ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ

HỌC PHẦN: HỌC MÁY
ĐỀ TÀI
TÓM TẮT VĂN BẢN TIẾNG VIỆT

Sinh viên thực hiện: **Đỗ Minh Tuấn - 20002175**
Lã Anh Trúc - 20002169
Phạm Hoàng An - 20002102
Lê Hồng Thạch - 20002162
Trần Kim Phượng - 20002154

Ngành kỹ thuật điện tử và tin học
(Chương trình đào tạo chuẩn)

Giảng viên hướng dẫn: **TS. Phạm Tiến Lâm**
Ths. Đặng Văn Báu

Hà Nội - 2023

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ

HỌC PHẦN: HỌC MÁY
ĐỀ TÀI
TÓM TẮT VĂN BẢN TIẾNG VIỆT

Sinh viên thực hiện: **Đỗ Minh Tuấn - 20002175**
Lã Anh Trúc - 20002169
Phạm Hoàng An - 20002102
Lê Hồng Thạch - 20002162
Trần Kim Phượng - 20002154

Ngành kỹ thuật điện tử và tin học
(Chương trình đào tạo chuẩn)

Giảng viên hướng dẫn: **TS. Phạm Tiến Lâm**
Ths. Đặng Văn Báu

Hà Nội - 2023

Lời cảm ơn

Trong quá trình nghiên cứu môn Học máy, nhóm chúng em đã nhận được sự giúp đỡ nhiệt tình của Thầy Phạm Tiến Lâm cùng Thầy Đặng Văn Báu. Nhóm chúng em rất trân trọng sự tận tâm và sự giúp đỡ mà các Thầy đã dành cho chúng em. Những kiến thức cơ bản và góp ý từ các Thầy đã giúp chúng em tiếp cận và nắm bắt những khái niệm quan trọng, đồng thời phát triển kỹ năng nghiên cứu. Thầy đã truyền đạt không chỉ kiến thức mà còn cung cấp những hướng dẫn quý giá, tạo điều kiện cho chúng em tự tin và tiến bộ trong quá trình thực hiện dự án này. Chúng em rất trân trọng khoảng thời gian cùng các Thầy học tập và nghiên cứu môn học .

Do những hạn chế về mặt kiến thức cũng như thời gian thực hiện, dự án này của chúng em cũng không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự đóng góp , phê bình của các Thầy để đề tài của chúng em được hoàn thiện hơn. Cuối cùng chúng em xin kính chúc các Thầy nhiều sức khỏe , thành công và hạnh phúc trong công việc và cuộc sống.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

MỞ ĐẦU.....1
CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI.....2
1.1. Lý do chọn đề tài.....2
1.2. Ý tưởng bài toán:.....2
CHƯƠNG II: GIẢI QUYẾT VẤN ĐỀ.....3
2.1. Cách tiếp cận bài toán:.....3
2.2. Các phương pháp và kỹ thuật sử dụng trong bài.....4
2.2.1. Phương pháp trích xuất (Extraction-based):.....4
2.2.2. Phương pháp trừu tượng(Abstraction-based):.....4
2.2.3. Phân cụm (Kmean).....5
2.2.4. Tokenize.....5
2.3.5. Word2Vec:.....9
2.2.6. Mô hình Long Short Term Memory (LSTM):.....11
2.2.7. Mô hình Auto Encoder:.....11
2.3. Thực hiện:.....14
2.3.1. Phương pháp trích dẫn.....14
2.3.2. Phương Pháp trừu tượng.....17
2.3.3. Tóm tắt văn bản bằng phân cụm.....23
2.4. Giao diện ứng dụng web.....26
2.4.1. Các công nghệ sử dụng.....26
2.4.2. Phương thức triển khai:.....26
2.4.3. Phương thức hoạt động.....27
2.4.4. Chi tiết hoạt động.....27
CHƯƠNG III: KẾT QUẢ ĐẠT ĐƯỢC.....30
3.1. Kết quả.....30
3.2. Các hạn chế và nhược điểm.....30
KẾT LUẬN.....31
TÀI LIỆU THAM KHẢO.....32

MỞ ĐẦU

Tất cả chúng ta đều biết rằng việc đọc và hiểu văn bản là một kỹ năng quan trọng trong cuộc sống hàng ngày của chúng ta. Tuy nhiên, với lượng thông tin lớn và ngày càng tăng trên internet và các nguồn tin tức khác, việc đọc và tiêu hóa các thông tin này trở nên khó khăn hơn bao giờ hết. Trong khi đó, việc tóm tắt văn bản là một cách hiệu quả để giải quyết vấn đề này. Tuy nhiên, việc tóm tắt văn bản thủ công là một công việc tốn thời gian và không thể áp dụng cho các nhu cầu tóm tắt lớn.

Trong bài tập lớn này, chúng em sẽ tập trung vào việc xây dựng một mô hình học máy để tự động tóm tắt văn bản tiếng Việt. Mục tiêu của bài tập lớn này là giúp hiểu rõ hơn về các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) và mô hình học máy, và áp dụng chúng để giải quyết một vấn đề thực tế. Chúng em đã tiến hành tiền xử lý dữ liệu, rút trích đặc trưng và xây dựng mô hình học máy để tự động tóm tắt các văn bản tiếng Việt. Sau khi xây dựng được mô hình, chúng em sẽ đánh giá và so sánh kết quả của mô hình với các phương pháp tóm tắt văn bản khác, và đưa ra nhận xét và đề xuất cải tiến cho mô hình của chúng ta.

Bài tập lớn này đã giúp chúng em cải thiện kỹ năng lập trình, phân tích dữ liệu và giải quyết vấn đề,đồng thời cũng giúp chúng em hiểu rõ hơn về cách hoạt động của các kỹ thuật NLP và mô hình học máy trong việc giải quyết các vấn đề thực tế. Ngoài ra, chúng ta cũng sẽ rèn luyện được kỹ năng làm việc độc lập và làm việc nhóm thông qua việc thực hiện bài tập lớn này.

Chúng em thấy rằng bài tập lớn này đã mang lại cho chúng em nhiều kiến thức bổ ích và kinh nghiệm quý báu để áp dụng vào các dự án và công việc trong tương lai.

Nội dung bài báo cáo này được chia làm 3 chương:

Chương 1: Giới thiệu đề tài

Chương 2: Phương pháp thực hiện

Chương 3: Kết quả đạt được

CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Với việc dữ liệu ngày càng nhiều và có độ phức tạp cũng như tốc độ cần tiếp thu càng ngày càng nhanh hơn nữa, vì vậy nhu cầu cần tóm tắt văn bản tăng cao. Đề tài "Tóm tắt văn bản tiếng Việt" là giải pháp hữu ích với nhu cầu tóm tắt văn bản tiếng Việt đang tăng cao trong cuộc sống hàng ngày, đặc biệt là trong lĩnh vực tin tức, nghiên cứu khoa học, và kinh doanh. Việc ứng dụng học máy để tóm tắt văn bản giúp dễ dàng tiếp cận thông tin và tiết kiệm thời gian đọc hiểu, tuy nhiên việc tóm tắt văn bản thủ công là rất tốn thời gian và công sức.

Do đó, việc ứng dụng học máy để tóm tắt văn bản tiếng Việt là một giải pháp tiện lợi và hiệu quả để giải quyết vấn đề này. Ngoài ra, việc thực hiện đề tài này còn giúp tôi rèn luyện và cải thiện kỹ năng lập trình, phân tích dữ liệu, và áp dụng kiến thức của mình vào các dự án và công việc trong tương lai.

Đặc biệt, đề tài này rất thú vị và có tính ứng dụng cao trong thực tế. Việc sử dụng các kỹ thuật NLP và tóm tắt văn bản tiếng Việt là một lĩnh vực đang được quan tâm và nghiên cứu rộng rãi trong cộng đồng khoa học và công nghệ. Tôi tin rằng việc thực hiện đề tài này sẽ giúp tôi hiểu rõ hơn về các kỹ thuật NLP và mô hình học máy, và áp dụng chúng để giải quyết một vấn đề thực tế. Ngoài ra, đề tài này còn đòi hỏi sự tìm hiểu và nghiên cứu thêm về ngôn ngữ và văn hóa tiếng Việt, giúp tôi có cơ hội tiếp cận với các tài liệu và nguồn thông tin mới.

Cuối cùng, tóm tắt văn bản tiếng Việt là một thử thách thú vị và có ý nghĩa trong việc giải quyết các vấn đề thực tế. Tôi hy vọng rằng đề tài này sẽ giúp tôi phát triển kỹ năng và kiến thức của mình, đồng thời đóng góp vào sự phát triển của lĩnh vực NLP và học máy ở Việt Nam.

1.2. Ý tưởng bài toán:

Với đầu vào là một văn bản gồm N câu, mô hình sẽ loại bỏ đi những câu có ý nghĩa tương tự nhau hay ko phải ý chính để tạo ra một văn bản tóm tắt gồm K câu và giữ lại được ý chính của đoạn.

CHƯƠNG II: GIẢI QUYẾT VẤN ĐỀ

2.1. Cách tiếp cận bài toán:

Có hai phương pháp chính thường được sử dụng để tóm tắt văn bản trong Xử Lý Ngôn Ngữ Tự Nhiên (NLP): Phương pháp Extraction-based (phương pháp dựa trên trích xuất) và phương pháp Abstraction-based (phương pháp dựa trên trừu tượng).

+ Phương pháp **Extraction-based:** là một phương pháp trong xử lý ngôn ngữ tự nhiên, trong đó các thông tin quan trọng được trích xuất ra từ văn bản ban đầu để tạo thành một tài liệu mới, thường là một tài liệu tóm tắt hoặc một bản tin. Ý tưởng chính của phương thức này thường sử dụng các thuật toán và kỹ thuật như sắp xếp theo mức độ quan trọng, đếm từ khóa, tính toán tần suất xuất hiện và phân tích ngữ cảnh để xác định những phần quan trọng trong văn bản gốc. Tóm tắt được tạo ra từ việc ghép các đoạn văn hoặc từ ngữ quan trọng này lại thành một tài liệu rút gọn.

Văn bản đầu vào → độ tương tự của các câu → đánh giá mức độ quan trọng của các câu → lựa chọn các câu với level đánh giá cao.

+ Phương pháp **Abstraction-based:** là một phương pháp trong xử lý ngôn ngữ tự nhiên, trong đó các thông tin quan trọng được tóm tắt với phương pháp này sử dụng các quy tắc ngữ nghĩa, cú pháp và ngữ cảnh để tạo ra những câu mới có khả năng diễn đạt ý nghĩa của văn bản gốc một cách trừu tượng. Các phương pháp này thường sử dụng các kỹ thuật như phân tích ngữ nghĩa, xử lý ngôn ngữ tự nhiên và học máy để tạo ra tóm tắt trừu tượng. Phương pháp này ưu việt hơn so với phương pháp extractive. Tuy nhiên, các thuật toán tóm tắt văn bản thực hiện theo phương pháp abstractive khó phát triển hơn và đó là lý do tại sao việc sử dụng phương pháp extractive vẫn còn phổ biến.

Văn bản đầu vào → hiểu nội dung → phân tích về mặt ngữ nghĩa → tạo văn bản tóm tắt.

2.2. Các phương pháp và kỹ thuật sử dụng trong bài

2.2.1. Phương pháp trích xuất (Extraction-based):

Đây là quy trình thực hiện của phương pháp trích xuất:

Bước 1: Chuyển đoạn văn thành các câu: chia đoạn văn thành các câu tương ứng. Trong những cách thực hiện, tách câu khi dấu chấm xuất hiện là cách tốt nhất.

Bước 2: Xử lý văn bản: Bằng cách loại bỏ các stopwords (những từ mang ít ý nghĩa cho câu), số, dấu chấm câu và các ký tự đặc biệt khác khỏi câu. Loại bỏ các thông tin dư thừa và không quan trọng có thể không cung cấp bất kỳ giá trị gia tăng nào cho ý nghĩa của văn bản.

Bước 3: Tokenization: tách các từ và mã hóa chúng thành các đơn vị nhỏ gọi là Tokens.

Bước 4: Đánh giá tần suất xuất hiện có trọng số của các từ: Chia tần suất xuất hiện của mỗi từ cho tần suất xuất hiện của từ xuất hiện nhiều nhất trong đoạn văn.

Bước 5: Thay thế các từ bằng tần số có trọng số của chúng: Sau khi thay thế từng từ tìm thấy trong câu gốc bằng tần suất có trọng số của chúng, chúng ta tính tổng của chúng. Từ tổng đó, chúng ta suy ra được thứ tự ưu tiên và thứ tự sắp xếp dựa trên tổng tần số có trọng số của mỗi câu. Câu có tổng cao hơn sẽ được ưu tiên trước.

Bước 6: Nhận văn bản tóm tắt: Từ đó ta có được văn bản tóm tắt dựa trên độ ưu tiên của các câu

2.2.2. Phương pháp trừu tượng(Abstraction-based):

Sử dụng thư viện Keras của python cùng với các thư viện khác để thiết kế và xử lý dữ liệu. Sau đó, chúng ta sử dụng ánh xạ rút gọn để xử lý các từ rút gọn của ngôn ngữ.

Tiếp đến là tiền xử lý dữ liệu, chúng ta làm sạch dữ liệu cùng với đó là loại bỏ đi các stopword để làm dữ liệu sẵn sàng cho mô hình. Giai đoạn này được thực hiện riêng cho mô hình tóm tắt và mô hình hoàn chỉnh của mô hình train.

Xây dựng mô hình bằng cách triển khai 3 lớp LSTM được thêm vào.

Tiếp đến là một lớp Attention và một lớp Dense được đặt trước giai đoạn biên dịch.

Cuối cùng sử dụng dữ liệu được tạo và thêm chúng một cách tuần tự bằng cách sử dụng phương thức decode_seq và seq2seq. Tiếp đến là phương thức seq2text để thêm văn bản vào chuỗi. Kết thúc chúng ta sẽ có được văn bản tóm tắt cần thiết.

2.2.3. Phân cụm (Kmean)

Bài toán phân cụm là 1 nhánh ứng dụng chính của lĩnh vực Unsupervised Learning (Học không giám sát), trong đó dữ liệu được mô tả trong bài toán không được dán nhãn (tức là không có đầu ra). Quá trình phân cụm bắt đầu bằng việc xây dựng một không gian đặc trưng (feature space) từ các văn bản. Các đặc trưng này thường là các thuộc tính của văn bản như từ vựng, tần suất xuất hiện của các từ trong văn bản, hoặc các đặc trưng ngữ nghĩa khác.

Sau đó, các thuật toán phân cụm được áp dụng để gom nhóm các văn bản dựa trên đặc trưng của chúng. Các phương pháp phân cụm thông thường bao gồm K-means, phân cụm hiệu suất cao (hierarchical clustering)

Kết quả của Thuật toán sẽ tìm cách phân cụm - chia dữ liệu thành từng nhóm có đặc điểm tương tự nhau, nhưng đồng thời đặc tính giữa các nhóm đó lại phải càng khác biệt càng tốt.

Và ở đây, chúng ta muốn phân cụm các vector đại diện cho từng câu trong văn bản vừa rồi để biết những câu nào mang ý nghĩa giống nhau.

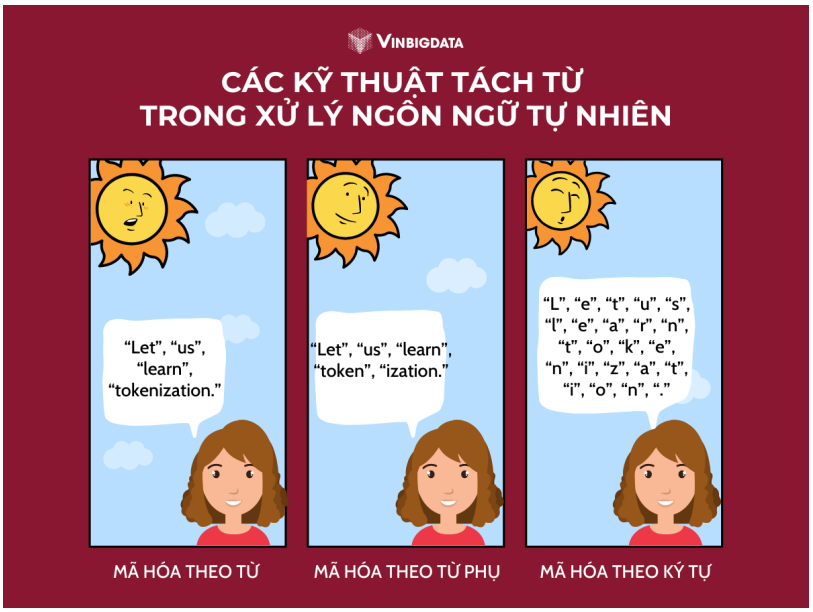
2.2.4. Tokenize.

Tokenization (tách từ) là một trong những bước quan trọng nhất trong quá trình tiền xử lý văn bản, đó là quá trình tách một cụm từ, câu, đoạn văn, một hoặc nhiều tài liệu văn bản thành các đơn vị nhỏ hơn. Mỗi đơn vị nhỏ hơn này được gọi là Tokens.

Có thể coi tokens là các khối xây dựng của NLP và tất cả các mô hình NLP đều xử lý văn bản thô ở cấp độ các Tokens. Chúng được sử dụng để tạo từ vựng trong một kho ngữ liệu (một tập dữ liệu trong NLP). Từ vựng này sau đó được chuyển thành số (ID) và giúp chúng ta lập mô hình.

Phân loại

Tokens có thể là bất cứ thứ gì – một từ (word), một từ phụ (sub-word) hoặc thậm chí là một ký tự (character).



- **Thuật toán mã hóa dựa trên từ** (word-based tokenization algorithm):

Đây là kĩ thuật tokenization được sử dụng phổ biến trong phân tích văn bản. Nó chia một đoạn văn bản thành các từ (ví dụ tiếng Anh) hoặc âm tiết (ví dụ tiếng Việt) dựa trên dấu phân cách.

Tách từ có thể được thực hiện dễ dàng bằng cách sử dụng phương thức split () của RegEx hoặc Python. Ngoài ra, có rất nhiều thư viện Python – NLTK, spaCy, Keras, Gensim, có thể giúp bạn thực hiện việc này một cách thuận tiện.

Thực tế, các mô hình NLP sử dụng các phương pháp tách từ phù hợp theo từng ngôn ngữ. Tuỳ thuộc vào từng bài toán, mà cùng một văn bản có thể được xử lý dưới các loại tokens khác nhau. Mỗi token thường có tính duy nhất và được biểu diễn bằng một ID, các ID này là một cách mã hoá hay cách định danh token trên không gian số.

Hạn chế của kỹ thuật này là nó dẫn đến một kho ngữ liệu khổng lồ và một lượng từ vựng lớn, khiến mô hình cồng kềnh hơn và đòi hỏi nhiều tài nguyên tính toán hơn. Bên cạnh đó, một hạn chế nữa là liên quan đến các từ sai chính tả.

Nếu kho ngữ liệu có từ "knowledge" viết sai chính tả thành "knowldge", mô hình sẽ gán token không thuộc từ vựng cho từ sau đó. Do đó, để giải quyết tất cả những vấn đề này, các nhà nghiên cứu đã đưa ra kỹ thuật mã hóa dựa trên ký tự.

Ví dụ: Câu gốc là "Let us learn tokenization."

- **Thuật toán mã hóa dựa trên từ** sẽ chia câu thành các từ: ["Let", "us", "learn", "tokenization."]
- **Thuật toán mã hóa dựa trên từ phụ** (subword-based tokenization algorithm):

Một kỹ thuật phổ biến khác là mã hóa dựa trên từ khóa phụ. Đây là một giải pháp nằm giữa mã hóa dựa trên từ và ký tự. Ý tưởng chính là giải quyết đồng thời các vấn đề của mã hóa dựa trên từ (kích thước từ vựng rất lớn, có nhiều tokens không thuộc từ vựng, sự khác biệt trong ý nghĩa của các từ rất giống nhau) và mã hóa dựa trên ký tự (chuỗi rất dài và token riêng lẻ ít ý nghĩa hơn).

Các thuật toán mã hóa dựa trên từ khóa phụ sử dụng các nguyên tắc sau:

- ❖ Không chia các từ thường dùng thành các từ phụ nhỏ hơn.
- ❖ Chia các từ hiếm thành các từ phụ có ý nghĩa.

Mã hóa dựa trên từ khóa phụ cho phép mô hình có kích thước từ vựng phù hợp và cũng có thể học các biểu diễn độc lập theo ngữ cảnh có ý nghĩa. Mô hình thậm chí có thể xử lý một từ mà nó chưa từng thấy trước đây vì sự phân tách có thể dẫn đến các từ phụ đã biết.

Như vậy, trên đây là cách các phương pháp mã hóa phát triển theo thời gian để đáp ứng nhu cầu ngày càng tăng của NLP và đưa ra các giải pháp tốt hơn cho các vấn đề.

- **Mã hóa dựa trên ký tự** chia văn bản thô thành các ký tự riêng lẻ. Logic đằng sau mã hóa này là một ngôn ngữ có nhiều từ khác nhau nhưng có một số ký tự cố định. Điều này dẫn đến một lượng từ vựng rất nhỏ. Ví dụ tiếng Anh có 256 ký tự khác nhau (chữ cái, số, ký tự đặc biệt) trong khi chứa gần 170.000 từ

trong vốn từ vựng. Do đó, mã hóa dựa trên ký tự sẽ sử dụng ít token hơn so với mã hóa dựa trên từ.

Một trong những lợi thế chính của mã hóa dựa trên ký tự là sẽ không có hoặc rất ít từ không xác định hoặc không thuộc từ vựng. Do đó, nó có thể biểu diễn các từ chưa biết (những từ không được nhìn thấy trong quá trình huấn luyện) bằng cách biểu diễn cho mỗi ký tự. Một ưu điểm khác là các từ sai chính tả có thể được viết đúng chính tả lại, thay vì có thể đánh dấu chúng là mã thông báo không thuộc từ vựng và làm mất thông tin.

Ví dụ: Câu gốc là “Let us learn tokenization.”

Thuật toán mã hóa dựa trên từ phụ sẽ chia câu thành các từ khóa phụ: [“Let”, “us”, “learn”, “token”, “ization.”]

2.3.5. Word2Vec:

Word2vec là một mô hình đơn giản và nổi tiếng giúp tạo ra các biểu diễn embedding của từ trong một không gian có số chiều thấp hơn nhiều lần so với số từ trong từ điển. Ý tưởng của word2vec đã được sử dụng trong nhiều bài toán với dữ liệu khác xa với dữ liệu ngôn ngữ.

Ý tưởng cơ bản của word2vec có thể được gói gọn trong các ý sau:

- Hai từ xuất hiện trong những văn cảnh giống nhau thường có ý nghĩa gần với nhau.
- Ta có thể đoán được một từ nếu biết các từ xung quanh nó trong câu. Ví dụ, với câu "Hà Nội là … của Việt Nam" thì từ trong dấu ba chấm khả năng cao là "thủ đô". Với câu hoàn chỉnh "Hà Nội là thủ đô của Việt Nam", mô hình word2vec sẽ xây dựng ra embeding của các từ sao cho xác suất để từ trong dấu ba chấm là "thủ đô" là cao nhất.

Hai cách khác nhau xây dựng mô hình word2vec:

- Skip-gram: Dự đoán những từ ngữ cảnh nếu biết trước từ đích.
- CBOW (Continuous Bag of Words): Dựa vào những từ ngữ cảnh để dự đoán từ đích.

Mỗi cách có những ưu nhược điểm khác nhau và áp dụng với những loại dữ liệu khác nhau.

Skip-gram: được sử dụng để học các biểu diễn từ vựng dựa trên ngữ cảnh của các từ đó. Ý tưởng cơ bản của Skip-gram là học các biểu diễn từ vựng bằng cách dự đoán các từ xung quanh từ đang xét.

Trong Skip-gram, một từ được biểu diễn dưới dạng một vector số thực có độ dài cố định. Các vector này được học bằng cách đưa vào mô hình một từ và yêu cầu mô hình dự đoán các từ trong văn bản xung quanh từ đó. Quá trình học các vector này sẽ được thực hiện thông qua việc tối ưu hàm mất mát giữa các từ dự đoán và các từ thực tế trong ngữ liệu huấn luyện.

Một trong những ứng dụng phổ biến của Skip-gram là trong bài toán xây dựng mô hình dự đoán từ khóa (keyword prediction) cho các bài viết trên mạng. Skip-gram có thể học được các biểu diễn từ vựng tốt và giúp mô hình dự đoán chính xác các từ khóa liên quan đến nội dung của bài viết.

Để xây dựng một mô hình Skip-gram, bạn có thể sử dụng các thư viện học máy như TensorFlow hoặc PyTorch. Các thư viện này cung cấp các lớp và phương thức để xây dựng mô hình Skip-gram và huấn luyện nó trên các tập dữ liệu văn bản.

Continuous Bag of Words (CBOW): được sử dụng để học các biểu diễn từ vựng dựa trên ngữ cảnh của các từ đó, tuy nhiên, CBOW hoạt động theo cách ngược lại so với Skip-gram.

Trong CBOW, mô hình được huấn luyện để dự đoán từ đang xét dựa trên các từ xung quanh nó trong ngữ cảnh. Ví dụ, nếu chúng ta đang xét từ "nhân viên", CBOW sẽ dự đoán từ này dựa trên các từ xung quanh nó trong câu như "lương", "chức vụ", "công ty", v.v. Điều này được thực hiện bằng cách lấy trung bình của các biểu diễn vector của các từ xung quanh từ đang xét, sau đó đưa trung bình này vào một mô hình neural network để dự đoán từ đang xét.

CBOW được sử dụng rộng rãi trong các ứng dụng NLP, chẳng hạn như trong bài toán phân tích cảm xúc (sentiment analysis) hoặc trong bài toán xây dựng mô hình dự đoán từ khóa (keyword prediction) cho các bài viết trên mạng.

Để xây dựng một mô hình CBOW, bạn có thể sử dụng các thư viện học máy như TensorFlow hoặc PyTorch. Các thư viện này cung cấp các lớp và phương thức để xây dựng mô hình CBOW và huấn luyện nó trên các tập dữ liệu văn bản.

2.2.6. Mô hình Long Short Term Memory (LSTM):

Là một kiến trúc mạng nơ-ron đặc biệt trong lĩnh vực học sâu (deep learning) được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên (NLP) và các bài toán chuỗi thời gian (time series). LSTM được thiết kế để giải quyết vấn đề biến mất gradient (vanishing gradient problem) trong quá trình huấn luyện mạng nơ-ron sâu.

LSTM được xây dựng dựa trên ý tưởng của mạng nơ-ron ngắn hạn (RNN), nhưng thêm vào đó một số đặc tính mới như cổng quên (forget gate) và bộ nhớ (memory cell) để giúp mô hình ghi nhớ và lưu trữ thông tin quan trọng trong quá trình huấn luyện. Cụ thể, LSTM có khả năng lưu trữ thông tin trong bộ nhớ dài hạn và lựa chọn thông tin quan trọng để truyền tiếp cho các tầng tiếp theo của mạng.

LSTM thường được sử dụng trong các bài toán NLP như dịch máy, phân loại văn bản, sinh văn bản, v.v. Nó cũng được sử dụng trong các bài toán chuỗi thời gian như dự đoán giá cổ phiếu, dự báo thời tiết, v.v.

Để xây dựng một mô hình LSTM, bạn có thể sử dụng các thư viện học máy như TensorFlow hoặc PyTorch. Các thư viện này cung cấp các lớp và phương thức để xây dựng mô hình LSTM và huấn luyện nó trên các tập dữ liệu NLP hoặc chuỗi thời gian.

2.2.7. Mô hình Auto Encoder:

Khái niệm: Mô hình autoencoder là một trong những mô hình học sâu (deep learning) phổ biến trong lĩnh vực xử lý ảnh và xử lý ngôn ngữ tự nhiên.
Autoencoder là một loại mạng neural nhân tạo được sử dụng để học các loại mã hóa dữ liệu không giám sát (unsupervised learning).

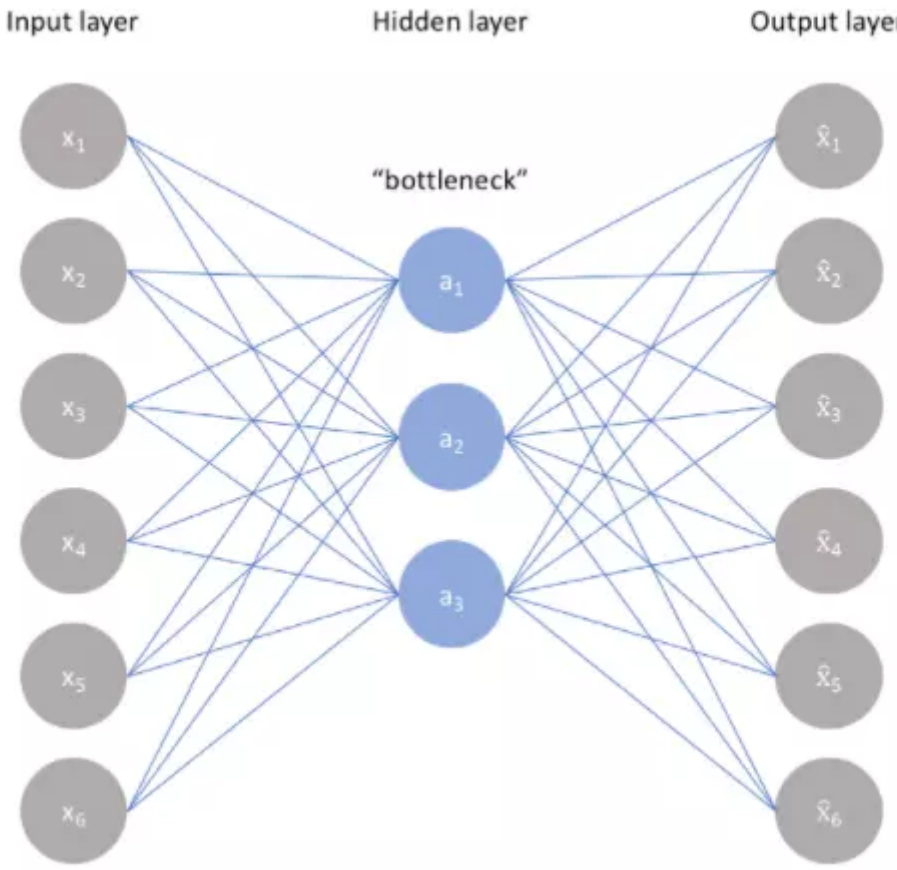
Mục đích của Autoencoder là học cách biểu diễn chiều nhỏ hơn (mã hóa) cho dữ liệu có chiều cao hơn. Đây cũng là lý do mà Autoencoder thường được dùng cho các bài toán giảm chiều dữ liệu hay trích xuất đặc trưng.

Autoencoder có khả năng học các biểu diễn nén (compressed representation) của dữ liệu đầu vào, trong đó dữ liệu có thể là ảnh, văn bản, âm

thanh, v.v. Autoencoder sử dụng một mạng nơ-ron để ánh xạ dữ liệu đầu vào thành các biểu diễn nén và sau đó giải mã để tái tạo lại dữ liệu ban đầu.

Kiến trúc Autoencoder gồm 3 phần:

- Encoder: Module có nhiệm vụ nén dữ liệu đầu vào thành một biểu diễn được mã hóa (coding), thường nhỏ hơn một vài bậc so với dữ liệu đầu vào.
- Bottleneck: Module chứa các biểu diễn tri thức được nén (chính là output của Encoder), đây là phần quan trọng nhất của mạng bởi nó mang đặc trưng của đầu vào, có thể dùng để tái tạo ảnh, lấy đặc trưng của ảnh, ….
- Decoder: Module giúp mạng giải nén các biểu diễn tri thức và tái cấu trúc lại dữ liệu từ dạng mã hóa của nó, mô hình học dựa trên việc so sánh đầu ra của Decoder với đầu vào ban đầu (Input của Encoder).



Cách thức hoạt động:

a) Encoder:

Mô hình bao gồm convolutional blocks và pooling modules để giảm kích thước đầu vào và đưa về dạng bottleneck. Sau đó, bộ giải mã sẽ sử dụng các module upsampling hoặc fully connected để giải mã đặc trưng và tái tạo lại đầu vào ban đầu. Trong những bài toán đơn giản, đầu ra mong đợi sẽ giống với đầu vào, tuy nhiên ở những bài toán phức tạp hơn, đầu ra mong muốn là một ảnh hoàn toàn mới, được hình thành từ đặc trưng của ảnh đầu vào. Tùy thuộc vào đầu vào và bài toán yêu cầu, thay vì sử dụng convolutional và pooling, có thể sử dụng các khối fully connected. Ví dụ, trong bài toán làm mờ chữ số viết tay, chỉ cần một vài lớp fully connected cũng có thể hoạt động hiệu quả.

b) Bottleneck:

Bottleneck được thiết kế để mã hóa thông tin của ảnh đầu vào và mang đặc trưng của ảnh đó. Qua cấu trúc mã hóa-giải mã, mô hình trích xuất được đặc trưng của ảnh và thiết lập được mối tương quan giữa input và output của mạng. Kích thước của Bottleneck càng nhỏ thì nguy cơ overfitting càng thấp, tuy nhiên quá nhỏ cũng sẽ hạn chế khả năng lưu trữ thông tin của ảnh đầu vào và gây khó khăn cho quá trình giải mã ở khối Decoder.

c) Decoder:

Khối cuối cùng, mang nghiệp vụ giải mã từ Bottleneck để tái tạo lại hình ảnh đầu vào dựa vào các đặc trưng “tiềm ẩn” bên trong Bottleneck.

2.3. Thực hiện:

2.3.1. Phương pháp trích dẫn

Bước 1: Chuẩn bị dữ liệu

```
#importing libraries
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import nltk
import bs4 as BeautifulSoup
import urllib.request

url = "https://dantri.com.vn/the-gioi/nga-tuyen-bo-pha-huy-tau-chien-cuoi-cu

#fetching the content from the URL
fetched_data = urllib.request.urlopen(url)

article_read = fetched_data.read()

#parsing the URL content and storing in a variable
article_parsed = BeautifulSoup.BeautifulSoup(article_read,'html.parser')

#returning <p> tags
paragraphs = article_parsed.find_all('p')

article_content = ''

#looping through the paragraphs and adding them to the variable
for p in paragraphs:
    article_content += p.text
print('text')
print(article_content)
```

Bước 2: Xử lý dữ liệu

```
def get_stopwords_list(stop_file_path):
    """load stop words """

    with open(stop_file_path, 'r', encoding="utf-8") as f:
        stopwords = f.readlines()
        stop_set = set(m.strip() for m in stopwords)
        return list(frozenset(stop_set))
```

```
def _create_dictionary_table(text_string) -> dict:

    #Chinh đường dẫn này stopword
    stopwords_path = "D:\study\HUS\machineLearning\Final\code\\vietnamese-stopwords.txt"
    stopwords = get_stopwords_list(stopwords_path)
    print('stopword')
    print(stopwords)
    words = word_tokenize(text_string)

    #reducing words to their root form
    stem = PorterStemmer()

    #creating dictionary for the word frequency table
    frequency_table = dict()
    for wd in words:
        wd = stem.stem(wd)
        if wd in stopwords:
            continue
        if wd in frequency_table:
            frequency_table[wd] += 1
        else:
            frequency_table[wd] = 1

    return frequency_table
```

Bước 3: Mã hóa bài viết thành câu

```
#tokenizing the sentences
sentences = sent_tokenize(article)
```

Bước 4: Tìm tần suất trọng số của các câu

```
def _calculate_sentence_scores(sentences, frequency_table) -> dict:

    #algorithm for scoring a sentence by its words
    sentence_weight = dict()

    for sentence in sentences:
        sentence_wordcount = (len(word_tokenize(sentence)))
        sentence_wordcount_without_stop_words = 0
        for word_weight in frequency_table:
            if word_weight in sentence.lower():
                sentence_wordcount_without_stop_words += 1
                if sentence[:7] in sentence_weight:
                    sentence_weight[sentence[:7]] += frequency_table[word_weight]
                else:
                    sentence_weight[sentence[:7]] = frequency_table[word_weight]

        sentence_weight[sentence[:7]] = sentence_weight[sentence[:7]] / sentence_wordcount_without_stop_words

    return sentence_weight
```

Bước 5: Tính ngưỡng của các câu: Là giá trị trung bình giúp tránh được các câu có giá trị nhỏ hơn giá trị trung bình đó:

```
def _calculate_average_score(sentence_weight) -> int:

    #calculating the average score for the sentences
    sum_values = 0
    for entry in sentence_weight:
        sum_values += sentence_weight[entry]

    #getting sentence average value from source text
    average_score = (sum_values / len(sentence_weight))

    return average_score
```

Bước 6: Nhận bản tóm tắt

```
def _get_article_summary(sentences, sentence_weight, threshold):
    sentence_counter = 0
    article_summary = ''

    for sentence in sentences:
        if sentence[:7] in sentence_weight and sentence_weight[sentence[:7]] >= (threshold):
            article_summary += " " + sentence
            sentence_counter += 1

    return article_summary
```

Bước 7: Thực hiện chương trình

```
def _run_article_summary(article):

    #creating a dictionary for the word frequency table
    frequency_table = _create_dictionary_table(article)

    #tokenizing the sentences
    sentences = sent_tokenize(article)

    #algorithm for scoring a sentence by its words
    sentence_scores = _calculate_sentence_scores(sentences, frequency_table)

    #getting the threshold
    threshold = _calculate_average_score(sentence_scores)

    #producing the summary
    article_summary = _get_article_summary(sentences, sentence_scores, 1 * threshold)

    return article_summary

if __name__ == '__main__':
    summary_results = _run_article_summary(article_content)
    print("summary:", summary_results)
```

2.3.2. Phương Pháp trừu tượng

Để có được dữ liệu, chúng tôi đã tham khảo dữ liệu có được từ một nhóm tác giả Nghiêm Quốc Minh đã tập hợp 300 Cụm văn bản tiếng Việt dùng cho tóm tắt đa văn bản có tên là "Bộ dữ liệu ViMS" từ tập san học thuật "ViMs: a high-quality Vietnamese dataset for abstractive multi-document summarization".

Chúng tôi đã xử lý lại dữ liệu để phù hợp hơn với nhu cầu và lưu vào file csv như sau:

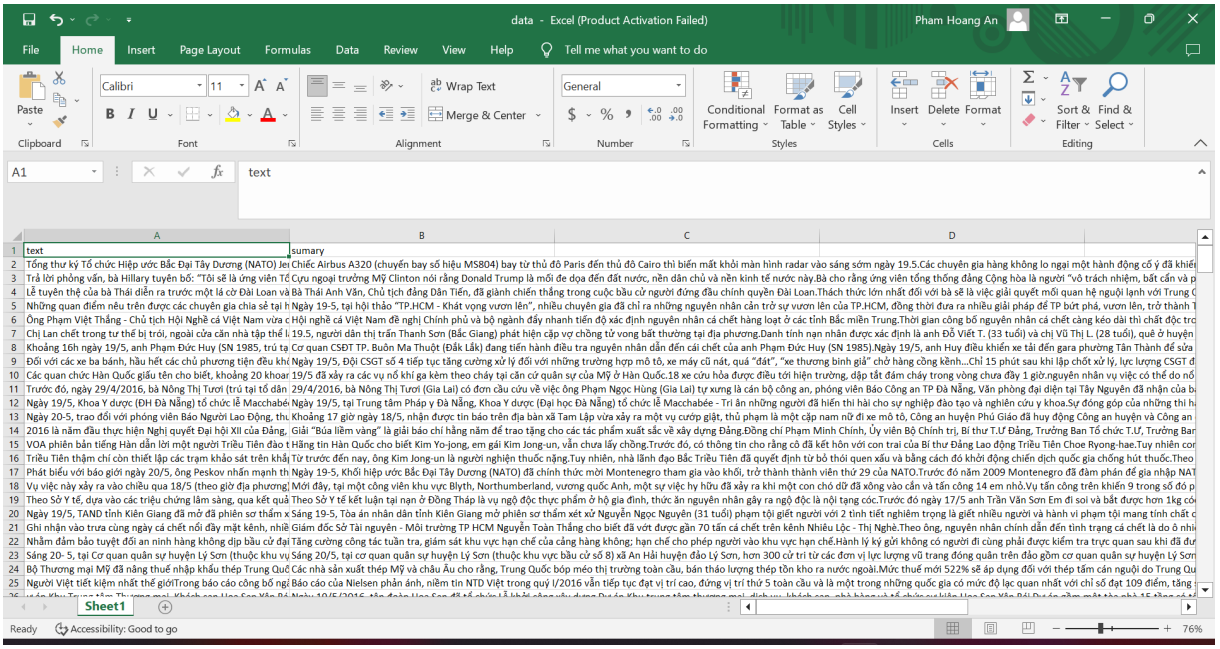
```
[ ]  url = 'D:\study\HUS\machineLearning\Final\ViMs-Dataset-master\ViMs-Dataset-master\ViMs\original'
     subdirectory_names = get_subdirectories(url)
     print("Danh sách thư mục con:")
     path = 'original'
     data = []
     # Mở workbook từ tệp Excel đã có
     workbook = openpyxl.load_workbook('D:\study\HUS\machineLearning\Final\code\data.xlsx')
     for name in subdirectory_names:
         paths = name+path
         listPath = get_files_in_directory(paths)
         for name in listPath:
             print(name)
             # Đọc nội dung từ tệp .txt từ dòng sau dòng chứa từ "Content"
             content = ''
             found_content = False

             with open(name, 'r') as file:
                 for line in file:
                     if found_content:
                         content += line.strip()  # Bỏ các khoảng trắng thừa và dòng mới
                     elif 'Content' in line:
                         found_content = True
             data.append(content)
             break
     # Chọn sheet active (sheet đầu tiên trong workbook)
         sheet = workbook.active

         # Ghi dữ liệu vào cột A
         for index, value in enumerate(data, start=1):
             cell = sheet.cell(row=index+1, column=1)
             cell.value = value

         # Lưu workbook thành tệp Excel
         excel_file_path = 'D:\study\HUS\machineLearning\Final\code\data.xlsx'
         workbook.save(excel_file_path)
```

Sau đó, chúng tôi thu được kết quả là tập dữ liệu phù hợp để sử dụng với mục đích của nghiên cứu như sau:



Bước 1: Chuẩn bị dữ liệu

Nhập các thư viện cần thiết. Tensorflow và keras là những thư viện chính mà chúng tôi sử dụng để triển khai RNN cùng với các thư viện thiết yếu khác và đọc dữ liệu

```
import numpy as np
import pandas as pd
import re
from bs4 import BeautifulSoup
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from nltk.corpus import stopwords
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate, TimeDistributed
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
import warnings
pd.set_option("display.max_colwidth", 200)
warnings.filterwarnings("ignore")
```

```
data = pd.read_excel('/content/data.xlsx')
data.head()
```

	Text	Summary	cleaned_text	cleaned_summary
0	Tổng thư ký Tổ chức Hiệp ước Bắc Đại Tây Dương (NATO) Jens Stoltenberg ngày 19/5 cho biết, nếu Ai Cập để nghị, liên minh này sẽ hỗ trợ công tác tìm kiếm chiếc máy bay mang số hiệu MS 804 của hãng ...	Chiếc Airbus A320 (chuyến bay số hiệu MS804) bay từ thủ đô Paris đến thủ đô Cairo thì biến mất khỏi màn hình radar vào sáng sớm ngày 19.5.Các chuyên gia hàng không lo ngại một hành động cố ý đã kh...	NaN	NaN
1	Trả lời phỏng vấn, bà Hillary tuyên bố: "Tôi sẽ là ứng viên Tổng thống đại diện cho đảng tôi. Điều đó đã ngã ngũ rồi, không có gì thay đổi được".Khẳng định của bà Hillary được đưa ra khi bà nắm ưu...	Cựu ngoại trưởng Mỹ Clinton nói rằng Donald Trump là mối đe dọa đến đất nước, nền dân chủ và nền kinh tế nước này.Bà cho rằng ứng viên tổng thống đảng Cộng hòa là người "vô trách nhiệm, bất cẩn và...	NaN	NaN
2	Lễ tuyên thệ của bà Thái diễn ra trước một lá cờ Đài Loan và chận dụng người sáng lập hòn đảo Tôn Dật Tiên.Cơ quan Ngoại giao Đài Loan cho biết gần 700 nguyên thủ quốc gia, các nhà ngoại giao và q...	Bà Thái Anh Vân, Chủ tịch đảng Dân Tiến, đã giành chiến thắng trong cuộc bầu cử người đứng đầu chính quyền Đài Loan.Thách thức lớn nhất đối với bà sẽ là việc giải quyết mối quan hệ nguội lạnh với ...	NaN	NaN
3	Những quan điểm nêu trên được các chuyên gia chia sẻ tại hội thảo "TP HCM - Khát vọng vươn lên" do báo Tuổi trẻ tổ chức ngày 19/5.Theo PGS. TS Trần Đình Thiên - Viện trưởng Viện kinh tế Việt Nam, ...	Ngày 19-5, tại hội thảo "TP.HCM - Khát vọng vươn lên", nhiều chuyên gia đã chỉ ra những nguyên nhân cần trở sự vươn lên của TP.HCM, đồng thời đưa ra nhiều giải pháp để TP bứt phá, vươn lên, trở th...	NaN	NaN
4	Ông Phạm Việt Thắng - Chủ tịch Hội Nghề cá Việt Nam vừa có công văn gửi Văn phòng Chính phủ, Bộ TN&MT, Bộ NN&PTNT, Bộ KH&CN, Bộ Tài chính sáng nay (27-5).Nội dung công văn đề nghị các cơ quan chức...	Hội nghề cá Việt Nam đề nghị Chính phủ và bộ ngành đẩy nhanh tiến độ xác định nguyên nhân cá chết hàng loạt ở các tỉnh Bắc miền Trung.Thời gian công bố nguyên nhân cá chết càng kéo dài thì chất độ...	NaN	NaN

Bước 2: Tiền xử lý văn bản

```
stop_words = set(stopwords.words('english'))

def text_cleaner(text,num):
    # lower
    newString = text.lower()
    # remove HTML
    newString = BeautifulSoup(newString, "lxml").text
    # Remove any text inside the parenthesis
    newString = re.sub(r'\([^)]*\)', '', newString)
    # remove double quotes
    newString = re.sub('"','', newString)
    # remove 's
    newString = re.sub(r"'s\b","",newString)
    # Eliminate punctuations and special characters
    #newString = re.sub("[^a-zA-Z]", " ", newString)
    # Remove stopwords
    if(num==0):
        tokens = [w for w in newString.split() if not w in stop_words]
    else:
        tokens=newString.split()
    long_words=[]
    # Remove short words
    for i in tokens:
        if len(i)>1:
            long_words.append(i)
    return (" ".join(long_words)).strip()
```

```
# Cleaning the "Text" Column

cleaned_text = []
for t in data['Text']:
    cleaned_text.append(text_cleaner(t,0))

# Cleaning the "Summary" Column

cleaned_summary = []
for t in data['Summary']:
    cleaned_summary.append(text_cleaner(t,1))
```

```
[ ] cleaned_text =np.array(data['cleaned_text'])
    cleaned_summary=np.array(data['cleaned_summary'])

    short_text=[]
    short_summary=[]

    for i in range(len(cleaned_text)):
        if(len(cleaned_summary[i].split())<=max_summary_len and len(cleaned_text[i].split())<=max_text_len):
            short_text.append(cleaned_text[i])
            short_summary.append(cleaned_summary[i])

    df=pd.DataFrame({'text':short_text,'summary':short_summary}) # new dataframe to use
```

Bước 4: Xây dựng mô hình

```
from sklearn.model_selection import train_test_split

x_tr,x_val,y_tr,y_val=train_test_split(np.array(df['text']), np.array(df['summary']),
                                       test_size=0.1, random_state=0, shuffle=True)

# A tokenizer builds the vocabulary and converts a word sequence to an integer sequence.
# We will now build tokenizers for text and summary.

x_tokenizer = Tokenizer()
x_tokenizer.fit_on_texts(list(x_tr))
```

Bộ mã hóa- Encoder_inputs được sử dụng để mã hóa các từ thành dữ liệu số để các lớp LSTM xử lý.Các lớp LSTM- Chúng tôi sử dụng 3 lớp LSTM để xử lý dữ liệu một cách hiệu quả.

```
latent_dim = 300
embedding_dim=100

# Encoder
encoder_inputs = Input(shape=(max_text_len,))

#embedding layer
enc_emb =  Embedding(x_voc, embedding_dim,trainable=True)(encoder_inputs)

#encoder lstm 1
encoder_lstm1 = LSTM(latent_dim,return_sequences=True,return_state=True,dropout=0.4,recurrent_dropout=0.4)
encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)

#encoder lstm 2
encoder_lstm2 = LSTM(latent_dim,return_sequences=True,return_state=True,dropout=0.4,recurrent_dropout=0.4)
encoder_output2, state_h2, state_c2 = encoder_lstm2(encoder_output1)

#encoder lstm 3
encoder_lstm3=LSTM(latent_dim, return_state=True, return_sequences=True,dropout=0.4,recurrent_dropout=0.4)
encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2)

# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None,))

#embedding layer
dec_emb_layer = Embedding(y_voc, embedding_dim,trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)

decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True,dropout=0.4,recurrent_dropout=0.2)
decoder_outputs,decoder_fwd_state, decoder_back_state = decoder_lstm(dec_emb,initial_state=[state_h, state_c])
```

Lớp Chú ý được sử dụng để chọn lọc thông tin liên quan trong khi loại bỏ thông tin không hữu ích

```
# Attention layer
attn_layer = Attention(name='attention_layer')
attn_out= attn_layer([encoder_outputs, decoder_outputs])

# Concat attention input and decoder LSTM output
decoder_concat_input = Concatenate(axis=-1, name='concat_layer')([decoder_outputs, attn_out])

#dense layer
decoder_dense =  TimeDistributed(Dense(y_voc, activation='softmax'))
decoder_outputs = decoder_dense(decoder_concat_input)

# Define the model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model.summary()
```

Giải mã chuỗi bộ giải mã để tạo văn bản và chuyển đổi chuỗi số nguyên thành chuỗi từ để tóm tắt.

```
def decode_sequence(input_seq):
    # Encode the input as state vectors.
    e_out, e_h, e_c = encoder_model.predict(input_seq)
    print("end")

    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))
    # Populate the first word of target sequence with the start word.
    target_seq[0,0] = target_word_index['sostok']

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        print("start0")
        output_tokens, h, c = decoder_model.predict([target_seq] + [e_out, e_h, e_c])
        print("start")
        # Sample a token
        print(output_tokens[0, -1, :].shape)
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_token = reverse_target_word_index[sampled_token_index]
        print("start2")
        if(sampled_token!='eostok'):
            decoded_sentence += ' '+sampled_token

        # Exit condition: either hit max length or find stop word.
        if (sampled_token == 'eostok'  or len(decoded_sentence.split()) >= (max_summary_len-1)):
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1,1))
        target_seq[0] = sampled_token_index

        # Update internal states
        e_h, e_c = h, c
        print("end2")
    return decoded_sentence
```

```
[ ]  def seq2summary(input_seq):
         newString=''
         for i in input_seq:
             if((i!=0 and i!=target_word_index['sostok']) and i!=target_word_index['eostok']):
                 newString=newString+reverse_target_word_index[i]+' '
         return newString

     def seq2text(input_seq):
         newString=''
         for i in input_seq:
             if(i!=0):
                 newString=newString+reverse_source_word_index[i]+' '
         return newString
```

Đầu ra dự đoán

```
[ ]  for i in range(0, 10,10):
         print("Review:",seq2text(x_tr[i]))
         print("Original summary:",seq2summary(y_tr[i]))
         test = x_tr[i].reshape(1,max_text_len)
         test = np.tile(test, (max_text_len,1))
         print(test.shape)
         print("Predicted summary:",decode_sequence(test))
         print("\n")
```

```
Review: chiều 28 5 lê văn công đến uống cà phê tại quán trên đường lý tự trọng lúc sau khi nữ nhân viên hoàng thị ngọc đi ra thì côr
Original summary: chiều 28 5 công an tp buôn ma thuột triển khai bắt giữ lê văn công để điều tra làm rõ hành vi dao nữ nhân viên tại
(300, 300)
10/10 [==============================] - 26s 3s/step
end
start0
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-97-80a2f5055123> in <cell line: 1>()
      5     test = np.tile(test, (max_text_len,1))
      6     print(test.shape)
----> 7     print("Predicted summary:",decode_sequence(test))
      8     print("\n")

                                  2 frames
/usr/local/lib/python3.10/dist-packages/keras/engine/data_adapter.py in _check_data_cardinality(data)
   1850           )
   1851       msg += "Make sure all arrays contain the same number of samples."
-> 1852       raise ValueError(msg)
   1853
   1854

ValueError: Data cardinality is ambiguous:
  x sizes: 1, 300, 300, 300
Make sure all arrays contain the same number of samples.

SEARCH STACK OVERFLOW
```

2.3.3. Tóm tắt văn bản bằng phân cụm

Ngoài ra chúng ta còn tìm hiểu được một cách tiếp cận khác bằng cách phân cụm như sau:

Bước 1: Tiền xử lý văn bản:

```
contents_parsed = content.lower() #Biến đổi hết thành chữ thường
contents_parsed = contents_parsed.replace('\n', '. ') #Đổi các ký tự xuống dòng thành chấm câu
contents_parsed = contents_parsed.strip() #Loại bỏ đi các khoảng trắng thừa
```

Đây là quá trình chuẩn bị dữ liệu để có thể xử lý dễ dàng hơn. Bước này có thể bao gồm loại bỏ các ký tự đặc biệt, dấu câu, số liệu, chuyển thành chữ thường, và loại bỏ các từ dừng (stop words) như "và", "là", "một",... Thực hiện lọc hỗ trợ loại bỏ thông tin dư thừa và không quan trọng có thể không cung cấp bất kỳ giá trị gia tăng nào cho ý nghĩa của văn bản.

Bước 2: Tách câu trong văn bản:

```
[7]  import nltk
     sentences = nltk.sent_tokenize(contents_parsed)
```

Ở đây, chúng ta sử dụng thư viện NLTK, hàm sent_tokenize để lấy ra danh sách các câu thành phần được tách ra từ văn bản.

Bước 3: Chuyển các câu sang vector:

Chúng ta sẽ sử dụng mô hình Word2Vec đã được training cho tiếng việt, chuyển đổi các từ đó sang các vector số thực có chiều dài cố định. Và vector của một câu sau khi chuyển đổi sang sẽ là tổng của các vector đại diện cho các từ trong câu.

```
[15] from gensim.models import KeyedVectors
```

```
[16] w2v = KeyedVectors.load_word2vec_format("/content/vi.vec")
```

Tải và nạp bộ nhớ mô hình word2vec đã được huấn luyện và các từ sẽ được biến đổi thành một vector 100 chiều. Chúng ta sẽ sử dụng thư viện gensim để load lại model.

```
[18] vocab = w2v.index_to_key #Danh sách các từ trong từ điển

     from pyvi import ViTokenizer

     X = []
     for sentence in sentences:
         sentence_tokenized = ViTokenizer.tokenize(sentence)
         words = sentence_tokenized.split(" ")
         sentence_vec = np.zeros((100))
         for word in words:
             if word in vocab:
                 sentence_vec += w2v[word]
         X.append(sentence_vec)
```

Thư viện pyVi với hàm ViTokenize được sử dụng để ghép các từ có nghĩa trong tiếng Việt lại với nhau nhằm đảm bảo nguyên ý nghĩa của từng từ. Sau đó khai một vector 100 chiều gồm toàn số 0, sau đó dùng hàm word2vec chuyển các từ trong câu (nếu được) thành vector rồi cộng vào vector này. Cuối cùng là thêm vào mảng X.

Bước 4: Phân cụm

Đặt số cụm (cluster) bằng với số câu mà chúng ta muốn tóm tắt. Sử dụng và cài đặt dễ dàng với sklearn.

```
[19] from sklearn.cluster import KMeans

     n_clusters = 5
     kmeans = KMeans(n_clusters=n_clusters)
     kmeans = kmeans.fit(X)
```

Bước 5:Xây dựng đoạn văn bản tóm tắt

Mỗi 1 cụm ta sẽ lấy ra 1 câu đại diện có khoảng cách gần với trung tâm của cụm nhất

Sau khi đã có được các câu của văn bản tóm tắt, giờ chúng ta quan tâm là sẽ sắp xếp thứ tự như thế nào cho hợp lý. Ở đây, với mỗi ý nghĩa, mình sẽ tính "thứ tự xuất hiện trung bình" của cụm đó.

Làm tương tự với các cụm khác, sau đó chúng ta sẽ lấy các câu đại diện trong các cụm theo thứ tự từ nhỏ đến lớn của thứ tự xuất hiện trung bình để tạo ra một văn bản tóm tắt

```
[20] from sklearn.metrics import pairwise_distances_argmin_min

     avg = []
     for j in range(n_clusters):
         idx = np.where(kmeans.labels_ == j)[0]
         avg.append(np.mean(idx))
     closest, _ = pairwise_distances_argmin_min(kmeans.cluster_centers_, X)
     ordering = sorted(range(n_clusters), key=lambda k: avg[k])
     summary = ' '.join([sentences[closest[idx]] for idx in ordering])
```

Trong đoạn code trên: Có sử dụng hàm pairwise_distances_argmin_min của sklearning metric để lấy ra khoảng cách vector giữa các vector và các trung tâm cụm của nó, sau đó chọn ra khoảng cách nhỏ nhất để lấy được câu đại diện. Sau đó tính "thứ tự xuất hiện trung bình" và sắp xếp lại để tạo ra văn bản tóm tắt.

2.4. Giao diện ứng dụng web

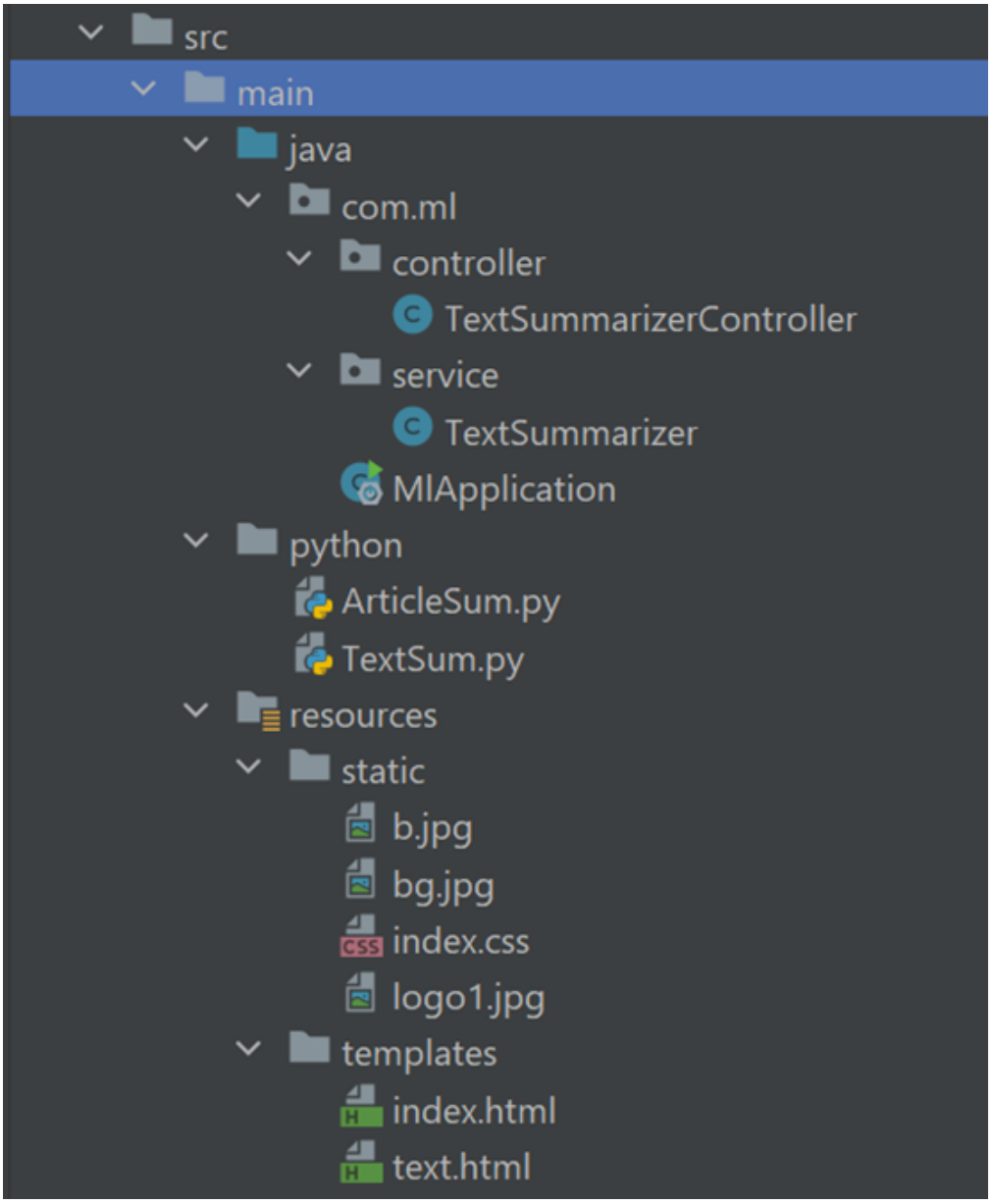
2.4.1. Các công nghệ sử dụng.

- Back-end: Python, Java Spring Boot.
- Front-end: Thymeleaf, HTML, CSS, JavaScript.

2.4.2. Phương thức triển khai:

* Mục tiêu: Tạo ứng dụng web theo mô hình MVC tóm tắt văn bản Tiếng Việt

* Cây thư mục Project:



Hình 1: Cấu trúc thư mục

2.4.3. Phương thức hoạt động.

- Package Java bao gồm:

+ Package Service: Chứa phương thức xử lý thực thi chương trình Python.

+ Package Controller: Bao gồm các phương thức xử lý controller (ánh xạ các phương thức trong controller với các yêu cầu HTTP tương ứng tới các trang giao diện HTML hoặc xử lý hệ thống để trả về các đối tượng mô hình đã triển khai).

+ Package Resource: Là phần View của chương trình, chứa các nội dung hiển thị (Front-end) của hệ thống.

- Package Python: Chứa nội dung mã Python của chương trình "Tóm tắt văn bản Tiếng Việt"

Hình 2: Mô hình thực thi của chương trình

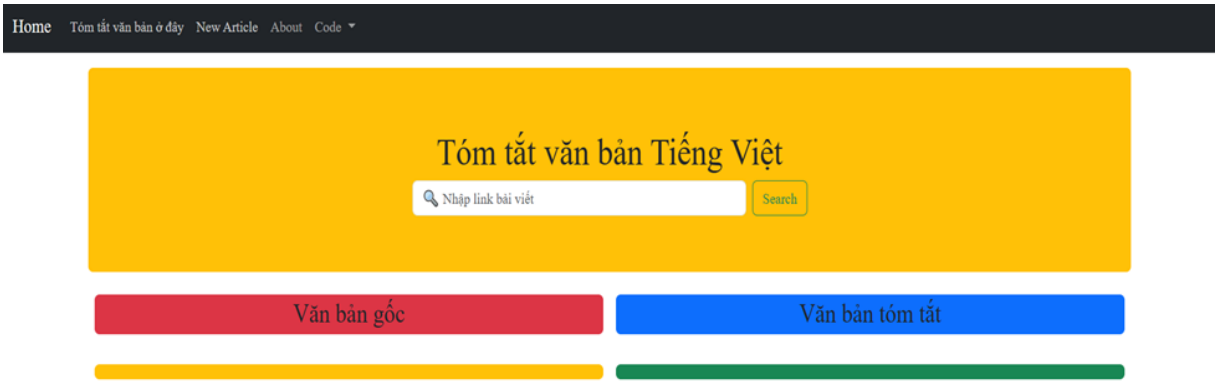
2.4.4. Chi tiết hoạt động.

- Triển khai trang web trên máy chủ Spring Boot (chạy localhost).

- Giao diện hệ thống: 2 chức năng bao gồm :

+ Tóm tắt bài báo Tiếng Việt qua URL.

- Input: Người dùng nhập đường dẫn bài viết (URL) cần tóm tắt vào form.



Hình 2: Input chức năng
"Tóm tắt bài báo Tiếng Việt qua URL".

- Output: Kết quả bài viết được tóm tắt.

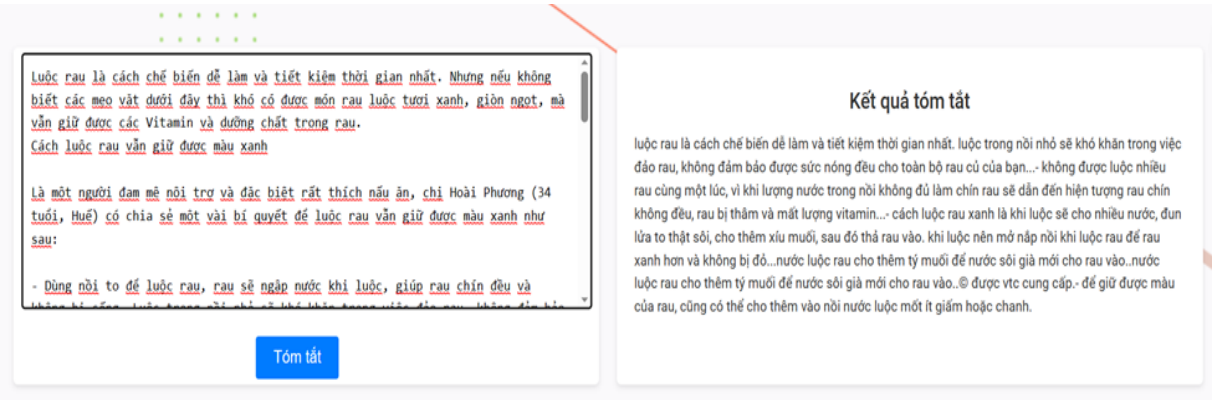
Hình 3: Output chức năng
"Tóm tắt bài báo Tiếng Việt qua URL".

+ Tóm tắt văn bản Tiếng Việt tùy chọn.

- Input: Người dùng nhập văn bản Tiếng Việt cần tóm tắt vào form.

Hình 4: Input chức năng
"Tóm tắt văn bản Tiếng Việt tùy chọn".

+ Output: Kết quả bài viết được tóm tắt.



Hình 5: Output chức năng

"Tóm tắt văn bản Tiếng Việt tùy chọn".

CHƯƠNG III: KẾT QUẢ ĐẠT ĐƯỢC

3.1. Kết quả

- Tốc độ tóm tắt nhanh hơn rất nhiều so với cách tóm tắt thủ công.
- Độ chính xác khá ổn khi kiểm tra, cho ra được kết quả là đoạn văn tóm tắt.
- Có giao diện web thân thiện với người dùng, giúp người dùng dễ dàng sử dụng.
- Người dùng có thể chọn 2 cách tóm tắt tùy thuộc nhu cầu tóm tắt cả bài hay một phần văn bản.

3.2. Các hạn chế và nhược điểm.

- Xử lý ngôn ngữ tự nhiên NLP là một lĩnh vực hay và hấp dẫn nhưng bên cạnh đó nó cũng là một thực thách đối với chúng tôi khi đây là một vấn đề khá lạ chưa được tìm hiểu nhiều về phần xử lý ngôn ngữ ở trên trường.
- Cả nhóm đã tập trung tìm hiểu, thảo luận rất nhiều nhưng do các tài liệu về xử lý ngôn ngữ Tiếng Việt khác hạn chế
- Do thời gian không có quá nhiều để hoàn thiện hơn sản phẩm.
- Độ chính xác: mặc dù đã đạt được độ chính xác nhất định những vẫn chưa hoàn toàn chính xác có thể dẫn đến sự sai sót và hiểu nhầm trong quá trình tóm tắt.
- Dữ liệu huấn luyện vẫn chưa đủ lớn, chưa đại diện cho tất cả loại văn bản dẫn đến chưa hoạt động hiệu quả.
- Vẫn còn gặp khó khăn trong việc hiểu các cấu trúc ngôn ngữ phức tạp và không gian ngữ nghĩa do vẫn còn thiếu những kiến thức xử lý ngôn ngữ tự nhiên.

KẾT LUẬN

Qua dự án bài tập lớn này, nhóm đã tìm hiểu về ứng dụng của học máy kết hợp với xử lý ngôn ngữ tự nhiên (NLP) đê tóm tắt văn bản tiếng việt qua hai phương pháp chính và các kỹ thuật . Nhóm đã nghiên cứu và thực nghiệm nhiều phương pháp khác nhau để thực hiện trong đó có phương pháp trích dẫn, phương pháp trừu tượng và phương pháp phân cụm.

Kết quả cho thấy rằng, chúng tôi đã thực hiện cơ bản được tóm tắt văn bản tiếng việt cùng với đó là thực hiện được giao diện web cho chương trình. Ngoài ra có thể thấy được rằng phương pháp trừu tượng đạt được kết quả tốt hơn so với phương pháp trích dẫn. Cùng với đó là giao diện web dễ dàng cho người dùng sử dụng cũng như sở hữu hai tùy chọn phù hợp hơn với nhu cầu người dùng.

Tuy nhiên, chúng tôi vẫn còn gặp những khó khăn và hạn chế khi thực hiện bài tập lớn này. Xử lý ngôn ngữ tự nhiên (NLP) cũng là một trong những lĩnh vực còn nhiều khó khăn với chúng tôi để tiếp cận và xử lý đạt kết quả tốt nhất. Những tài liệu về chủ đề và những tài liệu hỗ trợ còn hạn chế, cùng với đó là thời gian nghiên cứu không được nhiều dẫn đến chúng tôi chưa thể hoàn thiện dự án này một cách tốt nhất. Dữ liệu để huấn luyện gặp khó khăn trong việc thu thập cũng như xử lý để đảm bảo chất lượng.

Tóm lại, dự án này của chúng tôi đã chứng minh được việc ứng dụng học máy để tóm tắt văn bản tiếng Việt là khả thi và có triển vọng đạt hiệu quả tốt. Tuy nhiên, dự án này vẫn cần tiếp tục nghiên cứu cũng như có thêm những cải tiến, phát triển những phương pháp mới để giải quyết những hạn chế còn gặp phải và tạo ra giải pháp tối ưu hơn trong việc tóm tắt văn bản tiếng Việt.

TÀI LIỆU THAM KHẢO

Tóm tắt văn bản trong Machine Learning:

[1]. Comprehensive Guide to Text Summarization using Deep Learning in Python(analyticsvidhya.com)

[2].https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning

[3].https://viblo.asia/p/xay-dung-chuong-trinh-tom-tat-van-ban-tieng-viet-don-gian-voi-machine-learning

[4].https://hoctructuyen123.net/tom-tat-van-ban-trong-hoc-may/?fbclid=IwAR2g61kb-19-8hbxkrCwt9KEI7hDBsYQi2mNX3orbsVGfuzDqMl5h4a_f50

[5].Word2vec:https://machinelearningcoban.com/tabml_book/ch_embedding/word2vec.html

[6]. LSTM:https://websitehcm.com/long-short-term-memory-lstm-la-gi/

[7].Autoencoder:https://bizflycloud.vn/tin-tuc/autoencoder-la-gi-20220526165157229.htm

[8] N.-T. Tran, M.-Q. Nghiem, N. T. H. Nguyen, N. L.-T. Nguyen, N. Van Chi, and D. Dinh, "ViMs: a high-quality Vietnamese dataset for abstractive multi-document summarization." Language Resources and Evaluation, vol. 54, no. 4, pp. 893-920, 2020, doi: 10.1007/s10579-020-09495-4: https://github.com/CLC-HCMUS/ViMs-Dataset?fbclid=IwAR1gLec6907ZXKalU8nteQ4xsUDfIabxx2WL2RQWEIVHEHhUsaeyAktWUu4

[9] How to Make a Text Summarizer: https://youtu.be/ogrJaOIuBx4?t=325