

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023-2024
TÌM HIỂU VÀ CÀI ĐẶT THUẬT TOÁN ĐÁNH GIÁ SỰ
TƯƠNG ĐỒNG CỦA VĂN BẢN SỬ DỤNG ĐỘ ĐO CỦA
COSINE VÀ MANHATTAN

Giáo viên hướng dẫn:
Nguyễn Nhứt Lam

Sinh viên thực hiện:
Họ tên: Trần Bá Hiếu
MSSV: 110121024
Lớp: DA21TTA

Trà Vinh, 8 tháng 1 năm 2024

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023-2024
TÌM HIỂU VÀ CÀI ĐẶT THUẬT TOÁN ĐÁNH GIÁ SỰ
TƯƠNG ĐỒNG CỦA VĂN BẢN SỬ DỤNG ĐỘ ĐO CỦA
COSINE VÀ MANHATTAN

Giáo viên hướng dẫn:
Nguyễn Nhứt Lam

Sinh viên thực hiện:
Họ tên: Trần Bá Hiếu
MSSV: 110121024
Lớp: DA21TTA

Trà Vinh, 8 tháng 1 năm 2024

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

[illegible]

Trà Vinh, ngày tháng năm
Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

[illegible]

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Nhứt Lam về sự hỗ trợ và sự hướng dẫn quý báu trong quá trình thực hiện đề tài của mình trong khóa học cơ sở ngành. Sự tận tâm và kiến thức sâu rộng của thầy đã giúp tôi nắm vững kiến thức cơ bản và phát triển kỹ năng trong lĩnh vực này.

Thầy đã luôn sẵn sàng lắng nghe và hỗ trợ trong việc xác định đề tài, tìm hiểu và thực hiện dự án của em. Sự hướng dẫn chi tiết và những lời khuyên quý báu từ thầy đã giúp tôi vượt qua những khó khăn và hoàn thành đồ án một cách thành công.

Em xin cảm ơn thầy về sự đóng góp quý báu của mình trong quá trình học tập và mong rằng kiến thức và kinh nghiệm mà em đã học từ thầy sẽ luôn là nguồn động viên và phát triển trong tương lai.

Chân thành cảm ơn và kính chúc thầy một sức khỏe tốt và thành công trong sự nghiệp giảng dạy và nghiên cứu.

Trà Vinh, 8 Tháng 1 năm 2024

TRẦN BÁ HIẾU

MỤC LỤC

KHOA KỸ THUẬT VÀ CÔNG NGHỆ.....	1
BỘ MÔN CÔNG NGHỆ THÔNG TIN	1
THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH.....	1
KHOA KỸ THUẬT VÀ CÔNG NGHỆ.....	2
BỘ MÔN CÔNG NGHỆ THÔNG TIN	2
THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH.....	2
CHƯƠNG 1: TỔNG QUAN.....	9
1.1. Tổng quan về vấn đề.	9
1.2. Hướng giải quyết.....	10
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT.....	12
2.1. Phương pháp biến diễn một văn bản.	12
2.2. Mô hình Vec-Tơ.....	12
2.3. Các độ đo tương đồng.	13
2.4. Mô hình tổng quát.	14
2.5. TF-IDF là gì ?.....	15
2.6. Độ đo Cosine Similarity.....	16
2.7. Độ đo Manhattan Distance.	22
2.8. Python là gì ?.....	22
2.9. Scikit-Learn là gì?	23
2.10. Tại sao lại dùng Sklearn trong đề tài này ?	23
2.11. Flask là gì ?.....	23
2.12. HTML,CSS là gì?.....	24
2.13. JavaScript là gì?.....	24
2.14. Json là gì?	24
2.15. Visual Studio Code là gì ?.....	25

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	26
3.1. Hướng dẫn cài đặt Python.	26
3.2. Hướng dẫn cài đặt Visual Studio Code.	27
3.3. Hướng dẫn cài đặt Flask trên VSCode.	27
3.4. Hướng dẫn cài Sklearn trên VSCode.	27
3.5. Xây dựng thuật toán đánh giá sự tương đồng của văn bản Cosine và Manhattan hiển thị với giao diện Website.	28
3.6. Bộ mã trong mục chính web.py.	29
3.6.1. Tất cả các biến và công dụng trong web.py.	29
3.6.2 Toàn bộ code trong web.py.	30
3.6.3. Phân tích các đoạn code trong web.py.	32
3.7. Bộ mã trong mục index.html:	36
3.7.1. Toàn bộ bộ mã của trang Index.html.	36
3.7.2. Về HTML của trang Index.	39
3.7.3. Về JavaScript của trang Index:	40
3.8. Bộ mã trong mục Ketqua.html.	43
3.8.1. Toàn bộ bộ mã của trang Ketqua.html.	43
3.8.2. Chức năng của trang Ketqua.	44
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU	45
4.1. Tổng quan kết quả nghiên cứu.	45
4.2. Mô hình hoạt động của trang web tính toán Cosine và Manhattan.	46
4.3. Mô hình nút đếm từ xuất hiện.	46
4.4. Giao diện trang Index.html.	47
4.5. Giao diện trang Ketqua.html.	48
4.6. Giao diện đếm số từ trong Form.	49
4.7. Giao diện đếm từ xuất hiện.	49
4.8. Thông báo lỗi.	49

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	51
5.1. Kết quả đạt được:	51
5.2. Kết quả chưa đạt được:.....	51
5.3. Hướng phát triển:	51
DANH MỤC TÀI LIỆU THAM KHẢO	53

DANH MỤC HÌNH ẢNH

Hình 1. Ví dụ về gốc tạo bởi 2 vec-tơ.	13
Hình 2. Mô hình so sánh hai văn bản.	14
Hình 3. Hình ảnh minh họa về độ đo Cosine Similarity.	17
Hình 4. Ví dụ về Manhattan Distance.	22
Hình 5. Thư mục code của nghiên cứu.....	28
Hình 6. Mô hình hoạt động của website.....	46
Hình 7. Mô hình chức năng nút đếm từ xuất hiện.....	47
Hình 8. Giao diện trang Index.html.....	47
Hình 9. Giao diện trang ketqua.html.	48
Hình 10. Giao diện đếm số từ trong Form.....	49
Hình 11. Giao diện đếm từ xuất hiện.	49
Hình 12. Thông báo lỗi.....	50

BẢNG BIỂU

Bảng 1. Tính toán TF.....	19
Bảng 2. Kết quả TF.....	20
Bảng 3. Tính toán IDF.	20
Bảng 4. Tính toán IF-IDF.	21
Bảng 5. Tất cả các biến của web.py.	29

TÓM TẮT NIÊN LUẬN ĐỒ ÁN CƠ SỞ NGÀNH

Vấn đề nghiên cứu:

- Mục tiêu: Khám phá và cài đặt thuật toán để đánh giá mức độ tương đồng giữa các văn bản.
- Ý nghĩa: Giúp phát hiện đạo văn, phân loại văn bản, và cải thiện hệ thống tìm kiếm thông tin.

Các hướng tiếp cận:

- Về mặt lý thuyết: nghiên cứu về độ đo Cosine và Manhattan trong không gian vector để đánh giá sự tương đồng. Nghiên cứu thông qua các trang web video trên Internet.
- Ứng dụng: Phát triển thuật toán dựa trên cả hai độ đo này để xác định mức độ giống nhau giữa các văn bản, xây dựng thuật toán thông qua ngôn ngữ Python và các thư viện của python.

Cách giải quyết vấn đề:

- Xây dựng và thử nghiệm các thuật toán dựa trên ngôn ngữ lập trình Python và các tập dữ liệu thích hợp.
- So sánh hiệu quả và độ chính xác của các thuật toán thông qua các đoạn, câu văn và đánh giá.

Một số kết quả đạt được:

- Hiệu suất mô tả về khả năng xử lý và độ chính xác của thuật toán trong các tình huống thực tế.
- Ứng dụng đề xuất các lĩnh vực có thể áp dụng, như hệ thống kiểm tra đạo văn hoặc cải thiện công cụ tìm kiếm.

Kết cấu của bài báo cáo:

- ❖ Chương 1: Tổng quan.
- ❖ Chương 2: Nghiên cứu lý thuyết.
- ❖ Chương 3: Hiện thực hóa nghiên cứu.
- ❖ Chương 4: Kết quả nghiên cứu.

❖ Chương 5: Kết luận và hướng phát triển.

Github: <https://github.com/BHieuu/csn-da21tta-tranbahieu-cosine-manhattan>.

MỞ ĐẦU

Lý do chọn đề tài:

+ Trong thời đại kỹ nguyên số lượng văn bản được sản xuất và lưu trữ trực tuyến tăng trưởng nhanh chóng, đòi hỏi chúng ta cần có công cụ phân tích và xử lý văn bản một cách hiệu quả.

+ Có nhu cầu lớn về công cụ có thể nhanh chóng và chính xác để xác định sự tương đồng văn bản trong các lĩnh vực giáo dục và nghiên cứu như các bài kiểm tra tại trường, các đoạn văn câu văn cần tìm kiếm sự tương đồng trên nguồn dữ liệu khổng lồ trên Internet.

Mục đích nghiên cứu:

+ Mục tiêu chính là phát triển và thử nghiệm một thuật toán hiệu quả để đánh giá sự tương đồng của văn bản dựa trên độ đo Cosine và Manhattan. Góp phần vào việc cải thiện công nghệ xử lý văn bản và tìm kiếm thông tin.

Đối tượng nghiên cứu:

- Tập trung vào việc phân tích và xử lý các loại văn bản số như bài báo, bài văn của học sinh và các tài liệu khác nhau.
- Nghiên cứu và áp dụng hai độ đo này trong việc đánh giá sự tương đồng.

Phạm vi nghiên cứu:

+ Tập trung vào việc nghiên cứu và hiểu biết về cách thức hoạt động của các độ đo Cosine và Manhattan thông qua những nguồn tin có sẵn đáng tin cậy ở trên mạng Internet, từ đó hiểu biết thêm về hai độ đo Cosine và Manhattan và phát triển thành thuật toán hiệu quả.

+ Thử nghiệm thuật toán trên một số đoạn, câu văn bản cụ thể và đánh giá hiệu suất của nó trong các tình huống thực tế.

CHƯƠNG 1: TỔNG QUAN

+ Em đã tự hỏi rằng làm thế nào mà các máy tính có thể so sánh và phân biệt giữa hàng nghìn thậm chí hàng triệu văn bản khác nhau. Đó chính là nhờ vào những thuật toán thông minh, giúp máy tính “Hiểu” được văn bản giống như cách chúng ta làm. Trong thế giới ngập tràn thông tin như hiện nay, việc này trở nên cực kỳ quan trọng từ việc tìm kiếm thông tin chính xác trên internet cho đến việc phát hiện ra những bài báo khoa học sao chép nhau.

+ Đề tài của em sẽ tập trung vào hai công cụ chính đó là: độ đo Cosine và Manhattan. Nếu chúng ta nghĩ rằng chúng chỉ là những thuật toán học khô khan, thì chúng ta sẽ thật sự ngạc nhiên sau khi chúng ta thật sự hiểu nhiều hơn về chúng đấy. Chúng như là hai chiếc cân, giúp chúng ta ”cân đo” và “so sánh” sự giống và khác nhau giữa các văn bản đó. Hãy tưởng tượng chúng ta có một đồng sách trước mặt và chúng ta cần tìm ra những cuốn có nội dung tương tự nhau, chúng ta sẽ làm thế nào? Có thể chúng ta sẽ mất hàng đồng giờ thậm chí cả tuần liền để đọc và lọc chúng ra. Do đó độ đo Cosine và Manhattan chính là trợ thủ đắc lực giúp chúng ta làm việc đó một cách nhanh chóng và chính xác nhất.

+ Mục tiêu của chúng ta không chỉ là hiểu về cách thức hoạt động của hai độ đo này mà còn cài đặt chúng vào một thuật toán máy tính thuật sự. Đây sẽ là một hành trình thú vị, khi chúng ta vừa được “chơi” với số liệu mà chúng ta thường cho là nhàm chán và khô khan.

+ Như vậy đề tài này không chỉ dừng lại ở việc học thuật mà còn mở ra những khả năng ứng dụng rộng rãi từ việc giúp các bạn học sinh/sinh viên tìm kiếm thông tin một cách dễ dàng, cho đến việc hỗ trợ các giáo viên/giảng viên đánh giá các bài kiểm tra của sinh viên có sự tương đồng với nhau hay không.

+ Hy vọng rằng, qua quá trình nghiên cứu này chúng ta không chỉ học hỏi được nhiều điều mới mẻ mà còn góp phần vào sự phát triển của công nghệ thông tin và xử lý dữ liệu. Chúng ta sẽ cùng nhau khám phá thử nghiệm và tạo ra điều đó thực sự hữu ích cho cộng đồng.

1.1. Tổng quan về vấn đề.

Dựa trên nhu cầu phát hiện sao chép trong các tài liệu văn bản, hiện nay đã xuất hiện các giải pháp và công cụ phần mềm hỗ trợ việc này. Các phương pháp hiện hành thường tập trung vào việc tìm kiếm và so sánh các chuỗi văn bản, hiệu

quả chủ yếu khi nội dung được sao chép một cách chính xác, hay còn gọi là sao chép "Nguyên văn". Trong bối cảnh đó, một bài báo mới đã đề xuất một phương pháp tiên tiến hơn: sử dụng độ đo cosine để so sánh độ tương đồng giữa các văn bản. Phương pháp này không chỉ cải thiện khả năng phát hiện sao chép văn bản mà còn góp phần vào việc hạn chế xâm phạm quyền sở hữu trí tuệ trong lĩnh vực công bố khoa học. Phương pháp sử dụng độ đo Manhattan và Cosine cho thấy những tiềm năng lớn trong việc tăng cường độ chính xác của quá trình phát hiện sao chép, đặc biệt trong trường hợp các văn bản được biến đổi một cách tinh vi.

Ví dụ :

- Văn bản Kiểm tra (Văn bản A): Một bài luận của học sinh.
- Văn bản Kiểm Tra (Văn bản B): Nhiều bài báo cáo và tài liệu trực tuyến có sẵn.

1.2. Hướng giải quyết.

1. Chuẩn bị dữ liệu.

+ Xác định và thu thập Văn bản A và Văn bản B.

2. Xử lý Văn bản.

+ Thực hiện Tokenization.

Chú thích:

+ Tokenization là quá trình chia nhỏ văn bản thành các đơn vị nhỏ hơn, thường là từ, cụm từ, hoặc ký tự. trong ngôn ngữ, từ là đơn vị thông dụng nhất để dùng tokenization. Mục đích của tokenization là để phân tách văn bản thành các phần nhỏ hơn để có thể phân tích và xử lý dễ dàng hơn.

Ví dụ:

- Văn bản: “ Tôi là sinh viên của TVU.”
- Sau Tokenization:[“Tôi”, “là”, “sinh”, “viên”, “của”, “TVU”]

+ Sau khi thực hiện tokenization giúp đoạn văn bản trở nên gọn hơn, từ đó cung cấp cho một dữ liệu tốt hơn để cho các bước phân tích tiếp theo.

3. Tính toán Độ Đo Cosine.

+ Tính độ đo cosine giữa văn bản A và văn bản B.

+ Đánh giá mức độ tương đồng dựa trên giá trị Cosine (Giá trị gần 1 chỉ ra sự tương đồng cao).

4. Tính độ đo Manhattan.

- + Tính khoảng cách Manhattan giữa văn bản A và văn bản B.
- + Đánh giá mức độ tương đồng của chúng thông qua khoảng cách càng nhỏ, hai văn bản càng tương đồng.

5. Phân Tích và Kết Luận.

- + So sánh và phân tích kết quả từ hai phép đo.
- + Nếu cả hai chỉ số đều cho thấy mức độ tương đồng cao, có thể kết luận rằng bài luận có khả năng sao chép từ một hay nhiều nguồn khác.
- + Xây dựng độ đo hợp lý thông quan ngôn ngữ lập trình Python và các thư viện do Python hỗ trợ.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Phương pháp biến diễn một văn bản.

+ Trong xử lý văn bản có rất nhiều phương pháp để xử lý văn bản và có cách tính toán khác nhau, nhưng nói một cách tổng quan thì các phương pháp đó thường không được tương tác trực tiếp như trên tập dữ liệu thô ban đầu, mà chúng thường được thực hiện một số bước như là tiền xử lý văn bản hoặc mô hình hóa văn bản.

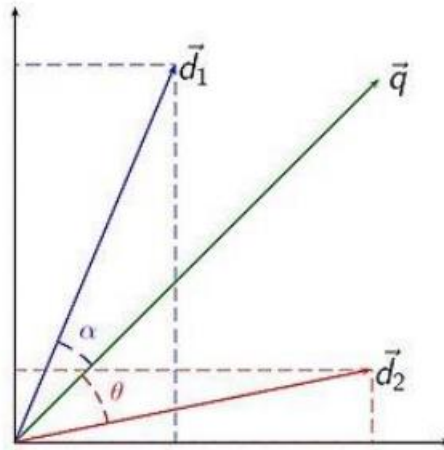
+ Một văn bản trước khi được mô hình hóa, ý có nghĩa là trước khi được sử dụng, cần phải trải qua được tiền xử lý. Quá trình tiền xử lý sẽ giúp nâng cao hiệu suất và còn giảm độ phức tạp khi sử dụng các phương pháp tính độ tương đồng. Nhưng tùy vào mục đích khai thác mà chúng ta sẽ có những phương pháp tiền xử lý văn bản khác nhau như là: chuyển chúng hẳn thành dạng chữ thường, loại bỏ các ký tự đặc biệt, các chữ số hay là tách các phép toán số học, tách văn bản thành các câu hoặc các từ riêng lẻ để sử dụng cho mục đích tính toán sau này, loại bỏ đi các từ dừng hay còn gọi là các từ không cần thiết, các từ vào một cấu trúc dữ liệu phù hợp... Nói chung, một văn bản ở dạng thô (dạng chuỗi) cần được chuyển sang một mô hình khác để tạo thuận lợi cho việc biểu diễn và tính toán chúng. Còn tùy vào từng thuật toán xử lý khác nhau mà chọn một mô hình biến diễn phù hợp.

+ Trong các cơ sở dữ liệu văn bản, mô hình vec-tơ là một mô hình biểu diễn văn bản được sử dụng phổ biến nhất. Mọi quan hệ giữa các tập văn bản sẽ được thực hiện thông qua việc tính toán trên các vec-tơ biểu diễn nên đem lại hiệu quả rất cao. Theo mô hình này, mỗi văn bản được biểu diễn thành một vec-tơ và mỗi thành phần của một vec-tơ là một từ khóa riêng biệt trong tập văn bản gốc và được gán một giá trị là hàm f , chỉ mật độ xuất hiện của từ (hay từ khóa) trong văn bản.

2.2. Mô hình Vec-Tơ.

+ Mô hình vec-tơ là một mô hình đại số thông dụng và đơn giản, thường được dùng để biểu diễn văn bản. Sau khi tiền xử lý, một văn bản được mô tả bởi một cụm từ hay tập các từ (gọi là từ chỉ mục). Tập các từ này chỉ mục xác định một không gian mà mỗi từ chỉ mục tượng trưng cho một chiều trong không gian đó. Các từ chỉ mục này cũng chính là các từ chứa nội dung chính của tập văn bản, mỗi từ chỉ mục này sẽ được gán một trọng số. Để tính toán tương đồng giữa văn bản truy vấn và các văn bản mẫu, chúng ta có thể sử dụng các phép toán của mô hình vec-tơ.

+ Với một văn bản d được biểu diễn dưới dạng \vec{d} với $\vec{d} \in R^m$ là một vec-tơ m chiều. Trong đó $\vec{d} = \{w_1, w_2, \dots, w_m\}$ và m là số chiều của vec-tơ văn bản d , mỗi chiều tương ứng với một từ trong tập hợp các từ, w_i là trọng số của đặc trưng thứ i (với $1 \leq i \leq m$). Độ tương tự của hai văn bản thường được định nghĩa là khoảng cách các điểm hoặc là góc giữa những vec-tơ trong không gian.



Hình 1. Ví dụ về góc tạo bởi 2 vec-tơ.

2.3. Các độ đo tương đồng.

+ Sự tương đồng giữa hai văn bản là sự giống nhau về nội dung giữa hai văn bản đó. Do đó, hai văn bản là bản sao hoặc gần giống nhau thì sẽ có nội dung giống nhau khá nhiều, hay gọi “độ tương đồng” giữa hai văn bản là cao. Độ tương đồng nằm trong khoảng giữa 0 và 1, như vậy độ tương đồng càng gần 1 thì khả năng các văn bản là bản sao hoặc sẽ gần giống nhau là rất cao và ngược lại. Do đó, để xem chúng có phải là bản sao hoặc gần giống nhau hay không ta phải tính độ tương đồng giữa chúng.

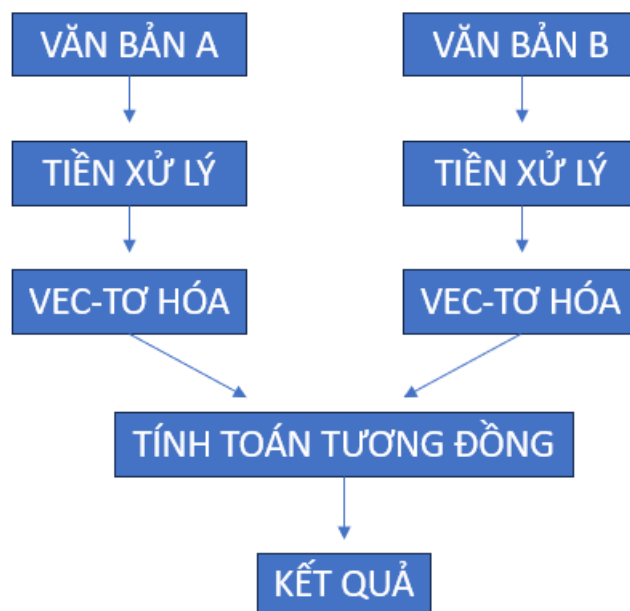
+ Để đánh giá mức độ tương đồng của văn bản trong tiếng Anh, các nghiên cứu đã áp dụng thành công một loạt phương pháp. Điển hình là sử dụng các bộ dữ liệu tiêu chuẩn về ngôn ngữ như Wordnet, Brown Đoạn văn bản, và Penn TreeBank. Những phương pháp này phân loại thành hai hướng tiếp cận chính: phương pháp dựa trên dữ liệu và phương pháp dựa trên tri thức, giúp xác định sự tương đồng ngữ nghĩa giữa các từ, trong khi phương pháp dựa trên chuỗi nhằm xác định sự tương đồng từ vựng. Trong đó, phương pháp dựa trên chuỗi chia thành hai loại: loại dựa trên ký tự và loại dựa trên từ. Các thuật toán dựa trên ký tự bao gồm Longest Common Subsequence (LCS), Damerau-Levenshtein, Jaro, Jaro-Winkler, Needleman-Wunsch, Smith-Waterman, và n-gram. Trong khi đó, các thuật toán dựa

trên từ bao gồm khoảng cách Manhattan, Cosine Similarity, Dice's coefficient, khoảng cách Euclidean, Jaccard Similarity, Matching coefficient và Overlap coefficient....

+ Trong nghiên cứu này, nhóm tác giả nghiên cứu, cài đặt và thực nghiệm dựa trên hai độ đo Cosine, Matthanan, để tính toán mức độ giống nhau của văn bản tiếng Việt.

2.4. Mô hình tổng quát.

+ Quá trình so sánh giữa hai văn bản trong nghiên cứu này diễn ra theo một loạt các bước cụ thể. Đầu tiên, cả hai văn bản đều trải qua giai đoạn tiền xử lý để chuẩn bị cho việc xử lý dữ liệu. Tiếp theo, chúng được chuyển đổi thành dạng vector thông qua quá trình vector hóa, cho phép biểu diễn mỗi văn bản dưới dạng một chuỗi số đại diện cho các đặc trưng của nó. Cuối cùng, hai vector này được so sánh với nhau để xác định độ tương đồng giữa chúng. Quá trình này được thực hiện dựa trên một mô hình đề xuất, được thiết kế để đánh giá mức độ giống nhau của văn bản một cách chính xác và hiệu quả.



Hình 2. Mô hình so sánh hai văn bản.

Quá trình so sánh giữa một văn bản truy vấn và tập văn bản nguồn được thực hiện theo mô hình sau:

- + Tiền xử lý dữ liệu nguồn: Tập các văn bản nguồn cần được xử lý trước để loại bỏ các ký tự không cần thiết, chuyển đổi văn bản thành dạng tiêu chuẩn và loại bỏ các từ không quan trọng.

- + Vector hóa dữ liệu nguồn: Sau khi tiền xử lý, các văn bản nguồn sẽ được chuyển đổi thành dạng vector, trong đó mỗi văn bản được biểu diễn bằng một vec-tơ số học. Quá trình này có thể sử dụng các phương pháp như TF-IDF hoặc Word Embeddings để biểu diễn văn bản.

- + Tiền xử lý văn bản truy vấn: Văn bản truy vấn cũng cần trải qua tiền xử lý tương tự như dữ liệu nguồn để đảm bảo tính nhất quán và đồng nhất.

- + Vector hóa văn bản truy vấn: Văn bản truy vấn sau tiền xử lý sẽ được chuyển thành vector để biểu diễn.

- + So sánh văn bản truy vấn với dữ liệu nguồn: Cuối cùng, văn bản truy vấn sẽ được so sánh với tất cả các văn bản trong tập dữ liệu nguồn thông qua việc tính toán độ tương đồng giữa vector biểu diễn của văn bản truy vấn và các vector biểu diễn của văn bản nguồn. Kết quả của quá trình này là việc xác định mức độ giống nhau giữa văn bản truy vấn và các văn bản nguồn, thường được biểu thị dưới dạng một giá trị số (điểm số) hoặc một danh sách các văn bản nguồn tương đồng.

2.5. TF-IDF là gì ?

TF-IDF, viết tắt của "Term Frequency-Inverse Document Frequency", là một phương pháp thống kê được sử dụng trong xử lý ngôn ngữ tự nhiên và trí tuệ nhân tạo thường dùng để đánh giá mức độ quan trọng của một từ trong một văn bản so với một tập hợp các văn bản hoặc một đoạn văn bản (Đoạn văn bản hay còn gọi NLP là một tập hợp lớn và cấu trúc của một văn bản. Ví dụ: sách, bài báo, ghi chép hội thoại...).

- + TF (Term Frequency) là sự đo lường tần suất xuất hiện của một từ trong một văn bản cụ thể nào đó. Đơn giản như là, nếu một từ xuất hiện nhiều lần trong một văn bản, nó có thể là quan trọng hoặc có liên quan đến văn bản đó.

- + IDF (Inverse Document Frequency) là sự giảm trọng số của các từ xuất hiện rất phổ biến trong đoạn văn bản và tăng trọng số cho các từ ít gặp hơn. Mục

đích của nó là để giảm tầm quan trọng của các văn bản và không mang nhiều thông tin đặc biệt của văn bản đó. [1]

Khi kết hợp TF và IDF, Chúng ta sẽ được một số liệu thống kê cho mỗi từ trong mỗi văn bản, phản ánh mức độ quan trọng của từ đó trong văn bản đó, so với toàn bộ một đoạn văn bản. TF-IDF thường được sử dụng trong các hệ thống trích xuất thông tin như máy tìm kiếm, phân loại văn bản,... giúp máy tính xác định được những từ quan trọng trong một văn bản cụ thể.

Ví dụ:

Ví dụ 3 đoạn văn bản:

1. Tôi yêu thích học lập trình.
2. Học lập trình là niềm đam mê của tôi.
3. Tôi thích chơi thể thao.

Văn bản cần phân tích: "Tôi yêu thích lập trình."

Bước 1: Tính TF.

TF của một từ là tần suất của từ đó trong văn bản. Ví dụ, trong văn bản "Tôi yêu thích lập trình," mỗi từ xuất hiện một lần. Vì văn bản có 5 từ, TF của mỗi từ là $1/5$.

Bước 2: Tính IDF.

IDF của một từ là logarit của tỷ lệ giữa tổng số văn bản trong đoạn văn bản và số văn bản chứa từ đó. Giả sử chúng ta sử dụng log cơ số 10.

Ví dụ, xét từ "thích":

Tổng số văn bản trong đoạn văn bản: 3.

Số văn bản chứa từ "thích": 2.

$IDF("thích") = \log_{10}(3/2)$

Bước 3: Tính TF-IDF

$1/5 * \log_{10}(3/2) = 0.034$.

2.6. Độ đo Cosine Similarity.

Phương pháp tính độ tương đồng văn bản dựa trên độ đo Cosine là một phương pháp tương đối đơn giản và hiệu quả. Trong phương pháp này, mỗi văn bản được biểu diễn theo mô hình "túi từ" (bag-of-words). Ý tưởng chính của mô hình này là mô tả một văn bản dưới dạng một túi các từ, không quan tâm đến ngữ pháp và thứ tự của từ. [2]

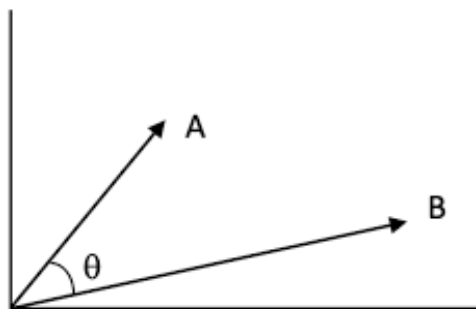
Cụ thể, quá trình biểu diễn văn bản bằng mô hình "bag-of-words" diễn ra như sau:

- + Văn bản được tách thành các từ hoặc cụm từ (n-gram).
- + Từng từ hoặc cụm từ được tính số lần xuất hiện trong văn bản.
- + Đối với tất cả các từ hoặc cụm từ khác nhau trong tập dữ liệu, tạo thành một vector có số chiều bằng số từ hoặc cụm từ đó. Mỗi phần tử trong vector biểu thị số lần xuất hiện của từ hoặc cụm từ tương ứng trong văn bản.

Sau khi hai văn bản đã được biểu diễn thành hai vector \vec{a} và \vec{b} , độ đo Cosine được sử dụng để tính toán độ tương đồng giữa chúng. Công thức tính độ tương đồng Cosine là:

$$\text{Cosine Similarity}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

1. \vec{a} và \vec{b} là hai vec-tơ đại diện cho hai văn bản khác nhau.
2. $\vec{a} \cdot \vec{b}$ là tích vô hướng của hai vec-tơ, được tính bằng cách nhân từng cặp vào phần tử tương ứng của hai vec-tơ và cộng tất cả sản phẩm đó.
3. $\|\vec{a}\|$ và $\|\vec{b}\|$ là chuẩn Euclidean (còn được gọi là độ lớn) của hai vec-tơ, được tính bằng căn bậc hai của tổng các bình phương các từ của vec-tơ.



Hình 3. Hình ảnh minh họa về độ đo Cosine Similarity.

Ví dụ minh họa.

ĐOẠN VĂN BẢN A

ĐOẠN VĂN BẢN B

Hôm nay trời nắng đẹp			Thời tiết hôm nay rất tốt		
Hôm	Nay	Trời	Nắng	Đẹp	Thời tiết

Trong đó:

Chỉ số 1 biểu thị từ : “Hôm”.

Chỉ số 2 biểu thị từ : “Nay”.

Chỉ số 3 biểu thị từ : “Trời”.

Chỉ số 4 biểu thị từ : “Nắng”.

Chỉ số 5 biểu thị từ : “Đẹp”.

Chỉ số 6 biểu thị từ : “Thời tiết”.

+ Từ "rất" và "tốt" đã bị bỏ qua trong ví dụ ban đầu là do mô hình "bag-of-words" (túi từ) không xem xét các từ nối (stop words) như "rất" và "tốt" và thường bỏ qua chúng trong quá trình biểu diễn văn bản. Các từ nối như "rất," "tốt," "và," "cũng," và "nữa" thường không mang nhiều ý nghĩa riêng lẻ trong việc phân tích tương đồng văn bản.

+ Chúng ta sẽ biểu diễn câu 1 thành dạng vector \vec{a} và câu 2 thành dạng vector \vec{b} như sau:

$$\vec{a} : [1 ; 1 ; 1 ; 1 ; 1 ; 0]$$

$$\vec{b} : [1 ; 1 ; 0 ; 0 ; 0 ; 1]$$

1. + Để tính tích vô hướng của hai vec-tơ \vec{a} và \vec{b} chúng ta sử dụng công thức sau:

$$\vec{a} \cdot \vec{b} = (a1 \cdot b1) + (a2 \cdot b2) + (a3 \cdot b3) + (a4 \cdot b4) + (a5 \cdot b5) + (a6 \cdot b6)$$

2. + Trong trường hợp này, các thành phần của vec-tơ \vec{a} và \vec{b} đã được cung cấp. Áp dụng vào ta có:

$$\vec{a} \cdot \vec{b} = (1 \cdot 1) + (1 \cdot 1) + (1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 1)$$

3. + Kết quả của tích vô hướng hai vec-tơ \vec{a} và \vec{b} là:

$$\vec{a} \cdot \vec{b} = (1 \cdot 1) + (1 \cdot 1) + (1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) = 1 + 1 + 0 + 0 + 0 + 0 = 2$$

→ Vậy tích vô hướng của vec- tơ \vec{a} và $\vec{b} = 2$.

+ Độ lớn của vec-tơ \vec{a} và \vec{b} (được ký hiệu là $||\vec{a}||, ||\vec{b}||$) được tính bằng cách lấy căn bậc hai của tổng bình phương các thành phần của vec-tơ \vec{a}, \vec{b} :

vec-tơ \vec{a} :

$$||\vec{a}|| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2} = \sqrt{5}$$

vec-tơ \vec{b} :

$$||\vec{b}|| = \sqrt{1^2 + 1^2 + 0^2 + 0^2 + 0^2 + 1^2} = \sqrt{3}$$

Đối với cách tính Cosine Similarity bằng TF-IDF ta làm như sau:

Bước 1: Tính toán TF:

Bảng 1. Tính toán TF.

Từ	TF văn bản 1	TF văn bản 2
Hôm	$\frac{1}{5}$	$\frac{1}{6}$
nay	$\frac{1}{5}$	$\frac{1}{6}$
trời	$\frac{1}{5}$	$\frac{0}{6}$
nắng	$\frac{1}{5}$	$\frac{0}{6}$
đẹp	$\frac{1}{5}$	$\frac{0}{6}$
Thời	$\frac{0}{5}$	$\frac{1}{6}$
tiết	$\frac{0}{5}$	$\frac{1}{6}$
rất	$\frac{0}{5}$	$\frac{1}{6}$
tốt	$\frac{0}{5}$	$\frac{1}{6}$

Bảng 2. Kết quả TF.

Từ	TF văn bản 1	TF văn bản 2
Hôm	0, 2	0, 16
nay	0, 2	0, 16
trời	0, 2	0
nắng	0, 2	0
đẹp	0, 2	0
Thời	0	0, 16
tiết	0	0, 16
rất	0	0, 16
tốt	0	0, 16

Bước 2: Tính IDF.

Bảng 3. Tính toán IDF.

Từ	IDF
Hôm	$\log(\frac{2}{2}) = 0$
nay	$\log(\frac{2}{2}) = 0$
trời	$\log(\frac{2}{1}) = 0,69$
nắng	$\log(\frac{2}{1}) = 0,69$
đẹp	$\log(\frac{2}{1}) = 0,69$

thời	$\log(\frac{2}{1}) = 0,69$
tiết	$\log(\frac{2}{1}) = 0,69$
rất	$\log(\frac{2}{1}) = 0,69$
tốt	$\log(\frac{2}{1}) = 0,69$

Bước 3: Tính IF-IDF.

Bảng 4. Tính toán IF-IDF.

Từ	TF x IDF văn bản 1	TF x IDF văn bản 2
Hôm	$0,5 * 0 = 0$	$0,16 * 0 = 0$
nay	$0,5 * 0 = 0$	$0,16 * 0 = 0$
trời	$0,2 * 0,69 = 0,138$	$0,16 * 0,69 = 0,11$
nắng	$0,2 * 0,69 = 0,138$	$0 * 0,69 = 0$
đẹp	$0,2 * 0,69 = 0,138$	$0 * 0,69 = 0$
thời	$0 * 0,69 = 0$	$0,16 * 0,69 = 0,11$
tiết	$0 * 0,69 = 0$	$0,16 * 0,69 = 0,11$
rất	$0 * 0,69 = 0$	$0,16 * 0,69 = 0,11$
tốt	$0 * 0,69 = 0$	$0,16 * 0,69 = 0,11$

Bảng 4. Tính toán TF-IDF.

Bước 4 : tính toán Cosine similarity

- Áp dụng công thức tính tử số:

$$(0*0) * (0*0) * (0,138*0,11) * (0,138*0,0) * (0,138*0,0) * (0*0,11) * (0*0,11) * (0*0,11) * (0*0,11) = 0.015.$$

- Áp dụng công thức tính mẫu số:

$$\text{Văn bản 1} : \sqrt{0^2 + 0^2 + 0,138^2 + 0,138^2 + 0,138^2 + 0^2 + 0^2 + 0^2 + 0^2}$$

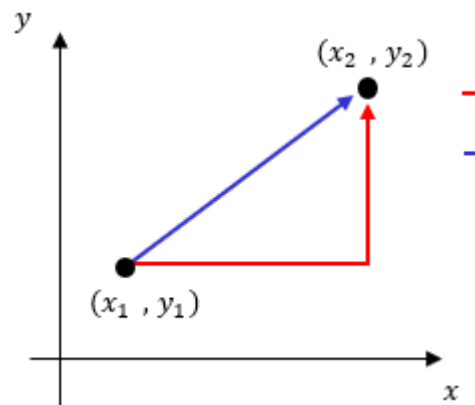
Nhân cho

$$\text{Văn bản 2} : \sqrt{0^2 + 0^2 + 0,11^2 + 0^2 + 0^2 + 0,11^2 + 0,11^2 + 0,11^2 + 0,11^2} \\ = 0.067$$

➤ Vậy ta có $\frac{0.015}{0.067} = 0,22$

2.7. Độ đo Manhattan Distance.

Khoảng cách Manhattan là một dạng khoảng cách giữa hai điểm trong không gian Euclid với hệ tọa độ Descartes. Đại lượng này được tính bằng tổng chiều dài của hình chiếu của đường thẳng nối hai điểm này trong hệ trục tọa độ Descartes. [3]



Hình 4. Ví dụ về Manhattan Distance.

Giả sử 2 đoạn văn bản “Hôm nay trời đẹp lắm” và “Thời tiết hôm nay khá tốt”, Trước hết hãy biểu diễn chúng dưới dạng TF-IDF. Tương tự như ví dụ của Cosine Similarity ở phía trên, đối với Manhattan distance cũng áp dụng.

Sau khi áp dụng công thức Manhattan:

$$(A,B) = \sum_{i=1}^n |a_i - b_i| \\ = |0 - 0| + |0 - 0| + |0.138 - 0,11| + |0.138 - 0,11| + |0.138 - 0| + |0.138 - 0| + |0 - 0.11| + |0 - 0.11| + |0 - 0.11| + |0 - 0.11|. \\ = 0.882$$

Như vậy độ tương đồng Manhattan giữa hai câu trên là 0.882.

2.8. Python là gì ?

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ.

- Công ty mẹ: Python Software Foundation

- Giấy phép: Giấy phép Python Software Foundation
- Kiểm tra kiểu: Vặt, động, mạnh từ phiên bản 3.5: dần dần
- Mẫu hình: Ngôn ngữ lập trình hướng đối tượng, lập trình câu lệnh, lập trình hàm, lập trình thủ tục, lập trình phản xạ
- Phần mở rộng tên tập tin: .py, .pyc, .pyd, .pyo (trước 3.5), .pyw, .pyz (từ 3.5)
- Phiên bản ổn định: 3.12.1 / 7 tháng 12 2023
- Thiết kế bởi: Guido van Rossum. [4]

2.9. Scikit-Learn là gì?

Sklearn tên đầy đủ là Scikit-learn đây là một thư viện mã nguồn mở trong Python được sử dụng rộng rãi cho máy học (machine learning). Nó cung cấp các công cụ hiệu quả cho việc phân tích dữ liệu và xây dựng các mô hình học máy. [5]

2.10. Tại sao lại dùng Sklearn trong đề tài này ?

Bởi vì Sklearn cung cấp một API trực quan và dễ hiểu. Điều này giúp cho việc tính toán độ tương đồng trở nên dễ dàng hơn, đặc biệt là đối với những người mới bắt đầu lập trình hoặc những người không thật sự giỏi về toán học.

1. Sklearn có các công cụ xử lý văn bản như TF-IDF Vectorizer, giúp biến đổi văn bản thành dạng số học, đây là một bước rất quan trọng trong Cosine Similarity và Manhattan Distance.
2. Ngoài Cosine Similarity và Manhattan Distance ra Sklearn cũng hỗ trợ nhiều thuật toán và phép đo khác, cho phép người dùng có thể thêm nhiều lựa chọn khi nghiên cứu và thử nghiệm.
3. Sklearn có một bộ tài liệu rất đầy đủ và chi tiết, hữu ích giúp người dùng có thêm nhiều lựa chọn khi nghiên cứu và thử nghiệm.
4. Sklearn còn có một cộng đồng người dùng và phát triển lớn mạnh, điều này có nghĩa sẽ có rất nhiều tài nguyên, bao gồm các hướng dẫn, ví dụ về mã nguồn và sự hỗ trợ từ cộng đồng khi gặp khó khăn trong quá trình nghiên cứu.

2.11. Flask là gì ?

Flask là một micro framework dành cho Python được sử dụng để xây dựng ứng dụng web. Nó còn được biết đến với sự đơn giản và linh hoạt, cho phép các nhà phát triển tạo ra các ứng dụng web một cách nhanh chóng và hiệu quả. [6]

+ Các tiện lợi khi sử dụng Flask:

1. Đơn giản về dễ dùng
2. Linh hoạt
3. Tích hợp sẵn dễ dàng với các công nghệ khác.
4. Cộng đồng lớn và tài liệu đầy đủ.
5. Phù hợp với các ứng dụng nhỏ đến trung bình.

2.12. HTML, CSS là gì?

+ HTML là viết tắt của "Hypertext Markup Language" trong tiếng Anh, dịch sang tiếng Việt là "Ngôn ngữ Đánh dấu Siêu văn bản". Đây là một ngôn ngữ đánh dấu được sử dụng để tạo và thiết kế các trang web. HTML mô tả cấu trúc của trang web bằng cách sử dụng các thẻ (tags) đánh dấu các phần khác nhau của nội dung trang web.

+ CSS là viết tắt của "Cascading Style Sheets" trong tiếng Anh, dịch sang tiếng Việt là "Bảng kiểu Trãi rộng". CSS là một ngôn ngữ lập trình sử dụng để mô tả cách mà các phần tử HTML sẽ được hiển thị trên trình duyệt. Nó giúp tách biệt phần nội dung (HTML) và phần kiểu dáng (CSS) của một trang web, làm cho mã nguồn trang web trở nên dễ quản lý và bảo trì hơn. [7] [8]

+ Sử dụng HTML trong nghiên cứu lần này là để dễ dàng sử dụng đối với người sử dụng thuật toán.

2.13. JavaScript là gì?

JavaScript (viết tắt là JS) là một ngôn ngữ lập trình dựa trên văn bản (text-based) thường được sử dụng để làm cho trang web trở nên tương tác và động đậy. JavaScript không chỉ được sử dụng trên trình duyệt web để thao tác với các yếu tố HTML và CSS, mà còn được sử dụng trong nhiều môi trường khác nhau như máy chủ thông qua Node.js để xây dựng ứng dụng web và dịch vụ. [9]

2.14. Json là gì?

JavaScript Object Notation (thường được viết tắt là JSON) là một kiểu dữ liệu mở trong JavaScript. Kiểu dữ liệu này bao gồm chủ yếu là văn bản, có thể đọc được theo dạng cặp "thuộc tính - giá trị". Về cấu trúc, nó mô tả một vật thể bằng cách bọc những vật thể con trong vật thể lớn hơn trong dấu ngoặc nhọn ({ }). JSON

là một kiểu dữ liệu trung gian, chủ yếu được dùng để vận chuyển thông tin giữa các thành phần của một chương trình. [10]

2.15. Visual Studio Code là gì ?

Visual Studio Code (thường được gọi là VS Code) là một trình soạn thảo mã nguồn mở và miễn phí được phát triển bởi Microsoft. Nó là một ứng dụng đa nền tảng, có thể chạy trên Windows, macOS và Linux. Visual Studio Code được thiết kế để hỗ trợ nhiều ngôn ngữ lập trình và cung cấp nhiều tính năng hữu ích cho nhà phát triển. [11]

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

❖ **Tất cả các môi trường, thư viện... đã sử dụng trong nghiên cứu lần này:**

1. Python.
2. Visual Studio Code.
3. Flask.
4. Sklearn.
5. HTML,CSS.
6. JavaScript.

Để chạy được bộ mã cần phải có những cài đặt quan trọng và môi trường ảo sau đây:

3.1. Hướng dẫn cài đặt Python.

1. Tải Python:

- Truy cập trang chính thức của Python tại python.org.
- Chọn phiên bản Python muốn cài đặt (đề xuất là phiên bản mới nhất).
- Chọn gói cài đặt phù hợp với hệ điều hành của bạn (*Windows*).

2. Chạy Trình cài đặt:

- Mở tệp tải về (.exe) sau khi đã tải xong.
- Chọn "Add Python to PATH" trước khi bấm "*Install Now*".

3. Cài Đặt:

- Bấm "*Install Now*" để bắt đầu quá trình cài đặt.
- Quá trình cài đặt có thể mất vài phút.

4. Kiểm Tra Cài Đặt:

- Mở Command Prompt hoặc PowerShell và nhập `python --version` hoặc `python -V` để kiểm tra phiên bản Python đã cài đặt.
- Nếu đã hiện: `python version: ...` vậy là đã cài đặt thành công.



3.2. Hướng dẫn cài đặt Visual Studio Code.

1. Tải VSCode:

- Truy cập trang chính thức của Visual Studio Code tại <https://code.visualstudio.com/> và tiến hành tải xuống.
- Nhấp vào nút "Download for Windows" đối với hệ điều hành Windows.

2. Chạy Trình Cài Đặt:

- Mở tệp tải về (.exe) sau khi đã tải xong.
- Làm theo hướng dẫn trên màn hình để hoàn tất quá trình cài đặt.

3. Khởi Chạy VSCode:

- Sau khi cài đặt xong, mở Visual Studio Code từ menu Start hoặc bằng cách tìm kiếm "Visual Studio Code" trong thanh Start.



3.3. Hướng dẫn cài đặt Flask trên VSCode.

- Mở cửa sổ terminal hoặc command prompt và chạy lệnh sau để cài đặt Flask:

```
pip install Flask
```

- Tại giao diện code trong Python sử dụng để nhập thư viện Flask vào:

```
from flask import Flask
```

3.4. Hướng dẫn cài Sklearn trên VSCode.

+ Để cài đặt sklearn trên VSCode, sử dụng môi trường để tạo ra môi trường cài đặt riêng biệt cho dự án.

1. Mở Terminal trong VSCode:

- Mở VSCode và mở một Terminal bằng cách chọn View -> Terminal.

2. Tạo môi trường ảo cho Python:

- Sử dụng lệnh sau đây để tạo một môi trường ảo:

```
python -m venv venv
```

3. Kích hoạt môi trường ảo:

```
.\venv\Scripts\activate
```

4. Cài đặt scikit-learn:

- Khi môi trường ảo đã được kích hoạt, sử dụng lệnh pip để cài đặt Sklearn:

```
pip install scikit-learn
```

Điều này sẽ cài đặt Sklearn và tất cả các gói phụ thuộc của Sklearn.

5. Kiểm tra cài đặt:

- Để kiểm tra xem Sklearn đã được cài đặt thành công hay chưa, có thể sử dụng Python trong Terminal:

```
python
```

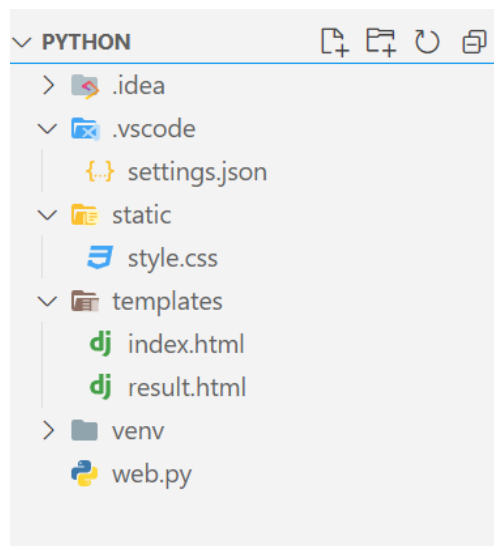
- Sau đó, trong Python, nhập các lệnh sau:

```
import sklearn  
print(sklearn.__version__)
```

- Nếu không có lỗi xuất hiện và phiên bản của Sklearn sẽ được hiển thị, điều đó có nghĩa là đã cài đặt thành công.

3.5. Xây dựng thuật toán đánh giá sự tương đồng của văn bản Cosine và Manhattan hiển thị với giao diện Website.

+ Để xây dựng một Website tính toán độ đo Cosine Similarity và Manhattan distance chúng ta cần có các mục như sau:



.idea: một thư mục được tạo tự động để chứa các thông tin dự án đang mở.

.vscode: chứa các tệp cấu hình của dự án.

Static: chứa file CSS để chỉnh giao diện cho trang web.

Templates: chứa mục index.html là trang chủ chính của trang web, ketqua.html là trang kết quả sau khi tính toán của trang web.

Venv: là môi trường ảo của python.

Web.py: thư mục chứa code chính của dự án.

Hình 5. Thư mục code của nghiên cứu.

+ Đối với nghiên cứu lần này sẽ sử dụng 4 mục chính để tiến hành xây dựng trang web so sánh bao gồm:

1. Web.py: chứa đoạn mã chính của đồ án.
2. Index.html: trang chính của website.
3. Result.html: trang hiển thị kết quả so sánh.

4. Style.css: tùy chỉnh giao diện cho website.

Lưu ý: việc phân tích trong bài báo cáo lần này, sẽ xoay quanh đến các đoạn mã chính và không đề cập nhiều đến các đoạn mã bổ sung.

3.6. Bộ mã trong mục chính web.py.

Vì đây là đoạn mã chính của nghiên cứu lần này, do đó sẽ phân tích kỹ hơn về đoạn mã.

3.6.1. Tất cả các biến và công dụng trong web.py.

Mô tả: đây là mục xử lý chính việc tính toán Cosine, Manhattan và tra cứu các từ giống nhau, khác nhau... được nhận từ trang index.html sau đó sẽ gửi về trang ketqua.html để hiện thị ra kết quả cho người dùng.

Bảng 5. Tất cả các biến của web.py.

TÊN BIẾN	CÔNG DỤNG
TXT1	Lấy nội dung văn bản 1 đã nhập vào bên trang index.html.
TXT2	Lấy nội dung văn bản 2 đã nhập vào bên trang index.html.
TAO_VECTOR	Biến để gán cho phương thức TfidfVectorizer của Sklearn dùng để chuyển đổi các đầu vào văn bản thành các vector TF-IDF.
TFIDF_MATRAN	Để gán ma trận đặc trưng TF-IDF là kết quả của việc tạo tao_vector lên txt1 và txt2.
KQ_CS	Kết quả của việc tính toán độ tương đồng Cosine giữa txt1 và txt2.
DIEM_COSINE	Số điểm cụ thể được trích từ tính toán Cosine của txt1 và txt2.
KC_MHT	Kết quả của việc tính toán khoảng cách độ tương đồng Manhattan giữa txt1 và txt2.

MANHATTAN_DIEM	Số điểm cụ thể được trích từ tính toán Manhattan giữa txt1 và txt2.
TXT_GIONGNHAU	Danh sách các từ xuất hiện trong cả hai văn bản.
TXT1_KHACNHAU	Danh sách các từ chỉ xuất hiện trong văn bản thứ nhất.
TXT2_KHACNHAU	Danh sách các từ chỉ xuất hiện trong văn bản thứ hai.
TXT_DACTRUNG	Danh sách các từ đặc trưng (features: là một phương thức lọc các từ đặc trưng trong python.). Đây là cơ sở để xác định giống nhau và khác nhau giữa hai văn bản.
TU_KIEMTRA	Từ được người dùng nhập từ trang index.html để kiểm tra số lần xuất hiện trong các văn bản.
DEM1	Số lần đếm từ “tu_kiemtra” xuất hiện trong txt1.
DEM2	Số lần đếm từ “tu_kiemtra” xuất hiện trong txt2.
DEM_KQ	Kết quả chứa thông tin về số lần từ “tu_kiemtra” xuất hiện trong cả hai văn bản. Là kết quả cuối cùng khi được trả về dưới dạng JSON để hiển thị cho người dùng.

3.6.2 Toàn bộ code trong web.py.

```
from flask import Flask, render_template, request, jsonify
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity, manhattan_distances

app = Flask(__name__)
```

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/ketqua', methods=['POST'])
def tinhtoan():
    txt1 = request.form['txt1']
    txt2 = request.form['txt2']

    # Tính toán độ đo Cosine và Manhattan
    tao_vector = TfidfVectorizer()
    tfidf_matran = tao_vector.fit_transform([txt1, txt2])

    kq_cs = cosine_similarity(tfidf_matran[0:1], tfidf_matran[1:2])
    diem_cosine = kq_cs[0][0]

    kc_mht = manhattan_distances(tfidf_matran[0:1], tfidf_matran[1:2])
    diem_manhattan = kc_mht[0][0]

    # Tìm từ giống nhau và từ riêng của mỗi văn bản
    txt_giongnhau = []
    txt1_khacnhau = []
    txt2_khacnhau = []

    txt_dacchung = tao_vector.get_feature_names_out()
    for tu in txt_dacchung:
        if tfidf_matran[0, tao_vector.vocabulary_[tu]] > 0 and tfidf_matran[1,
            tao_vector.vocabulary_[tu]] > 0:
            txt_giongnhau.append(tu)
        elif tfidf_matran[0, tao_vector.vocabulary_[tu]] > 0:
            txt1_khacnhau.append(tu)
        elif tfidf_matran[1, tao_vector.vocabulary_[tu]] > 0:
```

```
txt2_khacnhau.append(tu)

# Gửi kết quả về template
return render_template('ketqua.html', kq_cs=diem_cosine,
kc_mht=manhattan_diem, kq_txt1=txt_giongnhau, kq_txt2=txt1_khacnhau,
kq_txt3=txt2_khacnhau)

@app.route('/dem_tu', methods=['POST'])
def dem_tu_xuathien():

# Lấy từ từ form đã nhập
tu_kiemtra = request.json['tu']

# Lấy văn bản về từ js
txt1 = request.json['txt1']
txt2 = request.json['txt2']
dem1 = txt1.count(tu_kiemtra) # Đếm vb 1
dem2 = txt2.count(tu_kiemtra) # Đếm vb 2

dem_kq = f"Số lần xuất hiện của '{tu_kiemtra}':\nVăn bản 1: {
dem1 }\nVăn bản 2: {dem2}\n" # Chuẩn bị kq để chuyển

return jsonify({'dem_kq': dem_kq}) # Gửi kq về json

if __name__ == '__main__':
app.run(debug=True)
```

3.6.3. Phân tích các đoạn code trong web.py.

Nhập các thư viện cần thiết:

```
from flask import Flask, render_template, request, jsonify
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity, manhattan_distances
```

+ **Form flask import Flask, render_template, request, jsonify:** dùng để import Flask framework của python là một điểm khởi đầu cho việc tạo trang web trong python.

+ **Form Sklearn.feature_extraction.text import TfidfVectorizer:** đây dùng để import một class của Sklearn có công dụng là để tính toán TF-IDF của một văn bản.

+ **From Sklearn.metrics.pairwise import cosine_similarity, manhattan_distances:** dùng để import một class của Sklearn để tính độ tương đồng của Cosine và khoảng cách văn bản Manhattan.

Khởi tạo đối tượng Flask:

```
app = Flask(__name__)
```

+ Đây dùng để tạo một đối tượng của lớp Flask đại diện cho ứng dụng Flask đại diện cho Flask để tạo một trang web.

Liên kết với trang Index.html:

```
@app.route('/')
def index():
    return render_template('index.html')
```

+ Đoạn mã này dùng để liên kết với trang index.html sử dụng framework Flask **Def index():** là hàm xử lý cho đường dẫn chính nếu người dùng truy cập vào thì hàm index sẽ được gọi và **render_template('index.html')** là để hiển thị cho trang index.html.

Liên kết với trang Ketqua.html:

```
@app.route('/ketqua', methods=['POST'])
def tinhtoan():
    txt1 = request.form['txt1']
    txt2 = request.form['txt2']
```

+ Tạo một định tuyến route của Flask liên kết với trang ketqua.html và đặt một methods chỉ rằng hàm này sẽ xử lý các yêu cầu của “POST”.

+ **Def tinhtoan():** định nghĩa hàm tính toán sẽ được gọi khi có một yêu cầu từ methods “POST” đến trang ketqua.html.

+ txt1 và txt2 là hai biến dùng để lấy dữ liệu từ form bên trang index.html mà người dùng nhập vào thông qua yêu cầu "POST". Request.form là một đối tượng của Flask chứa dữ liệu từ nhận từ Form.

Chuyển đổi văn bản sang dạng vector TF-IDF:

```
tao_vector = TfidfVectorizer()
tfidf_matran = tao_vector.fit_transform([txt1, txt2])
```

Đây là một hàm dùng để chuyển đổi văn bản sang dạng TF-IDF trong đó:

+ Tao_vector: là một biến được gán dựa trên sự khởi tạo của đối tượng TfidfVectorizer() là một lớp của python để tính toán trọng số TF-IDF của một văn bản.

+ Sau đó tạo một biến Tfidf_matran được dựa trên giá trị của biến tao_vector và sử dụng phương thức fit_transform của đối tượng TfidfVectorizer trong python để nhận vào một danh sách các văn bản (txt1, txt2) và sau đó trả về ma trận TF-IDF. Nói chung nó sẽ học từ vựng từ tập hợp văn bản và tạo một ma trận TF-IDF đã học.

Tính Cosine similarity:

```
kq_cs = cosine_similarity(tfidf_matran[0:1], tfidf_matran[1:2])
diem_cosine = kq_cs[0][0]
```

+ Đoạn mã này sử dụng hàm cosine_similarity của thư viện Sklearn để tính độ tương đồng cosine của hai văn bản sau đó gán vào biến kq_cs.

+ Tfidf_matran[0;1] tượng trưng cho dữ liệu của đoạn văn bản đầu tiên và tfidf_matran[1;2] là dữ liệu của đoạn văn bản thứ hai.

+ kq_cs là kết quả của việc tính toán cosine dùng để chứa giá trị tương đồng cosine giữa txt1 và txt2.

Tính Manhattan distance:

```
kc_mht = manhattan_distances(tfidf_matran[0:1], tfidf_matran[1:2])
diem_manhattan = kc_mht[0][0]
```

+ Tương tự như việc tính toán cosine, Manhattan cũng sử dụng manhattan_distance của Sklearn để tính khoảng cách giữa hai văn bản txt1 và txt2.

+ Và kq_cs và là biến để chứa dữ liệu tính của Manhattan và được

Tìm từ giống nhau:

```
# Tìm từ giống nhau và từ riêng của mỗi văn bản
txt_giongnhau = []
```

```
txt1_khacnhau = []
txt2_khacnhau = []

txt_dactrung = tao_vector.get_feature_names_out()
for tu in txt_dactrung:
    if tfidf_matran[0, tao_vector.vocabulary_[tu]] > 0 and tfidf_matran[1,
    tao_vector.vocabulary_[tu]] > 0:
        txt_giongnhau.append(tu)
    elif tfidf_matran[0, tao_vector.vocabulary_[tu]] > 0:
        txt1_khacnhau.append(tu)
    elif tfidf_matran[1, tao_vector.vocabulary_[tu]] > 0:
        txt2_khacnhau.append(tu)
```

+ Trong bộ mã này là việc tính toán từ riêng và từ giống nhau của hai đoạn văn bản txt1 và txt2. Trong đó nó cần tạo 3 biến rỗng để chứa dữ liệu của các từ giống và khác nhau sau khi được lọc qua hai đoạn văn bản.

+Ban đầu sẽ sử dụng phương thức `get_feature_names_out()` trong lớp Sklearn để lọc các từ đặc trưng của đối tượng `TfidfVectorizer`, và sử dụng vòng lặp `for` để duyệt qua từng từ đặc trưng.

+ Sau đó đặt một số câu điều kiện kiểm tra từ đặc trưng nào xuất hiện trong văn bản nào sẽ gán vào biến tương ứng đã được tạo trước đó.

Ví dụ : “`Elif tfidf_matran[1, tao_vector.vocabulary_[tu]] > 0:`”

`Txt2_khacnhau.append(tu)`”

Kiểm tra từ đặc trưng thông qua “tu” từ vòng lặp `for` tức là kiểm tra giá trị TF-IDF tương ứng với từ đó trong txt2 chỉ lớn hơn 0 ở txt2 thì sẽ được thêm vào danh sách `txt2_khacnhau`.

Chức năng đếm từ:

```
@app.route('/dem_tu', methods=['POST'])
def dem_tu_xuathien():
    # Lấy từ từ form đã nhập
    tu_kiemtra = request.json['tu']
    # Lấy văn bản về từ js
    txt1 = request.json['txt1']
```



```
txt2 = request.json['txt2']
dem1 = txt1.count(tu_kiemtra) # Đếm vb 1
dem2 = txt2.count(tu_kiemtra) # Đếm vb 2
dem_kq = f"Số lần xuất hiện của '{tu_kiemtra}':\nVăn bản 1: {
dem1}\nVăn bản 2: {dem2}\n" # Chuẩn bị kq để chuyển
return jsonify({'dem_kq': dem_kq}) # Gửi kq về json
```

+ Đây là đoạn mã cho việc xử lý đếm từ được gửi về từ trang index.html và sẽ trả về kết quả cho trang index thông qua Json.

+ Lấy từ cần tìm từ dữ liệu của Json gửi từ Client thông qua yêu cầu của methods "POST" sau đó lấy hai văn bản txt1 và txt2 từ dữ liệu kiểm tra bằng request.json.

+ Sau khi đã lấy hai văn bản và từ cần đếm, sử dụng phương thức "count" là một phương thức chuỗi trong Python, và nó dùng để đếm số lần của một chuỗi con. Trong trường hợp này "Tu_kiemtra" là chuỗi con và txt1 hoặc txt2 là chuỗi gốc.

+ Từ việc đếm thông qua phương thức "count" dữ liệu cần đếm sẽ được gán vào hai biến tương ứng dem1 và dem2, sau đó gửi về Json để hiển thị.

3.7. Bộ mã trong mục index.html:

3.7.1. Toàn bộ bộ mã của trang Index.html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Máy tính Cosine và Manhattan</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <h1>Công cụ tính toán Manhattan và Cosine</h1>
```

```
<form action="/ketqua" method="post" onsubmit="return ThôngBao()">
  <label for="txt1">Văn bản 1:</label>
  <textarea name="txt1" id="txt1" rows="4" cols="50" oninput="demSoChu('txt1',
'demChuTxt1')"></textarea>
  <div id="demChuTxt1" style="color: grey">Số chữ: 0</div>
  <br />
  <label for="txt2">Văn bản 2:</label>
  <textarea name="txt2" id="txt2" rows="4" cols="50"
oninput="demSoChu('txt2', 'demChuTxt2')"> </textarea>
  <div id="demChuTxt2" style="color: grey">Số chữ: 0</div>
  <br />
  <button type="submit">Tính toán</button>
  <button type="button" onclick="XoaNoiDung()">Xóa</button>
</form>

<form id="formDemTu">
  <label for="tuNhap">Nhập từ cần đếm:</label>
  <input type="text" id="tuNhap" name="tuNhap" required />
  <button type="button" onclick="demTuXuatHien()">Đếm từ</button>
</form>
<div id="ketQuaDemTu"></div>

<script>
function ThôngBao() {
  var txt1 = document.getElementById("txt1").value;
  var txt2 = document.getElementById("txt2").value;
  if (txt1.trim() === "" || txt2.trim() === "") {
    alert("Vui lòng nhập nội dung cho cả hai văn bản.");
    return false;
  }
  return true;
}
```

```
function XoaNoiDung() {
    document.getElementById("txt1").value = "";
    document.getElementById("txt2").value = "";
}

function demTuXuatHien() {
    var txt1 = document.getElementById("txt1").value;
    var txt2 = document.getElementById("txt2").value;

    if (txt1.trim() === "" || txt2.trim() === "") {
        alert("Vui lòng nhập nội dung cần đếm.");
        return;
    }

    var tuNhap = document.getElementById("tuNhap").value;
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/dem_tu", true);
    xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 200) {
            document.getElementById("ketQuaDemTu").style.display = "block";
            document.getElementById("ketQuaDemTu").innerText = JSON.parse(
                xhr.responseText
            ).dem_kq;
        }
    };
    xhr.send(JSON.stringify({ tu: tuNhap, txt1: txt1,
txt2: txt2 }));
}

function demSoChu(idTextarea, idDemChu) {
    var noiDung =
document.getElementById(idTextarea).value;
```

```
var soTu = noiDung
    .trim()
    .split(/\s+/)
    .filter(function (n) {
        return n != "";
    }).length;
document.getElementById(idDemChu).innerText = "Số từ:
" + soTu;
}
</script>
</body>
</html>
```

3.7.2. Về HTML của trang Index.

Đối với trang index.html chủ yếu là để hiển thị Form hiển thị cho người dùng, để người người dùng nhập văn bản và gửi về trang web.py.

Cài đặt sự kiện cho Form:

```
<form action="/ketqua" method="post" onsubmit="return ThongBao()">
```

+ Sử dụng thuộc tính action để xác định URL mà chuyển đến sau khi người dùng nhấn và button “Tính Toán” trên trang web.

+ Thiết lập method là “POST” cho form để gửi về dữ liệu bên trang web.py.

+ Sự kiện ThongBao() sẽ được hiển thị khi hàm trả về “False” vì cả 2 đoạn văn bản chưa được nhập vào.

Tạo các lable dùng để nhập dữ liệu:

```
<label for="txt1">Văn bản 1:</label>
<textarea name="txt1" id="txt1" rows="4" cols="50" oninput="demSoChu('txt1',
'demChuTxt1')"></textarea>
<div id="demChuTxt1" style="color: grey">Số chữ: 0</div>
<br />
<label for="txt2">Văn bản 2:</label>
<textarea name="txt2" id="txt2" rows="4" cols="50" oninput="demSoChu('txt2',
'demChuTxt2')"></textarea>
<div id="demChuTxt2" style="color: grey">Số chữ: 0</div>
```

```
<br />
<button type="submit">Tính toán</button>
<button type="button" onclick="XoaNoiDung()">Xóa</button>
</form>

<form id="formDemTu">
  <label for="tuNhap">Nhập từ cần đếm:</label>
  <input type="text" id="tuNhap" name="tuNhap" required />
  <button type="button" onclick="demTuXuatHien()">Đếm từ</button>
</form>
<div id="ketQuaDemTu"></div>
```

+ Hai textarea để nhập văn bản với id là "txt1" và "txt2". Mỗi textarea có thêm một div bên dưới để hiển thị số chữ của văn bản tương ứng. Sự kiện oninput của mỗi textarea sẽ gọi hàm demSoChu() để đếm và cập nhật số chữ.

+ Hai button:

- Button đầu tiên có type="submit" để gửi form
- Button thứ hai có onclick gọi hàm XoaNoiDung() để xóa nội dung của các textarea.

+ Phần thứ hai của form với id="formDemTu":

- Một input text để nhập từ cần đếm
- Một button có sự kiện onclick gọi hàm demTuXuatHien() để đếm số lần xuất hiện của từ
- Một div có id="ketQuaDemTu" để hiển thị kết quả đếm từ

+ Tóm lại, form này cho phép người dùng:

- 1) Nhập 2 đoạn văn bản và đếm tổng số chữ
- 2) Xóa nội dung 2 văn bản
- 3) Nhập một từ và đếm số lần xuất hiện của từ đó trong toàn bộ nội dung.
- 4) Các chức năng được thực hiện bằng các hàm JavaScript được gọi từ các sự kiện của button và textarea.

3.7.3. Về JavaScript của trang Index:

Có 3 Function tại hàm này:

- 1) Xoanoidung(): dùng để xóa nội dung đã nhập trên form.
- 2) Demtuxuathien(): dùng để chuẩn bị dữ liệu và gửi về web.py xử lý, sau đó nhận lại dữ liệu là hiển thị lên trang.
- 3) Demsochu(): dùng để đếm số chữ mà người dùng đã nhập ở trên form.

Function xóa nội dung:

```
function XoaNoiDung() {  
    document.getElementById("txt1").value = "";  
    document.getElementById("txt2").value = "";  
}
```

+ Hàm sử dụng document.getElementById() để lấy ra các element dựa vào id và gán giá trị rỗng vào thuộc tính value để xóa nội dung hiện tại của textarea.

+ Xóa nội dung của textarea có id là "txt1", bằng cách gán giá trị rỗng "" cho thuộc tính value của element đó.

+ Tương tự như vậy, xóa nội dung của textarea có id là "txt2".

Function đếm từ:

```
function demTuXuatHien() {  
    var txt1 = document.getElementById("txt1").value;  
    var txt2 = document.getElementById("txt2").value;  
  
    if (txt1.trim() === "" || txt2.trim() === "") {  
        alert("Vui lòng nhập nội dung cần đếm.");  
        return;  
    }  
  
    var tuNhap = document.getElementById("tuNhap").value;  
    var xhr = new XMLHttpRequest();  
    xhr.open("POST", "/dem_tu", true);  
    xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");  
    xhr.onreadystatechange = function () {  
        if (xhr.readyState === 4 && xhr.status === 200) {  
            document.getElementById("ketQuaDemTu").style.display = "block";  
            document.getElementById("ketQuaDemTu").innerText = JSON.parse(  
                xhr.responseText
```

```
    ).dem_kq;  
  }  
};  
  
xhr.send(JSON.stringify({ tu: tuNhap, txt1: txt1, txt2: txt2 }));  
}
```

+ Sử dụng getElementbyID để lấy nội dung của 2 textarea có id là "txt1" và "txt2", gán vào các biến txt1 và txt2.

+ Kiểm tra nếu cả 2 văn bản đều rỗng (chỉ có khoảng trắng) thì hiển thị thông báo yêu cầu nhập nội dung và thoát khỏi hàm bằng lệnh return.

+ Sau đó tạo request POST đến đường dẫn "/dem_tu".

+ Đính kèm dữ liệu cần đếm là các biến txt1, txt2 và tuNhap đã lấy được ở bước 1.

+ Xử lý dữ liệu trả về Json, lấy kết quả đếm số lần xuất hiện từ Json đã xử lý ở web.py và hiển thị ra div #ketQuaDemTu.

Function đếm số chữ:

```
function demSoChu(idTextarea, idDemChu) {  
    var noiDung =  
document.getElementById(idTextarea).value;  
    var soTu = noiDung  
        .trim()  
        .split(/\s+/)  
        .filter(function (n) {  
            return n != "";  
        }).length;  
    document.getElementById(idDemChu).innerText = "Số từ:  
" + soTu;  
}
```

+ Hàm nhận vào 2 tham số:

- idTextarea: id của textarea chứa văn bản cần đếm
- idDemChu: id của element dùng để hiển thị kết quả đếm

+ Các bước thực hiện của hàm:

1. Lấy nội dung trong textarea có id tương ứng, gán vào biến `noiDung`.
2. Loại bỏ khoảng trắng thừa ở đầu và cuối bằng `.trim()`
3. Tách theo khoảng trắng để lấy các từ, gán vào mảng các từ.
4. Lọc ra các từ khác rỗng.
5. Đếm độ dài mảng để biết số từ.
6. Hiển thị kết quả ra element có id tương ứng.

3.8. Bộ mã trong mục Ketqua.html.

3.8.1. Toàn bộ bộ mã của trang Ketqua.html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Kết quả tính toán</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <h1>Kết quả tính toán</h1>
    <div class="giaodien">
      <p>Cosine Similarity: {{ kq_cs }}</p>

      {% if kq_cs > 0.8 %}
      <p>Hai văn bản có độ giống nhau rất cao.</p>
      {% elif kq_cs > 0.5 %}
      <p>Hai văn bản khá giống nhau.</p>
      {% else %}
      <p>Hai văn bản không giống nhau.</p>
      {% endif %}
    </div>
  </body>
</html>
```



```
<p>Manhattan Distance: {{ kc_mht }}</p>

{% if kc_mht < 0.5 %}
<p>Hai văn bản có độ giống nhau rất cao.</p>
{% elif kc_mht < 1.0 %}
<p>Hai văn bản khá giống nhau.</p>
{% else %}
<p>Hai văn bản không giống nhau.</p>
{% endif %}

<p>Từ giống nhau: {{ kq_txt1 }}</p>
<p>Từ khác nhau văn bản 1: {{ kq_txt2 }}</p>
<p>Từ khác nhau văn bản 2: {{ kq_txt3 }}</p>
</div>
</body>
</html>
```

3.8.2. Chức năng của trang Ketqua.

+ Hiển thị giá trị Cosine Similarity (kq_cs) và Manhattan Distance (kc_mht) giữa 2 văn bản.

+ Dựa vào giá trị của kq_cs và kc_mht, đưa ra từ web.py và nhận định mức độ giống nhau giữa 2 văn bản: giống nhau rất cao, khá giống và không giống. Sử dụng cú pháp điều kiện if/elif/else để hiển thị các giá trị liên quan tới từ xuất hiện trong 2 văn bản, bao gồm:

- kq_txt1: Số từ giống nhau.
- kq_txt2: Số từ chỉ có trong văn bản 1.
- kq_txt3: Số từ chỉ có trong văn bản 2.

+ Trang Ketqua sẽ hiển thị các kết quả tính toán độ tương đồng văn bản dưới dạng trực quan, dễ hiểu cho người dùng.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Tổng quan kết quả nghiên cứu.

Sau khi thực hiện nghiên cứu, đã cho thấy rằng kết quả nghiên cứu đề cho đề tài “Tìm hiểu và cài đặt thuật toán đánh giá sự tương đồng của văn bản sử dụng độ đo của Cosine và Manhattan” đưa ra một kết quả rất trực quan như là :

- + Thuật toán đánh giá sử dụng hai độ đo Cosine và Manhattan đã được tìm hiểu và cài đặt thành công. Cả hai đều là phương pháp hiệu quả để đánh giá sự tương đồng giữa các văn bản để áp dụng cho các tình huống thực tế.

- + Độ đo Cosine được xác định bằng việc tính góc giữa hai vector đại diện cho hai văn bản. Kết quả của độ đo Cosine nằm trong khoảng từ -1 đến 1, với 1 chỉ ra sự tương đồng hoàn hảo giữa hai văn bản và ngược lại.

- + Độ đo Manhattan hay được gọi là tính khoảng cách, được tính bằng tổng giá trị tuyệt đối giữa hai vector đại diện cho hai văn bản. Khoảng cách càng nhỏ, hai văn bản càng tương đồng với nhau.

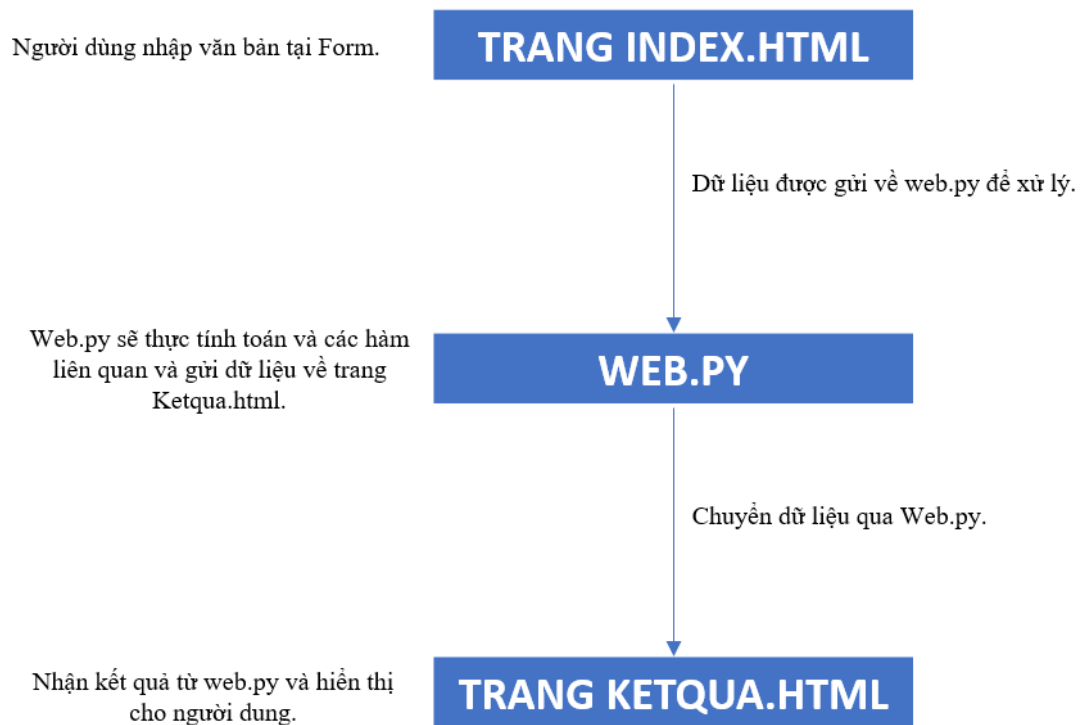
- + Kết quả nghiên cứu cho thấy cả hai phương pháp đều cho ra kết quả đánh giá sự tương đồng văn bản khá tốt. Tuy nhiên, độ đo Cosine thường cho kết quả tốt hơn khi so sánh các văn bản có độ dài khác nhau, trong khi độ đo Manhattan thường cho ra kết quả tốt hơn khi so sánh các văn bản độ dài tương tự, Bởi vì:

1. Độ đo Cosine tính góc giữa hai vector đại diện cho hai văn bản. Nó không quan tâm đến độ dài của vector, mà chỉ quan tâm đến hướng của chúng. Điều này có nghĩa là nếu một văn bản là một phiên bản mở rộng của văn bản khác (ví dụ, bằng cách lặp lại cùng một từ nhiều lần), độ đo Cosine vẫn sẽ cho rằng chúng giống nhau hoàn toàn. Điều này giúp độ đo Cosine hoạt động tốt khi so sánh các văn bản có độ dài khác nhau.
2. Độ đo Manhattan tính tổng giá trị tuyệt đối của sự khác biệt giữa các thành phần tương ứng của hai vector đại diện cho hai văn bản. Nó quan tâm đến độ dài của vector, nghĩa là nếu một văn bản là phiên bản mở rộng của văn bản khác, độ đo Manhattan sẽ cho rằng chúng khác nhau. Điều này giúp độ đo Manhattan hoạt động tốt khi so sánh các văn bản có độ dài tương tự.

+ Cả hai phương pháp đều có thể cài đặt môi trường lập trình Python và thông qua thư viện Sklearn, và có thể triển khai dưới dạng trang Web trong qua Flask một cách mượt mà.

4.2. Mô hình hoạt động của trang web tính toán Cosine và Manhattan.

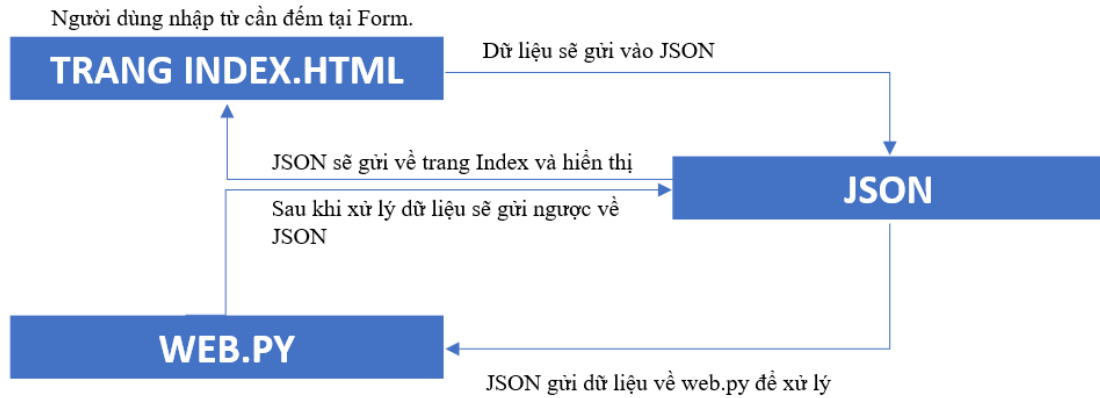
Người dùng sẽ nhập hai văn bản cần đánh giá tại trang Index và sau. Trang Index sẽ gửi về Web.py thực hiện tính toán, sau khi tính toán Web.py sẽ đưa kết quả tính toán đến trang Ketqua để hiển thị cho người dùng.



Hình 6. Mô hình hoạt động của website.

4.3. Mô hình nút đếm từ xuất hiện.

Sau khi người dùng đã nhập hai văn bản và nhập từ cần đếm vào form đếm từ, trang index sẽ gửi dữ liệu đến Json, từ đó JSON sẽ chuẩn bị dữ liệu và gửi đến web.py. Sau khi tính toán từ web.py sẽ gửi dữ liệu về JSON. JSON sẽ hiển thị trực tiếp kết quả trên trang Index.



Hình 7. Mô hình chức năng nút đếm từ xuất hiện.

4.4. Giao diện trang Index.html.

Giao diện hiển thị Form để người dùng nhập hai văn bản cần đếm, và các nút tính toán, xóa. Chức năng đếm từ.

Có sử dụng CSS để tùy chỉnh cho giao diện dễ nhìn và đẹp mắt hơn.

Công cụ tính toán Manhattan và Cosine

Văn bản 1:

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ.

Số từ: 47

Văn bản 2:

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

Số từ: 76

TÍNH TOÁN **XÓA**

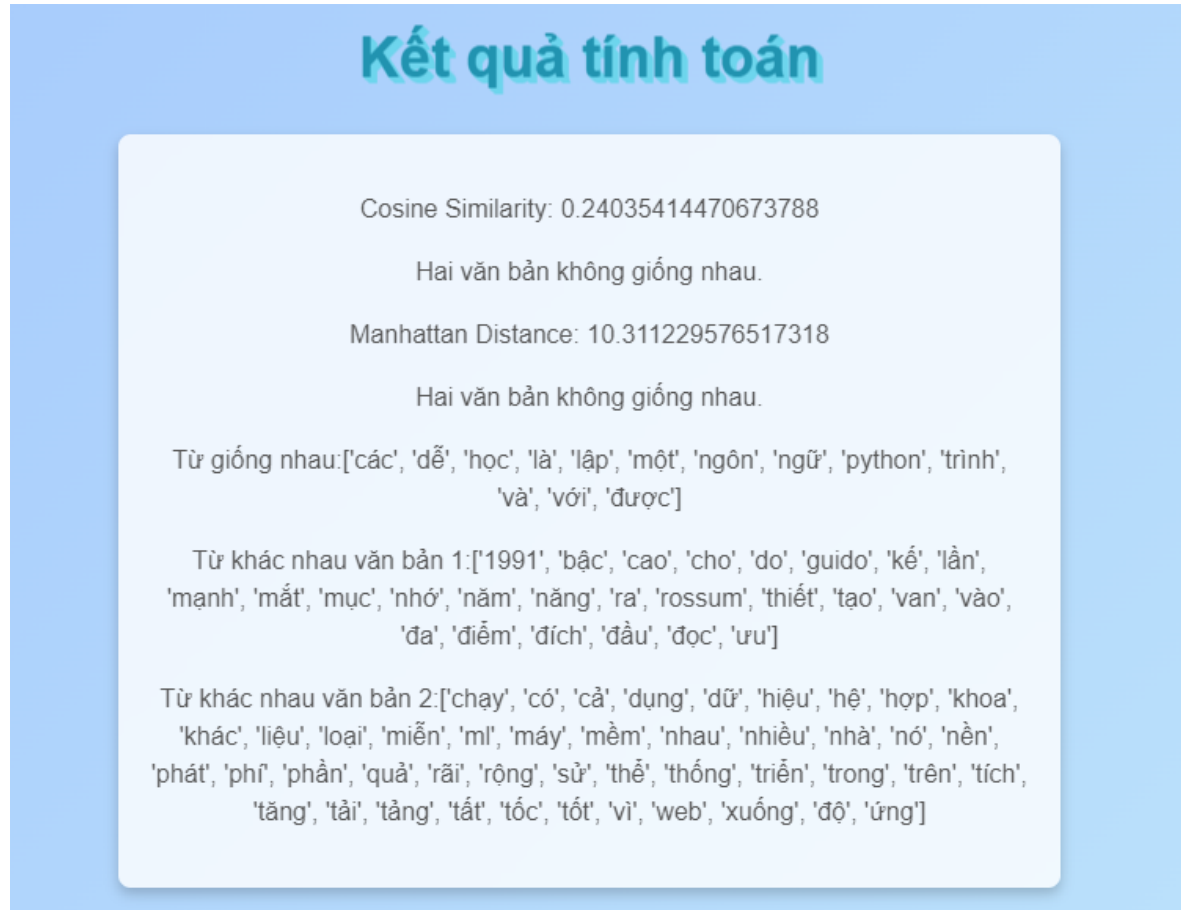
Nhập từ cần đếm:

ĐẾM TỪ

Hình 8. Giao diện trang Index.html.

4.5. Giao diện trang Ketqua.html.

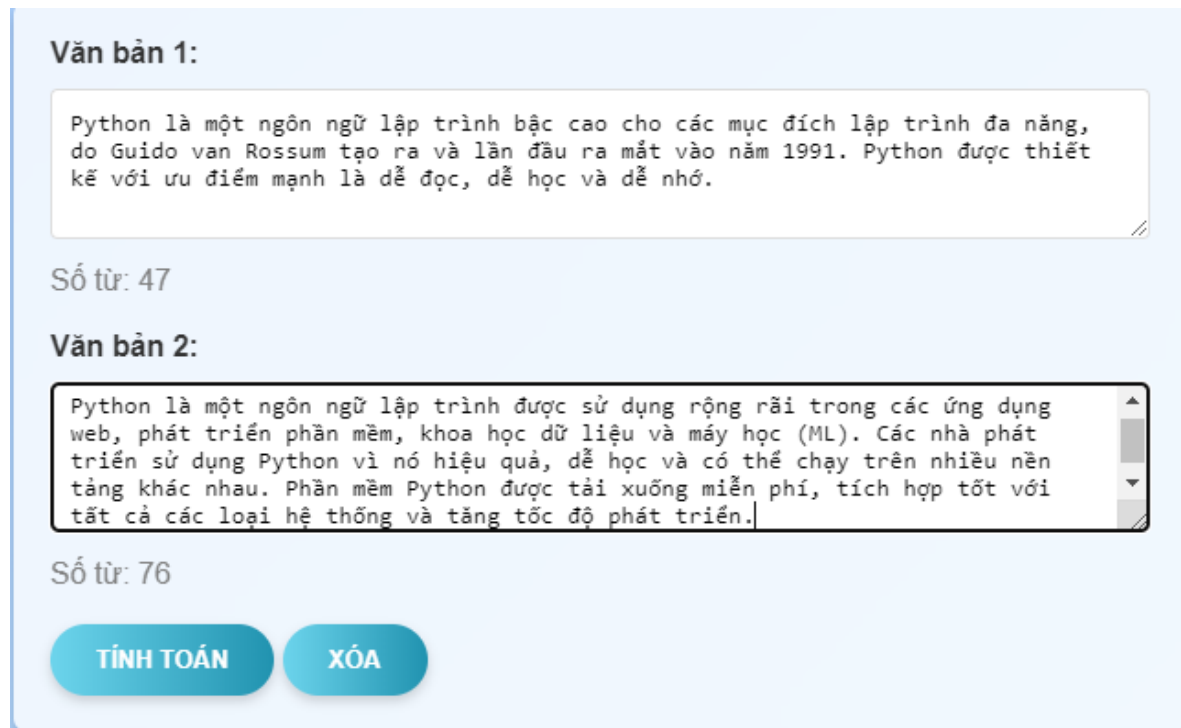
Sau khi người dùng nhấn vào nút tính toán kết quả tính toán của hai đoạn văn bản sẽ được đưa đến trang Ketqua.html và sẽ hiển thị kết quả tính toán, đánh giá, từ giống nhau và khác nhau cho người dùng.



Hình 9. Giao diện trang ketqua.html.

4.6. Giao diện đếm số từ trong Form.

Số từ đã nhập trên Form sẽ hiện thị dưới dạng số ở phía dưới Form.



Văn bản 1:

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ.

Số từ: 47

Văn bản 2:

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

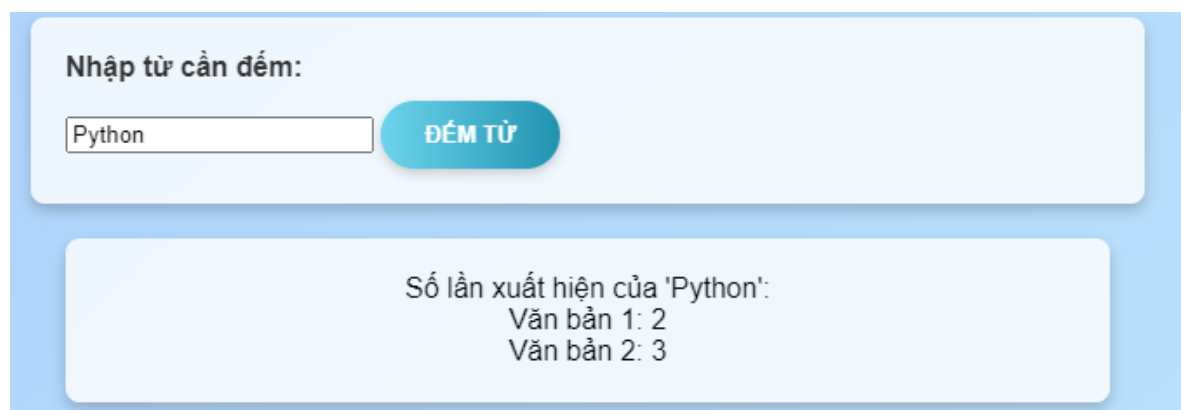
Số từ: 76

TÍNH TOÁN **XÓA**

Hình 10. Giao diện đếm số từ trong Form

4.7. Giao diện đếm từ xuất hiện.

Từ người dùng cần đếm sẽ được nhập tại Form và hiển thị kết quả phía bên dưới.



Nhập từ cần đếm:

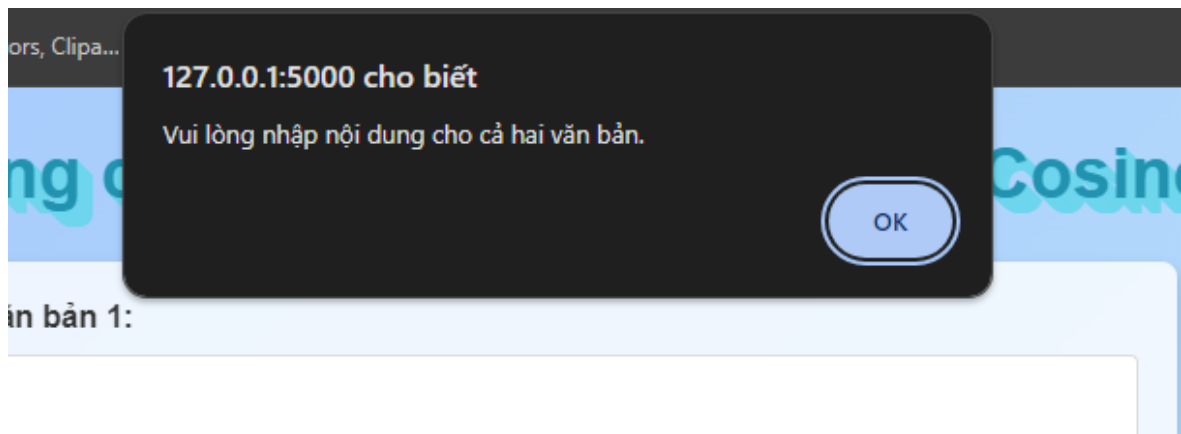
Python **ĐẾM TỪ**

Số lần xuất hiện của 'Python':
Văn bản 1: 2
Văn bản 2: 3

Hình 11. Giao diện đếm từ xuất hiện.

4.8. Thông báo lỗi.

Trang web sẽ hiện thị thông báo lỗi, nếu như người dùng không nhập văn bản vào Form nhưng lại nhấn vào nút tính toán



Hình 12. Thông báo lỗi.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được:

- + Tìm hiểu và cài đặt thành công thuật toán đánh giá sự tương đồng văn bản sử dụng độ đo Cosine và Manhattan.
- + Cả hai phương pháp đều cho ra kết quả đánh giá sự tương đồng văn bản khá tốt, mỗi phương pháp đều có ưu nhược điểm khác nhau.
- + Thực hiện được các thí nghiệm để đánh giá hiệu suất và độ chính xác của hai thuật toán.
- + So sánh và đánh giá được sự tương đồng của hai văn bản
- + Triển khai thành công thuật toán dưới dạng trang web thông qua Flask, giúp dễ sử dụng hơn.
- + Nắm được thêm kiến thức về Python và Cosine, Manhattan. Giúp mở ra tầm nhìn mới về việc so sánh giữa hai văn bản thông qua Python.

5.2. Kết quả chưa đạt được:

- + Có thể chưa tối ưu hóa được thuật toán để xử lý dữ liệu lớn hoặc dữ liệu phức tạp.
- + Chưa khám phá được tất cả các ứng dụng tiềm năng của thuật toán trong các lĩnh vực khác nhau.
- + Chưa thành công trong việc đưa trang web lên Host, Server hoặc Cloud.
- + Chưa nắm vững được việc tính toán bằng tay đối với đoạn văn bản quá lớn hoặc quá dài.

5.3. Hướng phát triển:

- + Đưa trang web lên Host, Server hoặc Cloud để trang web có thể hoạt động 24/7.
- + Cài đặt thêm tính năng giúp sau khi tính toán, các số liệu và dữ liệu liên quan có thể lưu vào cơ sở dữ liệu để lưu trữ.
- + Phát triển thành một công cụ tìm kiếm dùng để tra cứu các website có đoạn văn bản tương tự nhau.
- + Phát triển thành một công cụ đánh giá đạo văn giữa các văn bản và đưa ra nguồn văn bản đã đạo văn.

- + Tích hợp thêm nhiều thuật toán đánh giá như Jaccard, Levenshtein... giúp cho người dùng có thêm nhiều lựa chọn hơn.
- + Phát triển cho trang web không chỉ so sánh hai đoạn văn bản mà còn có thể so sánh nhiều hơn là hai đoạn.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] TF-IDF, "Viblo,":
<https://viblo.asia/p/doi-net-ve-tf-idf-trong-xu-ly-ngon-ngu-tu-nhien-vyDZO0e9lwj>.
- [2] C. Similarity, "geeksforgeeks,":
<https://www.geeksforgeeks.org/cosine-similarity/>.
- [3] Manhattan, "xlinux,":
<https://xlinux.nist.gov/dads/HTML/manhattanDistance.html>.
- [4] Python, "Wikipedia - Python," 1991. [Online]. Available:
[https://vi.wikipedia.org/wiki/Python_\(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_3%ACnh\)](https://vi.wikipedia.org/wiki/Python_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_3%ACnh)).
- [5] Sklearn, "scikit-learn,":
<https://scikit-learn.org/stable/>.
- [6] Flask, "topdev,":
<https://topdev.vn/blog/flask-python-la-gi-nhung-dieu-can-biet/>.
- [7] HTML, "glints,":
<https://glints.com/vn/blog/html-la-gi/>.
- [8] CSS, "topdev,":
<https://topdev.vn/blog/css-la-gi/>.
- [9] JavaScript, "hostinger,":
<https://www.hostinger.vn/huong-dan/javascript-la-gi>.
- [10] JSON, "WIKI,":
<https://vi.wikipedia.org/wiki/JSON>.
- [11] VSCode, "mona,":
<https://mona.media/visual-studio-code-la-gi/>.
- [12] M. h. Vector, "neliti,": <https://media.neliti.com/media/publications/453108-similarity-measurements-of-textual-docum-1a700e8b.pdf>.

