ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

ĐỒ ÁN CUỐI KÌ

SEMANTIC SEARCH ENGINE

Giảng viên hướng dẫn
ThS. Nguyễn Trọng Chỉnh

Thành viên nhóm
Phan Minh Nhật - 19521956
Trương Minh Sơn - 19522143

Lớp
CS221.N11.KHCL

Thành phố Hồ Chí Minh - 2023

Mục lục

1 Giới thiệu

1.1 Sơ lược về đồ án

Đồ án này hướng đến tìm hiểu quy trình và thực hiện một bài toán Search Engine (Công cụ tìm kiếm) nói chung và các bài toán con trong đó nói riêng. Cụ thể hơn là đồ án này sẽ tìm hiểu và thực hiện quy trình đọc và phân tích ngữ pháp của câu truy vấn, cũng như của bộ dữ liệu rồi từ đó trả về kết quả người dùng mong muốn.

1.2 Mục tiêu của đồ án

Để thực hiện được đồ án, chúng tôi đã đặt ra những mục tiêu như sau:

- Hiểu và nắm rõ ý tưởng cũng như vấn đề cần giải quyết của các bài toán con.
- Huấn luyện và đánh giá được một số mô hình con trong pipeline (S-BERT và FAISS). Từ đó rút ra được kết luận và nguyên nhân cho kết quả đó để phục vụ cho công việc cải tiến sau này.
- Xây dựng được ứng dụng demo Semntic Search Engine trong thực tế.

1.3 Ngữ cảnh bài toán

Nhờ có sự phát triển của công nghệ, mỗi ngày trôi qua luôn có rất nhiều thông tin được đăng tải và cập nhật thường xuyên trên các phương tiện thông tin, truyền thông. Với số lượng đồ sộ như vậy, sẽ rất khó khăn cho người dùng có thể tiếp cận với thông tin mình mong muốn mà không phải tìm kiếm tất cả lượng thông tin đó.
Để giải quyết vấn đề này, các tập đoàn công nghệ lớn đã và đang nghiên cứu và phát triển các hệ thống tìm kiếm thông tin. Khi người dùng thực hiện tìm kiếm, 1 công cụ tìm kiếm sẽ tìm trên mọi trang thông tin có trên Internet, bao gồm tựa đề (Title), nội dung (Content) và từ khoá (Keyword) liên quan đến câu truy vấn và dùng các thuật toán để trả về 1 danh sách chứa những kết quả tìm kiếm có độ liên quan cao nhất.
Có rất nhiều công cụ tìm kiếm với những phương pháp tìm kiếm khác nhau nhưng tất cả đều có điểm chung là nhằm tối ưu hoá trải nghiệm người dùng. Công cụ tìm kiếm dựa trên ngữ nghĩa (Semantic Search Engine) là một trong số đó.

Input/Output bài toán

- Input: Một câu Tiếng Anh bất kỳ, không có ký tự đặc biệt và có thê viết hoa tuỳ ý.
- Output: Top n (tuỳ ý người dùng) kết quả gần với câu Input ở trên.

Hình 1: Hình minh họa cho output của bài toán

1.4 Thách thức bài toán

- Bài toán còn gặp nhiều vấn đề nan giải như thiếu hụt dữ liệu, chương trình vẫn chưa được tối ưu,...
- Trong nhiều trường hợp, chương trình sẽ hiểu sai nghĩa của từ. Ví dụ như những từ đồng nghĩa, ẩn dụ, những câu thành ngữ, ...; cũng như không thể tiếp nhận ký tự đặc biệt, viết tắt.

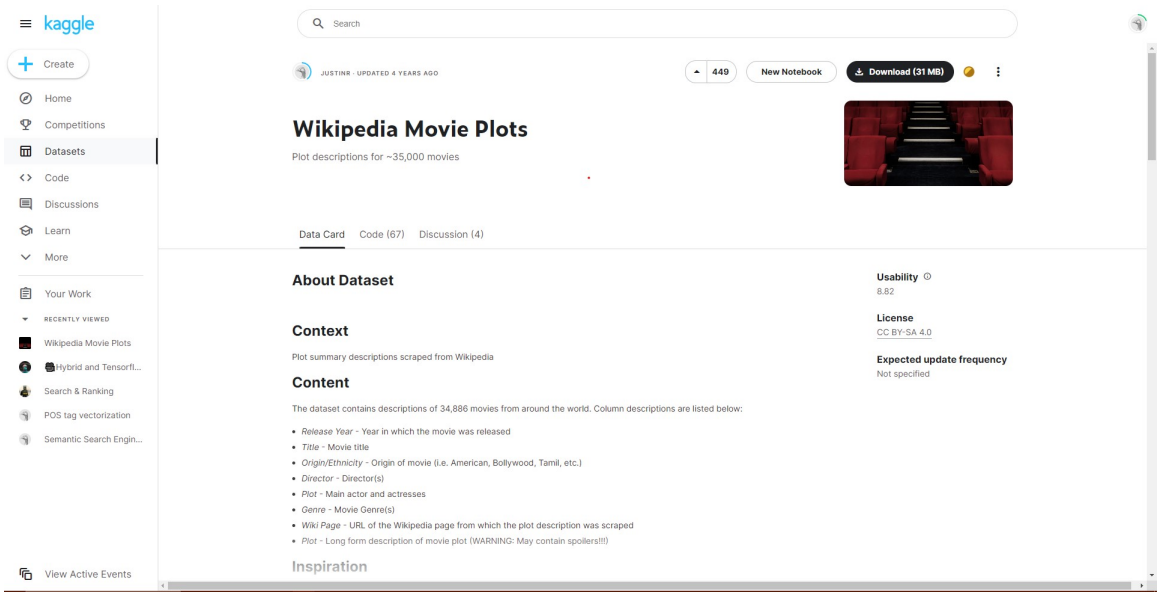
Mặc dù những khó khăn này, các thuật toán deep learning ngày nay hoạt động tương đối tốt và càng phát triển.

2 Bộ dữ liệu

Sử dụng bộ dữ liệu Wikipedia Movie Plots [1] chứa thông tin của hơn 35,000 bộ phim đã được trích xuất trực tiếp từ Wikipedia.
Cụ thể hơn, bộ dữ liệu này chứa miêu tả của 34,886 bộ phim trên khắp thế giới. Trong bộ dữ liệu này bao gồm:

- Realese year - năm bộ phim đó phát hành.

- Title - Tên bộ phim.
- Origin/Ethnicity - Nguồn gốc của bộ phim. (như Mỹ, Ấn Độ, Hàn Quốc,...)
- Director - Tên đạo diễn
- Cast - Tên các diễn viên chính.
- Genre - Thể loại của bộ phim
- Wiki Page - Đường dẫn đến trang Wikipedia của bộ phim
- Plot - Miêu tả sơ lược, hoặc tóm tắt, của bộ phim.



Hình 2: Wikipedia Movie Plots

3 Giới thiệu về Semantic Search

"Semantic search is a data searching technique in which a search query aims to not only find keywords but to determine the intent and contextual meaning of the words a person is using for a search." - *Bloomberg*

3.1 Semantic là gì ?

Theo Wikipedia, Ngữ nghĩa - Semantic là là nghiên cứu về quy chiếu, ý nghĩa hoặc sự thật. Thuật ngữ này có thể được sử dụng để chỉ các lĩnh vực con của một số ngành khác nhau, bao gồm triết học, ngôn ngữ học, ... Tuy thuật ngữ này thường khá ít khi được nhắc đến ở mảng Công nghệ thông tin, nhưng khái niệm không phải là không có liên quan. Và cụ thể ở đây là liên quan đến **tìm kiếm theo ngữ nghĩa - Semantic Search**.

Sự khác biệt của **Keyword** và **Semantic**

Keyword	Semantic
Từ đồng nghĩa có thể bị bỏ qua trong quá trình tìm kiếm	Kết hợp ý nghĩa của các từ để hiểu các từ đồng nghĩa
Cần lựa chọn cẩn thận các từ khóa để tìm kiếm	Truy vấn được tự động cải thiện bằng mã hoá
Thông tin được truy xuất phụ thuộc vào từ khóa và có thể sẽ trả về kết quả spam	Thông tin được truy xuất không phụ thuộc vào từ khóa và sẽ trả về kết quả chính xác thay vì bất kỳ kết quả không liên quan nào

3.2 Semantic Search

Semantic Search (Tìm kiếm ngữ nghĩa) áp dụng mục đích, ngữ cảnh và ý nghĩa khái niệm của người dùng để khớp truy vấn của người dùng với nội dung tương ứng. Điều này đồng nghĩa với việc, các công cục tìm kiếm khác (Google, . . .) không nhất thiết phải trích xuất từ khoá tương đương để cho ra kết quả.
Tìm kiếm ngữ nghĩa mang lại sự hiểu biết nâng cao về ý định của người tìm kiếm, khả năng trích xuất câu trả lời và mang lại nhiều kết quả được cá nhân hóa hơn.
So với tìm kiếm theo keyword, tìm kiếm theo ngữ nghĩa cải thiện độ chính xác của tìm kiếm bằng cách hiểu ý định của người tìm kiếm và ý nghĩa ngữ cảnh để cung cấp các kết quả có liên quan, được cá nhân hóa.
Tìm kiếm theo ngữ nghĩa cũng cho phép Google phân biệt giữa các thực thể khác nhau (người, địa điểm và sự vật) và diễn giải ý định của người tìm kiếm dựa trên nhiều yếu tố bao gồm:

1. Lịch sử tìm kiếm của người dùng
2. Vị trí của người dùng
3. Lịch sử tìm kiếm toàn cầu
4. Các biến thể chính tả.

Khác biệt giữa tìm kiếm theo từ khoá và ngữ nghĩa:

Mặc dù các công cụ tìm kiếm từ khóa cũng đưa vào quá trình xử lý ngôn ngữ tự nhiên để cải thiện việc khớp từ này sang từ khác – thông qua các phương pháp như sử dụng từ đồng nghĩa, loại bỏ stop words, bỏ qua số nhiều – quá trình xử lý đó vẫn dựa vào việc kết hợp từ với từ. Tuy nhiên, tìm kiếm ngữ nghĩa có thể trả về kết quả không có văn bản phù hợp, nhưng bất kỳ ai có kiến thức về miền đều có thể thấy rằng có những kết quả phù hợp rõ ràng.

VD: "*Soap*" sẽ luôn trả về kết quả là "*soap*" hoặc "*soapy*" do có sự trùng lặp về cấu trúc từ. Tuy nhiên, trong trường hợp bị thiếu dữ kiện, kết quả sẽ trả về là "*soap*". Ở trường hợp còn lại, kết quả trả về có thể sẽ kèm theo "*detergent*", vì "*soap*" và "*detergent*" có ý nghĩa tương đương nhau và trong trường hợp đó, công cụ tìm kiếm sẽ "giả vờ" rằng chất tẩy rửa thực sự là xà phòng khi nó đang xác định sự giống nhau.

3.3 Nguyên lý hoạt động

Semantic Search sử dụng **Vector Search** và **Machine Learning** để trả về kết quả nhằm khớp với truy vấn của người dùng, ngay cả khi không có từ nào khớp.

Các thành phần này hoạt động cùng nhau để truy xuất và xếp hạng kết quả dựa trên ý nghĩa.

Một trong những phần cơ bản nhất là ngữ cảnh.

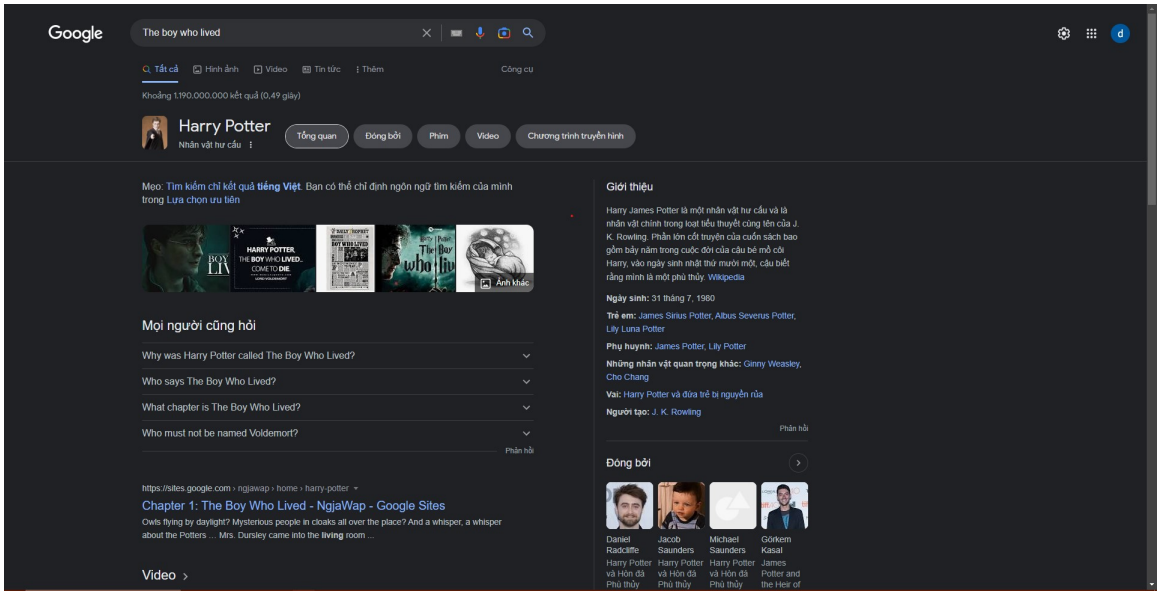
Đây là 1 phần rất quan trọng trong việc tìm hiểu những gì người tìm kiếm đang cố gắng tìm. 1 công cụ tìm kiếm sẽ sử dụng ngữ cảnh theo 2 cấp độ: theo cá nhân và theo nhóm

Cấp độ cá nhân (cá nhân hoá) sẽ sử dụng mối quan hệ của từng người tìm kiếm, các tìm kiếm trước đó và các tương tác trước đó để trả về nội dung phù hợp nhất với truy vấn hiện tại. Ở cấp độ nhóm, công cụ tìm kiếm có thể xếp hạng lại kết quả bằng cách sử dụng thông tin về cách tất cả người tìm kiếm tương tác với kết quả tìm kiếm, chẳng hạn như kết quả nào được nhấp vào thường xuyên nhất hoặc thậm chí tính thời vụ khi một số kết quả nhất định phổ biến hơn những kết quả khác.

Ngoài ra, tìm kiếm ngữ nghĩa cũng có thể tận dụng ngữ cảnh trong văn bản.

Như ta đã biết, các từ đồng nghĩa có thể hữu dụng trong cái search engine và có thể cải thiện tìm kiếm từ khóa bằng cách mở rộng kết quả khớp cho các truy vấn đến nội dung có liên quan. Tuy nhiên sẽ có những trường hợp hai từ tương đương nhau trong ngữ cảnh này nhưng lại không tương đương trong ngữ cảnh khác. Một ví dụ thường thấy là hai từ *soccer* và *football* đa số ở mọi nơi nghĩa 2 từ là như nhau,đều chỉ *[*bóng đá], tuy nhiên ở Mỹ, từ *football* mang nghĩa là *bóng bầu dục* và *socccer* mới là *bóng đá*. Điều này làm ảnh hưởng đến kết qả trả về tuỳ vào khu vực tìm kiếm.

Thứ hai là ý định của người dùng. Mục tiêu cuối cùng của bất kỳ công cụ tìm kiếm nào là giúp người dùng hoàn thành thành công một nhiệm vụ. Nhiệm vụ đó có thể là đọc các bài báo, mua quần áo hoặc tìm tài liệu. Công cụ tìm kiếm cần tìm ra những gì người dùng muốn làm hoặc ý định của người dùng là gì. Chúng ta có thể thấy điều này khi tìm kiếm trên một trang web thương mại điện tử. VD: Nếu tìm “The boy who lived” trên google sẽ hiển thị kết quả là “Harry Potter” - cụ thể hơn là chapter 1 của bộ truyện dù trong câu tìm kiếm không có nhắc gì đến “Harry Potter”.



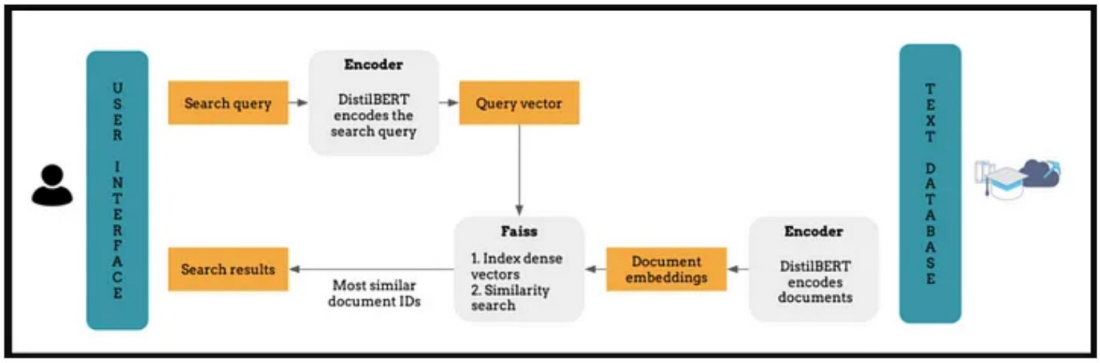
Hình 3: Hình ảnh minh họa cho ví dụ tìm kiếm "The boy who lived"

Bằng cách đi trước ý định của người dùng, công cụ tìm kiếm có thể trả về các kết quả phù hợp

nhất và không làm người dùng mất tập trung với các mục khớp về mặt văn bản nhưng không liên quan. Điều này có thể phù hợp hơn tất cả khi áp dụng một sắp xếp ở đầu tìm kiếm, chẳng hạn như giá từ thấp nhất đến cao nhất. Phân loại truy vấn và giới hạn tập hợp kết quả sẽ đảm bảo rằng chỉ những kết quả có liên quan mới xuất hiện.

4 Mô hình

4.1 Tổng quan mô hình



Hình 4: Hình minh họa cho workflow của mô hình

Workflow của mô hình bao gồm các giai đoạn chính sau:

1. Xử lý văn bản bằng các thư viện NLP.
2. Thực hiện vector hoá chuỗi văn bản từ bộ dữ liệu và câu truy vấn bằng mô hình S-BERT.
3. Lưu các vector trên vào bộ lưu trữ FAISS.
4. Thực hiện tính toán và trả về những kết quả tương đương với câu truy vấn.

4.2 Tiền xử lý dữ liệu

Đây là công đoạn xử lý dữ liệu bằng thư viện **NLTK**.

1. Tách từ (*Word-based tokenization*) 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. Dấu phân cách hay được dùng chính là dấu cách trắng. Tuy nhiên, cũng có thể tách văn bản không theo dấu phân cách. Ví dụ tách từ trong tiếng Việt vì một từ trong tiếng Việt có thể chứa 2 hoặc 3 âm tiết được nối với nhau bởi dấu cách trắng.

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.

Ví Dụ: *A bartender is working at a saloon, serving drinks to customers.* ==> ['A', 'bartender', 'is', 'working', 'at', 'a' ,'saloon', ',', 'serving', 'drinks', 'to', 'customers', '.']

2. Loại bỏ các từ không quan trọng và không ảnh hưởng đến nghĩa của từ và của câu (**Removing stopword**). (vd a, an, the, ...)

   Ví dụ: ['A', 'bartender', 'is', 'working', 'at', 'a' ,'saloon', ',', 'serving', 'drinks', 'to', 'customers', '.'] ==> ['A', 'bartender', 'working', 'saloon', ',', 'serving', 'drinks', 'customers', '.']

3. Biến đổi những từ đã tách về dạng gốc (**Stemming**) bằng cách cực kỳ đơn giản là loại bỏ 1 số ký tự nằm ở cuối từ mà nó nghĩ rằng là biến thể của từ. Ví dụ như chúng ta thấy các từ như **walked**, **walking**, **walks** chỉ khác nhau là ở những ký tự cuối cùng, bằng cách bỏ đi các hậu tố **-ed**, **-ing** hoặc **-s**, chúng ta sẽ được từ nguyên gốc là walk.

   Ví dụ: ['A', 'bartender', 'working', 'saloon', ',', 'serving', 'drinks', 'customers', '.'] ==> ['a', 'bartend', 'work', 'saloon', ',', 'serv', 'drink', 'custom', '.']

4. Cũng là 1 phương pháp biến đổi về dạng gốc nhưng thay vì thay đổi những chữ cuối cùng. Nhưng **Lemmatization** sẽ xử lý thông minh hơn bằng một bộ từ điển hoặc một bộ ontology nào đó. Điều này sẽ đảm bảo rằng các từ như "**goes**", "**went**" và "**go**" sẽ chắc chắn có kết quả trả về là như nhau. Kể các từ danh từ như **mouse**, **mice** cũng đều được đưa về cùng một dạng như nhau.

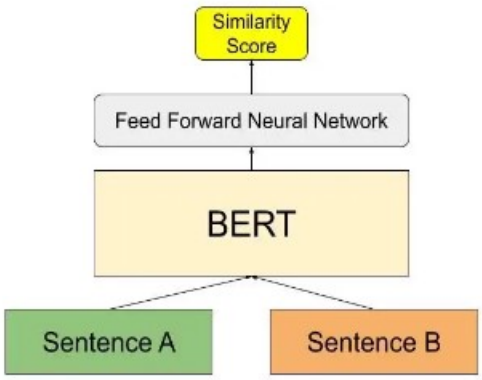
   Ví dụ: ['a', 'bartend', 'work', 'saloon', ',', 'serv', 'drink', 'custom', '.'] ==> ['a', 'bartend', 'work', 'saloon', ',', 'serv', 'drink', 'custom', '.']

4.3 Sentence-Bert

Hiện nay có rất nhiều transformer được sử dụng cho các bài toán NLP như question answering, language modeling và summarization. Một số transformer để giải quyết các bài toán này như BERT, RoBERTa,..., chúng giải quyết các bài toán trên bằng cách tính toán embedding ở cấp độ từ. Tuy nhiên đối với bài toán như semantic search, yêu cầu phải hiểu rõ được văn bản ở cấp độ câu, việc dùng word-level transformer để tính toán cho bài toán semantic search là không khả thi.

Trước khi tìm hiểu về Sentence-BERT (S-BERT), ta sẽ tìm hiểu qua về cách sử dụng BERT để giải quyết Semantic Search và lý do tại sao ta không dùng BERT cho bài toán này.

Để giải quyết Semantic Search, BERT sử dụng cấu trúc cross-encoder: đưa hai câu vào mô hình BERT và tính Similarity Score của hai câu đó. Dưới đây là cross-encoder network của BERT:



Hình 5: Hình minh họa cho Cross-encoder của BERT

BERT cross-encoder bao gồm model BERT tiêu chuẩn nhận vào hai input là hai câu A và B, chúng được phân tách nhau bởi [SEP] token. Ở trên mô hình BERT ta có Feed Forward Layer để có thể tính Similarity Score.

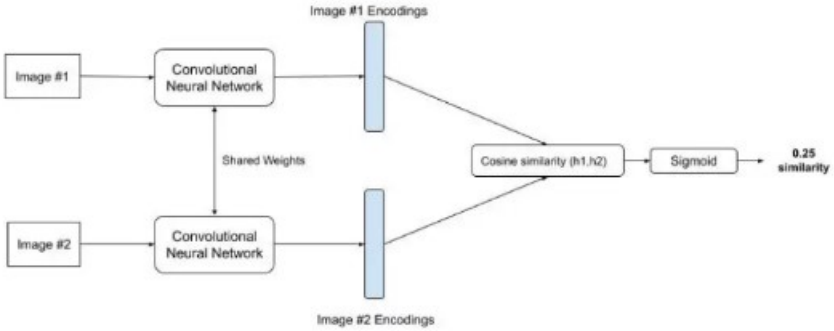
Tuy nhiên ở mô hình này, để so sánh toàn bộ các câu trong dữ liệu với nhau, ta sẽ cần tính toán (n)(n-1)/2 lần (với n là tổng số câu trong dữ liệu). Vậy nên khi ta có một bộ dữ liệu quá lớn, ta sẽ mất rất nhiều thời gian để huấn luyện mô hình.

Để giải quyết vấn đề này, các nhà nghiên cứu đã sử dụng BERT để tạo ra một mô hình sentence embedding. Cách thường thấy nhất là đưa từng câu vào BERT, và vì BERT tính toán word-level embedding nên mỗi từ trong câu sẽ có một embedding riêng, cách thông dụng nhất để tạo ra sentence embedding là tính trung bình tất cả word embedding hoặc sử dụng output của token đầu tiên ([CLS] token). Nhưng phương pháp này thường cho ra kết quả không tốt bằng GLoVE [**5**] (Global Vectors for Word Representation) embeddings.

Để có S-BERT, người ta đã fine-tune BERT sử dụng siamese và triplet network để cập nhật trọng số sao cho sentence embedding được tạo ra có ý nghĩa về mặt ngữ nghĩa và có thể so sánh được bằng cosine-similarity.

4.3.1 Siamese networks

Siamese neural networks là các mạng đặc biệt chứa hai hoặc nhiều hơn các mạng con/mô hình giống hệt nhau, có cùng tham số/trọng số. Khi cập nhật tham số/trọng số thì sẽ cập nhật giống nhau cho các mô hình con. Siamese networks thường được dùng để tính similarity score giữa các input.



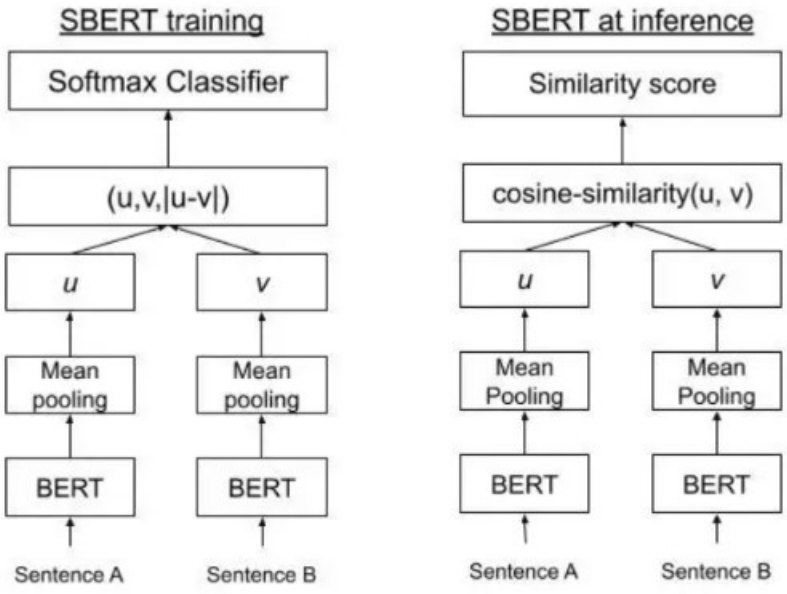
Hình 6: Hình minh họa cho Siamese network

4.3.2 Triplet loss

S-BERT dùng triplet loss để train kiến trúc siamese của nó. Để có thể hiểu hơn về triplet loss, ta sẽ lấy một ví dụ về bài toán tìm kiếm ảnh tương đồng. Đầu tiên mạng sẽ chọn một anchor image. Sau đó tìm thêm một cặp ảnh positive (một ảnh cùng lớp) và cặp ảnh negative (một ảnh khác lớp với anchor image). Tính similarity score giữa hai cặp ảnh này và dùng nó để tính loss và cập nhật lại trọng số (backpropagation). Tất cả ảnh trong dữ liệu sẽ được thực hiện tương tự, việc này giảm được chi phí tính toán thay vì phải so sánh từng cặp ảnh có thể có trong bộ dữ liệu. Tác giả của SBERT dùng mô hình tương tự như trên, nhưng khác với phân loại ảnh, semantic search không phân lớp. Tác giả dùng một kĩ thuật để tìm và minimizing khoảng cách euclidean giữa các câu và dùng metric này để xem xét cặp nào là positive, cặp nào là negative.

4.3.3 Kiến trúc S-BERT

Nếu nhìn qua kiến trúc cross-encoder của BERT, S-BERT có kiến trúc tương tự như vậy nhưng loại bỏ lớp classification cuối cùng. Không giống như BERT, S-BERT dùng kiến trúc siamese như đã nói ở trên, nó chứa hai kiến trúc BERT giống hệt nhau, chia sẻ cùng trọng số và SBERT sẽ xử lý hai câu theo cặp trong quá trình training. Giả sử ta đưa câu A vào BERT A và câu B vào BERT B. Mỗi BERT sẽ cho ra output là pooled sentence embedding. Trong paper của tác giả, họ đưa ra nhiều phương pháp pooling, nhưng họ nhận thấy mean-pooling là cách tiếp cận tốt nhất. Sau khi pooling, ta đã có 2 embedding, 1 cho câu A và 1 cho câu B. Khi model training, S-BERT ghép 2 embedding này với nhau và đưa qua softmax classifier và được train sử dụng hàm softmax-loss. Khi dự đoán, hai embedding sẽ được so sánh bằng cosine-similarity.

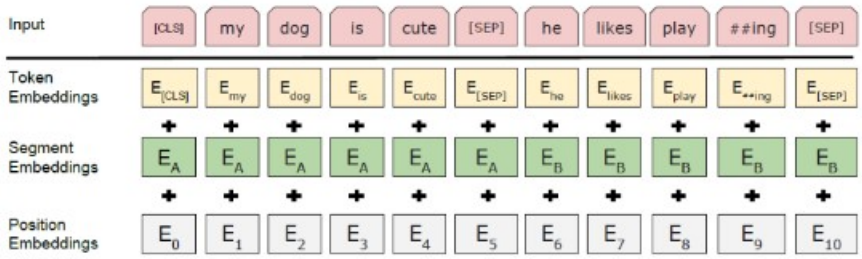


Hình 7: Kiến trúc của S-BERT

4.3.4 Cách S-BERT hoat động

Để cho máy tính có thể hiểu được sự quan trọng về mặt ngữ nghĩa của câu mà chúng ta đưa vào, ta cần phải chuyển câu thành vector, khi đó câu đã được biểu diễn bởi những con số và dễ dàng để tính toán hơn. Để làm được điều đó, đầu tiên ta cần phải tách các từ trong câu (tokenize). Ở đây sẽ có hai token đặc biệt là CLS và SEP, ta sẽ tìm hiểu về 2 token này trước:

- CLS token: viết tắt của classification, nó được thêm vào để thể hiện sự phân loại ở cấp độ câu. Nó được thêm vào đầu câu để đại diện cho toàn bộ câu.
- SEP token: là một token dùng để phân tách giúp model biết được đâu là câu tiếp theo. Trong trường hợp là câu đơn, SEP token sẽ nằm ở cuối câu.



Hình 8: Hình minh họa cách S-BERT thực hiện sentence embbeding

Bert dùng WordPiece để tách từ, các từ không có trong BERT Vocabulary (khoảng 30K tokens) sẽ được tách thành những từ con nhỏ hơn. WordPiece sẽ thêm vào trước các từ được tách ##.

Để có được embedded vector cho một từ, ta sẽ cần kết hợp Token Embedding, Segment Embedding và Position Embedding của từ đó.

- Token Embedding: Sau khi tách từ xong, ta sẽ ánh xạ các token vào Vocabulary để có được các chỉ mục của token đó.

[CLS]	101
this	2,023
is	2,003
a	1,037
sentence	6,251
i	1,045
want	2,215
to	2,000
em	7,861
##bed	8,270
##ding	4,667
[SEP]	102

Hình 9: Ví dụ cho Token embedding

- Segment Embedding: S-BERT cần biết được token này thuộc về câu nào. Giả sử ta có câu: I LIKE CATS, I LIKE DOGS, các từ thuộc câu đầu tiên sẽ được gán cho vector có giá trị 0, các từ thuộc câu thứ hai sẽ được gán cho vector có giá trị 1.

[CLS]	I	LIKE	CATS	[SEP]	I	LIKE	DOGS
0	0	0	0	0	1	1	1

Hình 10: Ví dụ cho Segment embedding

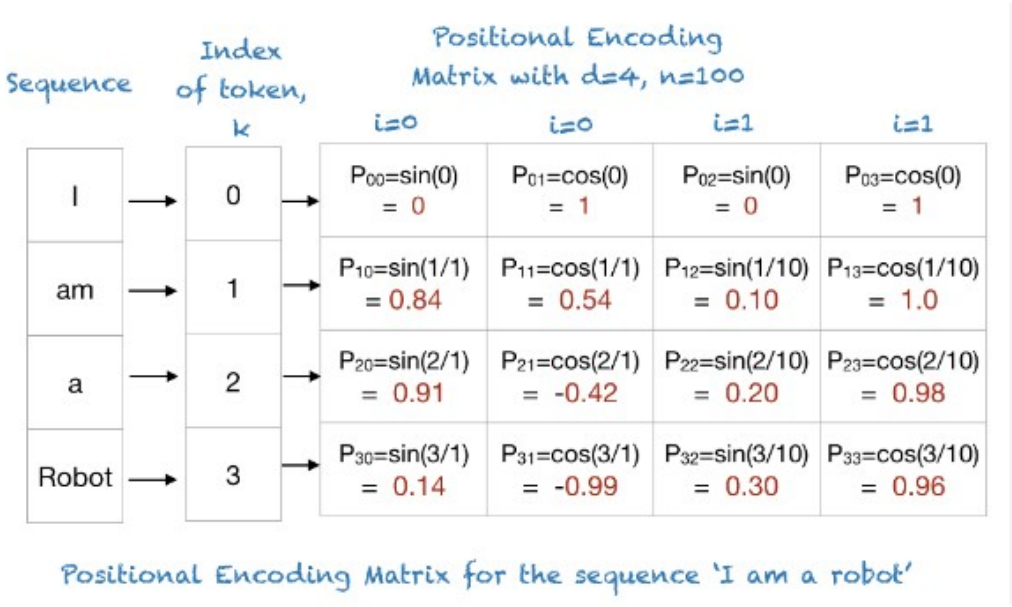
- Position Embedding: là vector đại diện cho vị trí của token. Ta có hai công thức cần phải biết để tính được position embbeding:

$$\mathrm{P}(\mathrm{k}, 2\mathrm{i}) = \sin(\frac{\mathrm{k}}{\mathrm{n}^{\frac{2\mathrm{i}}{\mathrm{d}}}}) \tag{1}$$

$$\mathrm{P}(\mathrm{k}, 2\mathrm{i}+1) = \cos(\frac{\mathrm{k}}{\mathrm{n}^{\frac{2\mathrm{i}}{\mathrm{d}}}}) \tag{2}$$

Trong đó:

  - **k** là vị trí của token đang xét trong văn bản
  - **d** là số chiều của output embedding space
  - **n** là giá trị do người dùng tự định nghĩa
  - **i** dùng để ánh xạ tới chỉ số cột



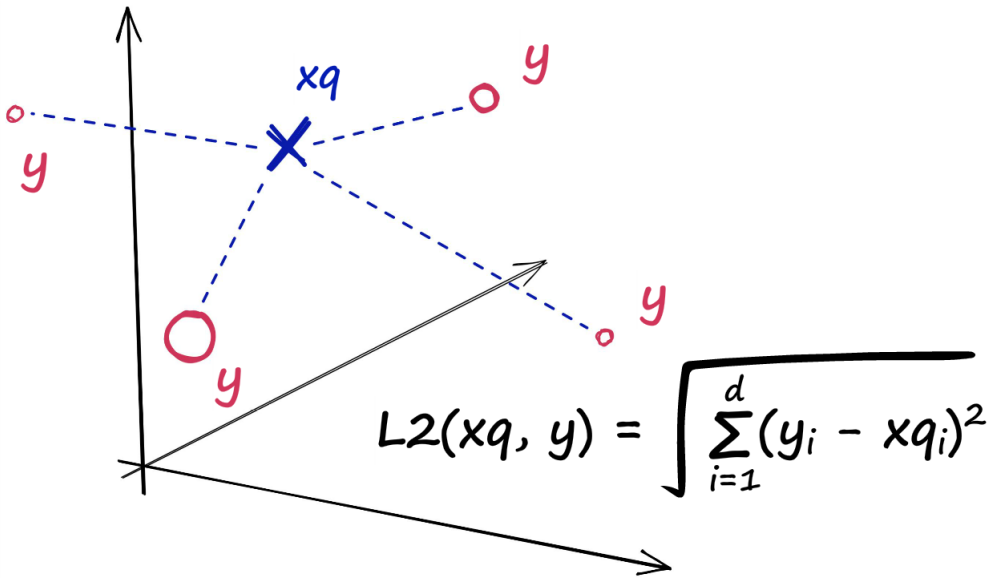
Hình 11: Ví dụ cho Position embedding

Khi có được cả ba embedding nêu trên, kết hợp lại ta sẽ được sentence embedding (có thể hiểu là vector biểu diễn cho câu).

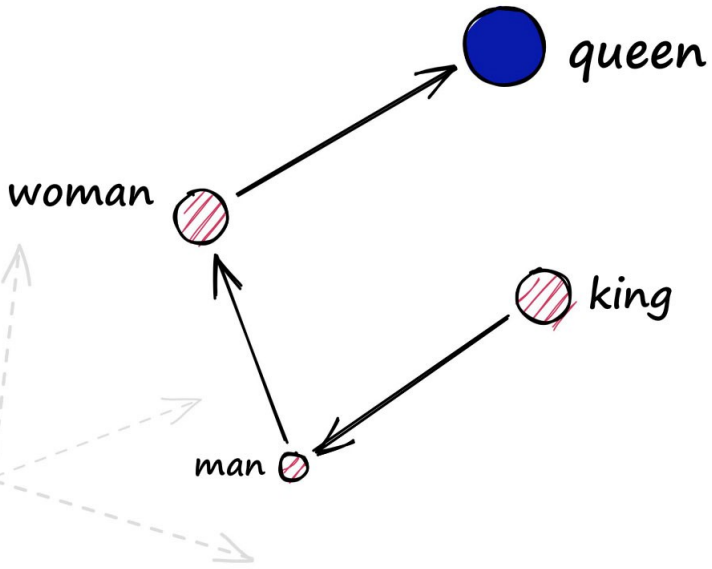
4.4 Similarity search và FAISS

4.4.1 Similarity search

Trong học máy, chúng ta thường biểu diễn các đối tượng và khái niệm trong thế giới thực dưới dạng một tập hợp các số liên tục, còn được gọi là vector embedding. Khi chúng ta biểu thị hình ảnh hoặc đoạn văn bản dưới dạng vector, thì sự giống nhau về ngữ nghĩa của chúng được thể hiện bằng mức độ gần của vector của chúng trong không gian vector. Similarity search [**9**] là chính xem xét khoảng cách giữa các vector của các đối tượng, sau đó chọn ra các vector gần nhất với vector mà ta đang cần xét.



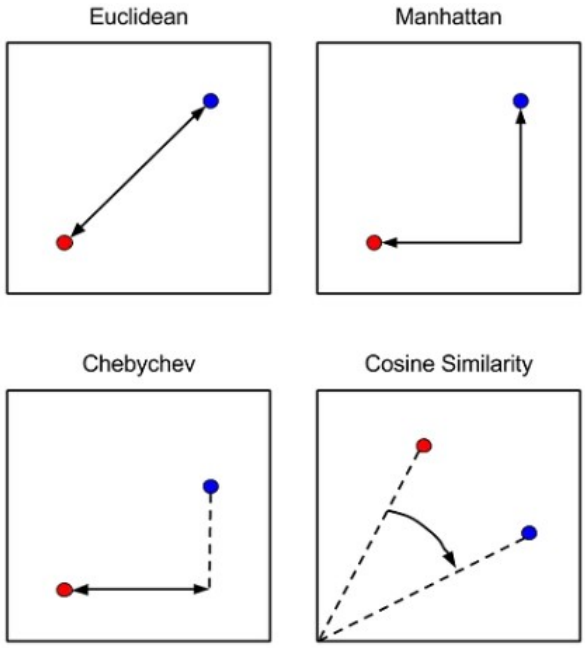
Hình 12: Độ liên quan của câu truy vấn và bộ dữ liệu



Hình 13: Độ liên quan ngữ nghĩa giữa **man-king** và **woman** là **queen**

Similarity Search có thể được sử dụng để so sánh dữ liệu một cách nhanh chóng. Đưa ra một truy vấn (có thể ở bất kỳ định dạng nào - văn bản, âm thanh, video, GIF,...), ta có thể sử dụng Similarity Search để trả về kết quả có liên quan. Có nhiều phương pháp để thực hiện việc so sánh độ liên quan cũng như tìm kiếm kế quả tốt nhất. Tuy nhiên, khi chúng ta có hàng triệu (hoặc thậm chí hàng tỷ) vật thể để so sánh - thì việc đó bắt đầu trở nên phức tạp. Vì thế, để tối ưu hoá công cụ tìm kiếm, đây là phương pháp hiệu quả nhất để cho kết quả tìm kiếm nhanh và chính xác nhất.

Một số thước đo khoảng cách thường được sử dụng trong ML là Euclidean, Manhattan, Cosine, and Chebyshev.

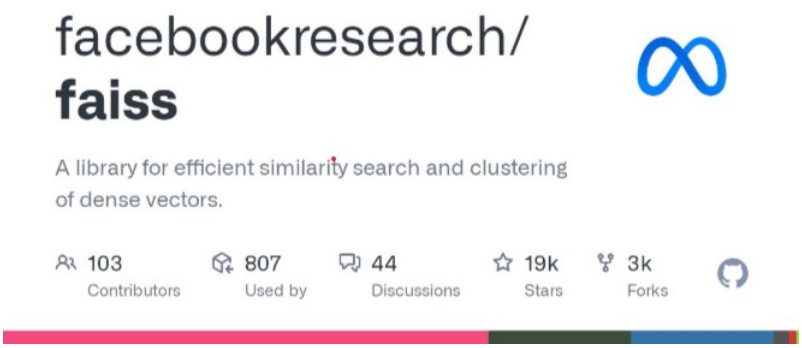


Hình 14: Distance Metrics

Sau khi biết trước 1 tập hợp các vector và 1 vector truy vấn, chúng ta áp dụng thuật toán **KNN (K Nearest Neighbors)** để thực hiện việc tìm các vectơ gần nhất trong một không gian cho vectơ truy vấn đó. Tuy nhiên, để tìm các vector gần nhất cho vector truy vấn, chương trình sẽ phải tính khoảng cách của nó với mọi vector có trong cơ sở dữ liệu .Điều này sẽ rất kém hiệu quả nếu chúng ta phải tìm kiếm trong hàng triệu vector. **ANN (Approximate Neighbor Search)** sẽ giải quyết vấn để trên bằng cách thay vì kiểm tra khoảng cách giữa mỗi vector trong cơ sở dữ liệu, chương trình sẽ truy xuất một "*dự đoán chính xác*" về neighbor gần nhất. Trong một số trường hợp, ta thà mất đi độ chính xác để tăng hiệu suất, do đó cho phép ta mở rộng quy mô tìm kiếm của mình. ANN cho phép tăng hiệu suất lớn cho tìm kiếm tương đồng khi xử lý các bộ dữ liệu khổng lồ.

4.4.2 FAISS

Facebook AI Similarity Search [**4**] (Faiss) là một thư viện sử dụng similarity search cùng với clustering các vector. Faiss được nghiên cứu và phát triển bởi đội ngũ Facebook AI Research, nó được viết bằng C++ và đóng gói trên môi trường Python. Bộ thư viện bao gồm các thuật toán tìm kiếm vector đa chiều trong Similarity search.



Hình 15: Facebook AI Similarity Search

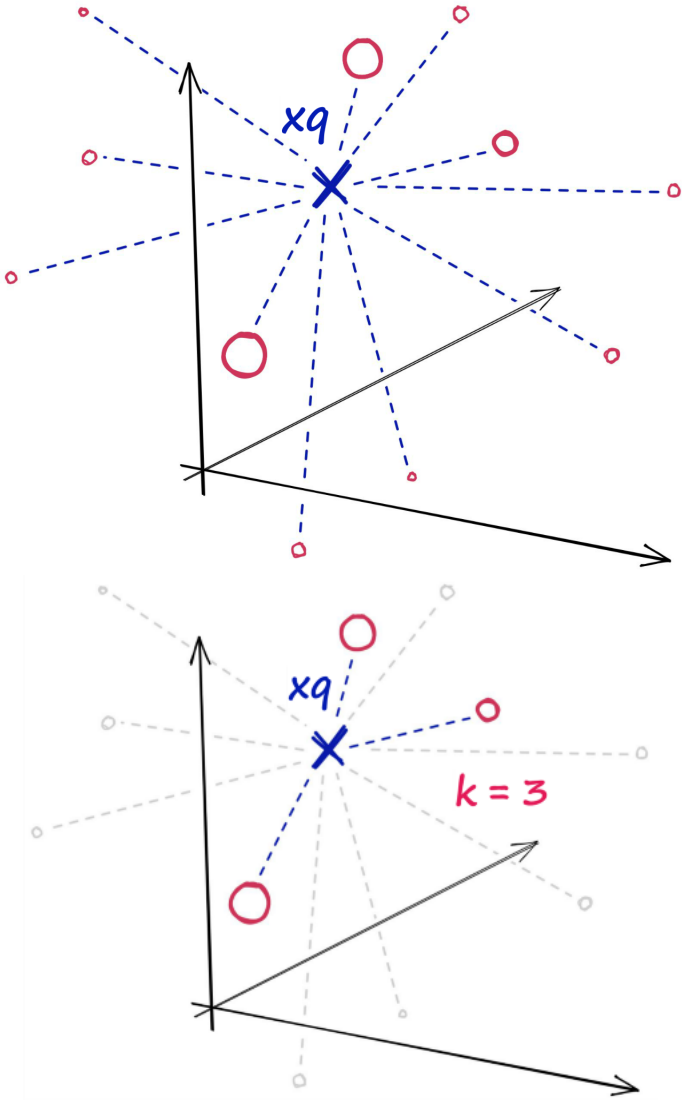
4.4.3 Similarity Search và FAISS

Thông qua các phương pháp thống kê hoặc học máy — chúng ta có thể xây dựng các vectơ mã hóa thông tin hữu ích, có ý nghĩa về dữ liệu gốc của mình. Sau đó, những vector này sẽ được lưu trữ bên trong các chỉ mục - index để sử dụng cho similarity search.

FAISS sẽ đóng vai trò như 1 bộ lưu trữ các vector và sẽ truy vấn tìm kiếm các index thông qua "vector truy vấn". Vectơ truy vấn này được so sánh với các vectơ chỉ mục khác để tìm kết quả khớp gần nhất - thường là với **Euclidean distance (L2)** hoặc **Cosine distance**.

FAISS có rất nhiều loại chỉ mục khác nhau, đối với bài này thì ta sẽ sử dụng chỉ mục **Flat** vì độ đơn giản, không cần phải thay đổi các vector đầu vào cũng như có có được chất lượng tìm kiếm tốt nhất và ít thời gian.

Với các **Flat index**, ta giới thiệu vector truy vấn **xq** và so sánh nó với mọi vector có kích thước đầy đủ khác trong chỉ mục và tính toán khoảng cách đến từng vector. Sau khi tính khoảng cách, ta sẽ trả về K chỉ mục gần nhất qua phương pháp **KNN** hoặc **ANN**. *Với K là 1 siêu tham số cho chúng ta biết dc nên cần tìnm bao nhiêu vector*



Hình 16: Tính khoảng cách

5 Thực nghiệm

5.1 Cross-encoder Re-ranking

Hệ thống truy vấn thông tin này có thể truy xuất những dữ liệu (bộ phim) không liên quan đến dữ kiện cần truy vấn. Do đó, ta sẽ sử dụng trình *re-renking* dựa trên bộ **cross-encoding** để chấm điểm mức độ liên quan của tất cả các ứng cử viên cho truy vấn tìm kiếm nhất định.

Cross Encodeer (C.E) tập trung chủ yếu (hoặc chéo) đối với một ứng cử viên đầu vào và nhãn nhất định, đồng thời có xu hướng đạt được độ chính xác cao hơn nhiều so với các đối tác của họ. Bộ **Cross encoder** có thể được sử dụng bất cứ khi nào ta có một tập hợp các cặp câu được xác định trước mà chúng tôi muốn cho điểm.

Most relevant movie	Cross-encoder Re-ranking Score
The Kentucky Fried Movie	0.00040342272
The Ugly Swans	0.00031254787
G-Force	0.00027860518
Twist	0.00027031262
Indian	0.00018003744

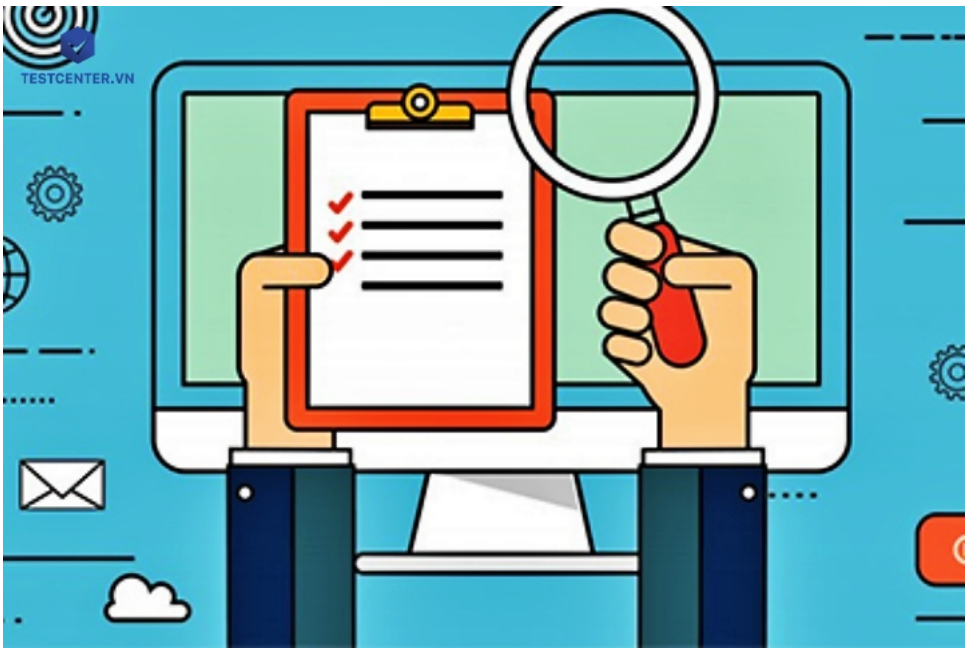
Bảng 1: Ví dụ cho cross-encoder re-ranking với query "*Artificial Intelligence based action movie*"

Trình Re-ranking dựa trên Bộ **Cross enoder** có thể cải thiện đáng kể kết quả cuối cùng cho người dùng. Câu truy vấn và một bộ tài liệu có thể được chuyển đồng thời đến mạng máy biến áp, sau đó mạng này sẽ xuất ra một điểm duy nhất trong khoảng từ 0 đến 1 cho biết mức độ liên quan của tài liệu đối với truy vấn đã cho.

5.2 Phương pháp đánh giá

Mô hình tìm kiếm sẽ được đánh giá bằng 5 câu query trên hai bộ dữ liệu (bộ dữ liệu đã qua xử lý NLP và bộ dữ liệu gốc). Ở mỗi câu query ta sẽ lấy top 15 kết quả mà mô hình cho là phù hợp với query nhất.

Vì bộ dữ liệu không có ground-truth sẵn nên nhóm phải tự đánh giá xem các kết quả trả về có phù hợp với query hay không. Sau đó tính Average Precision (AP) cho từng query, từ đó tính được MAP (Mean Average Precision) cho từng bộ dữ liệu.



Hình 17: Đánh giá mô hình tìm kiếm

5.3 Kết quả

Lưu ý: hàng được đánh dấu X ở cột [Phù hợp] có nghĩa là kết quả trả về phù hợp với query.

1. Kết quả cho câu query: *Artificial Intelligence based action movie* (xem kết quả ở **Bảng 2** và **Bảng 3**)

Ranking	Kết quả (tên phim)	Phù hợp
1	Universal Soldier: The Return	x
2	Armed Response	x
3	G-Force	x
4	The Cape Canaveral Monsters	
5	Best Defense	
6	Ten Years	
7	The Darwin Awards	
8	Impostor	x
9	Commando	
10	Chappie	x
11	Arjun — Kalimpong E Sitaharan	
12	Small Soldiers	x
13	Galactic Armored Fleet Majestic Prince: Genetic Awakening	x
14	Koi... Mil Gaya	
15	Nemesis	
AP = 0.72		

Bảng 2: Kết quả trên bộ data gốc

Ranking	Kết quả (tên phim)	Phù hợp
1	The Kentucky Fried Movie	
2	The Ugly Swans	x
3	G-Force	
4	Twist	
5	Indian	
6	Robot Overlords	x
7	Starship Invasions	x
8	The Day the Earth Stood Still	
9	Parasuram	
10	Species II	
11	Killers from Space	
12	Black Sunday	
13	Chappie	x
14	Ra.One	x
15	Nemesis	
AP = 0.39		

Bảng 3: Kết quả trên bộ data đã xử lý NLP

2. Kết quả cho câu query: *Movie about romance and pain of separation* (xem kết quả ở **Bảng 4** và **Bảng 5**)

Ranking	Kết quả (tên phim)	Phù hợp
1	Brothers	
2	2046	x
3	Young People Fucking	
4	Amar	x
5	Intentions of Murder	
6	I Love You, I Love You Not	x
7	Look Both Ways	x
8	Obaltan	
9	Ek Ke Baad Ek	
10	The Farm: En Veettu Thottathil	
11	Rites of Passage	
12	Sonali Cable	
13	Yearning	
14	About Love	x
15	I Like It Like That	
AP = 0.49		

Bảng 4: Kết quả trên bộ data gốc

Ranking	Kết quả (tên phim)	Phù hợp
1	Brothers	
2	Aandhali Koshimbir	x
3	Strings of Passion	
4	Camel Safari	x
5	Paanch Adhyay	x
6	2046	x
7	Obaltan	
8	Samooham	
9	U Me Aur Hum	x
10	Cold Heaven	
11	Rice Rhapsody	
12	Secrets	
13	You and I	x
14	Kannum Kannum	x
15	Swapaanam	
AP = 0.54		

Bảng 5: Kết quả trên bộ data đã xử lý NLP

3. Kết quả cho câu query: *Post apocalyptic movies* (xem kết quả ở **Bảng 6** và **Bảng 7**)

Ranking	Kết quả (tên phim)	Phù hợp
1	The Beach Party at the Threshold of Hell	x
2	World of the Dead: The Zombie Diaries	x
3	Kamen Rider vs Kamen Rider W and Decade: Movie War	x
4	Jesus Christ Superstar	
5	The Video Dead	x
6	Faces of Death IV	x
7	Slaughterhouse-Five,	
8	Flesheater	
9	Waxwork II: Lost in Time	x
10	Captain Video: Master of the Stratosphere	
11	Adhuri Kahani	
12	Commando Cody: Sky Marshal of the Universe	x
13	Alien Nation: Dark Horizon	x
14	Remote Control	x
15	The Errand Boy	
AP = 0.8		

Bảng 6: Kết quả trên bộ data gốc

Ranking	Kết quả (tên phim)	Phù hợp
1	The Bible: In the Beginning	
2	Kamen Rider vs Kamen Rider W and Decade: Movie War 2010	x
3	Rasta	
4	Waxwork II: Lost in Time	x
5	Vikingdom	
6	Slaughterhouse-Five	
7	Dangerous Parking	
8	Real Men	
9	My Summer Story	
10	Future War	x
11	Hunt Angels	
12	Alien Nation: Dark Horizon	x
13	The Shriek of Araby	
14	What a Night!	
15	Who Done It?	
AP = 0.41		

Bảng 7: Kết quả trên bộ data đã xử lý NLP

4. Kết quả cho câu query: *World war 2 movies* (xem kết quả ở **Bảng 8** và **Bảng 9**)

Ranking	Kết quả (tên phim)	Phù hợp
1	Special Delivery	
2	Till Death Us Do Part	
3	Back at the Front	
4	In the Army Now	
5	This Is the Army	
6	The Longest Day	x
7	In Love and War	
8	No Time to Die	
9	Innocents in Paris	
10	McHale's Navy	x
11	A Time of Destiny	x
12	The White Cliffs of Dover	
13	Allies	x
14	War and Remembrance	x
15	The Lion Has Wings	x
AP = 0.28		

Bảng 8: Kết quả trên bộ data gốc

Ranking	Kết quả (tên phim)	Phù hợp
1	Fantasy Mission Force	x
2	Jayne Mansfield's Car	
3	Back at the Front	
4	White Christmas	x
5	Midway	x
6	When Taekwondo Strikes	x
7	Kings Go Forth	x
8	Frankenstein vs. Baragon	x
9	Men Must Fight	
10	The White Cliffs of Dover	
11	Desert Bloom	
12	No Time to Die	
13	The Beach Party at the Threshold of Hell	
14	Uncommon Valor	
15	The Desert Fox	x
AP = 0.67		

Bảng 9: Kết quả trên bộ data đã xử lý NLP

5. Kết quả cho câu query: *Movie about treasure hunters* (xem kết quả ở **Bảng 10** và **Bảng 11**)

Ranking	Kết quả (tên phim)	Phù hợp
1	Anumanaspadam	x
2	Bangarada Jinke	x
3	Black Hunters	x
4	Lost Medallion: The Adventures of Billy Stone	x
5	Firewalker	x
6	Wind Across the Everglades	
7	Zombies of Mora Tau	x
8	Brotherhood of Blood	
9	Mackenna's Gold	x
10	Troublesome Night	
11	The Night of Counting the Years	
12	The Turkish Gambit	
13	Nayagi	
14	Salladin the Victorious (Al Nasser Salah Al Din)	
15	Yahşi Batı	
AP = 0.95		

Bảng 10: Kết quả trên bộ data gốc

Ranking	Kết quả (tên phim)	Phù hợp
1	Wind Across the Everglades	
2	Armour of God II: Operation Condor	x
3	Boxcar Bertha	
4	Brotherhood of Blood	
5	Gator Bait	
6	Headin' South	
7	Sekigahara	
8	Catalina Caper	
9	The Turkish Gambit	
10	Panther Girl of the Kongo	x
11	April Folly	
12	Getting Even	
13	Ashanti	
14	Black Hunters	x
15	The Deceivers	
AP = 0.3		

Bảng 11: Kết quả trên bộ data đã xử lý NLP

Dưới đây là kết quả tổng hợp được khi chạy thực nghiệm mô hình với 5 query.

Query	AP của data gốc	AP của data đã xử lý NLP
Artificial Intelligence based action movie	0.72	0.39
Movie about romance and pain of separation	0.49	0.54
Post apocalyptic movies	0.8	0.41
World war 2 movies	0.28	0.67
Movie about treasure hunters	0.95	0.3
MAP	**0.648**	**0.462**

Bảng 12: Bảng so sánh kết quả trên hai bộ dữ liệu

Từ kết quả AP từ các bảng trên ta tính được MAP của mô hình trên bộ dữ liệu gốc là **0.648**, hơn hẳn so với bộ data được xử lý NLP với MAP chỉ là **0.462**.

Nhìn chung thì kết quả của mô hình là khá thấp bới vì việc tìm kiếm dựa theo Plot của một bộ phim để cho kết quả đúng là khá khó. Ví dụ như Plot của phim chứa nhiều câu từ liên quan đến câu query, tuy nhiên đó lại không phải nội dung chính của bộ phim khiến cho mô hình dễ bị hiểu nhầm.

Nguyên nhân của việc MAP giảm có thể là do khi tách từ và chuyển về từ gốc, ngữ nghĩa của câu query phần nào đã bị giảm đi, nhưng thay vào đó việc tìm kiếm theo từ khóa có vẻ trở nên tốt hơn, ta có thể thấy điều này ở query "*World war 2 movies*" (xem kết quả ở **Bảng 8** và **Bảng 9**), thay vì phải hiểu được ý nghĩa của "*World war 2*" thì mô hình đã tìm kiếm các bộ phim có Plot chứa từ "*World war*", điều đó khiến cho kết quả trên bộ data đã xử lý NLP với query này tốt hơn kết quả trên bộ data gốc.

6 Kết luận

Trong bài báo cáo này, chúng em đã giới thiệu về Semantic Search, quy trình nó hoạt động cũng như xây dựng một hệ thống tìm kiếm tên phim dựa trên bản tóm tắt nội dung của nó. Các modules chính để tạo nên hệ thống tìm kiếm này cũng đã được trình bày chi tiết. Hệ thống tìm kiếm này được đánh giá trên bộ dữ liệu chưa qua xử lý NLP và đã qua xử lý NLP (tách từ, chuyển về từ gốc). Và dựa vào kết quả thực nghiệm, ta có thể kết luận rằng việc để có được kết quả tốt khi tìm kiếm phim dựa theo Plot là khá khó, đồng thời khi xử lý NLP cho bộ dữ liệu này thì kết quả sẽ không tốt bằng dữ liệu gốc.

Dưới đây là bảng phân chia công việc của nhóm.

		Trương Minh Sơn	Phan Minh Nhật
Lên ý tưởng		x	x
Tìm tài liệu	Semantic Search	x	
	S-BERT		x
	FAISS	x	
	Similarity Search	x	
Code		x	
Slide		x	x
Báo cáo		x	x
Đánh giá mô hình			x
Thuyết trình		x	x
Đóng góp		50%	50%

Tài liệu

[1] JustinR (2018) Wikipedia movie plots, Kaggle. Available at: https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots

[2] Semantic search là gì? Tối Ưu Nội Dung Cho semantic search (2021) GTV SEO. Available at: https://gtvseo.com/semantic-search/

[3] Barysevich, A. (2021) Semantic search: What it is amp; why it matters for seo Today, Search Engine Journal. Available at: https://www.searchenginejournal.com/semantic-search-seo/264037

[4] Introduction to facebook ai similarity search (Faiss) James Briggs. Available at: https://www.pinecone.io/learn/faiss-tutorial/.

[5] Pennington, J. Glove: Global vectors for word representation. Available at: https://nlp.stanford.edu/projects/glove/.

[6] Kotamraju, S. (2022) An intuitive explanation of sentence-bert, Medium. Towards Data Science. Available at: https://towardsdatascience.com/an-intuitive-explanation-of-sentence-bert-1984d144a868.

[7] Verma, S. (2022) Semantic search with S-Bert is all you need, Medium. MLearning.ai. Available at: https://medium.com/mlearning-ai/semantic-search-with-s-bert-is-all-you-need-951bc710e160.

[8] Sentence transformers and embeddings Pinecone. Available at: https://www.pinecone.io/learn/sentence-embeddings/.

[9] What is similarity search? Pinecone. Available at: https://www.pinecone.io/learn/what-is-similarity-search/.