đó có nghĩa là các cụm từ khóa quan trọng nhất thường xảy ra ở phần đầu của tài liệu.Vì SIFRank là một kiểu mô hình bag-of-word, nên cần xem xét thông tin về vị trí của các candidate trong trích xuất keyphrase cho tài liệu dài (đặc biệt là tài liệu có nhiều đoạn).

SIFRank+ sử dụng vị trí từ xuất hiện đầu tiên đề làm trọng số thiên vị (bias) cho 1 candidate, những từ xuất hiện càng sớm thì sẽ có trọng số càng lớn. Trọng số thiên vị là nghịch đảo của vị trí đầu tiên candidate xuất hiện, được tính theo công thức :

$$p(NP_i) = \frac{1}{p_1 + \mu}$$

Với p1 là vị trí đầu tiên candidate NPi xuất hiện, μ là siêu tham số để tối ưu trọng số thiên vị.

Để tiếp tục thu hẹp khoảng cách trọng số thiên vị vị trí của các cụm từ khóa candidate,, hàm softmax được sử dụng để chuẩn hóa :

$$\tilde{p}(NP_i) = \text{soft}\max(p(NP_i)) = \frac{\exp(p(NP_i))}{\sum_{k=1}^{N} \exp(p(NP_k))}$$

Cuối cùng, độ tương đồng cosine giữa 1 candidate NPi và document d được tính theo công thức :

$$\text{SIFRank+}(NP_i, d) = \tilde{p}(NP_i) \cdot \text{Sim}(v_{NP_i}, v_d)$$

5. SIFRank4VN

Sau khi tìm hiểu về cách hoạt động, các module cần thiết của SIFRank và test theo paper, nhóm quyết định thay đổi một số module để có thể chạy SIFRank cho tiếng Việt.

5.1. VncoreNLP, PhoBERT

Đối với task tách từ và gán nhãn thẻ pos, nhóm sử dụng VncoreNLP, là thư viện xử lý ngôn ngữ tự nhiên đạt được kết quả tốt nhất trên bộ dữ liệu chuẩn. Dữ liệu thô sẽ được tiền xử lý, loại bỏ stopword tiếng Việt, tách từ và gán nhãn thẻ Pos. Dựa vào thẻ pos, xây dựng các pattern để trích xuất cụm danh từ tiếng Việt.

Văn bản sau token được đi qua phoBERT, pretrain model language cho tiếng Việt để lấy embedding cho word (phoBERT thay thể cho ELMo trong bản tiếng Anh).

Corpus cho SIF gồm 2 bộ. Common corpus gồm 1G data được crawl trên wiki, tiền xử lý và tính term frequency, Dantri corpus gồm 18000 bài báo crawl trên trang dân trí, tiền xử lý và tính term frequency.

5.2. Pattern cho cụm danh từ tiếng Việt

Cụm danh từ là một nhóm các danh từ đi chung với nhau để làm thành một danh từ chung. Cụm danh từ có thể bao gồm từ hai đến vài danh từ. Khi mỗi danh từ đứng riêng thì mang một ý nghĩa đặc trưng nhưng khi chúng được kết hợp với nhau sẽ mang một ý nghĩa khác tuy nhiên ý nghĩa đặc trưng kia vẫn tồn tại ở một khía cạnh đủ để làm nên ý nghĩa cho một danh từ mới.

Cụ thể cụm danh từ gồm ba phần, được kết hợp ổn định với nhau theo thứ tự:

Cụm danh từ = phần phụ trước + danh từ trung tâm + phần phụ sau

- Phần phụ trước có cấu trúc tối đa gồm 3 định tố: D1 + D2 + D3:
    - D3 là định tố đứng ngay trước danh từ trung tâm có thể là danh từ loại thể hoặc danh từ chỉ đơn vị đo lường.
    - D2 là định tố có chức năng biểu thị ý nghĩa số lượng có thể là danh từ chỉ số lượng hoặc số từ chỉ số lượng
    - D1 là các đại từ chỉ tổng lượng

- Phần phụ sau có cấu trúc tối đa gồm 3 định tố: D4 + D5 + D6:
    - D4 là định tố đứng ngay sau danh từ trung tâm biểu thị ý nghĩa hạn định có thể là tính từ, danh từ, giới từ hoặc động từ.
    - D5 cũng là định tố đứng ngay sau danh từ trung tâm biểu thị ý nghĩa hạn định
    - D6 là là định tố biểu thị sự chỉ định về không/thời gian đối với danh từ trung tâm, do vậy ở vị trí này luôn là các đại từ chỉ định: này, kia, ấy, nọ, đó

6. Kiểm thử

6.1 SIFRank

Nhóm đã thực hiện kiểm thử SIFRank theo các phương pháp được nêu trong paper. Gồm có so sánh với các baseline model khác và so sánh các word embedding model khác nhau. Kết quả thống kê được thể hiện trong 2 bảng dưới.

	COMPARE WITH OTHER BASELINES								
	Inspec			Semeval2017			Duc2001		
	P	R	F1	P	R	F1	P	R	F1
Yake									
5	20.52	10.44	13.84	23.12	6.68	10.37	11.79	7.30	09.01
10	16.56	16.86	16.71	21.50	12.43	15.75	11.59	14.35	12.83
15	14.77	22.56	17.85	20.34	17.63	18.89	11.55	21.44	15.02
TextRank									
5	34.92	17.77	23.56	38.09	11.01	17.08	25.02	15.48	19.12
10	27.42	27.69	27.55	29.31	16.94	21.47	17.99	22.24	19.89
15	22	32.35	26.19	25.12	21.76	23.32	14.23	26.32	18.42
TopicRank									
5	26,28	13,37	17,72	31,72	9,16	14,22	23,19	14,34	17,72
10	21,76	21,74	21,75	25,78	14,9	18,88	17,88	22,04	19,75
15	19,16	27,3	22,52	22,38	19,31	20,73	14,68	27,04	19,03
SingleRank									
5	29,28	14,9	19,75	30,3	8,75	13,58	19,8	12,25	15,13
10	26,14	26,47	26,3	29,49	17,04	21,6	17,44	21,56	19,28
15	23,25	34,4	27,75	27,21	23,57	25,26	15,79	29,14	20,48
SIFrank									
5	**43.28**	**22.03**	**29.2**	**48.64**	**14.06**	**21.81**	**31.79**	**16.67**	**24.3**
10	**38.94**	**39.15**	**39.04**	**43.45**	**25.11**	**31.83**	**24.96**	**30.83**	**27.6**
15	**35.58**	**48.88**	**39.81**	**39.14**	**33.88**	**36.32**	**21.56**	**39.74**	**27.96**

Compare with other baselines

	COMPARE WITH OTHER PRE-TRAINED LANGUAGE MODELS AND ELMO'S DIFFERENT LAYERS								
	Inspec			SemEval2017			DUC2001		
	P	R	F1	P	R	F1	P	R	F1
Glove 100d	31,84	16,2	21,47	33,75	9,75	15,13	19,73	12,21	15,08
Glove 300d	35,8	18,22	24,15	38,05	10,99	17,06	23,51	14,55	17,97
Glove 200d	33,92	17,26	22,88	35,09	10,14	15,73	20,32	12,57	15,53
Bert 768	40,64	20,68	27,41	43,56	12,59	19,53	22,6	13,98	17,28
Bert 1024	40,28	20,5	27,17	42,67	12,33	19,13	22,73	14,06	17,38
ELMo-lstm-L0	41.04	20.89	27.68	**48.64**	**14.06**	**21.81**	**31.79**	**19.67**	**24.30**
ELMo-lstm-L1	**43.08**	**21.93**	**29.06**	45.48	13.14	20.39	20.59	12.74	15.74
ELMo-lstm-L2	42.20	21.48	28.47	43.34	12.81	19.88	16.87	10.44	12.90
ELMo-lstm-L0L1L2	42.76	21.76	28.85	48.03	13.88	21.54	27.43	16.97	20.97

Compare with other pre-trained language models and ELMo's different layers

So sánh kết quả test của nhóm và kết quả test trong paper, nhóm nhận thấy 2 kết quả gần như tương tự nhau. Một số trường hợp có kết quả lệch nhau như so sánh với embedding model BERT, sự khác biệt có thể là do nhóm không sử dụng cùng pretrain model với paper (paper sử dụng BERT_as_service còn nhóm sử dụng BERT_gluon-nlp)

6.2 SIFRank4VN

Nhóm thực hiện kiểm thử SIFRank4VN với 2 baseline là Tf-idf và Yake, mỗi phương pháp sẽ trích xuất ra 10 keyphrase có điểm số cao nhất. Kết quả được thống kê trong bảng dưới.

COMPARE WITH OTHER BASELINES			
	P	R	F
Tf-idf	29.74	22.03	25.31
Yake	42.52	34.01	37.79
SIFRank	**42.86**	**35.96**	**39.11**

Compare with other baselines

Kết quả cho thấy hiệu quả của SIFRank vẫn cao nhất khi sử dụng cho tiếng Việt. Tuy nhiên có thể thấy các chỉ số precision, recall và f1 không khác biệt nhiều lắm với Yake, điều này có thể do tính không khách quan khi gán nhãn cho bộ dữ liệu tiếng Việt hoặc nhóm chưa có nhiều thời gian tuning các hyperparameters cho SIFRank.

7. Tài liệu tham khảo

SIFRank:

+ Paper: https://ieeexplore.ieee.org/document/8954611

+ Github: https://github.com/sunyilgdx/SIFRank

VnCoreNLP:

+ paper: https://arxiv.org/pdf/1801.01331.pdf

+ git: https://github.com/vncorenlp/VnCoreNLP

phoBERT:

+ paper: https://arxiv.org/pdf/2003.00744.pdf

+ git: https://github.com/VinAIResearch/PhoBERT

ELMo:

+ paper: https://arxiv.org/pdf/1802.05365.pdf

Cụm danh từ tiếng việt: https://www.35express.org/danh-tu-la-gi-cum-danh-tu-la-gi/

Stanford CoreNLP: https://stanfordnlp.github.io/CoreNLP/