

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ



ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

PHÂN CỤM CHỦ ĐỀ YOUTUBE VIDEO BẰNG K-MEANS, FUZZY C-MEANS, AUTOENCODER VÀ DEEP EMBEDDED CLUSTERING

Học phần: Máy Học

Nhóm Sinh Viên:

1. NGUYỄN VĂN HOÀNG DŨNG
2. NGUYỄN QUỲNH KHÁNH HÀ
3. HUỲNH TRẦN ANH THY
4. NGUYỄN TRỊNH THU HUYỀN

Chuyên Ngành: KHOA HỌC DỮ LIỆU

Khóa: K46

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

TP. Hồ Chí Minh, Ngày 22 tháng 4 năm 2023

MỤC LỤC

| | |
|--|-----------|
| MỤC LỤC | 1 |
| CHƯƠNG 1. TỔNG QUAN | 3 |
| 1.1. Giới Thiệu Bài Toán | 3 |
| 1.2. Lý Do Chọn Lựa Đè Tài | 4 |
| CHƯƠNG 2. CƠ SỞ LÝ THUYẾT | 5 |
| 2.1. Các Phương Pháp Tiền Xử Lý Dữ Liệu | 5 |
| 2.1.1. Tiền Xử Lý Dữ Liệu | 5 |
| 2.1.2. Xử lý biến Text | 6 |
| 2.1.3. LDA + BERT | 8 |
| 2.2. Các Mô Hình Phân Tích Dữ Liệu | 8 |
| 2.2.1. Mô Hình K - Means | 8 |
| 2.2.2. Mô Hình Fuzzy C - Means | 9 |
| 2.2.3. Mô Hình Autoencoders | 10 |
| 2.2.4. Mô Hình Deep Embedded Clustering | 11 |
| CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM | 12 |
| 3.1. Bộ Dữ Liệu | 12 |
| 3.1.1. Tổng quan về bộ dữ liệu | 12 |
| 3.1.2. Tiền xử lý dữ liệu | 14 |
| 3.1.3. Xử lý biến text | 15 |
| 3.1.4. Trực quan hóa dữ liệu (EDA) | 17 |
| 3.1.5. WordCloud | 25 |
| 3.1.6. LDA + BERT | 27 |
| 3.2. Huấn Luyện Dữ Liệu với các mô hình | 30 |
| 3.2.1. Traditional Clustering | 30 |
| 3.2.2. Autoencoders | 33 |
| 3.2.3. Deep Embedded Clustering | 37 |
| 3.3. Các Kết Quả Thực Nghiệm | 37 |
| 3.3.1. Traditional Clustering | 37 |
| 3.3.2. Autoencoders | 37 |
| 3.3.3. Deep Embedded Clustering | 38 |
| 3.4. Phân Tích và Đánh Giá | 38 |
| CHƯƠNG 4. KẾT LUẬN | 39 |
| 4.1. Các Kết Quả Đạt Được | 39 |
| 4.2. Những Hạn Chế và Hướng Phát Triển | 39 |
| 4.2.1. Hạn chế | 39 |
| 4.2.2. Hướng phát triển | 39 |

| | |
|--------------------------------|-----------|
| PHỤ LỤC I (Link Github) | 40 |
| PHỤ LỤC II | 40 |
| TÀI LIỆU THAM KHẢO | 41 |

CHƯƠNG 1. TỔNG QUAN

1.1. Giới Thiệu Bài Toán

Với sự phát triển mạnh mẽ của các nền tảng mạng xã hội như: facebook, twitter, instagram, tiktok, youtube,... lượng thông tin lưu trữ cực kỳ lớn. Với lưu lượng dữ liệu lớn như vậy, thật khó để có thể tìm kiếm một chủ đề mà chúng ta quan tâm giữa hàng triệu thông tin kia. Chính vì lý do đó, chúng ta cần có một công cụ, mô hình có thể giúp khai phá lượng lớn thông tin và gộp các nhóm thông tin ấy theo từng chủ đề. Điện hình như YouTube là một nền tảng chia sẻ video trực tuyến lớn nhất thế giới. Hiện nay, YouTube có hơn 2 tỷ người dùng trên toàn thế giới và hàng triệu video được tải lên mỗi ngày. Người dùng trên toàn thế giới có thể tìm kiếm và xem các video về nhiều chủ đề khác nhau trên YouTube, từ giải trí, âm nhạc, thể thao, đến giáo dục và khoa học. Với lượng dữ liệu to lớn như vậy, việc phân loại các video trên YouTube thành các nhóm tương tự nhau dựa trên nội dung của chúng có thể giúp người dùng dễ dàng tìm kiếm và xem các video liên quan đến chủ đề mình quan tâm.

Mô hình Topic Modeling là một trong những công cụ giúp ta giải quyết vấn đề trên. Topic modeling là một kỹ thuật xử lý ngôn ngữ tự nhiên trong lĩnh vực học máy và khai phá dữ liệu, cho phép tự động phân loại các văn bản vào các chủ đề khác nhau dựa trên nội dung của chúng. Kỹ thuật này được sử dụng để phân tích và tổng hợp các thông tin từ nhiều văn bản khác nhau, giúp cải thiện quá trình tìm kiếm thông tin, phân tích dữ liệu và truy vấn dữ liệu. Thông qua, **bài toán phân cụm chủ đề Youtube video bằng K - Means, Fuzzy C - Means, Autoencoders và Deep Embedded Clustering** nhóm chúng em sẽ làm rõ hơn việc khai phá và gộp nhóm các chủ đề Youtube video để đề xuất cho người xem một cách hiệu quả nhất.

Để thực hiện bài toán này, nhóm chúng em sử dụng ngôn ngữ chủ yếu là *Python* để xây dựng các mô hình *K - Means*, *Fuzzy C - Means*, *Autoencoders* và *Deep Embedded Clustering*. Đầu tiên nhóm crawling data Youtube video để tạo bộ dữ liệu phục vụ cho bài nghiên cứu. Tiếp theo tiến hành xử lý bước tiền xử lý dữ liệu trước khi sử dụng vào các mô hình thuật toán. Trực quan hóa dữ liệu thông qua các biểu đồ, WordCloud các biến nghiên cứu. Cuối cùng là xây dựng các mô hình *K - Means*, *Fuzzy C - Means*,

Autoencoders và *Deep Embedded Clustering* tìm ra các chỉ số đánh giá của từng mô hình để lựa chọn phương án tốt nhất cho việc phân cụm chủ đề.

1.2. Lý Do Chọn Lựa Đề Tài

Bài toán này tập trung vào việc sử dụng các kỹ thuật liên quan đến việc kết hợp giữa xử lý ngôn ngữ tự nhiên và áp dụng các mô hình học máy để tìm ra những thông tin cũng thuộc một chủ đề nào đó để người dùng dễ dàng hơn trong việc tìm kiếm. Mục tiêu là phân cụm chủ đề, khai thác và vận dụng các mô hình học máy, học sâu kết hợp tiền xử lý bằng word embedding để đưa ra kết quả phân cụm tốt nhất. Một trong những thách thức của nghiên cứu này là trích xuất thông tin hữu ích từ biến Title sau đó tạo thành các vector để có thể đưa vào mô hình học máy. Nhóm đã sử dụng phương pháp LDA kết hợp với phương pháp BERT để tạo ra các vector đó.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Các Phương Pháp Tiền Xử Lý Dữ Liệu

2.1.1. Tiền Xử Lý Dữ Liệu

a. Khái niệm

Tiền xử lý đề cập đến các phép biến đổi được áp dụng cho dữ liệu của chúng ta trước khi đưa nó vào thuật toán.

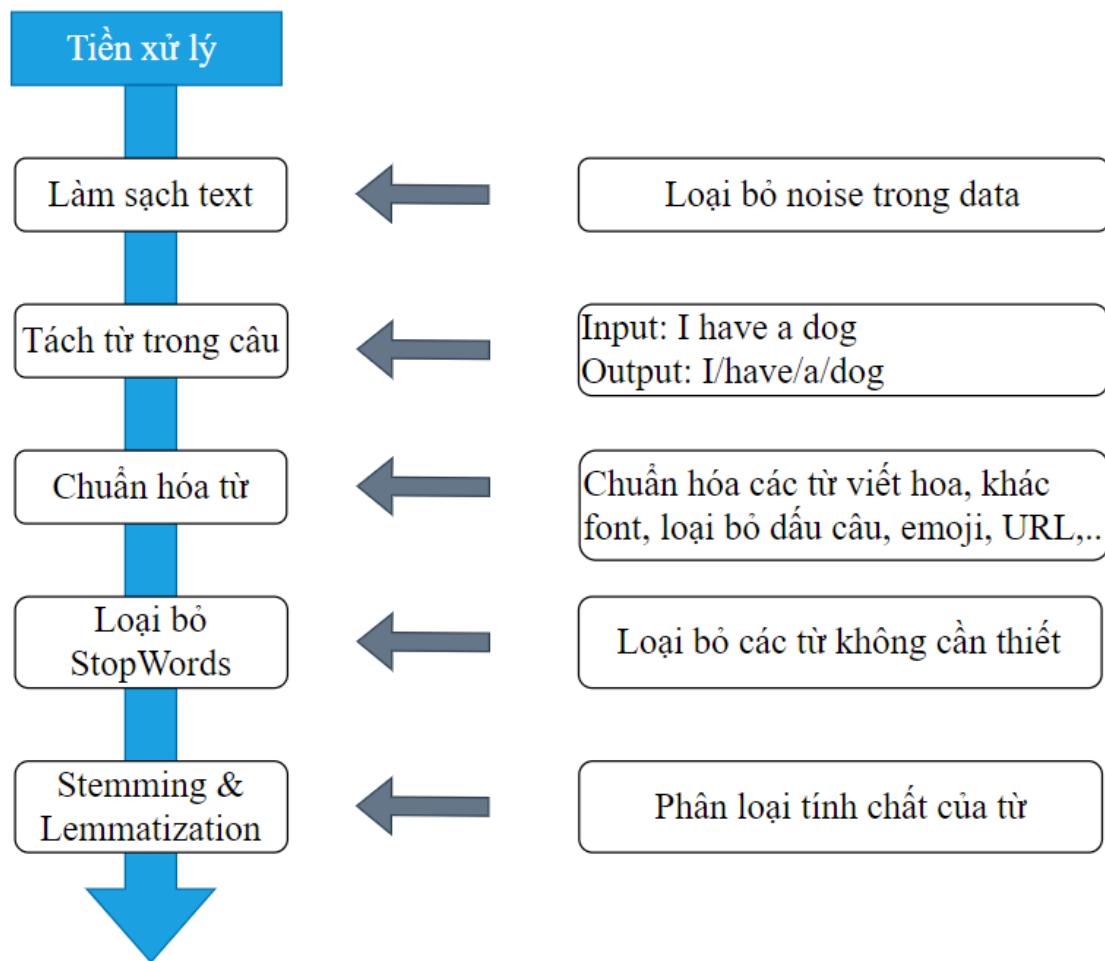
Tiền xử lý dữ liệu là một kỹ thuật được sử dụng để chuyển đổi dữ liệu thô thành một tập dữ liệu sạch. Nói cách khác, bất cứ khi nào dữ liệu được thu thập từ các nguồn khác nhau, nó được thu thập ở định dạng thô không khả thi cho việc phân tích.

b. Tại sao cần tiền xử lý dữ liệu

Để đạt được kết quả tốt hơn từ mô hình được áp dụng trong các dự án Học máy, định dạng của dữ liệu phải theo cách phù hợp. Một số mô hình Học máy được chỉ định cần thông tin ở định dạng được chỉ định, ví dụ: thuật toán Rừng ngẫu nhiên không hỗ trợ giá trị null, do đó, để thực thi thuật toán rừng ngẫu nhiên, giá trị rỗng phải được quản lý từ tập dữ liệu thô ban đầu.

Một khía cạnh khác là tập dữ liệu nên được định dạng theo cách mà nhiều hơn một thuật toán Học máy và Học sâu được thực thi trong một tập dữ liệu và tốt nhất trong số chúng được chọn.

2.1.2. Xử lý biến Text



Bước 1: Làm sạch biến text

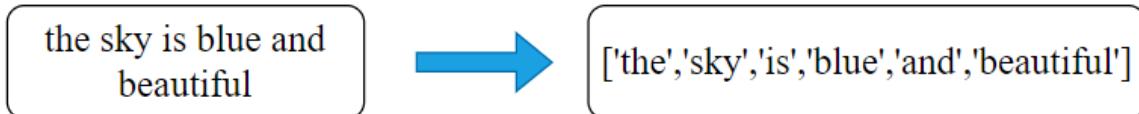
Mục đích bước này là loại bỏ noise trong data. Đầu tiên là biến đổi các câu thành kiểu chữ thường để thao tác loại bỏ các ký tự. Đa phần là loại bỏ các thẻ HTML và JS cũng có thể là những cụm từ không cần thiết, hay ký tự không có ý nghĩa (\$%&#").



Bước 2: Tách từ trong câu

Để câu thành nên đoạn văn bản bao giờ cũng bao gồm nhiều câu thành phần. Chính vì vậy để thực hiện các thao tác trên bộ dữ liệu text thì đầu tiên cần tách riêng biệt các

câu thành phần. Sử dụng hàm Tokenize để tách văn bản thành những câu riêng biệt. Thư viện NLTK cũng cung cấp hàm để tách các từ nltk.word_tokenize một cách chính xác.



Bước 3: Chuẩn hoá từ

Mục đích là đưa văn bản từ các dạng không đồng nhất về cùng một dạng. Dưới góc độ tối ưu bộ nhớ lưu trữ và tính chính xác cũng rất quan trọng.

Ví dụ: U.S.A = USA

Thực hiện loại bỏ các ký tự đặc biệt như dấu câu như ?, !, ", ;, v.v, loại bỏ các emoji và URL (nếu có)

Bước 4: Loại bỏ StopWords

Stop words thường là các từ xuất hiện nhiều lần và không đóng góp nhiều vào ý nghĩa của câu, chúng sẽ đóng vai trò như nhiễu, trong tiếng Anh các từ này có thể kể đến như the, is, at, on, which, in, some, many hay trong tiếng Việt là các từ cái, các, cả,....

Để sử dụng stop words của NLTK, trước tiên ta cần download bộ stop words nltk.download('stopwords').



Bước 5: Stemming & Lemmatization

- **Stemming** là quá trình biến đổi các từ về dạng gốc của nó (ví dụ: connected, connection khi stemming thu được connect hay moved, move khi stemming thu được mov). Nhưng chưa chắc từ này có trong từ vựng tiếng anh. Trong thư viện NLTK cũng có hỗ trợ thuật toán Porter để thực hiện nhiệm vụ này.
- **Lemmatization** về cơ bản là giống với stemming khi nó loại bỏ phần đuôi của từ để thu được gốc từ (ví dụ từ moved sau khi lemmatize sẽ thu được move). Chắc

chắn từ này có trong từ vựng tiếng anh. Sử dụng part-of-speech tagging (`nltk.pos_tag`) để thu được các tính chất của từ.

2.1.3. LDA + BERT

LDA (Latent Dirichlet Allocation) và BERT (Bidirectional Encoder Representations from Transformers) là hai công cụ khác nhau để xử lý ngôn ngữ tự nhiên và tạo ra các biểu diễn vector cho các đoạn văn bản. Tuy nhiên, chúng có phương pháp và mục đích khác nhau.

LDA là một thuật toán topic modeling, nó giúp phân tích và xác định các chủ đề tiềm ẩn trong một bộ dữ liệu văn bản. Đầu ra của LDA là một tập hợp các chủ đề và các từ liên quan đến chủ đề đó. Các vector đại diện cho các văn bản trong một tập dữ liệu được tạo ra bằng cách sử dụng phân phối của các từ trong từng chủ đề. Ví dụ, một văn bản có thể có phân phối chủ đề "thể thao" và "giải trí", trong đó phần lớn các từ trong văn bản liên quan đến chủ đề "thể thao".

BERT là một kiến trúc mạng nơ-ron biểu diễn một cách sâu sắc và đa chiều các thông tin trong văn bản. BERT được huấn luyện trên một lượng lớn các dữ liệu văn bản và đưa ra các biểu diễn vector cho từng từ trong văn bản, được gọi là "word embeddings". Các biểu diễn vector này không chỉ cố định đại diện cho từng từ trong văn bản mà còn đánh dấu sự tương quan giữa các từ trong câu và giữa các câu trong văn bản.

Để kết hợp LDA và BERT, bạn có thể sử dụng kỹ thuật "concatenation" (nối vector). Cụ thể, bạn có thể tạo ra các vector đại diện cho từng văn bản bằng cách sử dụng phân phối chủ đề của LDA và word embeddings của BERT, sau đó nối các vector này lại với nhau để tạo ra một vector đại diện cho văn bản đó. Vector này có thể được sử dụng để huấn luyện các mô hình học máy như bình thường.

2.2. Các Mô Hình Phân Tích Dữ Liệu

2.2.1. Mô Hình K - Means

a. Khái niệm

Core Idea của K-means là phân cụm dữ liệu đặc trưng bởi các tâm (centroids). Tâm đại diện cho mỗi cụm và bằng trung bình các điểm dữ liệu nằm trong cụm. Khoảng cách

từ các điểm dữ liệu đến các tâm sẽ xác định xem điểm dữ liệu đó thuộc tâm nào. Khi thực hiện phân cụm K-means, chúng ta có những bước chính như sau:

- + Khởi tạo ngẫu nhiên k tâm cụm ban đầu
- + Tạo vòng lặp quá trình cập nhật tâm:
 - Xác định khoảng cách từ từng điểm dữ liệu đến tâm
 - Xác định lại các tâm theo trung bình khoảng cách của các điểm dữ liệu trong 1 cụm

★ *Về mặt toán học:*

Giả sử ta có N điểm dữ liệu là $X = [x_1, x_2, \dots, x_N]$ và k cụm dữ liệu ($k < N$). Với mỗi điểm dữ liệu x có label vector y cho nó đặc trưng có cho x thuộc cụm nào. Nếu x_j có vector $y_j = [1, 0, 0, \dots, 0]$ thì x thuộc cụm 1. Vì mỗi điểm dữ liệu chỉ thuộc 1 cụm đồng nghĩa với việc chỉ có 1 phần tử của vector y có giá trị = 1.

2.2.2. Mô Hình Fuzzy C - Means

a. Khái niệm

Fuzzy C-means (FCM) là một thuật toán dùng để phân cụm dữ liệu, nó là một phương pháp dựa trên lý thuyết mờ (fuzzy theory). FCM có thể được áp dụng trên một tập dữ liệu với các đặc trưng khác nhau và có thể phân loại chúng thành các nhóm khác nhau. FCM được sử dụng trong nhiều lĩnh vực, như khai thác dữ liệu, nhận dạng mẫu, xử lý ảnh và nhiều ứng dụng khác.

Các bước thực hiện của thuật toán FCM như sau:

- Xác định số lượng nhóm cần phân loại và giá trị mờ ban đầu của mỗi đối tượng trong mỗi nhóm.
- Tính toán ma trận khoảng cách giữa mỗi đối tượng và tất cả các trung tâm nhóm.
- Tính toán một ma trận mờ (fuzzy matrix) bằng cách sử dụng hàm mất mát (objective function) định nghĩa trước để đánh giá độ mờ của mỗi đối tượng cho mỗi nhóm.
- Cập nhật lại giá trị trung tâm của các nhóm bằng cách sử dụng ma trận mờ đã tính được.

- Lặp lại các bước trên cho đến khi kết quả phân nhóm không thay đổi hoặc đạt được điều kiện dừng.

2.2.3. Mô Hình Autoencoders

a. Khái niệm

Autoencoder (tiếng việt hay gọi là bộ tự mã hóa) là mạng ANN có khả năng học hiệu quả các biểu diễn của dữ liệu đầu vào mà không cần nhãn hoặc dữ liệu huấn luyện từ trước, tức là nó có khả năng học không giám sát (Unsupervised Learning)

Mục tiêu của AutoEncoder là tạo ra giá trị output gần giống với input, thực hiện tái tạo và nén dữ liệu vào, sao cho data được nén đó có kích thước nhỏ hơn data đầu vào nhưng vẫn giữ được những cái đặc trưng nhất của dữ liệu, đó là lý do Autoencoder có thể dùng trong các bài toán giảm chiều dữ liệu hoặc trích xuất đặc trưng (feature extraction).

→ Đối với bài toán Topic Modeling, nhóm em sẽ áp dụng AutoEncoder để trích xuất các đặc trưng văn bản, giúp phân cụm và nhóm các từ có những đặc tính tương tự nhau, từ đó giúp khai phá các chủ đề tiềm ẩn trong tập dữ liệu.

**Cấu trúc của AutoEncoder gồm có 3 phần chính: Encoder (bộ mã hóa), Bottleneck (không gian tiềm ẩn latent space) và Decoder (bộ giải mã)

**Hoạt động của nó được mô tả như sau:

1. Encoder: Nhận một tập dữ liệu đầu vào và có nhiệm vụ nén và ánh xạ dữ liệu từ miền không gian này lên không gian tiềm ẩn có kích thước nhỏ hơn, tức là chiều của bottleneck này sẽ nhỏ hơn data đầu vào và sẽ chứa cái đặc trưng nhất của data đầu vào
2. Bottleneck: Điều đặc biệt trong kiến trúc của Autoencoder là tạo ra 1 nút thắt cổ chai (bottleneck) giữa encoder và decoder. Dữ liệu khi đi qua nút thắt cổ chai sẽ được mô hình cổ gắng khôi phục lại giống với dữ liệu gốc. Vì vậy các thông tin tại nút thắt đó là những thông tin đặc trưng tốt nhất của dữ liệu, từ đó có thể được dùng để trích xuất các đặc trưng tiềm ẩn của văn bản.
3. Decoder: có nhiệm vụ giải mã data từ bottleneck đó để tái tạo lại dữ liệu đầu vào dựa trên các đặc trưng tiềm ẩn bên trong Bottleneck.

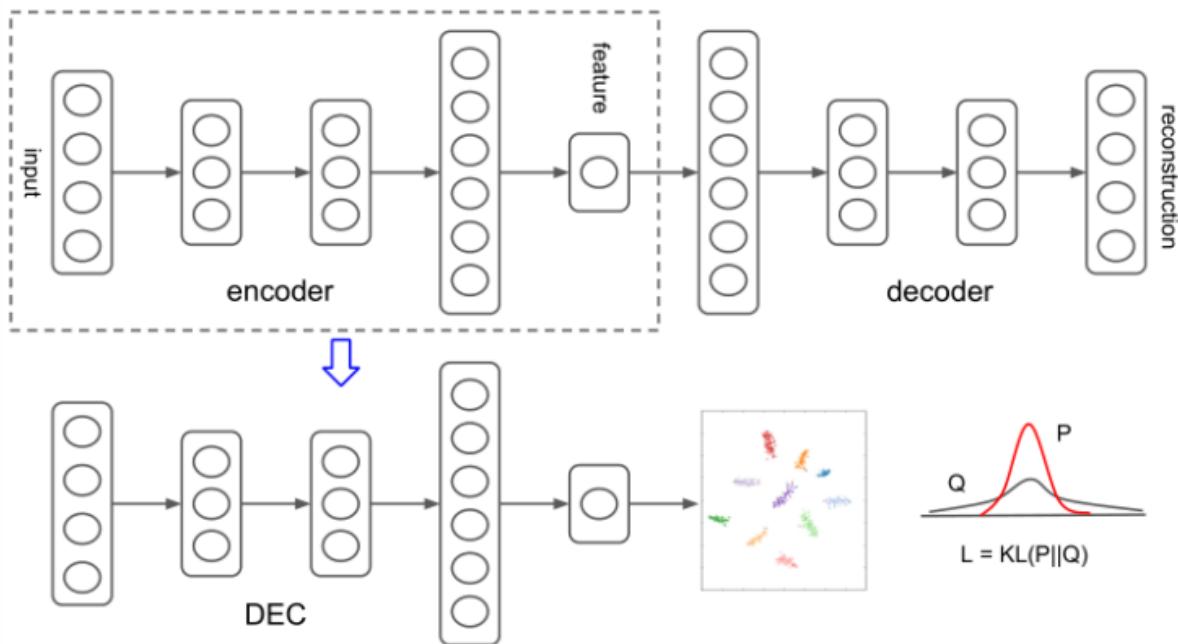
(Để huấn luyện mô hình Autoencoder, chúng ta đưa cho nó input data và nó sẽ cố gắng tái hiện lại input data bằng cách tối thiểu hóa Mean Squared Error giữa output và input. Hay nói cách khác input data cũng là nhãn của chính nó.)

Vì giá trị thực sự của Autoencoder model nằm ở output của Encoder ở trong không gian tiềm ẩn Latent Space. Hay nói cách khác, chúng ta thường sẽ chú trọng đến phần Encoder và Latent Space đối với bài toán trích xuất đặc trưng của văn bản.

2.2.4. Mô Hình Deep Embedded Clustering

a. Khái niệm

Mô hình Deep Embedded Clustering (DEC) là một mô hình phân cụm không giám sát (unsupervised clustering). Mô hình này kết hợp giữa Autoencoder và hiệu chỉnh lớp phân cụm truyền thống để tạo ra một mô hình phân cụm hiệu quả hơn.



DEC sử dụng một mạng nơ-ron sâu (Deep Neural Network) để học biểu diễn dữ liệu từ các đặc trưng (features) đầu vào. Mạng nơ-ron này được đào tạo trước (pre-trained) bằng thuật toán autoencoder, giúp mô hình học cách giảm chiều dữ liệu và trích xuất các đặc trưng quan trọng. Sau đó, mô hình sẽ sử dụng các đặc trưng này để phân cụm dữ liệu.

DEC cũng sử dụng một hàm mất mát đặc biệt gọi là KL-divergence (Kullback-Leibler divergence) để đo độ tương đồng giữa các đối tượng và các nhóm. Mục tiêu của mô hình là tối thiểu hóa hàm mất mát này để đạt được phân cụm tốt nhất.

Tổng quan, mô hình phân cụm Deep Embedded Clustering là một mô hình phân cụm không giám sát kết hợp giữa Autoencoder và hiệu chỉnh lớp phân cụm truyền thống. Mô hình sử dụng một mạng nơ-ron sâu để học biểu diễn dữ liệu và một hàm mất mát đặc biệt để đo độ tương đồng giữa các đối tượng và các nhóm. Ngoài ra, mô hình cũng sử dụng một chiến lược cập nhật động để cập nhật trung tâm nhóm cho mỗi vòng lặp. Mô hình DEC đã chứng tỏ được hiệu quả của nó trong nhiều ứng dụng thực tế.

2.3. Các phương pháp đánh giá:

2.3.1. Silhouette Score:

Silhouette Score là một độ đo được sử dụng để đánh giá chất lượng phân cụm của các thuật toán phân cụm. Độ đo này đo lường độ tách biệt của các cụm và sự tương đồng của các điểm trong cùng một cụm. Điểm số Silhouette càng cao thì hiệu suất phân cụm càng tốt.

Điểm số Silhouette cho một điểm dữ liệu i được tính bằng công thức:

$$s(i) = \frac{(b(i) - a(i))}{\max(a(i), b(i))}$$

Trong đó:

- $a(i)$ là khoảng cách trung bình giữa điểm i và các điểm trong cùng một cụm.
- $b(i)$ là khoảng cách trung bình giữa điểm i và các điểm trong cụm gần nhất khác với điểm i .
- $\max(a(i), b(i))$ là giá trị lớn nhất giữa $a(i)$ và $b(i)$.

Điểm số Silhouette cho toàn bộ tập dữ liệu được tính bằng trung bình của các điểm số Silhouette cho tất cả các điểm dữ liệu trong tập. Điểm số Silhouette có giá trị từ -1 đến 1, trong đó giá trị gần 1 cho thấy cụm tốt, giá trị gần 0 cho thấy các cụm gần như không tốt và giá trị gần -1 cho thấy cụm không tốt. Nếu điểm số Silhouette cho toàn bộ tập dữ liệu là dương, thì phân cụm được coi là tốt. Tuy nhiên, cần lưu ý rằng độ đo này không

phải lúc nào cũng phù hợp và cần được sử dụng cùng với các độ đo khác để đánh giá chất lượng phân cụm.

2.3.2. Davies-Bouldin Score:

Davies-Bouldin Score là một độ đo được sử dụng để đánh giá hiệu suất của các thuật toán phân cụm. Đó là một độ đo của sự phân tách và tính gọn của các cụm, trong đó điểm số thấp hơn cho thấy phân cụm tốt hơn.

Điểm số được tính bằng cách tính trung bình của khoảng cách giữa tất cả các cặp cụm, chia cho tổng khoảng cách giữa trung tâm của mỗi cụm và hàng xóm gần nhất của nó. Theo cách này, nó đo lường tỷ lệ giữa khoảng cách giữa các cụm và khoảng cách trong cụm.

Công thức tính điểm Davies-Bouldin cho một tập hợp các cụm C như sau:

$$DB(C) = \left(\frac{1}{n}\right) * \sum_{i=1}^n \max_{j \neq i} \frac{(R_i + R_j)}{D_{ij}}$$

Trong đó:

- n là số lượng các cụm trong tập hợp C.
- R_i là khoảng cách trung bình giữa các điểm trong cụm i và trung tâm của cụm đó.
- D_{ij} là khoảng cách giữa trung tâm của cụm i và trung tâm của cụm j.
- R_j là khoảng cách trung bình giữa các điểm trong cụm j và trung tâm của cụm đó.

Điểm Davies-Bouldin thấp hơn cho thấy hiệu suất phân cụm tốt hơn. Tuy nhiên, cần lưu ý rằng độ đo này có một số hạn chế và không luôn cung cấp một đánh giá chính xác về chất lượng phân cụm.

2.3.3. Calinski-Harabasz Score:

Calinski-Harabasz Score là một độ đo được sử dụng để đánh giá hiệu suất của các thuật toán phân cụm. Đó là một độ đo của tính phân tách của các cụm và được tính bằng cách so sánh khoảng cách giữa các cụm và sự đồng nhất của các điểm trong mỗi cụm. Một điểm số cao hơn cho thấy phân cụm tốt hơn.

Điểm số Calinski-Harabasz được tính bằng cách tính tỷ lệ giữa biến thể giữa các cụm và trong các cụm. Cụ thể, điểm số là tổng của các giá trị giữa-cụm và chia cho giá trị trong-cụm. Công thức tính điểm Calinski-Harabasz cho một tập hợp các cụm C như sau:

$$CH(C) = \frac{\frac{B}{(k-1)}}{\frac{W}{n-k}}$$

Trong đó:

- k là số lượng cụm trong tập hợp C.
- n là tổng số điểm dữ liệu.
- B là tổng khoảng cách giữa trung tâm của các cụm và trung điểm của toàn bộ dữ liệu.
- W là tổng của các khoảng cách giữa các điểm và trung tâm của cụm trong đó chúng thuộc.
- CH(C) là điểm số Calinski-Harabasz cho tập hợp các cụm C.

Điểm số Calinski-Harabasz càng cao thì hiệu suất phân cụm càng tốt. Tuy nhiên, cần lưu ý rằng độ đo này cũng có những hạn chế và không luôn đưa ra kết quả chính xác về chất lượng phân cụm.

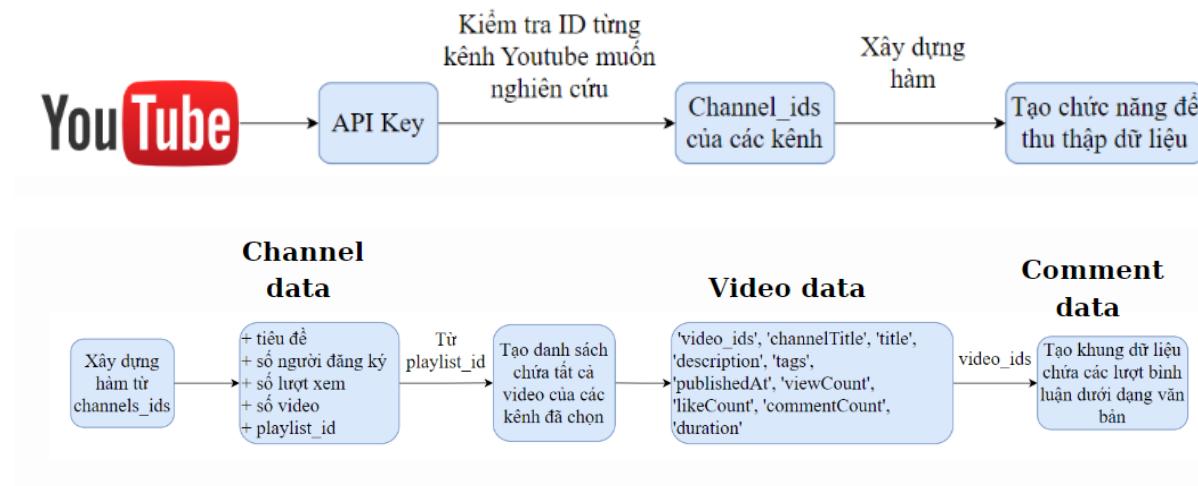
CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM

3.1. Bộ Dữ Liệu

3.1.1. Tổng quan về bộ dữ liệu

a. Crawling data

Bộ dữ liệu mà nhóm sử dụng cho bài toán lần này được crawling từ nền tảng Youtube thông qua phương pháp API. Các bước thực hiện theo mô hình dưới đây:



Bước đầu tiên là đăng nhập vào kênh Youtube cá nhân sau đó tiến hành lấy được API Key để thực hiện quá trình crawling data. Sau đó truy cập Youtube và kiểm tra ID channel của từng kênh muốn đưa vào phạm vi nghiên cứu (sử dụng channel_ids của chúng). Sau đó, nhóm đã tạo các chức năng để lấy số liệu thống kê kênh thông qua API. Cụ thể lấy được những số liệu thống kê như sau:

- Xây dựng hàm thu thập thống kê về các kênh được chọn bao gồm các thông tin tiêu đề, số người đăng ký, số lượt xem, số video, playlist ID. Hàm được dựa trên thông số channels_ids: danh sách ID kênh.

→ **Kết quả thu được:**

Khung dữ liệu chứa thống kê kênh cho tất cả các kênh trong danh sách được cung cấp: tiêu đề, số người đăng ký, số lượt xem, số video, playlist ID.

- Tiếp theo, tạo ra một danh sách chứa các ID video của tất cả video trong danh sách kênh đã cho. Thông số nhận được là playlist_id: ID video danh sách phát của kênh.

→ **Kết quả thu được:**

Danh sách ID video của tất cả video trong danh sách phát của các kênh.

- Sau đó, xây dựng hàm dựa trên playlist_id để nhận số liệu thống kê video của tất cả các video có ID như trong danh sách.

→ **Kết quả thu được:**

Khung dữ liệu với số liệu thống kê về video gồm: ‘video_ids’, ‘channelTitle’, ‘title’, ‘description’, ‘publishedAt’, ‘viewCount’, ‘likeCount’, ‘commentCount’, ‘thời lượng’, ‘định nghĩa’.

- Cuối cùng là tạo hàm thu về các lượt bình luận dưới dạng văn bản từ tất cả các video có ID như trong danh sách thông qua thông số video_ids: danh sách ID video.

→ **Kết quả thu được:**

Khung dữ liệu có ID video và các lượt bình luận được thể hiện dưới dạng văn bản.

b. Bộ dữ liệu

Bộ dữ liệu bao gồm các video xoay quanh 3 chủ đề chính: marketing, finance, accounting. Với số lượng channel thì nhóm thu được tổng cộng khoảng 3000 video với 9 biến mô tả. Cụ thể:

- + video_id: Mã video
- + channelTitle: Tên của kênh (channel)
- + title: Tiêu đề của video
- + description: Mô tả của video
- + publishedAt: Ngày giờ upload video
- + viewCount: Tổng lượt xem
- + likeCount: Tổng lượt like
- + commentCount: Tổng lượt bình luận
- + duration: Thời lượng video

| | channelName | subscribers | views | totalVideos | playlistId | Major | channel_ids |
|---|--|-------------|-----------|-------------|--------------------------|------------|----------------------------|
| 0 | Accounting Stuff | 553000 | 20097137 | 69 | UUYJLdSmyKoXCbnd-pklMn5Q | accounting | NaN |
| 1 | Silicon Valley Girl | 1100000 | 352488582 | 357 | UUiq1FlgtEK7LRAOB1JXTPig | marketing | [UCS3lFRhXKfCZ8FvRqPjy9NA] |
| 2 | Social Media Examiner | 273000 | 13690207 | 779 | UUS3lFRhXKfCZ8FvRqPjy9NA | marketing | [UCiq1FlgtEK7LRAOB1JXTPig] |
| 3 | Ryan Scribner | 808000 | 65720221 | 532 | UU3mjMoJuFnjYRBLon_6njbQ | finance | [UC3mjMoJuFnjYRBLon_6njbQ] |
| 4 | CPA Strength | 217000 | 13150965 | 564 | UUpHdE9yhgSuM4FezX8dmPw | accounting | [UCxFRGG-_23Kqxe0YexDc1eg] |
| 5 | Else Grech Accounting | 27100 | 2892407 | 356 | UU3CgiPd99Lea-Y0ubgdsI6g | accounting | [UCYJLdSmyKoXCbnd-pklMn5Q] |
| 6 | We Profit Day and Night with Stock Curry | 63500 | 4377898 | 541 | UUxFRGG-_23Kqxe0YexDc1eg | finance | [UC3CgiPd99Lea-Y0ubgdsI6g] |

| video_id | channelTitle | title | description | publishedAt | viewCount | likeCount | commentCount | duration |
|----------|--------------|--|---|--|----------------------|-----------|--------------|--------------|
| 0 | FeANJVZNFTA | Accounting Stuff | TAX BRACKETS: a Simple Guide for Beginners | 🌟 Tax Brackets Cheat Sheet → https://accounting... | 2023-02-13T11:00:09Z | 7842 | 301 | 67 PT6M16S |
| 1 | AMXGBH7hoJY | Accounting Stuff | TAX BASICS: a Beginner's Guide to Everything | 🌟 Tax Basics Cheat Sheet → https://accounting... | 2023-01-17T11:00:05Z | 16322 | 657 | 112 PT19M58S |
| 2 | Fi1wkUczuyk | Accounting Stuff | FINANCIAL STATEMENTS: all the basics in 8 MINS! | 🌟 Financial Statement Cheat Sheets → https://ac... | 2022-10-11T11:00:18Z | 134421 | 3988 | 333 PT9M6S |
| 3 | 3W_LwpeG8c8 | Accounting Stuff | FINANCIAL RATIOS: How to Analyze Financial Sta... | 🌟 Financial Ratios Cheat Sheets → https://accou... | 2022-08-15T11:00:03Z | 142001 | 4249 | 242 PT23M57S |
| 4 | Kh7bfwY89Go | Accounting Stuff | RETURN ON ASSETS (ROA) Financial Ratios | 🌟 Profitability Ratios Cheat Sheet → https://ac... | 2022-05-30T10:00:18Z | 18581 | 617 | 98 PT7M8S |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3196 | o5QFiTvhaWI | We Profit Day and Night with Stock Curry | NIO, PSTH, MARA - Top YouTuber Stock Picks for... | I watched hours of YouTube videos from the top... | 2021-01-08T04:44:46Z | 1365 | 110 | 25 PT16M59S |
| 3197 | pi7yAGEmkHI | We Profit Day and Night with Stock Curry | SKLZ, NIO, SPCE - Top YouTuber Stock Picks for... | I watched hours of YouTube videos from the top... | 2021-01-07T05:21:23Z | 594 | 55 | 12 PT17M59S |
| 3198 | vt4No-TuJE4 | We Profit Day and Night with Stock Curry | CHEK, SKLZ, BNGO, SBE, FVRR - Top YouTuber Sto... | I watched hours of YouTube videos from the top... | 2021-01-06T05:36:20Z | 966 | 95 | 21 PT17M47S |
| 3199 | uxqBU_uMD3k | We Profit Day and Night with Stock Curry | NIO, JAGX, BNGO - Top YouTuber Stock Picks for... | I watched hours of YouTube videos from the top... | 2021-01-05T04:35:33Z | 1379 | 125 | 19 PT21M |
| 3200 | vzbPaPJYr3A | We Profit Day and Night with Stock Curry | BNGO, NIO, FUBO - Top YouTuber Stock Picks for... | I watched hours of YouTube videos from the top... | 2021-01-04T03:13:53Z | 1518 | 92 | 13 PT32M7S |

3201 rows × 9 columns

3.1.2. Tiền xử lý dữ liệu

Bộ dữ liệu video Youtube sau khi tiến hành kiểm tra thì phát hiện có một vài cột có xuất hiện giá trị bị thiếu như biến description, viewCount, likeCount, commentCount. Nhưng vì những giá trị thiếu này chỉ chiếm một tỷ lệ rất nhỏ (dưới 20%) nên nhóm đã bỏ đi các giá trị không có ý nghĩa áy để thực hiện các bước tiếp theo.

| video_id | channelTitle | title | description | publishedAt | viewCount | likeCount | commentCount | duration |
|-------------------|--------------|--------|-------------|-------------|-----------|-----------|--------------|----------------|
| Total | 0 | 0 | 0 | 102 | 0 | 2 | 2 | 6 0 |
| Percent(%) | 0.0 | 0.0 | 0.0 | 3.186504 | 0.0 | 0.06248 | 0.06248 | 0.187441 0.0 |
| Types | object | object | object | object | object | float64 | float64 | float64 object |

Định dạng lại một số cột dữ liệu nhằm phục vụ cho việc trực quan hóa dữ liệu.

```

cols = ['viewCount', 'likeCount', 'commentCount']
video_df[cols] = video_df[cols].apply(pd.to_numeric, errors='coerce', axis=1)

# Create publish day (in the week) column
video_df['publishedAt'] = video_df['publishedAt'].apply(lambda x: parser.parse(x))
video_df['pushblishDayName'] = video_df['publishedAt'].apply(lambda x: x.strftime("%A"))
# convert duration to seconds
video_df['durationSecs'] = video_df['duration'].apply(lambda x: isodate.parse_duration(x))
video_df['durationSecs'] = video_df['durationSecs'].astype('timedelta64[s]')
# Comments and likes per 1000 view ratio
video_df['likeRatio'] = video_df['likeCount']/ video_df['viewCount'] * 1000
video_df['commentRatio'] = video_df['commentCount']/ video_df['viewCount'] * 1000
# Title character length
video_df['titleLength'] = video_df['title'].apply(lambda x: len(x))

```

Cụ thể:

- publishDayName: Ngày upload video trong tuần
- durationSecs: Chuyển đổi thời lượng thành giây
- likeRatio: Lượt thích trên tổng lượt xem
- commentRatio: Bình luận trên tổng lượt xem
- titleLength: Độ dài của title video

3.1.3. Xử lý biến text

Để khai thác được những thông tin từ các biến text một cách chính xác nhất, đầu tiên ta cần làm sạch các dữ liệu ấy. Tạo hàm ‘cleaning’ với bộ dữ liệu data chứa các câu lệnh: loại bỏ các yếu tố không có ý nghĩa như: hashtags, links, các ký tự đặc biệt,..., tokenize (ngăn cách các từ trong văn bản bởi dấu phẩy), Remove Puncs (loại bỏ đi các dấu chấm câu), Removing Stopwords (loại bỏ các từ như “the”, “a”, “an”, “in”,...), lemmatization (giảm số lượng từ), cuối cùng joining (return lại dữ liệu biến text sau khi đã xử lý).

```

def cleaning(data):

    data = data.lower() # Preprocessing The Data
    data = re.sub(r'\B#\S+', '', str(data)) # Code to remove the Hashtags from the text
    data = re.split('https:\/\.*|http:\/\.*', str(data)) # Code to remove the links from the text
    data = ' '.join(re.findall(r'\w+', str(data))) # Code to remove the Special characters from the text
    data = re.sub(r'\s+[a-zA-Z]\s+', ' ', str(data)) # Code to remove all the single characters in the text
    data = re.sub(r'\s+', ' ', str(data), flags=re.I) # Code to substitute the multiple spaces with single spaces
    data = re.sub('@[\^\s]+', '', str(data)) # Remove the twitter handlers
    data = re.sub(r"HTTP\S+", "", data)

    #1. Tokenize
    text_tokens = word_tokenize(data)

    #2. Remove Puncs
    tokens_without_punc = [w for w in text_tokens if w.isalpha()]

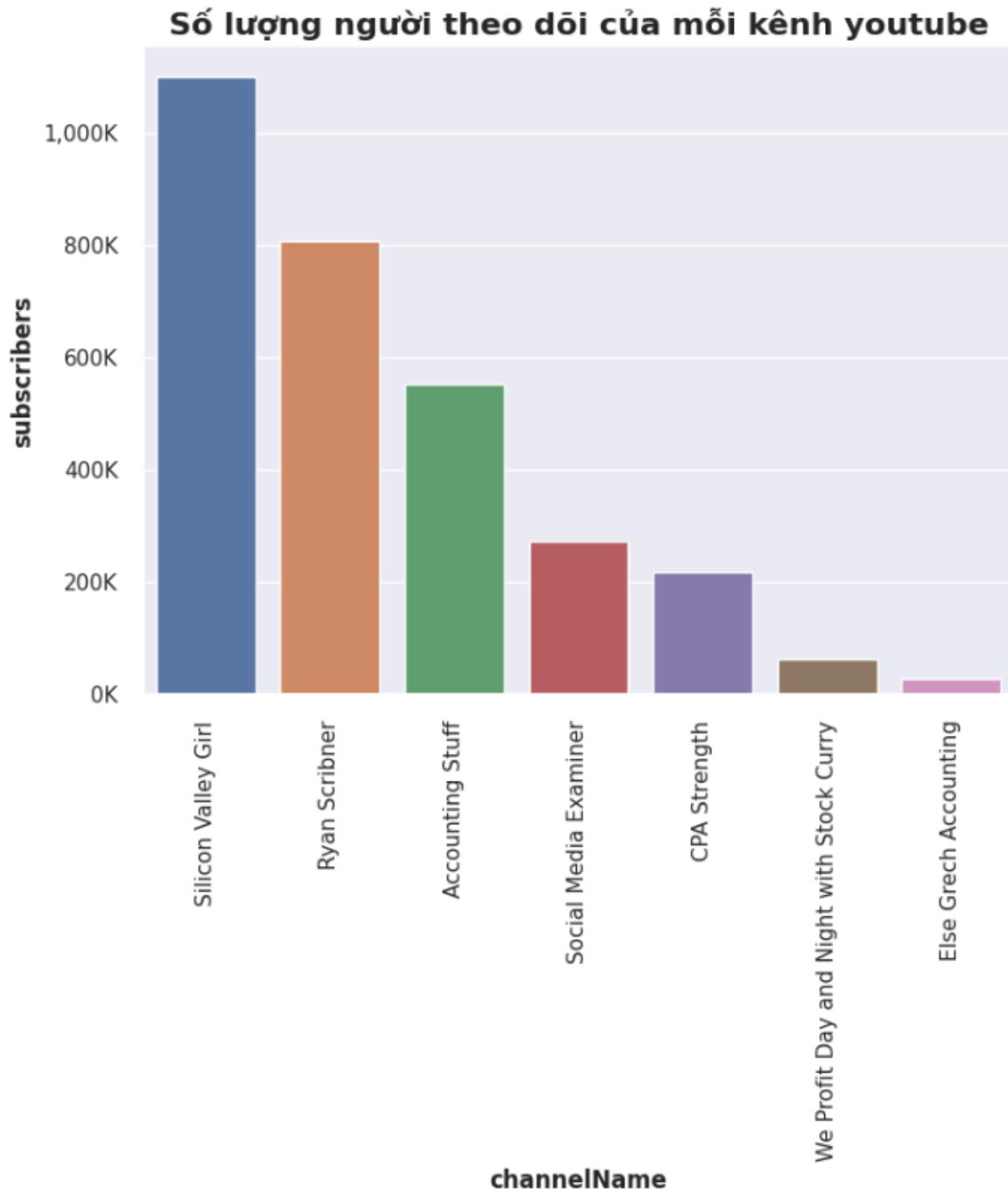
    #3. Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #4. lemma
    text_cleaned = [WordNetLemmatizer().lemmatize(t) for t in tokens_without_sw]

    #joining
    cleaning = " ".join(text_cleaned)
    return cleaning

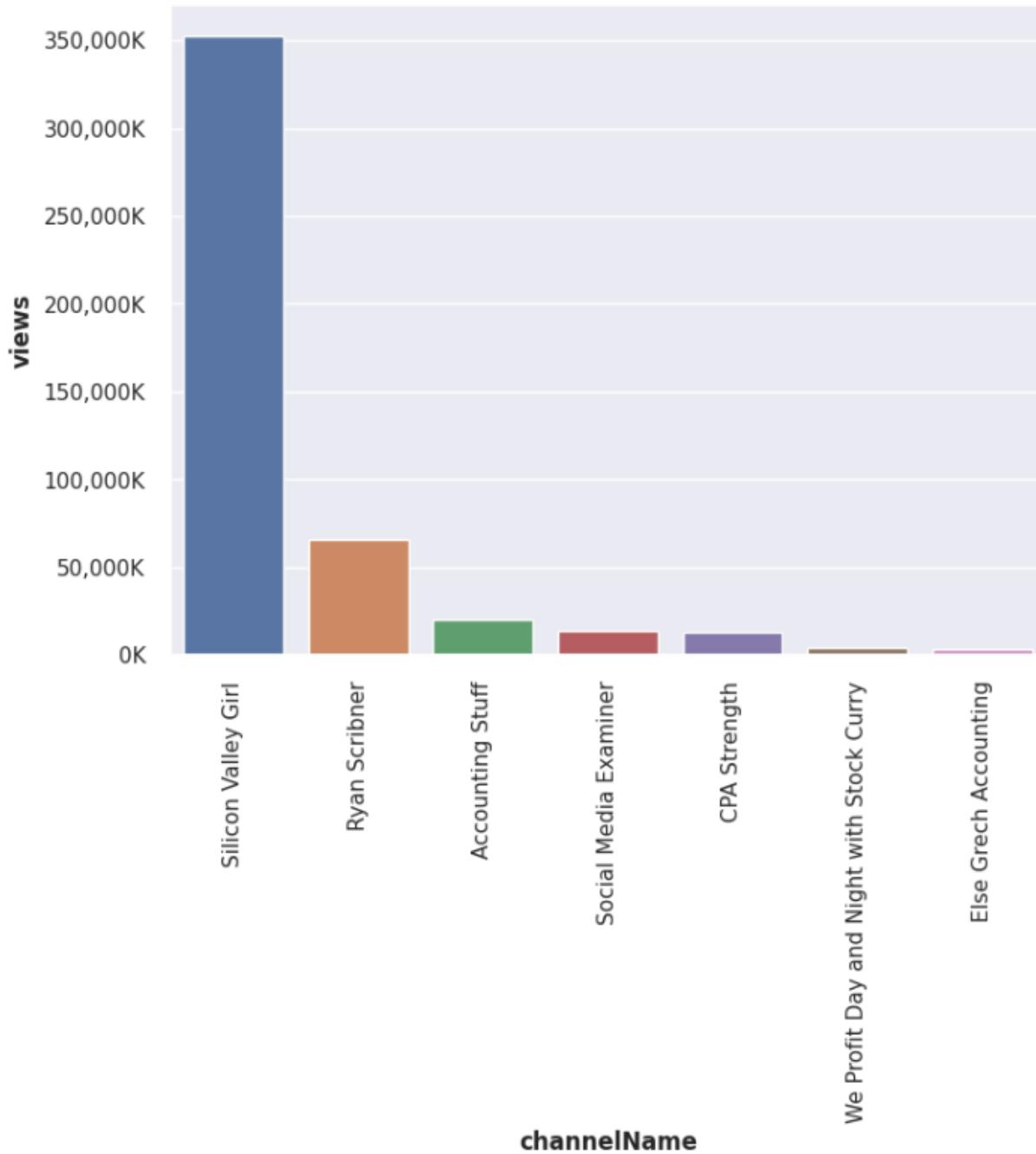
```

3.1.4. Trực quan hóa dữ liệu (EDA)

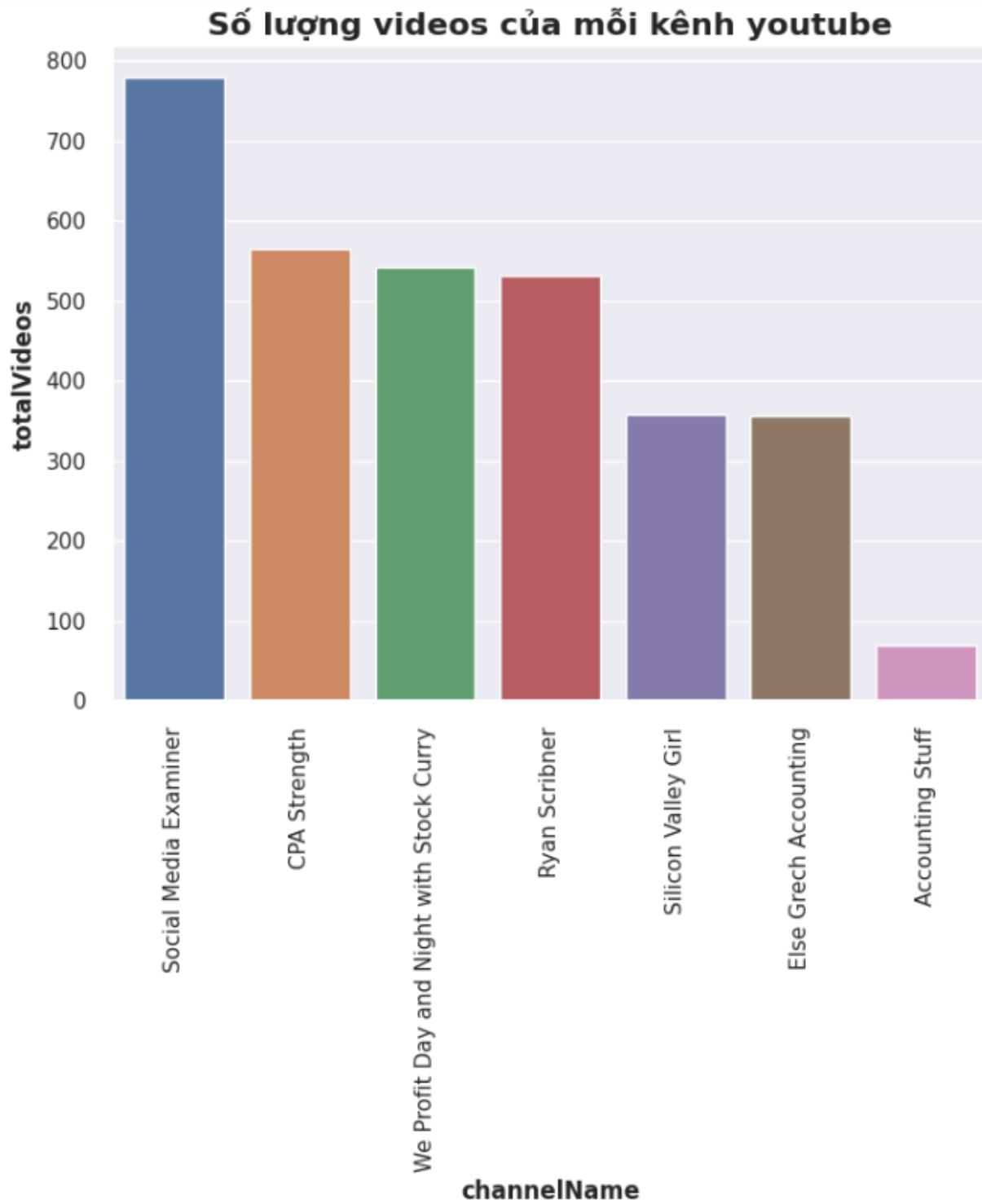


Biểu đồ trên cho thấy số lượt theo dõi của mỗi kênh Youtube trong bộ dữ liệu. Tổng quan lượt theo dõi của các kênh có sự chênh lệch rõ ràng. Kênh Silicon Valley Girl có tổng lượt theo dõi chạm mốc khoảng 1,5 triệu subscribe chiếm tỷ lệ cao nhất. Đứng thứ hai là kênh Ryan Scribner đạt 800 nghìn lượt subscribes. Chiếm tỷ lệ thấp nhất là kênh Else Grech Accounting với 20 nghìn lượt subscribes.

Số lượng views của mỗi kênh youtube

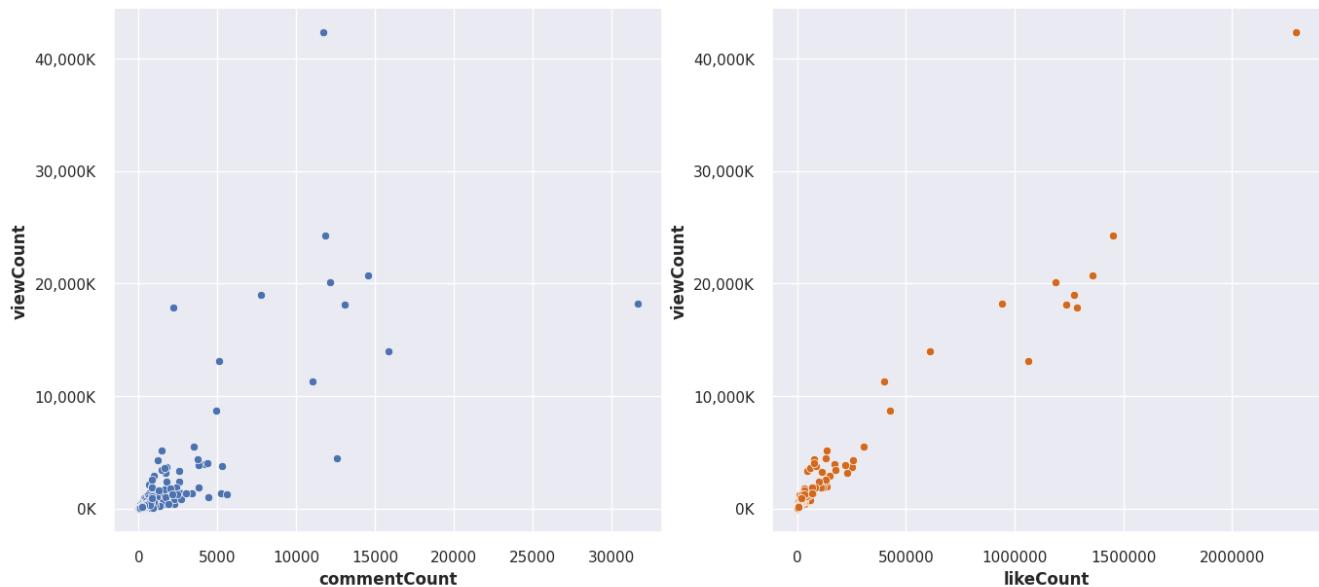


Biểu đồ thể hiện số lượng views của mỗi kênh youtube ta thấy không có sự thay đổi gì trong thứ tự so với biểu đồ thể hiện số lượt người theo dõi của các kênh Youtube. Số lượng views của Silicon Valley Girl tổng cộng là 350 triệu - đây cũng là kênh có số lượng views cao nhất. Với các kênh còn lại chỉ có tổng cộng khoảng từ 60 triệu views trở xuống.



Với biểu đồ này thứ tự của các kênh đã có sự thay đổi rõ rệt. Kênh có số video đăng tải nhiều nhất là Social Media Examiner với gần khoảng 800 video. Đứng thứ hai là kênh CPA Strength có hơn 550 video được đăng tải. Thấp nhất là kênh Accounting Stuff với 70 video.

Bình luận và lượt thích có tương quan với lượt xem hay không?



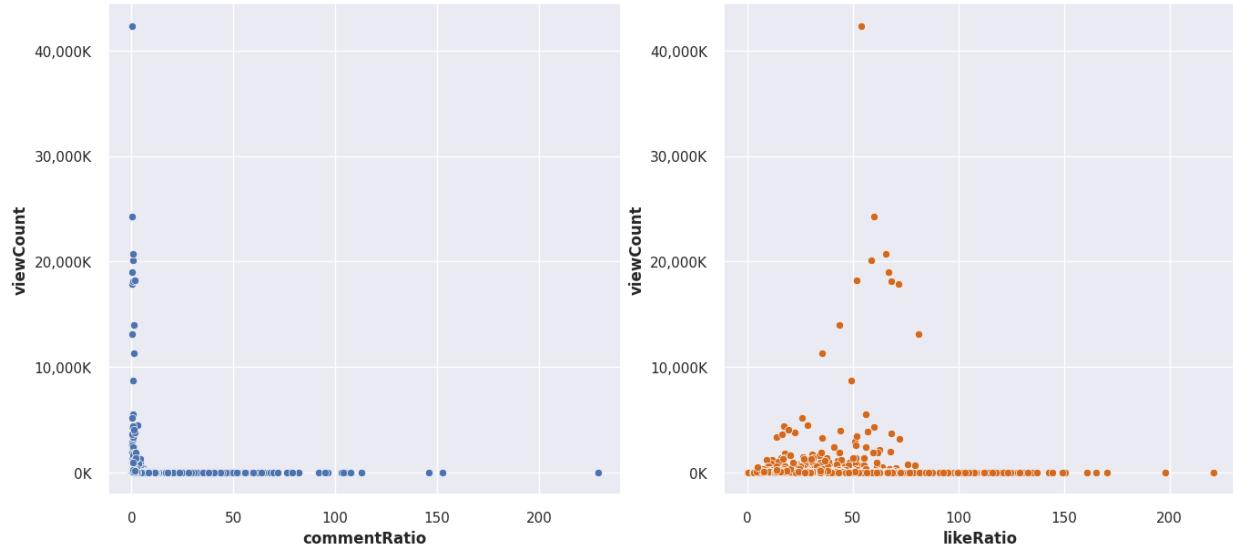
+ Từ biểu đồ phân tán giữa biến lượt bình luận và lượt xem, có vẻ như có một mối tương quan thuận giữa hai biến. Điều này có nghĩa là khi số lượng bình luận về video tăng lên, thì số lượt xem mà video đó nhận được cũng có xu hướng tăng lên. Tuy nhiên, mối tương quan giữa lượt bình luận và lượt xem dường như không mạnh lắm, vì có khá nhiều thay đổi trong các điểm dữ liệu. Dường như có nhiều video nhận được số lượt xem tương đối cao nhưng lại có số lượng bình luận thấp, điều này có thể cho thấy rằng có những yếu tố khác ngoài số lượng bình luận ảnh hưởng đến mức độ phổ biến của một video.

Nhìn chung, mặc dù có mối quan hệ thuận giữa số lượng bình luận và lượt xem, nhưng mối quan hệ này không đặc biệt mạnh và các yếu tố khác có thể đóng vai trò quyết định mức độ phổ biến của video.

+ Biểu đồ thứ hai cho thấy mối quan hệ giữa số lượt thích và lượt xem. Tương tự như biểu đồ thứ nhất, dường như có mối tương quan thuận giữa hai biến. Khi số lượt thích tăng lên thì số lượt xem cũng có xu hướng tăng lên. Tuy nhiên, không giống như biểu đồ đầu tiên, mối tương quan giữa likeCount và viewCount dường như mạnh hơn, với ít biến động hơn trong các điểm dữ liệu.

Nhìn chung, có vẻ như số lượt thích có thể là yếu tố dự báo mạnh mẽ hơn về mức độ phổ biến của video so với số lượng nhận xét, vì có mối tương quan tích cực mạnh mẽ hơn giữa số lượt thích và lượt xem. Tuy nhiên, như với biểu đồ đầu tiên, có thể có các yếu tố khác cũng ảnh hưởng đến mức độ phổ biến của video.

Tỷ lệ bình luận và lượt thích có tương quan với lượt xem hay không?

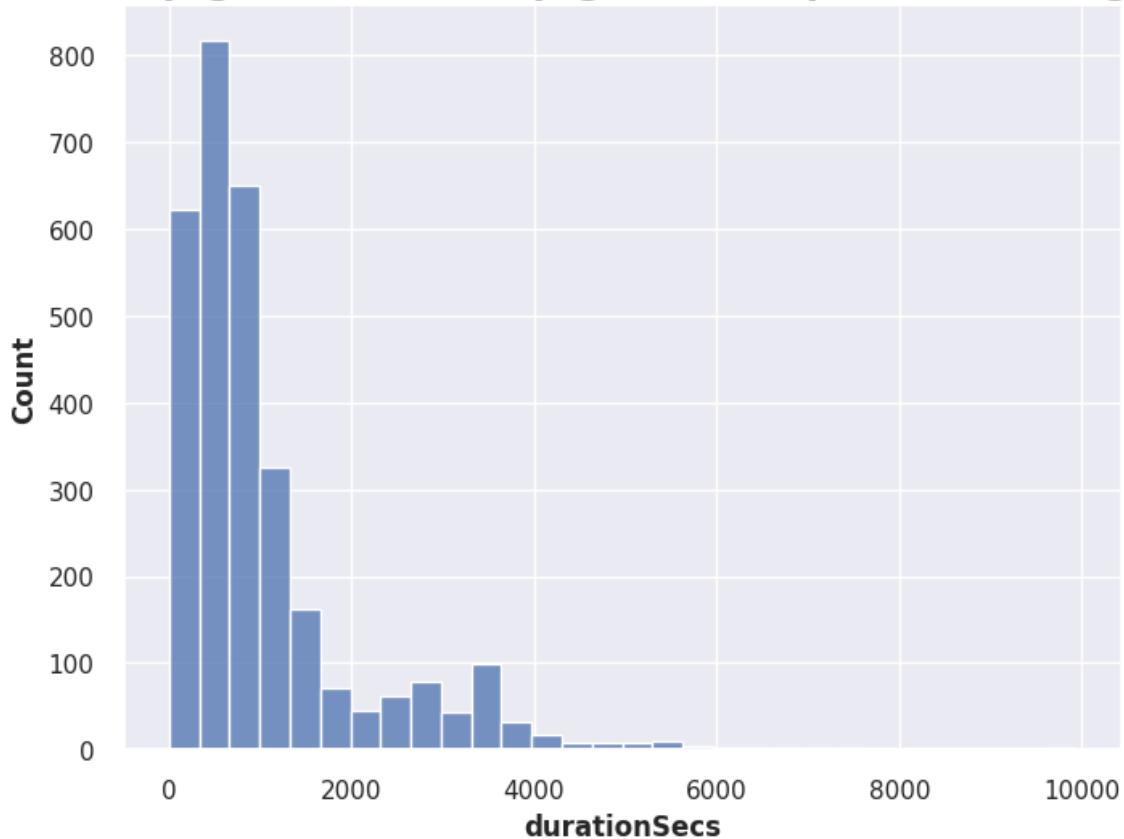


+ Biểu đồ phân tán của commentRatio và viewCount cho thấy rằng tỷ lệ bình luận của một video không có tương quan mạnh với lượt xem của video. Có thể thấy rằng dù video có tỷ lệ bình luận cao hoặc thấp thì vẫn có thể có lượt xem cao hoặc thấp tùy thuộc vào các yếu tố khác của video. Tuy nhiên, có một số video có tỷ lệ bình luận cao hơn so với các video khác và có lượt xem thấp hơn so với các video có tỷ lệ bình luận thấp hơn, cho thấy rằng tỷ lệ bình luận có thể ảnh hưởng đến lượt xem một cách nhất định, nhưng không phải là tương quan mạnh giữa chúng.

+ Biểu đồ phân tán của likeRatio và viewCount cho thấy một mối tương quan tích cực giữa tỷ lệ lượt thích và lượt xem của video. Cụ thể, chúng ta có thể thấy rằng các video có tỷ lệ lượt thích cao hơn thường có số lượt xem cũng cao hơn, và ngược lại, các video có tỷ lệ lượt thích thấp thường có số lượt xem thấp hơn. Tuy nhiên, có một số điểm ngoại lệ, tức là có một số video có tỷ lệ lượt thích rất cao nhưng số lượt xem thấp hơn so với một số video có tỷ lệ lượt thích thấp hơn nhưng lại có số lượt xem cao hơn.

Nhìn chung, từ 2 biểu đồ trên có thể thấy rằng tỷ lệ bình luận và like có mối tương quan với lượt xem, tuy nhiên mối tương quan này không rõ ràng, có rất nhiều điểm dữ liệu phân tán quanh đường tuyến tính.

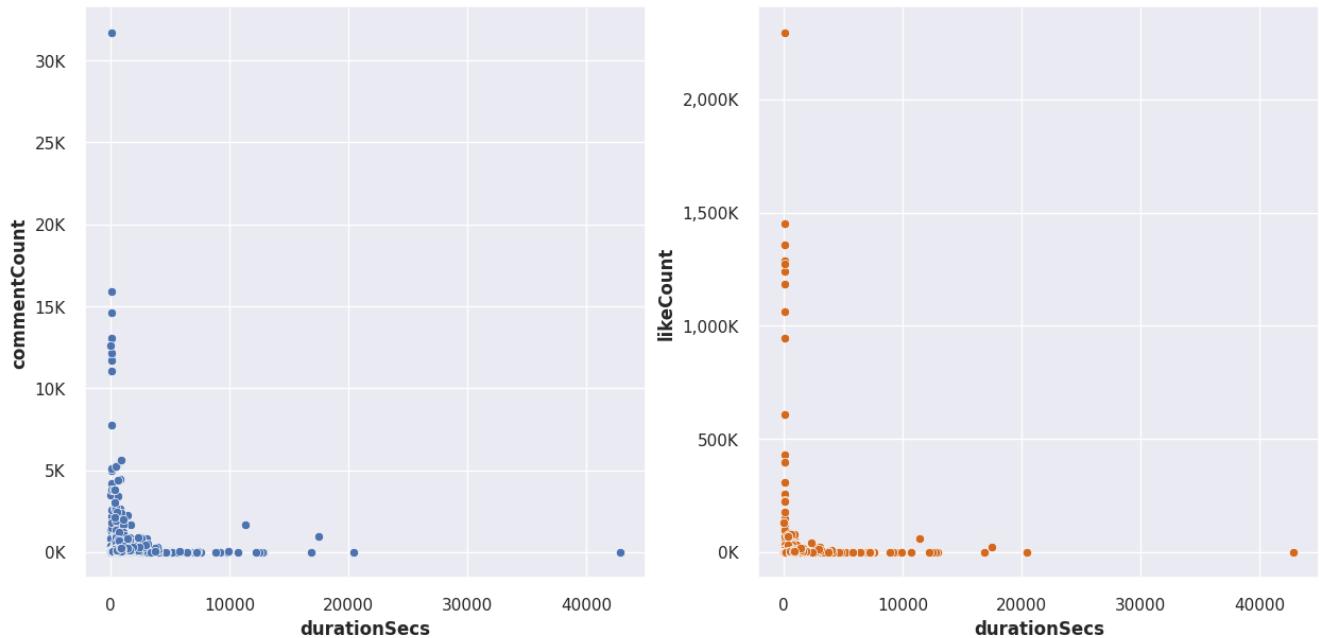
Thời lượng video có tác động đối với lượt xem và tương tác?



| Phần trăm | |
|---------------|-----------|
| (0, 2000] | 85.571013 |
| (2000, 4000] | 11.679068 |
| (4000, 6000] | 1.811711 |
| (6000, 8000] | 0.355872 |
| (8000, 10000] | 0.161760 |

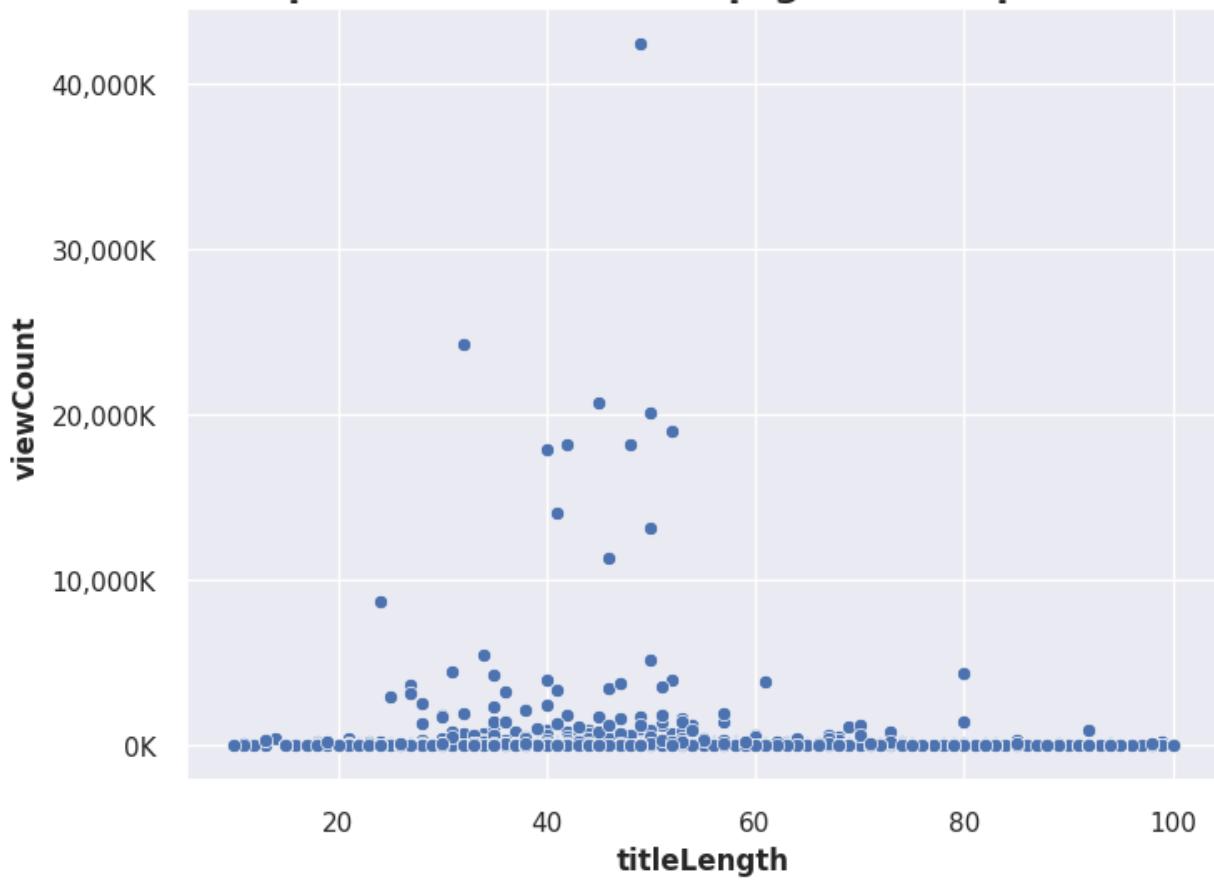
Nhìn vào bảng số liệu thống kê được, có thể thấy rằng số lượng video có thời lượng dưới 2000 giây (khoảng 33 phút) chiếm tỷ trọng lớn nhất với 85.57%. Điều này cho thấy rằng hầu hết các video trên YouTube có thời lượng ngắn.

Thời lượng video có tác động đối với lượt xem và tương tác?



Biểu đồ trên cho thấy sự tương quan giữa thời lượng video và số lượng bình luận, số lượng lượt thích của video. Với biểu đồ bên trái, ta thấy số lượng bình luận tăng dần khi thời lượng video tăng lên, nhưng với biểu đồ bên phải, ta thấy sự tương quan giữa số lượng lượt thích và thời lượng video không rõ ràng, có thể do một số video có thời lượng ngắn nhưng lại có nội dung thu hút nhiều lượt thích. Tuy nhiên, nhìn chung có thể thấy rằng những video có thời lượng dưới 1500 giây (tương đương khoảng 25 phút) thường có số lượng bình luận và lượt thích cao hơn so với các video có thời lượng ngắn hơn hoặc dài hơn.

Độ dài tiêu đề có tác động lên số lượt xem?



Phần trăm

(40, 60] 40.375283

(80, 100] 23.422841

(60, 80] 17.728890

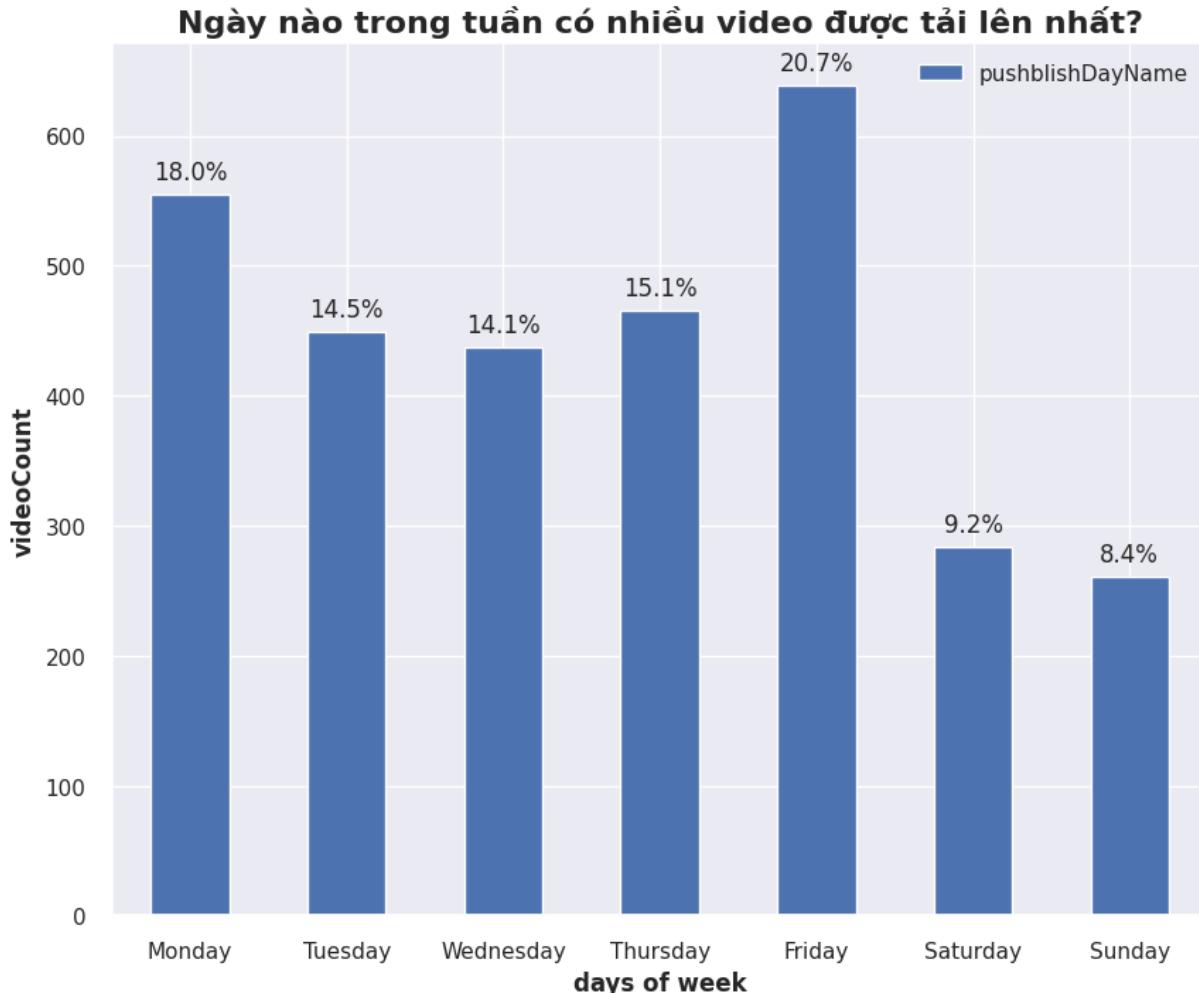
(20, 40] 17.308314

(0, 20] 1.164672

Từ bảng thống kê số liệu thu được ta có:

- Khoảng từ 40 - 60 chiếm khoảng 40.38% tổng số video.
- Khoảng từ 80 - 100 chiếm khoảng 23.42% tổng số video.
- Khoảng từ 60 - 80 chiếm khoảng 17.72% tổng số video.
- Khoảng từ 20 - 40 chiếm khoảng 17.31% tổng số video.
- Khoảng từ 0 - 20 chiếm khoảng 1.16% tổng số video.

Biểu đồ phân tán cho thấy không có mối tương quan rõ ràng giữa độ dài tiêu đề và số lượt xem. Có thể thấy rằng độ dài tiêu đề phổ biến là khoảng từ 40 đến 60 ký tự (khoảng gần một nửa tổng số video) và số lượt xem phân bố rộng rãi từ vài trăm nghìn đến vài triệu lượt xem.



Biểu đồ này thể hiện số lượng video được tải lên trên YouTube theo từng ngày trong tuần. Cụ thể, biểu đồ cột cho thấy thứ Sáu là ngày có nhiều video được tải lên nhất (chiếm 20.7% tổng số video). Tiếp theo là thứ Hai và thứ Năm lần lượt chiếm 18% và 15.1% trong tổng số video. Trong khi đó, thứ Bảy và Chủ Nhật là 2 ngày có ít video được tải lên nhất với 9.2% và 8.4% trong tổng số video.

3.1.5. WordCloud

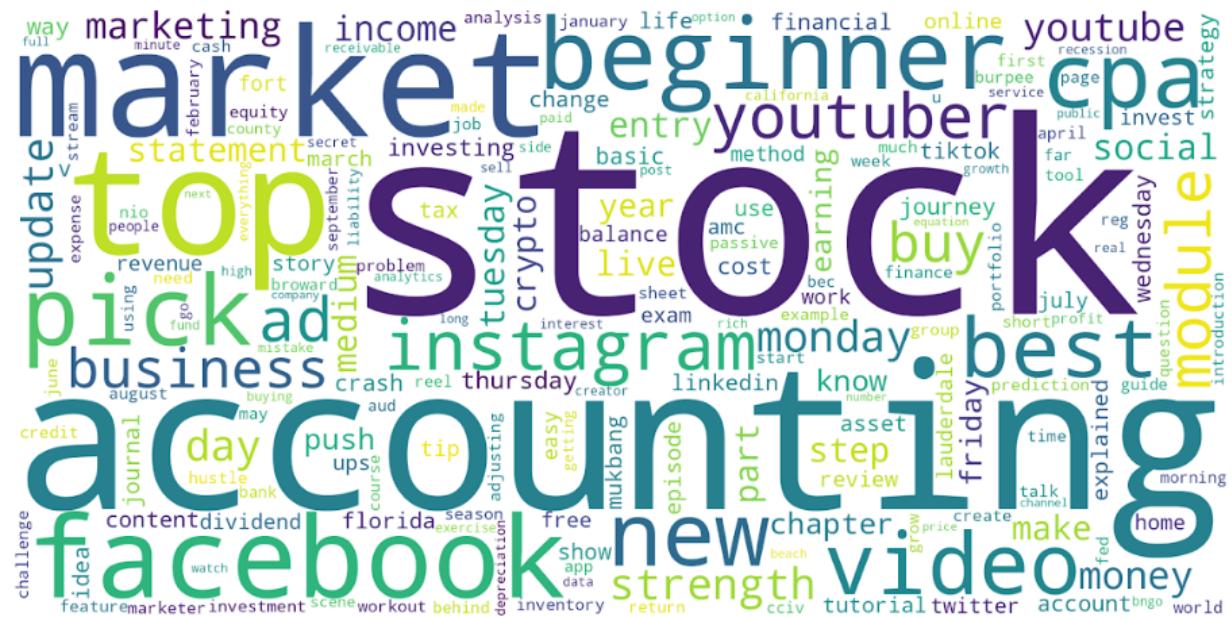
a. WordCloud của biến description



WordCloud trên được tạo ra từ các từ xuất hiện trong phần mô tả của các video trên YouTube. Biểu đồ này trực quan hóa các từ được sử dụng nhiều nhất trong mô tả của các video.

Từ các từ xuất hiện trên WordCloud, có thể thấy rằng các từ như "video", "beginners accounting", "stock", "marketing", "investment", "free",... là những từ phổ biến trong các mô tả video trên các kênh YouTube crawling ban đầu. Điều này cho thấy rằng các từ này là những từ quan trọng trong việc thu hút người xem và tăng tương tác của người dùng trên kênh YouTube.

b. WordCloud của biến title



WordCloud trên được tạo ra từ các từ xuất hiện trong tiêu đề của các video trên YouTube. Biểu đồ này trực quan hóa các từ được sử dụng nhiều nhất trong tiêu đề của các video.

Từ các từ xuất hiện trên WordCloud, có thể thấy rằng các từ như "accounting", "stock", "market", "video", "top" và "new",... là những từ xuất hiện nhiều nhất trong tiêu đề của các video. Các từ này có thể là những từ quan trọng để thu hút người xem và tăng cường sự quan tâm của khán giả đến video.

⇒ Tổng quan, WordCloud này là một cách thú vị để trực quan hóa và khám phá các từ phổ biến xuất hiện trong tiêu đề video trên YouTube, từ đó giúp người dùng có thể nắm bắt được xu hướng và sở thích của khán giả trên nền tảng này.

3.1.6. LDA + BERT

Để có thể thực hiện các mô hình thuật toán ở phía dưới, trước hết chúng ta cần giải quyết vấn đề cấp thiết nhất là biến đổi, xử lý các biến text. Dùng phương pháp LDA và BERT để biến đổi các văn bản của biến text thành các vector đại diện.

```
data = pd.read_csv("/content/video_data_channels_V2.csv", index_col = 0)
data["clean_text"] = data["title"].apply(lambda x: simple_preprocess(x))
dictionary = Dictionary(data["clean_text"])
corpus = [dictionary.doc2bow(text) for text in data["clean text"]]
```

Đọc dữ liệu từ tệp "video_data_channels_V2.csv" và lưu trữ vào biến data. Sau đó, tiến hành tạo một cột mới trong data là "clean_text", bằng cách áp dụng hàm 'simple_preprocess' từ 'gensim.utils' lên cột "title" trong data. 'simple_preprocess' là một hàm xử lý văn bản đơn giản, được sử dụng để tách từ và loại bỏ các ký tự đặc biệt.

Tiếp đó, sử dụng lớp 'Dictionary' từ 'gensim.corpora' để tạo từ điển từ tất cả các từ xuất hiện trong cột "clean_text". Mỗi từ được mã hóa thành một số nguyên duy nhất để có thể đưa vào mô hình LDA. Câu lệnh 'corpus = [dictionary.doc2bow(text) for text in data["clean_text"]]' để tạo một danh sách các tài liệu, mỗi tài liệu là một danh sách các cặp (id_từ, số_lần_xuất_hiện_từ_trong_tài_liệu), tương ứng với mỗi tài liệu trong data["clean_text"].

```
# Train LDA model and obtain topic assignment vectors
lda_model = LdaModel(corpus, num_topics=10, id2word=dictionary, passes=10)
lda_vectors = []
for i, text in enumerate(data["clean_text"]):
    lda_vector = lda_model.get_document_topics(corpus[i])
    lda_vector = [v[1] for v in lda_vector]
    lda_vectors.append(lda_vector)
```

Mô hình LDA được huấn luyện bằng cách sử dụng LdaModel từ gensim.models. Để tạo mô hình này, chúng ta cần truyền vào corpus (tài liệu đã được biểu diễn dưới dạng từ điển), num_topics (số lượng chủ đề cần được tìm thấy) và id2word (từ điển mà ta đã tạo trong bước trước). passes chỉ định số lần lặp lại để huấn luyện mô hình.

Sau khi mô hình LDA đã được huấn luyện, đoạn code sử dụng lda_model.get_document_topics(corpus[i]) để lấy ra danh sách các chủ đề và mức độ xuất hiện của chúng trong tài liệu thứ i trong corpus. Kết quả được lưu trữ trong lda_vector dưới dạng một danh sách các cặp (id_chủ_đề, mức độ xuất hiện chủ đề trong tài liệu). Chỉ số thứ hai của mỗi cặp được lưu trữ trong danh sách lda_vector bằng cách sử dụng v[1] từ v là biến lặp lại lda_vector. Đoạn code cuối cùng là lưu trữ các mức độ xuất hiện của các chủ đề trong tất cả các tài liệu vào danh sách lda_vectors.

```

# Clean and pre-process data for BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = BertModel.from_pretrained('bert-base-uncased')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
bert_model.to(device)

bert_vectors = []
for text in data["title"]:
    input_ids = torch.tensor([tokenizer.encode(text, add_special_tokens=True)]).to(device)
    with torch.no_grad():
        outputs = bert_model(input_ids)
    bert_vector = outputs[0][:,0,:].cpu().numpy()
    bert_vector = normalize(bert_vector)
    bert_vectors.append(bert_vector[0])

```

Đoạn code này sử dụng pretrained BERT model để biểu diễn các tiêu đề video dưới dạng vector. Đầu tiên, nó sử dụng BertTokenizer từ thư viện transformers để tạo ra các mã token cho mỗi tiêu đề, thêm các token đặc biệt như [CLS] ở đầu và [SEP] ở cuối và chuyển đổi chúng thành dạng tensor input_ids của PyTorch. Sau đó, nó truyền input_ids vào BertModel để tính toán đầu ra, tức là các vector biểu diễn của các token trong tiêu đề. Nó chỉ giữ lại vector đầu tiên (tức vector biểu diễn của [CLS] token) và chuẩn hóa nó bằng cách sử dụng hàm normalize từ sklearn.preprocessing. Kết quả cuối cùng là một danh sách các vector biểu diễn, mỗi vector biểu diễn là một mảng NumPy có kích thước bằng số chiều của BERT model

```

# Concatenate LDA and BERT vectors with a weight hyperparameter
weight = 0.5
concat_vectors = []
for i in range(len(data)):
    concat_vector = np.concatenate((weight*np.array(lda_vectors[i]), (1-weight)*np.array(bert_vectors[i])))
    concat_vectors.append(concat_vector)

```

Đoạn code này thực hiện việc ghép vector đặc trưng của LDA và BERT thành một vector mới dựa trên trọng số weight (trọng số này là số thực thuộc [0;1]). Để thực hiện việc này, đầu tiên tạo ra một danh sách rỗng concat_vectors. Sau đó, với mỗi một văn bản trong tập dữ liệu, lấy vector LDA tương ứng (được lưu trong danh sách lda_vectors ở bước trước đó) và vector BERT tương ứng (được lưu trong danh sách bert_vectors ở bước trước đó), ghép chúng lại theo công thức $weight * np.array(lda_vectors[i]) + (1-weight) * np.array(bert_vectors[i])$. Kết quả là một vector mới concat_vector, sau đó vector này được thêm vào danh sách concat_vectors. Cuối cùng, danh sách concat_vectors chứa các vector mới được ghép từ LDA và BERT theo trọng số weight.

```
max_len = 100 # Maximum length of concatenated vectors  
padded_vectors = pad_sequences(concat_vectors, padding='post', maxlen=max_len, dtype='float32')
```

Đoạn code này sử dụng hàm pad_sequences từ package tensorflow.keras.preprocessing.sequence để đưa tất cả các vector có độ dài khác nhau trong concat_vectors về cùng một độ dài max_len. Các vector có độ dài nhỏ hơn max_len sẽ được thêm các giá trị 0 vào phía cuối cho đến khi độ dài bằng max_len, còn các vector có độ dài lớn hơn max_len sẽ bị cắt đi các giá trị cuối cùng để độ dài bằng max_len. Kết quả đầu ra là một ma trận 2 chiều với kích thước (số lượng vector, max_len).

```
bert_vectors_np = np.array(bert_vector)  
np.save('bert_vector_V2.npy', bert_vectors_np)
```

```
lda_vectors_np = np.array(lda_vectors)  
np.save('lda_vectors_V2.npy', lda_vectors_np)
```

```
padded_vectors_np = np.array(padded_vectors)  
np.save('padded_vectors_V2.npy', padded_vectors_np)
```

Sau khi thực hiện xong các bước, tiến hành lưu trữ các vector được tính toán vào các mảng numpy riêng biệt để sử dụng lại trong các mô hình bên dưới. Cụ thể, nó sử dụng hàm np.save() để lưu trữ các mảng numpy chứa bert_vectors, lda_vectors và padded_vectors vào các file bert_vector_V2.npy, lda_vectors_V2.npy và padded_vectors_V2.npy, tương ứng. Việc lưu trữ dữ liệu giúp tiết kiệm thời gian và tài nguyên tính toán khi chạy lại các bước trong quá trình xử lý dữ liệu.

3.2. Huấn Luyện Dữ Liệu với các mô hình

3.2.1. Traditional Clustering

a. K - Means

K-means là một thuật toán phân nhóm (clustering) dữ liệu phổ biến trong học máy và khai phá dữ liệu. Nó là một thuật toán unsupervised learning, nghĩa là không cần có dữ liệu huấn luyện được gán nhãn trước.

Mục tiêu chính của thuật toán là phân tập dữ liệu thành k cụm hay còn gọi là cluster được đại diện bởi điểm trung tâm (centroid). Mỗi điểm dữ liệu sẽ được label vào cụm có điểm trung tâm gần nhất.

Thuật toán k - mean được triển khai bằng việc xác định ngẫu nhiên các điểm trung tâm của số cụm cho trước ban đầu, sau đó thực hiện vòng lặp sau để xác định các điểm dữ liệu thuộc vào cụm nào cũng như xác định lại điểm trung tâm của từng cụm:

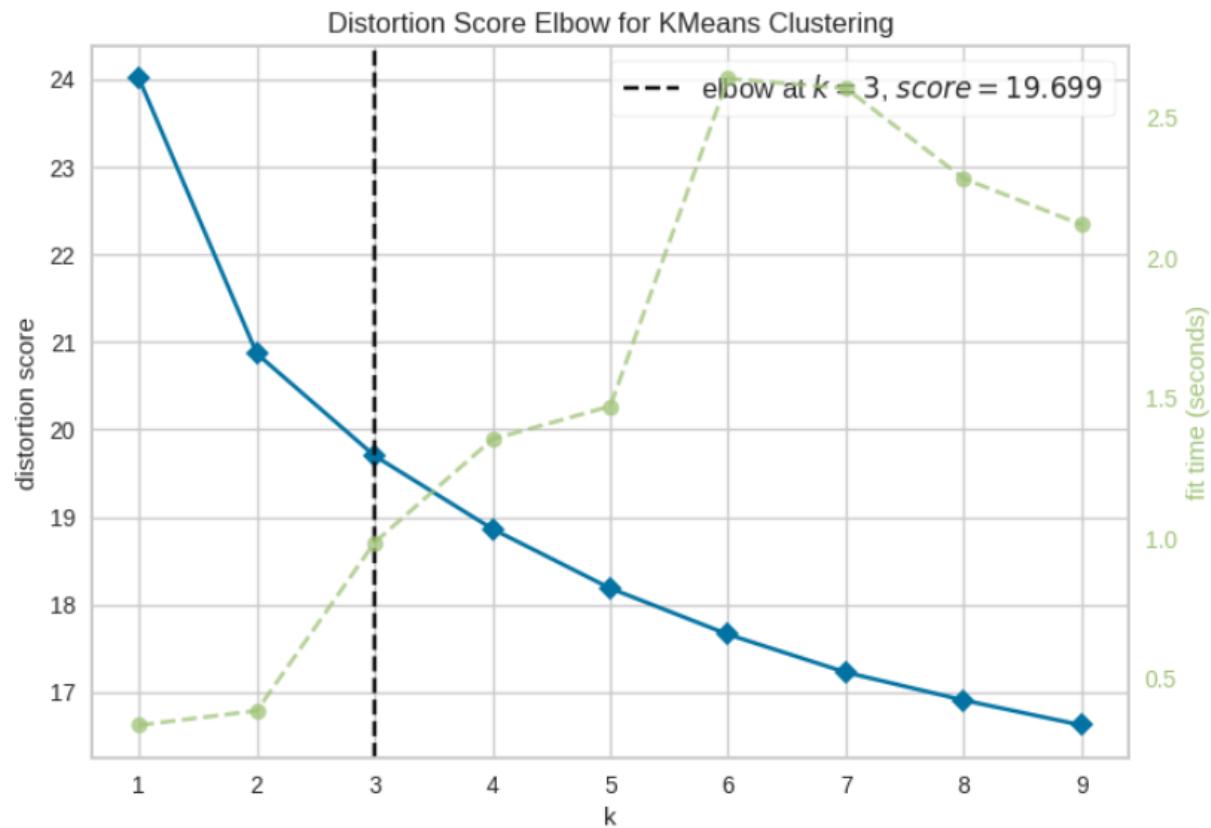
- *Gán cụm cho từng điểm dữ liệu:* Tính toán khoảng cách của từng điểm dữ liệu đến các cụm → chọn ra cụm có khoảng cách ngắn nhất để gắn cho điểm dữ liệu đó
- *Tái xác định điểm trung tâm:* Sau khi xác định được các điểm dữ liệu thuộc cụm nào, tính khoảng cách trung bình của các điểm dữ liệu trong 1 cụm để xác định lại điểm trung tâm của cụm đó

Vòng lặp này sẽ dừng lại khi các điểm trung tâm không còn có thể điều chỉnh nữa hoặc có điều chỉnh nhưng không đáng kể.

Với phương pháp k - means, ta cần xác định được số cụm ban đầu để triển khai hiệu quả. Để có được số cụm tối ưu nhất, ta áp dụng phương pháp elbow phổ biến thường gặp nhưng hiệu quả. Với bộ dữ liệu này, 3 cụm là đang được xem là tối ưu, tuy nhiên độ dốc từ 2 cụm lên 3 cụm không có sự thay đổi đáng kể so với từ 3 cụm lên 4 cụm → phương pháp phân cụm k - means đối với bộ dữ liệu này đang thể hiện không tốt và vẫn có thể cải thiện thêm.

```
padded_vectors = np.load('/content/padded_vectors_V2.npy')
```

```
Elbow_M = KElbowVisualizer(KMeans(), k=(1,10))
Elbow_M.fit(padded_vectors)
Elbow_M.show()
```



```

true_k = 3

def clustering(model, X):
    labels_pred = model.fit_predict(X)

    print(' > silhouette:', str(silhouette_score(X, labels_pred)))
    print(' > calinski harabasz:', str(calinski_harabasz_score(X, labels_pred)))
    print(' > davies bouldin:', str(davies_bouldin_score(X, labels_pred)), '\n')

    result_df = pd.DataFrame(list(zip(data['title'],labels_pred)), columns = ['text', 'kmeans_ml'])

    # Create wordclouds for clusters
    fig, ax = plt.subplots(figsize=(20, 8))

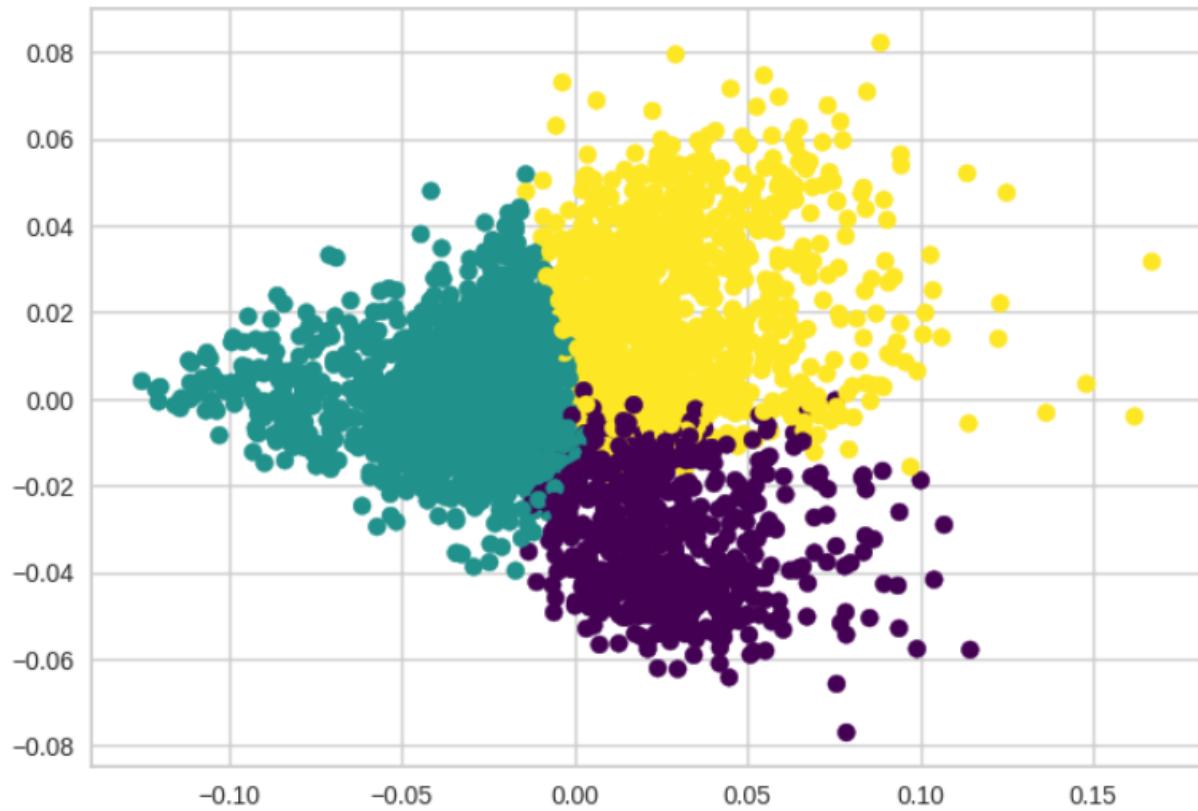
    for k in range(true_k):
        text = result_df[result_df.kmeans_ml == k]['text'].str.cat(sep=' ')
        wordcloud = WordCloud(max_font_size=50, max_words=200, background_color="white",
                              random_state=1905).generate(text)

        # Create subplot
        plt.subplot(2, 3, k+1).set_title("Cluster " + str(k))
        plt.plot()
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis("off")
        plt.show()

    # scatter plot
    print('\n')
    pca = PCA(n_components=2)
    pca_vectors = pca.fit_transform(X)
    plt.scatter(pca_vectors[:, 0], pca_vectors[:, 1], c=labels_pred, s=50, cmap='viridis')
    plt.show()

```

Tạo hàm clustering cho K-Means, với tính năng fit_predict mô hình để lấy ra predictive labels, tính toán các chỉ số, vẽ wordcloud của từng cụm, giảm chiều dữ liệu bằng PCA và vẽ scatter plot.



Kết quả phân cụm:

Ta có thể thấy với phương pháp k-means trả kết quả về 3 cụm khá mơ hồ, không thực sự rõ ràng topic của 3 cụm khi cụm nào cũng có các keyword chung như *stock*, *youtube*,... và vẫn đè chòng chéo lên nhau cũng thể hiện ở một rìa của giữa các cụm trong biểu đồ scatter phía trên.

Tuy nhiên kết quả vẫn cho chúng ta hy vọng để tiếp tục cải thiện về sau khi ở cụm 0 có thể đoán được là về chủ đề **finance** với key *stock* xuất hiện khá nhiều. Tương tự với cụm 1 là về chủ đề **marketing** với các key word quan trọng *journey* *episode*, *youtube*, *video*, v.v và cụm 2 là về **accounting** với các keyword như *cpa*, *accounting*, *balance sheet*, v.v



b. Fuzzy C - Means

Như đã đề cập ở phần trên, k-means chưa phải là phương pháp tối ưu nhất cho bộ dữ liệu này và vẫn có thể được cải thiện tốt hơn. Trước tiên ta sẽ xem qua phương pháp fuzzy c - means.

```
def fcm_clustering(model, X):
    # fit FCM model
    fcm = model
    fcm.fit(X)

    # obtain cluster centers and labels
    fcm_centers = fcm.centers
    fcm_labels = fcm.predict(X)

    # print clustering scores
    print('Silhouette:', silhouette_score(X, fcm_labels))
    print('Calinski Harabasz:', calinski_harabasz_score(X, fcm_labels))
    print('Davies Bouldin:', davies_bouldin_score(X, fcm_labels), '\n')

    result = pd.DataFrame(list(zip(data['title'],fcm_labels)), columns = ['text', 'fcm_labels'])

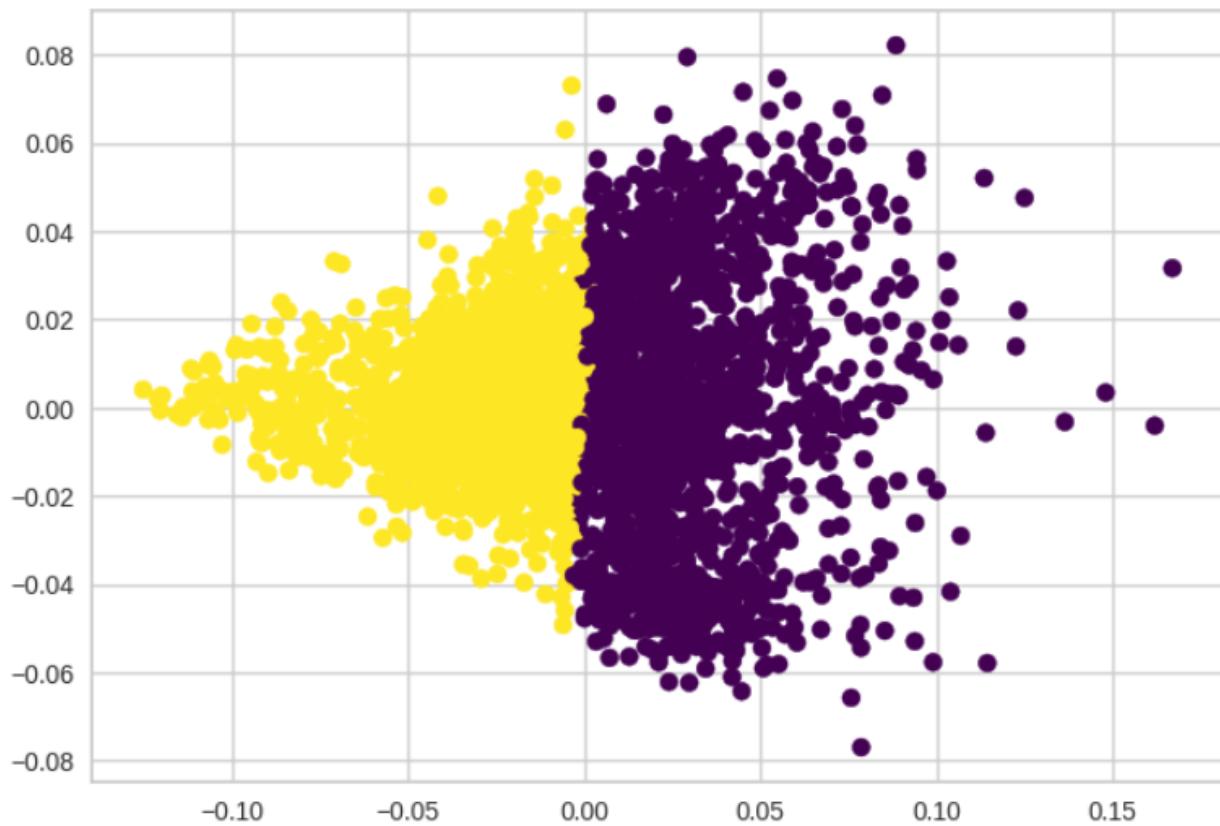
    # Create wordclouds for clusters
    from wordcloud import WordCloud

    fig, ax = plt.subplots(figsize=(20, 8))

    for k in range(true_k):
        text = result[result.fcm_labels == k]['text'].str.cat(sep=' ')
        wordcloud = WordCloud(max_font_size=50, max_words=200, background_color="white",
                             random_state=1905).generate(text)

        # Create subplot
        plt.subplot(2, 3, k+1).set_title("Cluster " + str(k))
        plt.plot()
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis("off")
        plt.show()

    # scatter plot
    print('\n')
    pca = PCA(n_components=2)
    pca_vectors = pca.fit_transform(X)
    plt.scatter(pca_vectors[:, 0], pca_vectors[:, 1], c=fcm_labels, s=50, cmap='viridis')
    plt.show()
```



Kết quả phân cụm:

Với fuzzy c - means, ta chỉ nhận lại được kết quả tối đa là 2 cụm (dù đã khởi tạo với số cụm lớn hơn là 3). Điều này có thể là do các điểm dữ liệu có điểm membership degree chỉ thuộc về tối đa là 2 cụm. Ở 2 cụm này ta thấy có sự tách biệt rõ ràng hơn trong biểu đồ scatter, không bị chồng chéo lên nhau như ở k-means truyền thống. Điều này được thể hiện rõ ràng hơn ở phần word cloud phía dưới khi ở cụm 0, ta có thể dễ dàng nhận ra chủ đề của nó là về **accounting** và **finance** với các keyword trọng điểm như *accounting, stock, cpa, business,....*. Ở cụm 1 với các keyword *video, social, youtube, instagram* ta có thể dễ dàng xác định chủ đề của nó là về **marketing**.

Tuy đã có sự cải thiện về độ chồng chéo nhau giữa các cụm nhưng 2 cụm không phải là target mà nhóm hướng tới, nên dù cho ra một kết quả khả quan hơn so với k-means truyền thống, Fuzzy c-means vẫn còn có thể cải thiện tốt hơn với các phương pháp học sâu được áp dụng đính kèm dưới đây.



3.2.2. Autoencoders

```
# Define the autoencoder model
input_layer = Input(shape=(X.shape[1],))
encoded_layer = Dense(32, activation='relu')(input_layer)
decoded_layer = Dense(X.shape[1], activation='sigmoid')(encoded_layer)
autoencoder = Model(input_layer, decoded_layer)
autoencoder.compile(optimizer='adam', loss='mse')

# Define a callback to save the best model
checkpoint = ModelCheckpoint("best_autoencoder_ver(2).hdf5", monitor='val_loss', verbose=1, save_best_only=True, mode='min')
# Train the autoencoder
autoencoder.fit(X, X, epochs=200, batch_size=128, validation_split=0.2, callbacks=[checkpoint])
# Load the best autoencoder model
autoencoder.load_weights("best_autoencoder_ver(2).hdf5")
# Obtain the encoded data
encoder = Model(autoencoder.input, autoencoder.layers[1].output)
encoded_data = encoder.predict(X)
```

Đào tạo mô hình bằng thư viện Keras mô hình này gồm có:

1. Mô hình có một lớp đầu vào (input_layer) có hình dạng giống như dữ liệu đầu vào ($X.shape[1]$)
2. Một lớp ẩn (hidden_layer) với 32 nodes và hàm kích hoạt đơn vị tuyến tính được chỉnh lưu (ReLU)
3. Lớp đầu ra (output_layer) có hình dạng giống như lớp đầu vào (input_layer) và chức năng kích hoạt SigMoid.

Sau đó, bộ tự động được biên dịch bằng trình tối ưu hóa Adam và hàm mất mát Mean Squared Error (MSE). Modelcheckpoint callback được dùng để tự động lưu mô hình tốt nhất dựa trên validation loss.

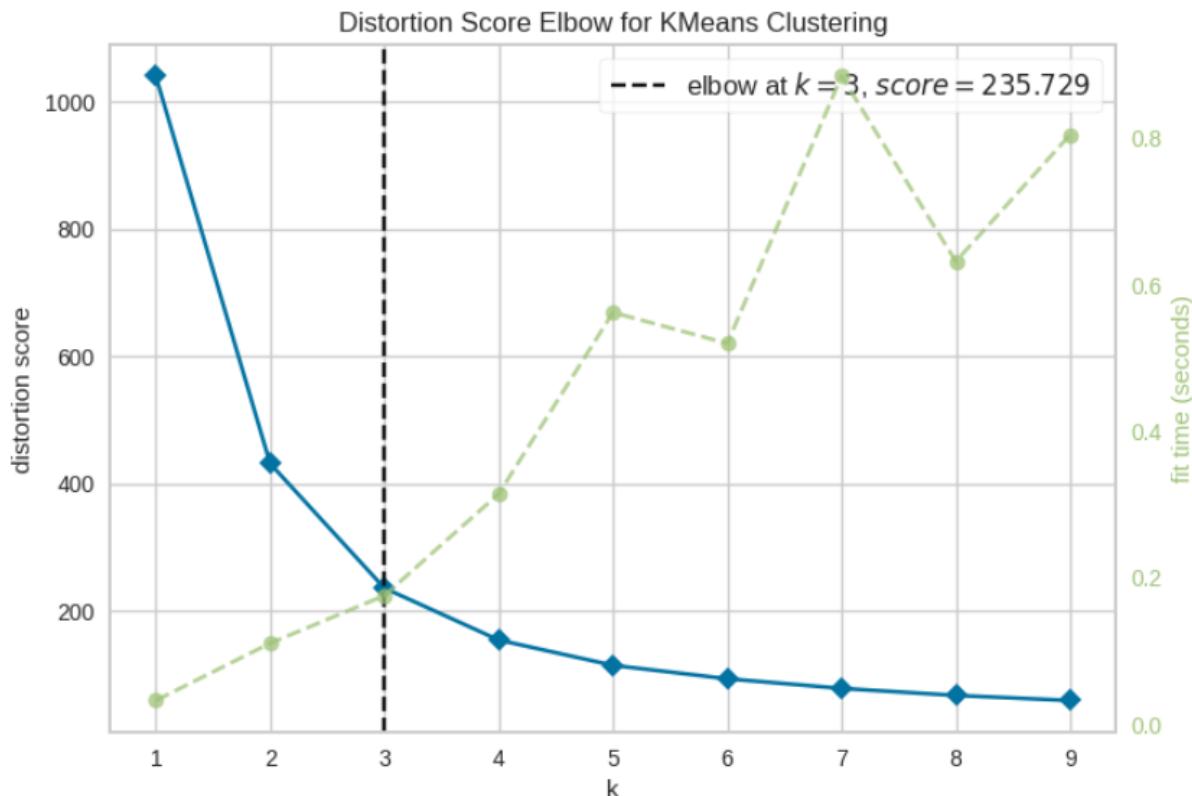
Trong quá trình đào tạo, AutoEncoder được đào tạo trên dữ liệu với epoch là 200 với batch_size là 128 và validation split là 0,2.

Sau khi đào tạo, mô hình AutoEncoder tốt nhất được load bằng phương pháp load_weights. Dữ liệu được mã hóa (encoded_data) sau đó được lấy bằng cách xác định

mô hình Keras mới có tên là encoder lấy lớp đầu vào (autoencoder input layer) và lớp ẩn đầu tiên (hidden layer) làm đầu vào của nó và đầu ra dữ liệu được mã hóa. Phương pháp dự đoán sau đó được gọi trên mô hình bộ mã hóa này để có được dữ liệu được mã hóa.

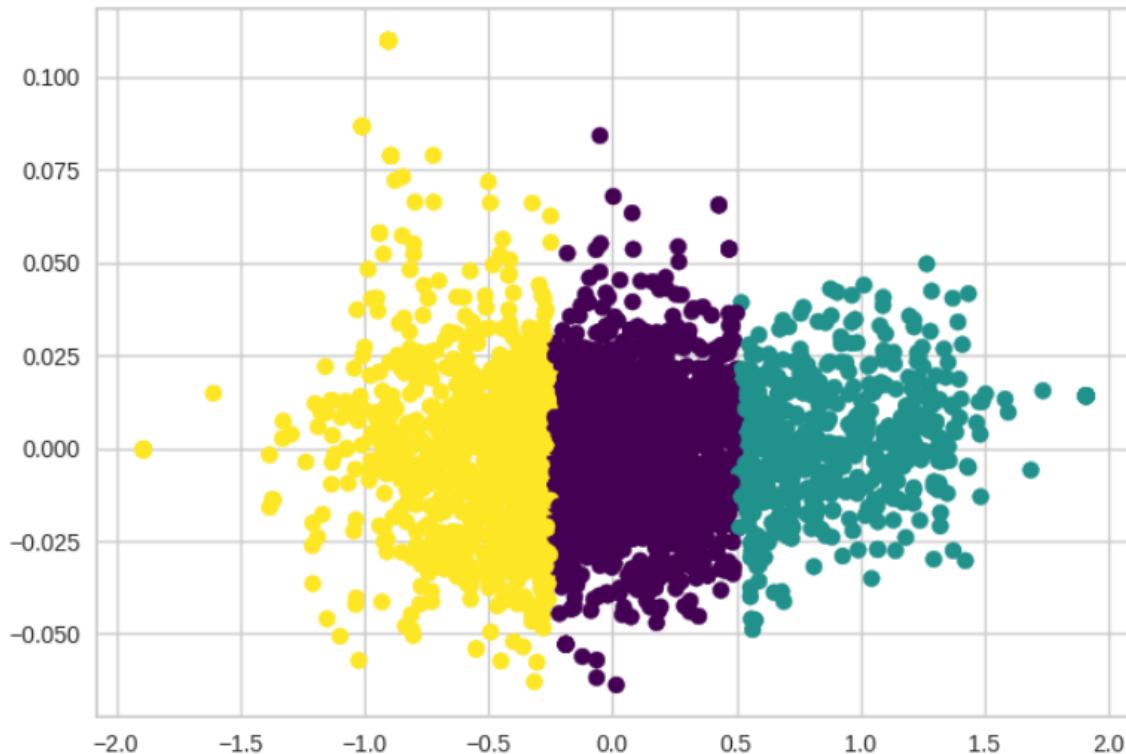
a. Phương pháp Elbow

```
Elbow_M = KElbowVisualizer(KMeans(), k=(1,10))
Elbow_M.fit(encoded_data)
Elbow_M.show()
```



Điểm khuỷu tay (điểm gãy) là điểm mà ở đó tốc độ suy giảm của hàm biến dạng sẽ thay đổi nhiều nhất. Tức là kể từ sau vị trí này thì gia tăng thêm số lượng cụm cũng không giúp hàm biến dạng giảm đáng kể. Nếu thuật toán phân chia theo số lượng cụm tại vị trí này sẽ đạt được tính chất phân cụm một cách tổng quát nhất mà không gặp các hiện tượng vị khớp (overfitting). Trong hình trên thì ta thấy vị trí của điểm khuỷu tay chính là $k = 3$ vì khi số lượng cụm lớn hơn 3 thì tốc độ suy giảm của hàm biến dạng dường như không đáng kể so với trước đó.

b. K - Means



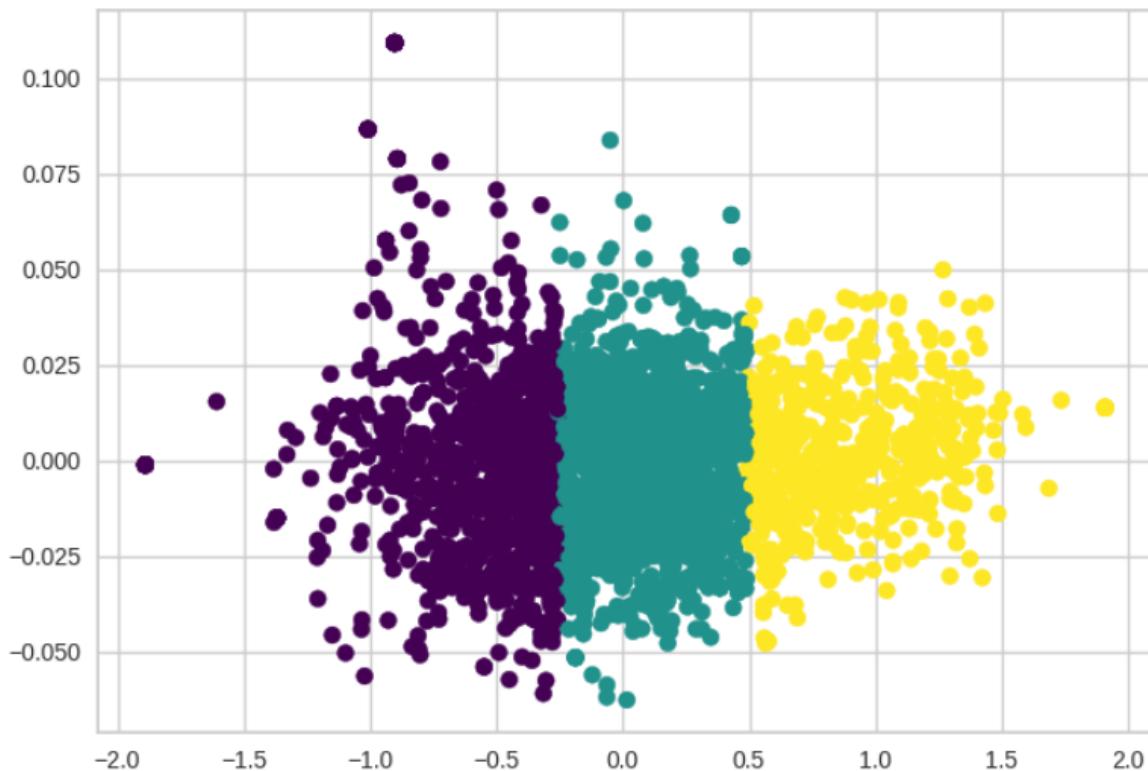
Kết quả phân cụm:



- Topic 0: topic về marketing với các từ khóa được đề cập đến nhiều như business, marketing, các trang mạng xã hội phổ biến dành cho marketing như instagram, facebook, tiktok, twitter,... (bị lẫn từ khóa liên quan đến chứng khoán)
- Topic 1: topic về chứng khoán với các từ khóa được đề cập đến nhiều như stock pick, stock market, top stock, và những từ khóa về tháng như january, april, march, august,... cũng xuất hiện (thể hiện những topic phân tích chứng khoán theo các tháng)
- Topic 2: topic về accounting với các từ khóa được đề cập đến nhiều như accounting beginner, accounting basic, journal entry (thuật ngữ kế toán ghi nhận giao dịch vào sổ kế toán), balance sheet (bảng cân đối kế toán) hay các từ khóa về

chứng chỉ CPA (chứng chỉ dùng để chỉ những kiểm toán viên có trình độ được chứng nhận trên toàn cầu) như: cpa strength, cpa exam,.. (nhưng cũng bị lẫn một số từ khóa liên quan đến marketing)

c. Fuzzy C - Means



Kết quả phân cụm:



- cluster 0: có thể thấy ở cụm 0 này thì khá khó để xác định chủ đề khi mà các chủ đề bị lẫn lộn với nhau (xuất hiện những từ khóa liên quan về accounting và cả những từ khóa về marketing cũng như chứng khoán)
- cluster 1: các topic về marketing với các từ khóa được đề cập đến nhiều như business, marketing, các trang mạng xã hội phổ biến dành cho marketing như

instagram, facebook, tiktok, twitter,... (cũng đang bị lẩn một số từ khóa về chứng khoán)

- cluster 2: các topic về chứng khoán với các từ khóa được đề cập đến nhiều như stock pick, stock market, top stock, và những từ khóa về tháng như january, april, march, august,... cũng xuất hiện (thể hiện những topic phân tích chứng khoán theo các tháng)

Và cũng có thể thấy dù đã tiến hành cải thiện chất lượng phân cụm bằng cách kết hợp Autoencoder ở cả 2 phương pháp K - Means và Fuzzy C - Means nhưng việc phân cụm vẫn chưa hẳn là tốt nên nhóm đã tiếp tục cải thiện chất lượng phân cụm bằng cách áp dụng phương pháp Deep Embedded Clustering để cho ra những chỉ số tốt nhất để cho việc phân cụm được chính xác hơn.

3.2.3. Deep Embedded Clustering

```
# Define the clustering layer
class ClusteringLayer(Layer):

    def __init__(self, n_clusters, alpha=1.0, **kwargs):
        self.n_clusters = n_clusters
        self.alpha = alpha
        super(ClusteringLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 2
        self.centroids = self.add_weight(name='centroids',
                                         shape=(self.n_clusters, input_shape[1]),
                                         initializer='glorot_uniform',
                                         trainable=True)

        super(ClusteringLayer, self).build(input_shape)

    def call(self, inputs):
        q = 1.0 / (1.0 + (K.sum(K.square(K.expand_dims(inputs, axis=1) - self.centroids), axis=2) / self.alpha))
        q = q ** (self.alpha + 1.0 / 2.0)
        q = K.transpose(K.transpose(q) / K.sum(q, axis=1))

        return q

    def compute_output_shape(self, input_shape):
        # Output shape is (batch_size, n_clusters)
        assert input_shape and len(input_shape) == 2
        return input_shape[0], self.n_clusters

    def get_config(self):
        config = {'n_clusters': self.n_clusters,
                  'alpha': self.alpha}
        base_config = super(ClusteringLayer, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))
```

Tạo lớp Clustering Layer tùy chỉnh, được kế thừa từ lớp Layer trong Keras để phân cụm dữ liệu. Gồm 5 phương thức “`__init__`”, “`build`”, “`call`”, “`compute_output_shape`” và “`get_config`”.

Trong đó phương thức “`__init__`” khởi tạo lớp với 2 tham số “`n_clusters`” là số lượng cụm mà lớp sẽ nhận dạng, và “`alpha`” là tham số được sử dụng trong quá trình phân cụm. Phương thức “`build`” xây dựng lớp bằng cách thêm một trọng số có thể huấn luyện được gọi là “`centroids`” là một ma trận kích thước (`n_clusters, input_shape[1]`). Ma trận này đại diện cho các trung tâm của các cụm và được khởi tạo bằng phương pháp khởi tạo Glorot uniform.

Phương thức “`call`” tính toán phân bổ cụm bằng cách sử dụng ma trận `centroids`.

Đầu ra là một có kích thước (`batch_size, n_clusters`), trong đó mỗi phần tử đại diện cho xác suất của đầu vào thuộc về một cụm cụ thể.

Phương thức “`compute_output_shape`” tính toán kích thước đầu ra của lớp, là (`batch_size, n_clusters`).

Phương thức “`get_config`” trả về một từ điển chứa cấu hình của lớp, bao gồm các tham số `n_clusters` và `alpha`. Nó cũng gọi phương thức `get_config` của lớp cơ sở để lấy cấu hình của lớp Layer.

```

def train_model(model, x, y, epochs, batch_size):
    model.fit(x, y, epochs=epochs, batch_size=batch_size)

# Define the deep embedded clustering model
clustering_layer = ClusteringLayer(n_clusters=3, name='clustering')(encoded_layer)
deep_embedded_clustering = Model(inputs=input_layer, outputs=clustering_layer)
deep_embedded_clustering.compile(optimizer=SGD(0.01, 0.9), loss='kld')

# Initialize the centroids using k-means
kmeans = KMeans(n_clusters=3, n_init=20)
kmeans.fit(encoder.predict(X))
deep_embedded_clustering.get_layer(name='clustering').set_weights([kmeans.cluster_centers_])

dummy_target = np.zeros((X.shape[0], 3))

# Call the function to create variables before running the decorated function
train_model(deep_embedded_clustering, X, dummy_target, epochs=100, batch_size=128)

# Obtain the cluster assignments for the encoded data
cluster_assignments = deep_embedded_clustering.predict(X).argmax(axis=1)

97/97 [=====] - 0s 1ms/step
Epoch 1/100
25/25 [=====] - 6s 225ms/step - loss: -4.5058e-06
Epoch 2/100
25/25 [=====] - 0s 3ms/step - loss: -4.5058e-06
Epoch 3/100
25/25 [=====] - 0s 3ms/step - loss: -4.5058e-06
Epoch 4/100
25/25 [=====] - 0s 3ms/step - loss: -4.5058e-06
Epoch 5/100
25/25 [=====] - 0s 3ms/step - loss: -4.5058e-06
Epoch 6/100

```

Huấn luyện mô hình Deep Embedded Clustering (DEC) với lớp phân cụm tùy chỉnh Clustering Layer. Mô hình được tối ưu hóa với SGD optimizer và hàm mất mát Kullback-Leibler divergence. Với trọng tâm cụm được gán dựa trên K-Means.

Vì mô hình học không giám sát không có nhãn nên ta sẽ tạo biến giả dummy_target với kích thước (X.shape[0], 3) với 3 là số lượng phân cụm. Sau đó tiến hành huấn luyện mô hình và thực hiện phương thức predict để tạo mảng cluster_assignments tương ứng với kết quả thực hiện phân cụm.

3.3. Các Kết Quả Thực Nghiệm

Để đánh giá các kết quả thực nghiệm nhận được, nhóm tiến hành kiểm tra các chỉ số bao gồm Silhouette, Calinski Harabasz và Davies Bouldin. So sánh sự hiệu quả của các mô hình học máy và học sâu dựa trên số liệu mà các chỉ số mang lại.

3.3.1. Traditional Clustering

a. K - Means

```
silhouette: 0.10793718
calinski harabasz: 393.22282420028904
davies bouldin: 2.779954058913868
```

Khi thực hiện K-Means truyền thống ta thu được kết quả với Silhouette xấp xỉ 0.1, Calinski-Harabasz 393.2 và Davies-Bouldin 2,77 cho thấy các cụm không được phân chia rõ ràng, có sự chồng chéo đáng kể giữa các cụm và hiệu suất phân cụm không tối ưu. Nhìn chung các chỉ số này cho thấy thuật toán K-Means không phù hợp cho tập dữ liệu đang được xử lý và cần điều chỉnh hoặc sử dụng một thuật toán khác.

b. Fuzzy C - Means

```
Silhouette: 0.11769526
Calinski Harabasz: 497.8333450989962
Davies Bouldin: 2.38641622391207
```

Phân cụm bằng Fuzzy C-Means truyền thống cũng không đem lại hiệu suất gom cụm tốt. Tuy nhiên đã có sự cải thiện nhẹ so với K-Means. Với Silhouette là 0.117 và Calinski Harabasz cao hơn và Davies Bouldin thấp hơn so với K-Means. Cho thấy các cụm có sự phân chia tốt hơn so với kết quả của K-Means, tuy nhiên cả 2 kết quả đều cho thấy thuật toán gom cụm truyền thống không tối ưu đối với tập dữ liệu mà nhóm nghiên cứu.

3.3.2. Autoencoders

a. K - Means

```
silhouette: 0.47000018
calinski harabasz: 5267.692433045319
davies bouldin: 0.662025306325431
```

Khi thực hiện Autoencoders kết hợp với K-Means, nhóm thu được kết quả phân cụm tương đối tốt và cải thiện hơn so với K-Means truyền thống. Với Silhouette score bằng

0.47 và Calinski Harabasz 5267.69 khá cao, cho thấy phân cụm được định vị khá tốt trong không gian đa chiều

b. Fuzzy C - Means

Silhouette: 0.46919057

Calinski Harabasz: 5262.726894802293

Davies Bouldin: 0.6631241112281506

Kết hợp Autoencoders với Fuzzy C-Means cũng đem lại kết quả cải thiện hơn so với Fuzzy C-Means truyền thống. Chỉ số Silhouette cho thấy mức độ tách biệt giữa các cụm không cao, chỉ là 0.47, cho thấy các điểm dữ liệu trong các cụm có sự chồng chéo nhất định. Chỉ số Calinski Harabasz cũng có giá trị khá cao, hơn 5000, cho thấy sự tách biệt giữa các cụm. Chỉ số Davies Bouldin là 0.663, không thấp nhưng cũng không cao, cho thấy độ tương đồng giữa các cụm không quá cao hoặc thấp. Vì vậy, kết quả clustering có thể được xem là khá tốt và đáng tin cậy.

3.3.3. Deep Embedded Clustering

Silhouette: 0.8700455999561516

Calinski Harabasz: 9076.146571297873

Davies Bouldin: 0.5309554647746934

Chỉ số Silhouette của Deep Embedded Clustering là 0.87, cho thấy phân cụm được xây dựng khá tốt. Chỉ số Calinski Harabasz của 9076.15 cũng cho thấy sự tách biệt rõ ràng giữa các cụm. Chỉ số Davies Bouldin của 0.53 cũng cho thấy rằng các cụm có độ tách biệt tốt và sự giống nhau giữa các điểm trong cùng một cụm thấp. Tổng quan, kết quả phân cụm là tốt và đáng tin cậy.

3.3.4. Nhận xét và đánh giá

| | Silhouette | Calinski Harabasz | Davies Bouldin |
|---------------------------|------------|-------------------|----------------|
| K-Means | 0.108 | 393.223 | 2.779 |
| Fuzzy C-Means | 0.117 | 497.833 | 2.386 |
| AE + K-Means | 0.470 | 5267.692 | 0.662 |
| AE + Fuzzy C-Means | 0.469 | 5272.726 | 0.663 |
| DEC | 0.870 | 9076.146 | 0.530 |

Dựa trên 3 chỉ số Silhouette, Calinski Harabasz và Davies Bouldin, ta thấy kết quả của 2 phương pháp phân cụm truyền thống (K-Means và Fuzzy C-Means) không đem lại kết quả phân cụm hiệu quả đối với bộ dữ liệu YouTube mà nhóm thực hiện nghiên cứu. Khi kết hợp với Autoencoders để thực hiện trích xuất đặc trưng văn bản, kết quả phân cụm đã có sự cải thiện rõ rệt, tuy nhiên chỉ số Silhouette vẫn < 0.5 cho thấy vẫn có sự chồng chéo giữa các cụm. Kết quả cuối cùng cho thấy Deep Embedded Clustering đem lại hiệu quả tốt nhất trên bộ dữ liệu. Với chỉ số Silhouette, Calinski Harabasz cao nhất và Davies Bouldin thấp nhất

CHƯƠNG 4. KẾT LUẬN

4.1. Các Kết Quả Đạt Được

Thông qua quá trình tìm hiểu và nghiên cứu, nhóm đã xây dựng được các mô hình học máy phân cụm Youtube video. Tìm ra những cụm từ thường xuyên xuất hiện trong tên video cho một chủ đề cụ thể để người dùng dễ dàng nhận biết cũng như tìm kiếm thông tin giữa lượng dữ liệu vô cùng to lớn.

Đồng thời nhóm đã ứng dụng những kiến thức đã học về Máy học (Machine Learning) để thực hiện Phân cụm các chủ đề qua biến ‘title’ trong bộ dữ liệu Youtube video, ngoài ra còn ứng dụng được kiến thức đã học là Xử lý ngôn ngữ tự nhiên (khai thác và xử lý dữ liệu văn bản), ứng dụng kiến thức về crawling data thông qua phương pháp API.

4.2. Những Hạn Chế và Hướng Phát Triển

4.2.1. *Hạn chế*

- Vấn đề về hiệu suất tính toán: Các phương pháp như BERT, Autoencoders đòi hỏi một lượng lớn tính toán và bộ nhớ để hoạt động hiệu quả. Nếu tăng số lượng video từ 3.000 lên hơn 10.000 videos thì sẽ mất hơn 30 phút để thực hiện BERT.
- Sự thiếu chính xác trong kết quả phân cụm: Các phương pháp phân cụm truyền thống không đem lại kết quả tốt trên bộ dữ liệu thực nghiệm.
- Khó khăn trong việc lựa chọn số lượng phân cụm phù hợp: Các phương pháp phân cụm này đòi hỏi việc lựa chọn số lượng phân cụm phù hợp để đem lại kết quả tốt nhất. Tuy nhiên việc lựa chọn này có thể khó khăn và không đem lại hiệu quả như kỳ vọng.

4.2.2. *Hướng phát triển*

- Sử dụng các mô hình tiên tiến: Có thể áp dụng các mô hình deep learning khác để cải thiện hiệu quả của phân cụm. Ví dụ như sử dụng mạng Transformer hoặc mô hình GPT.
- Áp dụng phương pháp phân cụm tăng cường (ensemble clustering) hoặc kết hợp nhiều mô hình phân cụm. Một số phương pháp phân cụm tăng cường phổ biến bao gồm bagging, boosting và stacking.
- Tăng cường dữ liệu: Để tăng độ chính xác của mô hình phân cụm, có thể áp dụng các kỹ thuật tăng cường dữ liệu như data augmentation, oversampling, undersampling.

- Thử nghiệm mô hình hóa chủ đề trên các YouTube video bằng nhiều ngôn ngữ khác nhau như Tiếng Việt, đồng thời áp dụng các phương pháp NLP xử lý ngôn ngữ tự nhiên phù hợp cho Tiếng Việt như PhoBERT, VnCoreNLP,...

PHỤ LỤC I (Link Github)

Link github mã nguồn:

https://github.com/KhanhHa1109/MachineLearning_YouTube_TopicModeling

PHỤ LỤC II

| Bảng phân công công việc | |
|---------------------------------|---|
| Nguyễn Quỳnh Khánh Hà | Data Crawling, LDA+BERT, Deep Embedded Clustering |
| Huỳnh Trần Anh Thy | Tiền xử lý, EDA & Visualization |
| Nguyễn Trịnh Thu Huyền | Autoencoder + K-Means/ Fuzzy C-Means |
| Nguyễn Văn Hoàng Dũng | Data Crawling, Traditional clustering (K-Means, Fuzzy C-Means) |

TÀI LIỆU THAM KHẢO

[1] Deep Clustering.

<https://deepnotes.io/deep-clustering>

[2] K-means Clustering: Simple Applications.

<https://machinelearningcoban.com/2017/01/04/kmeans2/>

[3] Topic Modeling with BERT, LDA and Clustering.

<https://github.com/AravindR7/Topic-Modeling-BERT-LDA>

[4] Keras implementation for Deep Embedded Clustering (DEC).

<https://github.com/XifengGuo/DEC-keras>

[5] Alghamdi R, Alfalqi K. A survey of topic modeling in text mining. Int J Adv Comput Sci Appl. 2015;6(1):7.

[6] P. Kherwa and P. Bansal, "Topic modeling: A comprehensive review," EAI Endorsed transactions on scalable information systems, vol. 7, no. 24, pp. 1-16, 2019.

[7] Building Autoencoders in Keras.

<https://blog.keras.io/building-autoencoders-in-keras.html>

[8] Tiễn xử lý trong xử lý ngôn ngữ tự nhiên.

<https://blog.vietnamlab.vn/ban-ve-cong-doan-tien-xu-ly-trong-xu-ly-ngon-nu-tu-nhien/>

<https://viblo.asia/p/tien-xu-li-du-lieu-van-ban-voi-nltk-Az45b0LgZxY>

[9] Davies–Bouldin index.

https://en.wikipedia.org/wiki/Davies%2680%93Bouldin_index