**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI: PHÂN TÍCH HÀNH VI VÀ PHÂN KHÚC KHÁCH HÀNG TRÊN SÀN THƯƠNG MẠI ĐIỆN TỬ AMAZON DỰA TRÊN DỮ LIỆU ĐÁNH GIÁ SẢN PHẨM**

Nhóm sinh viên thực hiện:

|  |  |
| --- | --- |
| Nguyễn Thị Thanh Vân | 2251061922 |
| Nguyễn Đức Chiến | 2251061727 |
| Lê Hà Phương | 2251061863 |
| Lưu Hiểu Khánh | 2251061807 |

Lớp: 64CNTT1

Khoa: Công nghệ thông tin

Giảng viên hướng dẫn: **TS. Lê Thị Tú Kiên**

***Hà Nội, 2025***

## **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc212649104)

[**DANH MỤC BẢNG BIỂU** 4](#_Toc212649105)

[**DANH MỤC HÌNH ẢNH** 5](#_Toc212649106)

[**DANH MỤC TỪ VIẾT TẮT** 5](#_Toc212649107)

[**LỜI NÓI ĐẦU** 7](#_Toc212649108)

[**CHƯƠNG 1: MỞ ĐẦU** 9](#_Toc212649109)

[1.1. Bối cảnh và đặt vấn đề 9](#_Toc212649110)

[1.2. Mục tiêu nghiên cứu 10](#_Toc212649111)

[1.3. Phương pháp nghiên cứu 10](#_Toc212649112)

[1.4. Đối tượng và phạm vi nghiên cứu 13](#_Toc212649113)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN** 15](#_Toc212649114)

[2.1. Tổng quan về khai phá dữ liệu trong TMĐT 15](#_Toc212649115)

[2.2. Các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) 16](#_Toc212649116)

[2.3. Các thuật toán Phân cụm 19](#_Toc212649117)

[**CHƯƠNG 3: DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU** 21](#_Toc212649118)

[3.1. Nguồn dữ liệu và mô tả 22](#_Toc212649119)

[3.2. Quy trình tiền xử lý dữ liệu 25](#_Toc212649120)

[**CHƯƠNG 4: THỰC HIỆN TRÍCH XUẤT ĐẶC TRƯNG, XÂY DỰNG HỒ SƠ VÀ GIẢM CHIỀU** 29](#_Toc212649121)

[4.1. Trích xuất đặc trưng định tính (Review Vector) 29](#_Toc212649122)

[4.2. Trích xuất đặc trưng định lượng (User-level) 31](#_Toc212649123)

[4.3. Xây dựng hồ sơ người dùng 33](#_Toc212649124)

[**CHƯƠNG 5: PHÂN CỤM VÀ PHÂN KHÚC KHÁCH HÀNG** 37](#_Toc212649125)

[5.1. Phân cụm bằng K-Means 37](#_Toc212649126)

[5.2. Phân cụm bằng DBSCAN 45](#_Toc212649127)

[5.3. Đánh giá và So sánh Hiệu quả Phân cụm 60](#_Toc212649128)

[5.4. Diễn giải và mô tả các phân khúc khách hàng 60](#_Toc212649129)

[**CHƯƠNG 6: KẾT LUẬN VÀ KIẾN NGHỊ** 70](#_Toc212649130)

[6.1. Kết luận 70](#_Toc212649131)

[6.2. Hạn chế của đề tài 70](#_Toc212649132)

[6.3. Kiến nghị và hướng phát triển 70](#_Toc212649133)

[**TÀI LIỆU THAM KHẢO** 72](#_Toc212649134)

## **DANH MỤC BẢNG BIỂU**

[**Bảng 2.1.** Các loại dữ liệu phi cấu trúc phổ biến 16](#_Toc212651021)

[**Bảng 2.2.** Các phương pháp chuẩn hóa và biểu diễn dữ liệu phi cấu trúc 17](#_Toc212651022)

[**Bảng 2.3.** Ưu, nhược điểm của K-Means và DBSCAN 21](#_Toc212651023)

[**Bảng 5.1.** Bảng đánh giá 2 phương pháp TF-IDF và Word2Vec 44](#_Toc212651024)

[**Bảng 5.2.** Bảng so sánh chỉ số Silhouette 60](#_Toc212651025)

## **DANH MỤC HÌNH ẢNH**

[**Hình 3.1.** Thông tin các cột của bộ dữ liệu sau khi đọc vào 23](#_Toc212653334)

[**Hình 3.2.** Kết quả thống kê cơ bản tập dữ liệu ban đầu 24](#_Toc212653335)

[**Hình 3.3.** Bảng phân phối điểm đánh giá sao của người dùng 24](#_Toc212653336)

[**Hình 3.4.** Các từ mang ý nghĩa bộc lộ cảm xúc nhưng vô nghĩa 26](#_Toc212653337)

[**Hình 3.5.** Đánh giá sau khi làm sạch và các tính từ thu được sau khi lọc 28](#_Toc212653338)

[**Hình 4.1.** Ma trận TF-IDF và các đặc trưng đầu tiên 29](#_Toc212653339)

[**Hình 4.2.** Ma trận Word2Vec và ví dụ biểu diễn vector từ 31](#_Toc212653340)

[**Hình 4.3.** Thống kê mô tả các đặc trưng hành vi người dùng 32](#_Toc212653341)

[**Hình 4.4.** Biểu diễn người dùng bằng vector TF-IDF và Word2Vec 34](#_Toc212653342)

[**Hình 4.5.** Kết quả giảm chiều dữ liệu và tạo vector đặc trưng cuối cùng của người dùng 35](#_Toc212653343)

[**Hình 5.1.** Biểu đồ Elbow (TF-IDF) 38](#_Toc212653344)

[**Hình 5.2.** Biểu đồ Elbow (Word2Vec) 39](#_Toc212653345)

[**Hình 5.3.** Kết quả Silhouette 40](#_Toc212653346)

[**Hình 5.4.** Kết quả phân cụm (TF-IDF) với k = 6 41](#_Toc212653347)

[**Hình 5.5.** Kết quả phân cụm (Word2Vec) với k = 6 42](#_Toc212653348)

[**Hình 5.6.** Kết quả phân cụm sử dụng phương pháp TF-IDF 42](#_Toc212653349)

[**Hình 5.7.** Kết quả phân cụm sử dụng phương pháp Word2Vec 43](#_Toc212653350)

[**Hình 5.8.** Kết quả tìm phương sai và số chiều tương ứng 46](#_Toc212653351)

[**Hình 5.9.** Kết quả tìm phương sai và số chiều tương ứng 47](#_Toc212653352)

[**Hình 5.10.** Kết quả chạy tìm kết quả tìm min\_simple 48](#_Toc212653353)

[**Hình 5.11.** Kết quả chạy tìm epsilon tối ưu 50](#_Toc212653354)

[**Hình 5.12.** Biểu diễn với PCA (DBSCAN + TF-IDF) 53](#_Toc212653355)

[**Hình 5.13.** Biểu diễn cụm theo các cặp chiều 54](#_Toc212653356)

[**Hình 5.14.** Biểu đồ K-distance 56](#_Toc212653357)

[**Hình 5.15.** Hình minh họa các cụm 59](#_Toc212653358)

[**Hình 5.16.** Sử dụng UMAP (DBSCAN + TF-IDF) 62](#_Toc212653359)

[**Hình 5.17.** Biểu đồ nhiệt kết quả khi dùng phương pháp TF-IDF 63](#_Toc212653360)

[**Hình 5.18.** Biểu đồ nhiệt kết quả khi dùng phương pháp Word2Vec 64](#_Toc212653361)

[**Hình 5.19.** Biểu đồ nhiệt DBSCAN + TF-IDF 65](#_Toc212653362)

[**Hình 5.20.** Biểu đồ nhiệt cho cụm -1 (DBSCAN + Word2Vec) 67](#_Toc212653363)

[**Hình 5.21.** Biểu đồ nhiệt cho cụm 0 (DBSCAN + Word2Vec) 68](#_Toc212653364)

[**Hình 5.22.** Biểu đồ nhiệt cho cụm 1 (DBSCAN + Word2Vec) 68](#_Toc212653365)

[**Hình 5.23.** Biểu đồ nhiệt cho cụm 2 (DBSCAN + Word2Vec) 69](#_Toc212653366)

## **DANH MỤC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Ký hiệu chữ viết tắt** | **Chữ viết đầy đủ** |
| 1 | OOM | Out of Memory (Hết bộ nhớ) |
| 2 | TMĐT | Thương mại điện tử |
| 3 | RDBMS | Relational Database Management System |
| 4 | NLP | Neuro – Linguistic Programming |
| 5 | POS Tagging | Part of Speech Tagging |
| 6 | TF-IDF | Term Frequency – Inverse Document Frequency |
| 7 | SVD | Singular Value Decomposition |
| 8 | UMAP | Uniform Manifold Approximation and Projection |
| 9 | BERT | Bidirectional Encoder Representations from Transformers |

## 

## **LỜI NÓI ĐẦU**

Trong bối cảnh thương mại điện tử phát triển mạnh mẽ, việc thấu hiểu hành vi và nhu cầu của khách hàng trở thành yếu tố sống còn đối với các doanh nghiệp. Amazon - một trong những nền tảng mua sắm trực tuyến lớn nhất thế giới - sở hữu khối lượng dữ liệu khổng lồ từ hàng triệu người dùng, tạo ra cơ hội quý giá để khai thác thông tin nhằm nâng cao chất lượng dịch vụ và tối ưu hóa chiến lược kinh doanh.

Đề tài “Phân tích hành vi và phân khúc khách hàng trên sàn thương mại điện tử Amazon dựa trên dữ liệu phân cụm” được thực hiện với mục tiêu xây dựng một mô hình phân cụm hiệu quả, giúp nhóm các người dùng có đặc điểm và sở thích tương đồng. Thông qua việc áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên và học máy không giám sát, đề tài hướng đến việc khám phá cấu trúc tiềm ẩn trong dữ liệu đánh giá của người dùng, từ đó hỗ trợ doanh nghiệp trong việc cá nhân hóa gợi ý sản phẩm, tối ưu hóa quảng cáo và nâng cao trải nghiệm khách hàng.

Quá trình nghiên cứu bao gồm các bước: thu thập và tiền xử lý dữ liệu đánh giá từ Amazon, biểu diễn văn bản bằng các phương pháp như TF-IDF và Word2Vec, áp dụng thuật toán phân cụm như DBSCAN, và đánh giá chất lượng phân cụm bằng các tiêu chí định lượng. Kết quả đạt được không chỉ mang giá trị học thuật mà còn có tiềm năng ứng dụng thực tiễn cao trong lĩnh vực thương mại điện tử hiện đại.

Hy vọng rằng đề tài này sẽ đóng góp một góc nhìn mới trong việc khai thác dữ liệu người dùng, đồng thời mở ra hướng tiếp cận hiệu quả cho các doanh nghiệp trong việc hiểu và phục vụ khách hàng một cách thông minh hơn.

Sau đây là bảng phân chia công việc của nhóm chúng em:

|  |  |  |  |
| --- | --- | --- | --- |
| **Họ và tên** | **Mã sinh viên** | **Nhiệm vụ** | **Đánh giá** |
| Nguyễn Thị Thanh Vân | 2251061922 | - Thu thập dữ liệu  - Thuật toán DBSCAN  - Phân tích kết quả  - Trình bày báo cáo | Hoàn thành |
| Nguyễn Đức Chiến | 2251061727 | - Thu thập dữ liệu  - Phương pháp Word2Vec  - Trình bày báo cáo | Hoàn thành |
| Lê Hà Phương | 2251061863 | -Thu thập dữ liệu  - Thuật toán K-Means  - Trình bày kết quả và báo cáo | Hoàn thành |
| Lưu Hiểu Khánh | 2251061807 | -Thu thập dữ liệu  - Phương pháp TF-IDF  - Trình bày báo cáo | Hoàn thành |

## 

## **CHƯƠNG 1: MỞ ĐẦU**

### 1.1. Bối cảnh và đặt vấn đề

Nghiên cứu này tập trung vào việc khai thác và phân tích các mô hình sở thích tiêu dùng phức tạp của tập khách hàng quy mô lớn trên nền tảng thương mại điện tử hàng đầu toàn cầu Amazon.

Trong bối cảnh dữ liệu lớn (Big Data), việc phân tích hành vi khách hàng không chỉ dừng lại ở các chỉ số giao dịch định lượng mà còn mở rộng sang việc thấu hiểu ý định và gu thẩm mỹ tiềm ẩn. Đề tài hướng đến mục tiêu chiến lược là định hình các phân khúc thị trường ẩn (latent market segments) - tức là các nhóm khách hàng không dễ dàng được xác định bằng các tiêu chí phân loại trực tiếp, truyền thống (như tuổi tác, giới tính, hay thu nhập) hoặc các chỉ số hành vi cơ bản (như tần suất mua hàng đơn thuần).

Thay vào đó, các phân khúc này được khám phá thông qua các mô hình hành vi và sở thích phức tạp, tiềm ẩn (latent) trong dữ liệu. Đây chính là yếu tố then chốt để xây dựng trải nghiệm mua sắm cá nhân hóa và tối ưu hóa hiệu suất kinh doanh của doanh nghiệp thương mại điện tử.

Để đáp ứng yêu cầu về độ chính xác và chiều sâu của việc phân loại, đề tài sử dụng bộ dữ liệu hỗn hợp (Hybrid Feature Set) đa chiều, tích hợp cả các thuộc tính cấu trúc (Structured Features) và phi cấu trúc (Unstructured Features). Toàn bộ dữ liệu được trải qua quy trình tiền xử lý dữ liệu (Data Preprocessing) và kỹ thuật vector hóa (Vectorization) nghiêm ngặt để đảm bảo sự đồng nhất và chất lượng đầu vào.

Nghiên cứu triển khai song song hai phương pháp phân cụm tiêu biểu để đánh giá hiệu quả, bao gồm K-Means và DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

* **K-Means** dựa trên khoảng cách và giả định các cụm có hình dạng gần cầu, giúp tạo ra các nhóm rõ ràng với tâm cụm dễ diễn giải.
* **DBSCAN** dựa trên mật độ, có khả năng phát hiện các vùng có mật độ cao trong không gian đặc trưng phức tạp và nhận diện các hành vi mua sắm độc đáo hoặc bất thường.

Thông qua việc so sánh các chỉ số đánh giá nội tại (Internal Evaluation Metrics) và phân tích khả năng diễn giải (Interpretability) của từng mô hình, nghiên cứu sẽ xác định thuật toán nào phản ánh cấu trúc phân bổ khách hàng Amazon chân thực và giá trị nhất. Việc so sánh này không chỉ giúp tìm ra phương pháp phân cụm tối ưu mà còn làm nổi bật đặc trưng riêng của từng nhóm sở thích tiêu dùng.

### 1.2. Mục tiêu nghiên cứu

**Mục tiêu tổng quát:**

Xây dựng mô hình phân cụm giúp nhóm các người dùng có đặc điểm và sở thích tương đồng, hỗ trợ doanh nghiệp trong việc cá nhân hóa gợi ý sản phẩm, tối ưu chiến lược marketing và nâng cao trải nghiệm người dùng.

**Mục tiêu cụ thể:**

* **Thu thập dữ liệu người dùng từ Amazon** bao gồm Id, ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text.
* **Tiền xử lý dữ liệu** bao gồmlàm sạch, chuẩn hóa và mã hóa dữ liệu văn bản, số liệu để sẵn sàng cho bước phân tích.
* **Áp dụng các thuật toán phân cụm** (như K-means, DBSCAN) để nhóm người dùng theo mức độ tương đồng về sở thích và hành vi mua sắm.
* **Đánh giá kết quả phân cụm** bằng các tiêu chí định lượng (Silhouette Score, Davies–Bouldin Index) và trực quan hóa cụm dữ liệu.
* **Đề xuất ứng dụng thực tiễn** của kết quả phân cụm trong việc cá nhân hóa gợi ý sản phẩm, tối ưu quảng cáo và nâng cao hiệu quả kinh doanh.

### 1.3. Phương pháp nghiên cứu

Nghiên cứu này áp dụng phương pháp tiếp cận dựa trên dữ liệu (data-driven) kết hợp với các kỹ thuật học máy (machine learning) để phân cụm người dùng dựa trên dữ liệu đánh giá sản phẩm thu thập từ Amazon (nguồn: Kaggle). Quy trình nghiên cứu gồm các bước chính sau:

#### 1.3.1. Thu thập dữ liệu

* **Nguồn dữ liệu:** Tập dữ liệu *Amazon\_find\_food\_review.csv* về các đánh giá người dùng trên sàn thương mại điện tử Amazon được thu thập từ trang dữ liệu Kaggle (tập dữ liệu được công khai).
* **Dữ liệu thu thập:**

1) **Id** – Mã định danh duy nhất cho mỗi đánh giá.

2) **ProductId** – Mã sản phẩm được đánh giá.

3) **UserId** – Mã người dùng thực hiện đánh giá.

4) **ProfileName** – Tên hiển thị của người dùng.

5) **HelpfulnessNumerator** – Số lượt đánh giá đánh giá này là “hữu ích”.

6) **HelpfulnessDenominator** – Tổng số lượt đánh giá hữu ích mà người dùng khác bỏ phiếu.

7) **Score** – Điểm đánh giá sản phẩm (từ 1 đến 5 sao).

8) **Time** – Thời điểm đánh giá (dưới dạng Unix timestamp).

9) **Summary** – Tóm tắt ngắn gọn về đánh giá.

10) **Text** – Nội dung chi tiết của đánh giá.

* **Mục tiêu:** Xây dựng tập dữ liệu phản ánh sở thích và hành vi tiêu dùng của người dùng trên sàn thương mại điện tử Amazon thông qua cả các đặc trưng định lượng (Score, HelpfulnessNumerator, ...) và đặc trưng phi cấu trúc (Text, Summary).

#### 1.3.2. Tiền xử lý dữ liệu

* **Làm sạch cơ bản:** 
  + Loại bỏ các bản ghi bị trùng lặp hoặc thiếu thông tin quan trọng (UserId, Text).
* **Chuẩn hóa và xử lý văn bản:** 
  + Chuyển toàn bộ nội dung đánh giá (Text, Summary) về chữ thường.
  + Loại bỏ ký tự đặc biệt, URL, emoji, dấu câu và số.
  + Thực hiện tokenization (tách từ).
  + Loại bỏ stopwords (các từ ít mang nghĩa như “the”, “and”, “is”, “very”, “really”, ...).
  + Thực hiện lemmatization hoặc stemming để đưa từ về gốc (ví dụ: “running” → “run”).
* **Vector hóa dữ liệu văn bản:** Để chuyển dữ liệu văn bản thành dạng số học (vector) phục vụ cho việc học máy, đề tài cân nhắc ba hướng:
  + **TF-IDF (Term Frequency – Inverse Document Frequency):** Phù hợp với dữ liệu văn bản lớn và ít nhiễu, giúp đo mức độ quan trọng của từ trong từng review.
  + **Word2Vec:** Dựa trên ngữ cảnh từ vựng, giúp biểu diễn ngữ nghĩa sâu hơn.
  + **Transformer embeddings:** Biểu diễn câu hoặc đoạn đánh giá trong không gian ngữ nghĩa có độ chính xác cao, thích hợp cho dữ liệu đánh giá dài.
* **Tổng hợp đặc trưng ở cấp người dùng (User-level features):** Vì mục tiêu là phân cụm người dùng, Mỗi UserId được biểu diễn bởi một vector đặc trưng gồm:
  + Trung bình điểm đánh giá (Avg\_score)
  + Số lượng sản phẩm mà người dùng đánh giá
  + Số lượng đánh giá của người dùng
  + Vector ngữ nghĩa tổng hợp của tất cả đánh giá
* **Chuẩn hóa số liệu:** 
  + Áp dụng StandardScaler hoặc MinMaxScaler để đưa các đặc trưng số về cùng thang đo.
  + Với các biến phân bố lệch (skewed) như HelpfulnessNumerator, có thể dùng log-transform.

#### 1.3.3. Trích chọn đặc trưng & giảm chiều

* **Lựa chọn đặc trưng:** Giữ lại các biến có khả năng phân biệt cao (ví dụ: mean Score, số lượng review, tỷ lệ hữu ích, embedding trung bình).
* **Giảm chiều:** Sử dụng PCA (Principal Component Analysis) hoặc UMAP/t-SNE để rút gọn số chiều dữ liệu trước khi phân cụm hoặc trực quan hóa.

#### 1.3.4. Phân cụm

**Đề tài thử nghiệm và so sánh hai thuật toán:**

* **K-Means:** Phân cụm dựa trên khoảng cách Euclidean, cần xác định trước số cụm k.
* **DBSCAN:** Phân cụm theo mật độ, không cần biết trước số cụm, phát hiện nhiễu tốt.

**Tham số lựa chọn:**

* **Với K-Means:** Xác định k bằng Elbow Method hoặc Silhouette Score.
* **Với DBSCAN:** Chọn eps và minPts dựa trên k-distance plot.

#### 1.3.5. Đánh giá phân cụm và trực quan hoá

* **Chỉ số đánh giá:** 
  + **Silhouette Score:** Đo mức độ tách biệt giữa các cụm.
  + **Davies–Bouldin Index:** Đánh giá sự cô đặc của cụm.
  + **Calinski–Harabasz Score:** Tỷ lệ giữa độ phân tán giữa các cụm và trong cụm.
* **Trực quan hoá:** 
  + Sử dụng PCA, t-SNE hoặc UMAP để biểu diễn dữ liệu trong không gian 2D/3D.
  + Biểu đồ thanh (bar chart) cho danh mục, boxplot cho rating trung bình theo cụm.

#### 1.3.6. Ứng dụng kết quả

Kết quả phân cụm có thể được ứng dụng trong:

* **Cá nhân hóa gợi ý sản phẩm:** Gợi ý mặt hàng phù hợp với nhóm sở thích.
* **Chiến lược marketing mục tiêu:** Xây dựng chiến dịch riêng cho từng phân khúc khách hàng.
* **Phân tích hành vi tiêu dùng:** Hiểu rõ hơn về thói quen, mức độ trung thành và xu hướng tiêu dùng.

### 1.4. Đối tượng và phạm vi nghiên cứu

#### 1.4.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là người dùng trên sàn thương mại điện tử Amazon và hành vi tiêu dùng của họ thông qua các đánh giá sản phẩm. Dữ liệu được thu thập từ các thông tin công khai trong phần đánh giá (review) của người dùng đối với các sản phẩm.

Dựa trên các thông tin về sản phẩm, nghiên cứu hướng đến việc phân cụm người dùng thành các nhóm có sở thích và hành vi tiêu dùng tương đồng, phục vụ cho mục tiêu cá nhân hóa đề xuất sản phẩm hoặc phân tích hành vi khách hàng.

#### 1.4.2. Phạm vi nghiên cứu

Phạm vi nghiên cứu được giới hạn trong:

* **Không gian nghiên cứu:** Dữ liệu được thu thập từ sàn thương mại điện tử Amazon, một trong những nền tảng mua sắm trực tuyến lớn nhất thế giới.
* **Thời gian nghiên cứu:** Dữ liệu được thu thập trong khoảng thời gian nhất định (từ năm 1999 đến năm 2012).
* **Phạm vi dữ liệu:** Nghiên cứu chỉ tập trung vào dữ liệu đánh giá sản phẩm (review data), không bao gồm dữ liệu giao dịch chi tiết hoặc thông tin cá nhân nhạy cảm khác của người dùng.
* **Phạm vi kỹ thuật:** Đề tài áp dụng các thuật toán phân cụm (clustering) như K-Means, DBSCAN để nhóm người dùng theo sở thích tiêu dùng, dựa trên các thuộc tính thu thập được.

## 

## **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN**

### 2.1. Tổng quan về khai phá dữ liệu trong TMĐT

Khai phá dữ liệu (Data Mining) là quá trình trích xuất các tri thức, mô hình, và mối quan hệ hữu ích từ một lượng lớn dữ liệu. Trong kỷ nguyên Big Data, khai phá dữ liệu đã mở rộng phạm vi ứng dụng từ các tập dữ liệu cấu trúc truyền thống sang các tập dữ liệu phi cấu trúc, chiếm tới hơn 80% tổng lượng dữ liệu được tạo ra hàng ngày. Sự gia tăng của các nền tảng trực tuyến, mạng xã hội và sàn thương mại điện tử đã khiến dữ liệu phi cấu trúc trở thành nguồn thông tin phong phú nhất để hiểu sâu về hành vi và sở thích của người dùng. Việc tích hợp dữ liệu phi cấu trúc vào quá trình khai phá là thiết yếu để xây dựng các mô hình dự đoán và phân tích chính xác hơn.

#### 2.1.1. Đặc điểm của dữ liệu phi cấu trúc

Dữ liệu phi cấu trúc là thông tin không tuân theo một mô hình dữ liệu (data model) định trước hoặc một tổ chức dạng bảng cố định. Điều này tạo ra cả thách thức và cơ hội lớn trong khai phá dữ liệu:

* **Thiếu mô hình cố định:** Dữ liệu không được tổ chức trong các cột và hàng của cơ sở dữ liệu quan hệ (RDBMS). Ví dụ điển hình là các đoạn văn bản tự do, email, hoặc tệp đa phương tiện.
* **Không đồng nhất (Heterogeneity):** Cùng một loại dữ liệu phi cấu trúc (ví dụ: văn bản) có thể được biểu diễn dưới nhiều định dạng và cấu trúc bên trong khác nhau, đòi hỏi các công cụ xử lý chuyên biệt (như NLP cho văn bản).
* **Tính ngữ cảnh cao:** Giá trị của dữ liệu phi cấu trúc thường phụ thuộc vào ngữ cảnh. Ví dụ, ý nghĩa của một từ trong đánh giá sản phẩm có thể thay đổi tùy theo ngành hàng.
* **Cần xử lý chuyên biệt:** Để có thể phân tích, dữ liệu phi cấu trúc phải trải qua một quy trình chuyển đổi phức tạp gọi là Vector hóa (Vectorization) hoặc Nhúng (Embedding), nhằm biểu diễn thông tin thành các vector số học trong không gian đa chiều.

#### 2.1.2. Các loại dữ liệu phi cấu trúc phổ biến (văn bản, hình ảnh, log hành vi, …)

Dữ liệu phi cấu trúc rất đa dạng và có vai trò khác nhau trong việc phân tích sở thích tiêu dùng trên sàn thương mại điện tử:

|  |  |  |
| --- | --- | --- |
| **Loại Dữ liệu** | **Mô tả** | **Vai trò** |
| Văn bản (Text) | Nội dung đánh giá sản phẩm (Review Text), bình luận, từ khóa tìm kiếm (Search Queries), mô tả sản phẩm, email. | Cung cấp **Vector ngữ nghĩa** về ý kiến, cảm xúc và ý định mua hàng. Đây là nguồn dữ liệu cốt lõi cho đề tài. |
| Log Hành vi (Behavioral Logs) | Chuỗi clickstream (thứ tự người dùng nhấp chuột), lịch sử duyệt web, nhật ký phiên giao dịch (session logs). | Mặc dù bán cấu trúc, nhưng chuỗi hành vi được xem là phi cấu trúc vì nó không có độ dài và cấu trúc cố định. Cần mã hóa chuỗi (Sequence Encoding) để tạo **Vector hành vi.** |
| Hình ảnh (Images) | Hình ảnh sản phẩm, hình ảnh đánh giá (do người dùng tải lên). | Dùng **Thị giác máy tính (Computer Vision)** để trích xuất các đặc trưng về phong cách, màu sắc, và thương hiệu. |
| Âm thanh/Video | Video giới thiệu sản phẩm, video review của người dùng. | Dùng **nhận dạng giọng nói** hoặc **xử lý video** để trích xuất cảm xúc và nội dung thông tin. |
| Dữ liệu Mạng xã hội | Các bài đăng, lượt thích, chia sẻ liên quan đến thương hiệu hoặc sản phẩm. | Mở rộng ngữ cảnh sở thích và ảnh hưởng của cộng đồng đối với quyết định mua sắm. |

**Bảng 2.1.** Các loại dữ liệu phi cấu trúc phổ biến

### 2.2. Các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP)

Dữ liệu đánh giá sản phẩm của Amazon chủ yếu là **dữ liệu văn bản phi cấu trúc**. Do đó, việc tiền xử lý đóng vai trò nền tảng cho mọi bước phân tích sau này.

#### 2.2.1. Làm sạch dữ liệu văn bản:

Quy trình làm sạch bao gồm các bước:

1. **Chuẩn hóa ngôn ngữ:** Chuyển toàn bộ văn bản về chữ thường (lowercase), loại bỏ ký tự đặc biệt, dấu câu, số và URL.
2. **Xử lý lỗi chính tả:** Sử dụng thư viện TextBlob hoặc SymSpellPy để phát hiện và sửa các lỗi chính tả phổ biến.
3. **Tách từ (Tokenization):** Dùng nltk.word\_tokenize() hoặc spaCy để tách từ trong câu.
4. **Loại bỏ stopwords:** Dùng tập từ dừng tiếng Anh có sẵn trong nltk.corpus.stopwords.
5. **Chuẩn hóa từ vựng:** Áp dụng stemming (ví dụ: PorterStemmer) hoặc lemmatization (WordNetLemmatizer).

Kết quả là mỗi review được chuyển thành danh sách từ khóa có ý nghĩa, phục vụ cho giai đoạn vector hóa.

#### 2.2.2. Chuẩn hóa và biểu diễn dữ liệu (TF-IDF, Word2Vec, One-Hot, Embedding)

Sau khi làm sạch, dữ liệu được biểu diễn bằng các kỹ thuật vector hóa:

|  |  |  |
| --- | --- | --- |
| **Phương pháp** | **Mô tả** | **Ứng dụng** |
| One-Hot Encoding | Mỗi từ là một chiều trong vector, giá trị 0/1 thể hiện sự xuất hiện. | Đơn giản, dùng cho dữ liệu nhỏ. |
| TF-IDF | Đo tần suất và tầm quan trọng của từ trong tập tài liệu. | Phù hợp cho dữ liệu đánh giá lớn. |
| Word2Vec | Học biểu diễn ngữ nghĩa từ ngữ cảnh. | Hiểu sâu mối quan hệ giữa các từ. |
| Transformer | Biểu diễn cả câu hoặc đoạn văn bản trong không gian vector dày đặc. | Hiệu quả cao với dữ liệu review dài, ngữ nghĩa phức tạp. |

**Bảng 2.2.** Các phương pháp chuẩn hóa và biểu diễn dữ liệu phi cấu trúc

#### 2.2.3. Trích chọn đặc trưng hành vi người dùng

Trích chọn đặc trưng (Feature Selection/Extraction) là quá trình xác định và chuyển đổi dữ liệu thô về hành vi người dùng thành một tập hợp các đặc điểm (features) đại diện, có ý nghĩa và hiệu quả cho việc phân cụm. Đây là bước cực kỳ quan trọng vì chất lượng của các đặc trưng ảnh hưởng trực tiếp đến kết quả phân cụm.

Mục tiêu là thu thập những thông tin cô đọng nhất, giúp thuật toán phân cụm dễ dàng nhận diện sự khác biệt giữa các nhóm người dùng. Đặc trưng hành vi người dùng thường được xây dựng dựa trên mô hình 5W1H (Who, What, When, Where, Why, How) trong ngữ cảnh tương tác của người dùng với hệ thống (ví dụ: một trang web, ứng dụng, nền tảng thương mại điện tử).

**Các loại đặc trưng hành vi thường gặp:**

* **Đặc trưng Hoạt động (Activity Features):** Đo lường mức độ tương tác.
  + **Tần suất (Frequency):** Số lần đăng nhập, số lần xem sản phẩm, số lần nhấp chuột trong một khoảng thời gian.
  + **Thời lượng (Duration):** Tổng thời gian sử dụng, thời gian trung bình cho mỗi phiên (session).
  + **Tính kịp thời (Recency):** Khoảng thời gian từ lần tương tác gần nhất.
* **Đặc trưng Nội dung (Content Features):** Liên quan đến loại nội dung người dùng quan tâm.
  + **Danh mục quan tâm:** Tỷ lệ xem các danh mục sản phẩm/bài viết khác nhau.
  + **Từ khóa tìm kiếm:** Các từ khóa người dùng thường xuyên sử dụng.
  + **Loại hành động:** Tỷ lệ giữa hành động xem, hành động thêm vào giỏ hàng, hành động mua.
* **Đặc trưng Xã hội (Social Features):** Nếu có tương tác xã hội.
  + **Mức độ ảnh hưởng:** Số lượt thích (likes), chia sẻ (shares), bình luận (comments).
  + **Mạng lưới:** Số lượng bạn bè, người theo dõi.
* **Đặc trưng Hồ sơ (Profile Features):**
  + **Nhân khẩu học:** Tuổi, giới tính, vị trí địa lý (nếu có).
  + **Thiết bị:** Loại thiết bị sử dụng (desktop, mobile).

**Kỹ thuật trích chọn / Giảm chiều dữ liệu:**

* **Phân tích thống kê:** Sử dụng các chỉ số thống kê (trung bình, phương sai) để chọn ra các đặc trưng có độ biến thiên cao (biến thiên càng cao càng hữu ích cho phân cụm).
* **Phân tích các thành phần chính (PCA - Principal Component Analysis):** Giảm số chiều của tập dữ liệu bằng cách tìm ra các tổ hợp tuyến tính của các đặc trưng ban đầu (thành phần chính) nhưng vẫn giữ lại phần lớn phương sai.
* **Mã hóa tự động (Autoencoders):** Sử dụng mạng nơ-ron để học cách mã hóa dữ liệu đầu vào thành một biểu diễn có số chiều thấp hơn.

### 2.3. Các thuật toán Phân cụm

#### 2.3.1. K-Means

**Ý tưởng chính:**

K-Means là thuật toán phân cụm dựa trên khoảng cách, có mục tiêu chia tập dữ liệu thành k cụm sao cho các điểm trong cùng một cụm gần nhau nhất và các cụm khác nhau xa nhau nhất.

Thuật toán hoạt động bằng cách tìm các tâm cụm (centroids) và tối thiểu hóa tổng khoảng cách bình phương giữa các điểm dữ liệu và tâm cụm của chúng.

**Hàm chi phí (Objective Function):**

Trong đó:

* : Cụm thứ i
* : Điểm dữ liệu thuộc cụm
* : Tâm của cụm i

**Các bước cơ bản của thuật toán K-Means:**

1. Chọn số cụm k và khởi tạo ngẫu nhiên k tâm cụm ban đầu.
2. Gán mỗi điểm dữ liệu vào cụm có tâm gần nhất (theo khoảng cách Euclidean).
3. Cập nhật lại tâm cụm bằng cách tính trung bình của các điểm trong từng cụm.
4. Lặp lại bước 2 và 3 cho đến khi tâm cụm hội tụ (không thay đổi đáng kể).

**Xác định số cụm k tối ưu:** Thường sử dụng hai phương pháp:

* **Elbow Method:** Tìm điểm “khuỷu tay” trên biểu đồ thể hiện sự giảm của hàm chi phí theo k.
* **Silhouette Score:** Đo mức độ gắn kết của điểm trong cùng cụm và sự tách biệt với cụm khác.

#### 2.3.2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

**Ý tưởng chính:**

DBSCAN là thuật toán phân cụm dựa trên mật độ, nghĩa là một cụm được hình thành khi tồn tại vùng có mật độ điểm dữ liệu cao.

Không giống K-Means, DBSCAN không cần biết trước số cụm k, và có khả năng phát hiện nhiễu (outliers) cũng như các cụm có hình dạng bất kỳ (không chỉ hình cầu).

**Các tham số chính:**

* **ε (Epsilon):** bán kính lân cận - khoảng cách tối đa để hai điểm được xem là gần nhau.
* **MinPts:** số điểm tối thiểu trong vùng lân cận ε để được coi là một vùng có mật độ cao.

**Phân loại điểm dữ liệu trong DBSCAN:**

* **Core point:** có ít nhất MinPts điểm trong vùng ε.
* **Border point:** nằm gần một core point nhưng có ít hơn MinPts lân cận.
* **Noise point (outlier):** không thuộc vùng mật độ nào.

**Các bước cơ bản của DBSCAN:**

1. Chọn một điểm dữ liệu chưa được thăm.
2. Tìm tất cả các điểm nằm trong vùng ε của điểm đó.
   * Nếu số lượng điểm ≥ MinPts, điểm đó là core point, và tạo một cụm mới.
   * Nếu không, đánh dấu điểm đó là noise (tạm thời).
3. Mở rộng cụm bằng cách lặp lại quá trình với các điểm trong vùng lân cận, thêm các điểm đủ điều kiện vào cùng cụm.
4. Tiếp tục cho đến khi tất cả các điểm đều được gán vào cụm hoặc được đánh dấu là nhiễu.

#### 2.3.3. Ưu - nhược điểm của từng phương pháp:

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Ưu điểm** | **Nhược điểm** |
| K-Means | Dễ hiểu, tính toán nhanh, hiệu quả với dữ liệu lớn. | Cần biết trước số cụm *k*, nhạy với nhiễu và điểm ngoại lai, chỉ phù hợp với cụm có dạng cầu. |
| DBSCAN | Tự xác định số cụm, phát hiện được nhiễu, xử lý tốt cụm có hình dạng bất kỳ. | Khó chọn tham số *epsilon* và *minPoints*, hoạt động kém khi mật độ cụm khác nhau. |

**Bảng 2.3.** Ưu, nhược điểm của K-Means và DBSCAN

## **CHƯƠNG 3: DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU**

### 3.1. Nguồn dữ liệu và mô tả

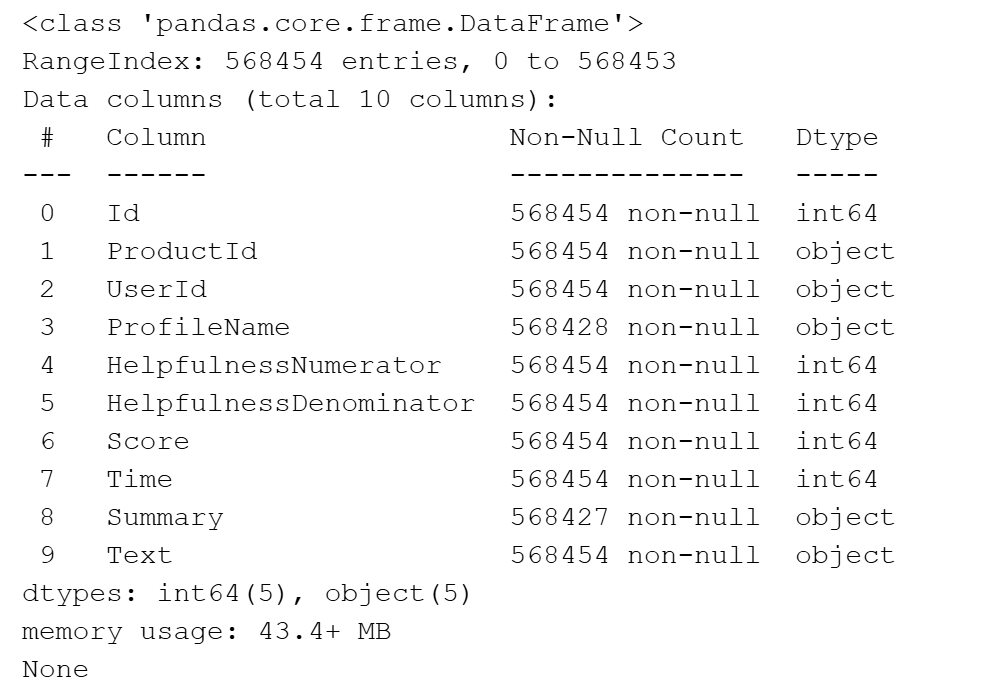
* **Giới thiệu:** Tập dữ liệu Amazon Fine Food Reviews (các trường dữ liệu quan trọng gồm: UserID, ProductID, Score (điểm đánh giá), Text, …).

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import nltk  from nltk.corpus import stopwords  from nltk import pos\_tag, word\_tokenize  import re |

|  |
| --- |
| nltk.download('punkt')  nltk.download('stopwords')  nltk.download('averaged\_perceptron\_tagger') |

|  |
| --- |
| # 3.1 Đọc dữ liệu và mô tả  df = pd.read\_csv(r'D:\DataMining\CK\_DBSCAN\Reviews.csv\Reviews.csv')  print(df.info()) |

Kết quả có được là:



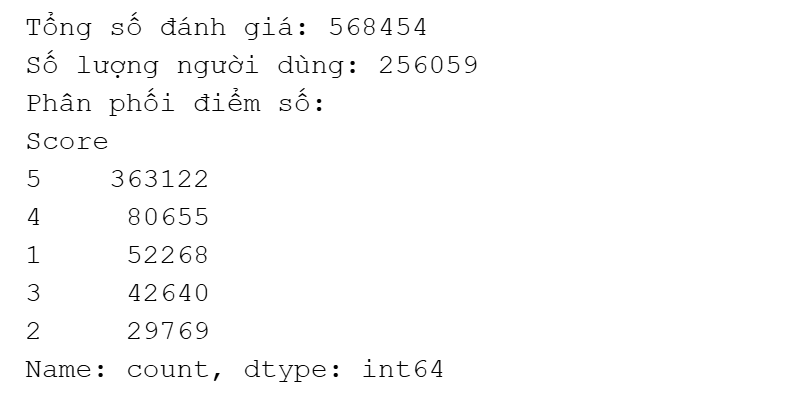
**Hình 3.1.** Thông tin các cột của bộ dữ liệu sau khi đọc vào

Các dữ liệu ở dạng Numeric (int64) hoặc object và được đọc ra từ file .csv.

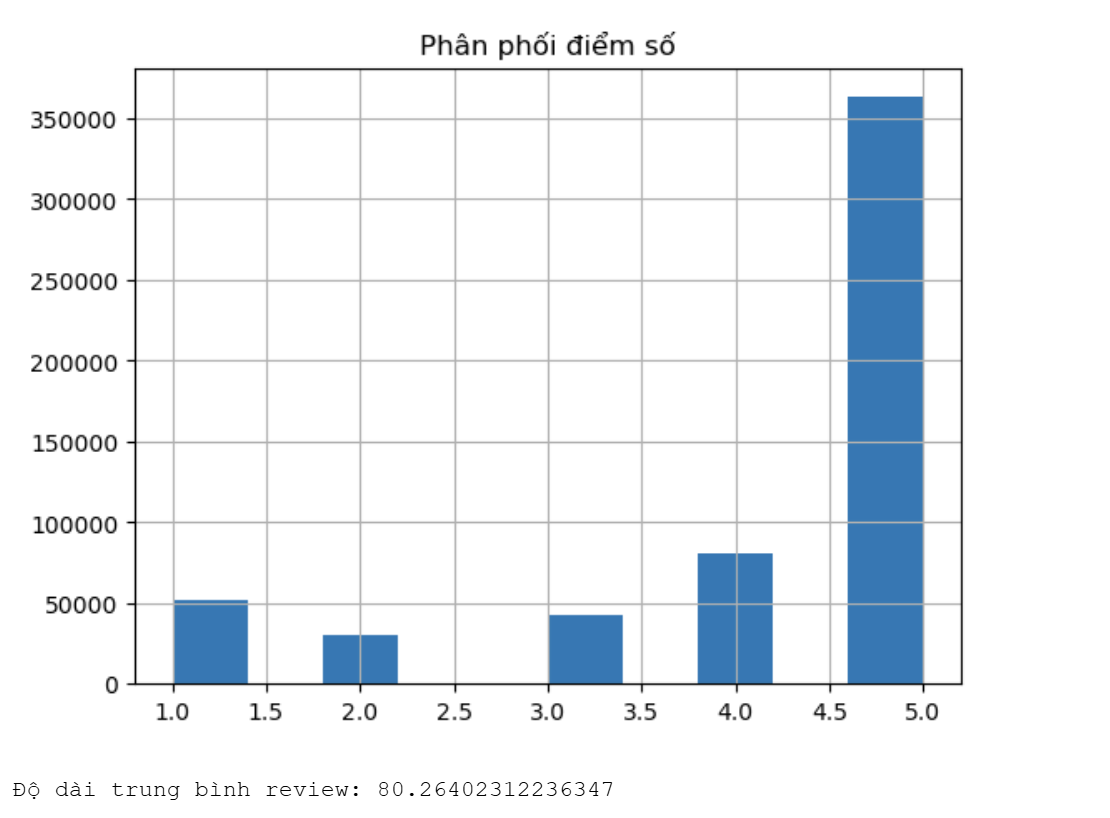
* **Thống kê mô tả:** Số lượng đánh giá (500k+) và Số lượng người dùng (260k+), Phân phối điểm đánh giá, độ dài trung bình của review.

|  |
| --- |
| # Thống kê cơ bản  print("Tổng số đánh giá:", len(df))  print("Số lượng người dùng:", df['UserId'].nunique())  print("Phân phối điểm số:")  print(df['Score'].value\_counts())  df['Score'].hist()  plt.title('Phân phối điểm số')  plt.show()  print("Độ dài trung bình review:", df['Text'].apply(lambda x: len(str(x).split())).mean()) |

Kết quả thu được như sau:



**Hình 3.2.** Kết quả thống kê cơ bản tập dữ liệu ban đầu



**Hình 3.3.** Bảng phân phối điểm đánh giá sao của người dùng

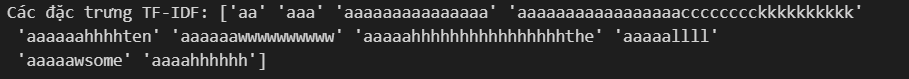
Từ hình trên, chúng ta thấy được rằng số lượng người dùng đánh giá 5 sao và 4 sao là khá cao (có khoảng trên 400 nghìn đánh giá từ 4 sao trở lên), còn số lượng người dùng cảm thấy chưa hài lòng về sản phẩm (từ dưới 4 sao) là khoảng hơn 100 nghìn đánh giá.

### 3.2. Quy trình tiền xử lý dữ liệu

* **Làm sạch dữ liệu:** Xử lý giá trị thiếu, loại bỏ đánh giá trùng lặp.
* **Tiền xử lý văn bản (NLP Preprocessing):**
  + Chuyển về chữ thường.
  + Xử lý các từ sai chính tả
  + Loại bỏ các ký tự đặc biệt, HTML tags, …
  + Tokenization (tách từ).
  + Loại bỏ Stop-words (từ dừng).

|  |
| --- |
| # 3.2 Tiền xử lý dữ liệu  import pandas as pd  import re  import nltk  from nltk.corpus import stopwords  from nltk.tokenize import word\_tokenize  from nltk.stem import WordNetLemmatizer  from symspellpy import SymSpell, Verbosity  # Tải dữ liệu cần thiết từ NLTK  nltk.download('punkt')  nltk.download('stopwords')  nltk.download('wordnet')  # Khởi tạo stopwords và lemmatizer  stop\_words = set(stopwords.words('english'))  lemmatizer = WordNetLemmatizer()  # Khởi tạo SymSpell để sửa lỗi chính tả  sym\_spell = SymSpell(max\_dictionary\_edit\_distance=2, prefix\_length=7)  sym\_spell.load\_dictionary("frequency\_dictionary\_en\_82\_765.txt", term\_index=0, count\_index=1)  # Hàm sửa lỗi chính tả bằng SymSpell  def correct\_spelling\_symspell(text):  suggestions = sym\_spell.lookup\_compound(text, max\_edit\_distance=2)  return suggestions[0].term if suggestions else text  # Đọc dữ liệu và loại bỏ trùng lặp, thiếu dữ liệu  df = df.drop\_duplicates(subset=['UserId', 'ProductId', 'Text'])  df = df.dropna(subset=['Text'])  # Hàm làm sạch văn bản  def clean\_text(text):  text = str(text).lower()  text = re.sub(r'<.\*?>', '', text) # Xóa HTML tags  text = re.sub(r'[^a-zA-Z ]', '', text) # Xóa ký tự đặc biệt  text = correct\_spelling\_symspell(text) # Sửa lỗi chính tả  tokens = word\_tokenize(text)  cleaned\_tokens = []  for w in tokens:  if (  w in stop\_words or # là stop-word  len(w) > 20 or # quá dài  re.search(r'(.)\1{2,}', w) # có ký tự lặp lại ≥ 3 lần  ):  continue  cleaned\_tokens.append(lemmatizer.lemmatize(w))  return ' '.join(cleaned\_tokens)  # Áp dụng tiền xử lý  df['clean\_text'] = df['Text'].apply(clean\_text)  print(df[['Text', 'clean\_text']].head()) |

Trong mỗi bình luận không chỉ có các từ dừng như: a, an, the, and, this… mà còn các từ mang ý nghĩa bộc lộ cảm xúc nhưng lại không thể dùng để phân cụm, bởi chúng vô nghĩa như hình sau:



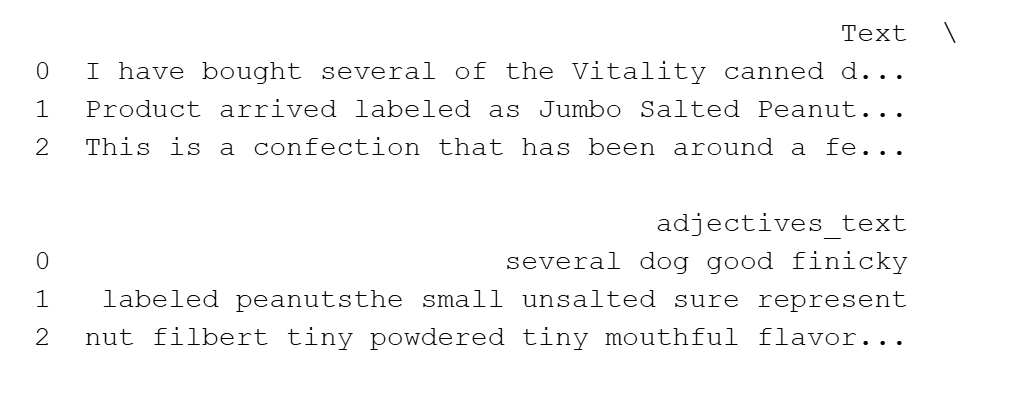
**Hình 3.4.** Các từ mang ý nghĩa bộc lộ cảm xúc nhưng vô nghĩa

Chúng gây ảnh hưởng tới kết quả phân cụm đối với TF-IDF (dùng tần suất xuất hiện của từ trong văn bản). Do vậy, chúng ta cần phải loại bỏ các từ này bằng:

* Lemmatization/Stemming (nếu có).
* Gắn thẻ từ loại (POS Tagging) và lọc tính từ: Tập trung vào các tính từ mô tả hương vị, kết cấu, chất lượng.

|  |
| --- |
| import nltk  # 1. Đảm bảo các tài nguyên cần thiết đã được tải  # Mặc dù bạn đã chạy chúng, hãy thêm lại để chắc chắn.  # Hoặc chạy chúng một lần nữa trong một ô riêng biệt.  try:  nltk.data.find('tokenizers/punkt')  except LookupError:  print("Downloading punkt...")  nltk.download('punkt')  try:  nltk.data.find('corpora/stopwords')  except LookupError:  print("Downloading stopwords...")  nltk.download('stopwords')  try:  nltk.data.find('taggers/averaged\_perceptron\_tagger')  except LookupError:  print("Downloading averaged\_perceptron\_tagger...")  nltk.download('averaged\_perceptron\_tagger')  try:  nltk.data.find('taggers/averaged\_perceptron\_tagger\_eng')  except LookupError:  print("Downloading averaged\_perceptron\_tagger...")  nltk.download('averaged\_perceptron\_tagger\_eng')  # NLP Preprocessing & Lọc tính từ  stop = set(stopwords.words('english'))  def extract\_adj(text):  if not isinstance(text, str) or pd.isna(text): # Xử lý NaN và non-string  return ''  tokens = word\_tokenize(text)  tokens = [w for w in tokens if w not in stop]  tags = pos\_tag(tokens)  adjectives = [w for w, tag in tags if tag.startswith("JJ")]  return ' '.join(adjectives)  df['adjectives\_text'] = df['clean\_text'].apply(extract\_adj)  print(df[['Text', 'adjectives\_text']].head(3)) |

Kết quả thu được như sau:



**Hình 3.5.** Đánh giá sau khi làm sạch và các tính từ thu được sau khi lọc

Từ kết quả này, chúng ta thấy rõ và thu được các tính từ. Điều này thể hiện rõ hơn cảm xúc của mỗi đánh giá của người dùng.

Sau đó, tiến hành xuất file dữ liệu sau khi đã được làm sạch:

|  |
| --- |
| # Trích xuất và lưu dữ liệu đã xử lý  df.to\_csv('processed\_reviews.csv', index=False) |

## 

## **CHƯƠNG 4: THỰC HIỆN TRÍCH XUẤT ĐẶC TRƯNG, XÂY DỰNG HỒ SƠ VÀ GIẢM CHIỀU**

### 4.1. Trích xuất đặc trưng định tính (Review Vector)

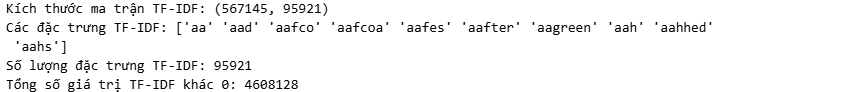
Thực hiện TF-IDF và Word2Vec trên dữ liệu văn bản đã lọc (Review Vector).

Với TF-IDF:

|  |
| --- |
| from sklearn.feature\_extraction.text import TfidfVectorizer  from gensim.models import Word2Vec  import numpy as np  from sklearn.preprocessing import MinMaxScaler  from sklearn.decomposition import PCA  import pandas as pd |

|  |
| --- |
| # 4.1 TF-IDF  df1 = pd.read\_csv('processed\_reviews.csv')  df1['adjectives\_text'] = df1['adjectives\_text'].fillna('')  tfidf = TfidfVectorizer()  tfidf\_matrix = tfidf.fit\_transform(df1['adjectives\_text'])  # Hiển thị kích thước và 10 đặc trưng đầu tiên  print("Kích thước ma trận TF-IDF:", tfidf\_matrix.shape)  print("Các đặc trưng TF-IDF:", tfidf.get\_feature\_names\_out()[:10])  # Đánh giá độ thưa (sparsity) của ma trận TF-IDF.  print("Số lượng đặc trưng TF-IDF:", len(tfidf.get\_feature\_names\_out()))  print("Tổng số giá trị TF-IDF khác 0:", tfidf\_matrix.nnz) |

Kết quả thu được như sau:



**Hình 4.1.** Ma trận TF-IDF và các đặc trưng đầu tiên

Đánh giá kết quả:

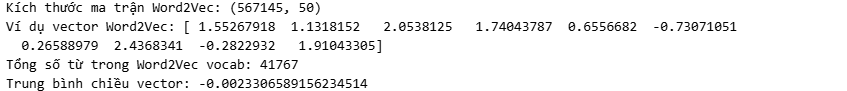
* Ma trận TF-IDF có kích thước rất lớn (567.145 dòng tương ứng với số lượng review, và 95.921 cột tương ứng với số từ đặc trưng trong toàn bộ tập dữ liệu).
* Tỷ lệ các giá trị khác 0 (~4.6 triệu trên tổng 54 tỷ phần tử) cho thấy độ thưa (sparsity) rất cao, điều này là đặc trưng phổ biến trong biểu diễn TF-IDF.
* Tập đặc trưng lớn giúp mô hình có khả năng phân biệt các từ mang tính biểu cảm (tính từ), tuy nhiên cũng khiến chi phí tính toán và bộ nhớ tăng mạnh.

=> Kết luận: Phương pháp TF-IDF đã trích xuất thành công các đặc trưng văn bản định tính từ các câu đánh giá. Tuy ma trận đầu ra có kích thước lớn và thưa, song đây là biểu diễn hiệu quả để nắm bắt tầm quan trọng tương đối của các tính từ trong từng review.

Với Word2Vec:

|  |
| --- |
| # 4.1 Word2Vec  # Chuyển mỗi câu thành danh sách các từ, đồng thời huấn luyện mô hình  sentences = [txt.split() for txt in df1['adjectives\_text']]  w2v\_model = Word2Vec(sentences, vector\_size=50, min\_count=2, workers=4)  # Tính vector trung bình của các từ trong một câu  def get\_w2v\_vector(tokens):  vectors = [w2v\_model.wv[w] for w in tokens if w in w2v\_model.wv]  if len(vectors):  return np.mean(vectors, axis=0)  else:  return np.zeros(50)  # Lưu vector biểu diễn cho từng câu  w2v\_features = np.array([get\_w2v\_vector(sent) for sent in sentences])  # In ra kích thước và ví dụ vector giúp đảm bảo Word2Vec chạy ổn.  print("Kích thước ma trận Word2Vec:", w2v\_features.shape)  print("Ví dụ vector Word2Vec:", w2v\_features[0][:10])  # Cho biết kích thước vocab và kiểm tra có bias hoặc dữ liệu rỗng.  print("Tổng số từ trong Word2Vec vocab:", len(w2v\_model.wv))  print("Trung bình chiều vector:", np.mean(w2v\_features)) |

Kết quả thu được như sau:



**Hình 4.2.** Ma trận Word2Vec và ví dụ biểu diễn vector từ

Đánh giá kết quả:

* Số lượng từ vựng (41.767) cho thấy dữ liệu có độ đa dạng cao, phản ánh nhiều sắc thái cảm xúc của người dùng.
* Kích thước vector 50 chiều là hợp lý để cân bằng giữa khả năng biểu diễn ngữ nghĩa và hiệu năng xử lý.
* Trung bình chiều vector gần 0 chứng tỏ mô hình không bị lệch (bias) mạnh về hướng nào, tức là huấn luyện ổn định.

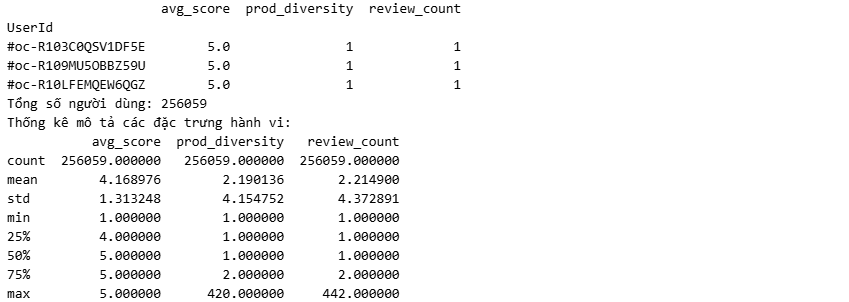
=> Kết luận: Word2Vec đã biểu diễn tốt mối quan hệ ngữ nghĩa giữa các từ, giúp chuyển đổi văn bản thành dạng vector liên tục. Đây là bước quan trọng để sau này tổng hợp thành hồ sơ người dùng theo đặc trưng ngữ nghĩa thay vì chỉ tần suất xuất hiện.

### 4.2. Trích xuất đặc trưng định lượng (User-level)

* **Tính toán các chỉ số hành vi cho mỗi người dùng:**
  + Điểm đánh giá trung bình (Average Score).
  + Số lượng đánh giá (Review Count).
  + (Tùy chọn) Độ đa dạng sản phẩm đã đánh giá (Diversity of Product IDs).

|  |
| --- |
| # 4.2 Đặc trưng định lượng  # Gom dữ liệu theo từng UserId để tính 3 đặc trưng hành vi  user\_stats = df1.groupby('UserId').agg({  'Score':'mean',  'ProductId':'nunique',  'Text':'count'  }).rename(columns={  'Score':'avg\_score',  'ProductId':'prod\_diversity',  'Text':'review\_count'  })  # Hiển thị 3 dòng đầu tiên và xác nhận dữ liệu người dùng và kiểm tra phân phối các chỉ số hành vi.  print(user\_stats.head(3))  print("Tổng số người dùng:", user\_stats.shape[0])  print("Thống kê mô tả các đặc trưng hành vi:")  print(user\_stats.describe())  # Chuẩn hóa các giá trị về khoảng [0, 1]  scaler = MinMaxScaler()  quant\_features = scaler.fit\_transform(user\_stats) |

Kết quả thu được như sau:



**Hình 4.3.** Thống kê mô tả các đặc trưng hành vi người dùng

* **Chuẩn hóa (Scaling):** Sử dụng MinMaxScaler để chuẩn hóa các đặc trưng này, tránh việc các giá trị lớn (ví dụ: Review Count) lấn át các giá trị nhỏ (ví dụ: Average Score).

|  |
| --- |
| # Chuẩn hóa các giá trị về khoảng [0, 1]  scaler = MinMaxScaler()  quant\_features = scaler.fit\_transform(user\_stats) |

Đánh giá kết quả:

* Phần lớn người dùng chỉ để lại 1–2 đánh giá và chủ yếu chấm 4–5 sao, thể hiện xu hướng tích cực trong tập dữ liệu.
* Một số ít người dùng có độ đa dạng sản phẩm rất cao (tới 420) và số lượng đánh giá lên tới 442, cho thấy sự tồn tại của nhóm người dùng “nhiệt tình” hoặc “chuyên nghiệp”.

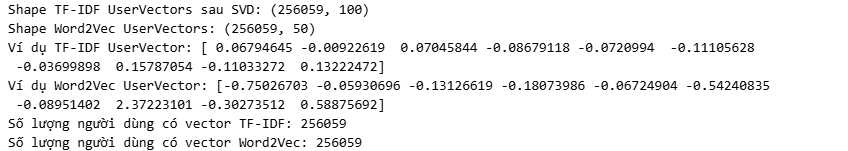
=> Kết luận: Bộ đặc trưng định lượng (Average Score, Product Diversity, Review Count) phản ánh rõ hành vi của người dùng, làm cơ sở để kết hợp với đặc trưng ngữ nghĩa ở bước tiếp theo.

### 4.3. Xây dựng hồ sơ người dùng

* **Tổng hợp đặc trưng văn bản:** Tính trung bình cộng (Mean Aggregation) các Review Vector (từ TF-IDF và Word2Vec) thuộc về cùng một UserId để tạo ra Textual User Vector.

|  |
| --- |
| from sklearn.decomposition import TruncatedSVD  import numpy as np  # 1. Giảm chiều ma trận TF-IDF  n\_components = 100 # Chọn số chiều phù hợp với cấu hình máy  svd = TruncatedSVD(n\_components=n\_components, random\_state=42)  tfidf\_reduced = svd.fit\_transform(tfidf\_matrix) # TF-IDF gốc là sparse matrix, đầu ra là dense  # 2. Tổng hợp User Vector từ ma trận giảm chiều  user\_indices = df1.groupby('UserId').indices  tfidf\_means = []  w2v\_means = []  for reviewer, idx in user\_indices.items():  # Trung bình TF-IDF đã giảm chiều (chỉ còn ~100 chiều)  tfidf\_user = tfidf\_reduced[idx].mean(axis=0)  tfidf\_means.append(tfidf\_user)  # Trung bình Word2Vec  w2v\_user = np.mean(w2v\_features[idx], axis=0)  w2v\_means.append(w2v\_user)  tfidf\_means = np.array(tfidf\_means)  w2v\_means = np.array(w2v\_means)  # Mỗi user chỉ còn vector ~100 chiều => dễ phân cụm, dễ phân tích!  print("Shape TF-IDF UserVectors sau SVD:", tfidf\_means.shape)  print("Shape Word2Vec UserVectors:", w2v\_means.shape)  print("Ví dụ TF-IDF UserVector:", tfidf\_means[0][:10])  print("Ví dụ Word2Vec UserVector:", w2v\_means[0][:10])  # Xác nhận khớp với số lượng user trong user\_stats.  print("Số lượng người dùng có vector TF-IDF:", len(tfidf\_means))  print("Số lượng người dùng có vector Word2Vec:", len(w2v\_means)) |

Kết quả thu được như sau:



**Hình 4.4.** Biểu diễn người dùng bằng vector TF-IDF và Word2Vec

Đánh giá kết quả:

* Sau khi giảm chiều TF-IDF bằng SVD, mỗi người dùng được biểu diễn bằng vector 100 chiều (TF-IDF) và 50 chiều (Word2Vec).
* Việc tính trung bình các review vector theo từng người dùng giúp làm nổi bật phong cách ngôn ngữ tổng thể của họ thay vì từng nhận xét đơn lẻ.
* Hai loại vector này phản ánh song song mức độ quan trọng của từ (TF-IDF) và mối quan hệ ngữ nghĩa giữa các từ (Word2Vec).

=> Kết luận: Hồ sơ người dùng (User Profile) được xây dựng đầy đủ, mỗi người giờ đây có biểu diễn ngữ nghĩa và hành vi riêng, sẵn sàng cho các bước phân cụm hoặc gợi ý tiếp theo.

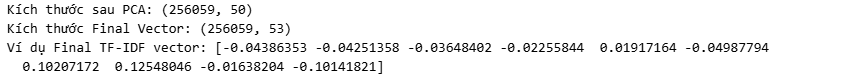
**4.4. Giảm chiều dữ liệu (Principal Component Analysis - PCA)** và tổng hợp vector (User Profile Construction)

* **Lý do:** Áp dụng PCA trên Textual User Vector trước khi kết hợp để giảm nhiễu và chi phí tính toán.
* **Thực hiện:** Giảm chiều Textual User Vector xuống k chiều (k nằm trong khoảng 50 - 100).
* **Kết hợp đặc trưng (Feature Concatenation):** Ghép Textual User Vector (đã giảm chiều) với Đặc trưng Định lượng (đã chuẩn hóa) để tạo ra Final User Vector cho phân cụm.

|  |
| --- |
| import pandas as pd  # Lấy danh sách user\_id đúng thứ tự với đặc trưng (đã tạo ở trên)  user\_ids = list(user\_indices.keys())  # Tạo DataFrame cho TF-IDF  tfidf\_profile\_df = pd.DataFrame(tfidf\_means)  tfidf\_profile\_df['UserId'] = user\_ids  tfidf\_profile\_df.to\_csv('user\_profile\_tfidf.csv', index=False)  # Tạo DataFrame cho Word2Vec  w2v\_profile\_df = pd.DataFrame(w2v\_means)  w2v\_profile\_df['UserId'] = user\_ids  w2v\_profile\_df.to\_csv('user\_profile\_word2vec.csv', index=False) |

|  |
| --- |
| # 4.4 PCA giảm chiều  pca = PCA(n\_components=50)  tfidf\_pca = pca.fit\_transform(tfidf\_means)  w2v\_pca = pca.fit\_transform(w2v\_means)  print("Kích thước sau PCA:", tfidf\_pca.shape)  # Nối đặc trưng văn bản + định lượng cho mỗi user  final\_tfidf = np.hstack([tfidf\_pca, quant\_features])  final\_w2v = np.hstack([w2v\_pca, quant\_features])  print("Kích thước Final Vector:", final\_tfidf.shape) |

Kết quả thu được như sau:



**Hình 4.5.** Kết quả giảm chiều dữ liệu và tạo vector đặc trưng cuối cùng của người dùng

Đánh giá kết quả:

* PCA giúp giảm số chiều của TF-IDF từ 100 → 50, giảm nhiễu và chi phí tính toán trong khi vẫn giữ phần lớn thông tin (thường >80% phương sai).
* Việc ghép 3 đặc trưng định lượng (đã chuẩn hóa) với vector PCA cho ra Final User Vector 53 chiều, đủ nhỏ để xử lý nhanh trong các mô hình phân cụm hoặc gợi ý sản phẩm.
* Dữ liệu sau PCA có tính ổn định và dễ dàng trực quan hóa hoặc sử dụng cho bước đánh giá tiếp theo (ví dụ: K-Means, DBSCAN,…).

=> Kết luận: Quy trình giảm chiều và ghép đặc trưng đã hoàn tất thành công, giúp tạo ra hồ sơ người dùng cô đọng, đầy đủ thông tin từ cả khía cạnh ngôn ngữ và hành vi. Đây là đầu vào quan trọng cho bước phân tích hoặc mô hình gợi ý ở chương sau.

|  |
| --- |
| import pandas as pd  # Nếu bạn đã có final\_tfidf, final\_w2v là mảng numpy, có thể export như sau:  pd.DataFrame(final\_tfidf).to\_csv('final\_tfidf.csv', index=False)  pd.DataFrame(final\_w2v).to\_csv('final\_w2v.csv', index=False) |

Sau khi đã hoàn tất quá trình giảm chiều và kết hợp đặc trưng định lượng, bước tiếp theo là lưu trữ kết quả biểu diễn người dùng để phục vụ cho các giai đoạn phân tích và phân cụm sau này.

Hai tệp dữ liệu được xuất ra gồm:

* final\_tfidf.csv: Hồ sơ người dùng dựa trên đặc trưng TF-IDF đã giảm chiều.
* final\_w2v.csv: Hồ sơ người dùng dựa trên đặc trưng Word2Vec đã giảm chiều.

Mỗi dòng trong hai tệp này tương ứng với một người dùng duy nhất, được biểu diễn bằng vector đặc trưng 53 chiều (gồm 50 chiều từ PCA và 3 chiều định lượng).

Dữ liệu này đóng vai trò như đầu vào cho các mô hình học không giám sát (như K-Means, DBSCAN) hoặc các hệ thống gợi ý sản phẩm cá nhân hóa ở các chương sau.

## **CHƯƠNG 5: PHÂN CỤM VÀ PHÂN KHÚC KHÁCH HÀNG**

### 5.1. Phân cụm bằng K-Means

* **Xác định k tối ưu:** Phân tích biểu đồ Elbow, hoặc chỉ số Silhouette.

**Elbow:**

* TF-IDF:

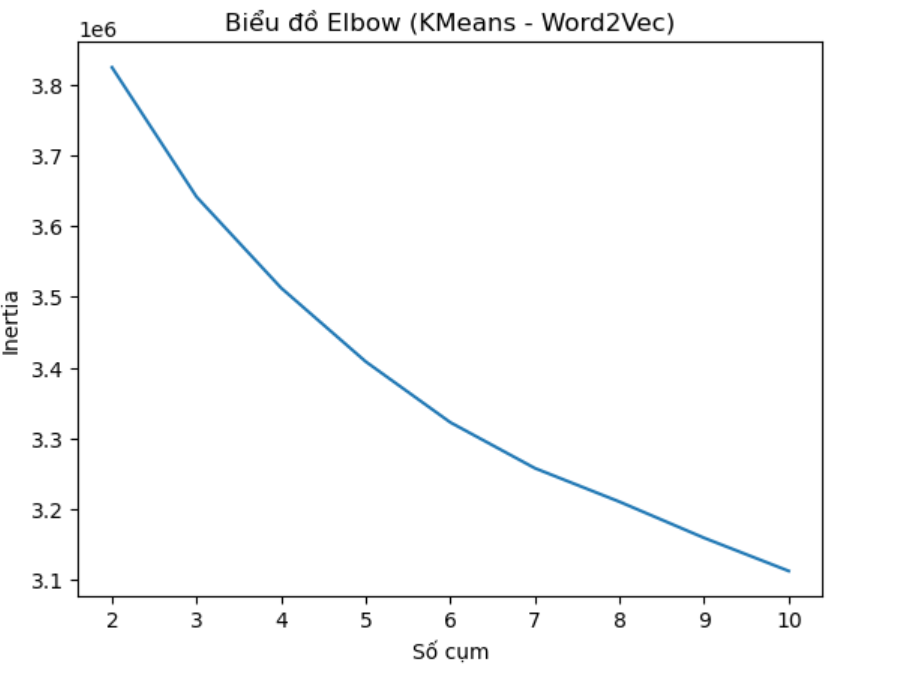
|  |
| --- |
| df\_final = pd.read\_csv('final\_tfidf.csv')  final\_tfidf = df\_final.values  # 5.1 Phân cụm KMeans (Elbow & Silhouette)  inertias = []  K = range(2,11)  for k in K:  km = KMeans(n\_clusters=k, random\_state=42)  km.fit(final\_tfidf)  inertias.append(km.inertia\_)  plt.plot(K, inertias)  plt.xlabel('Số cụm')  plt.ylabel('Inertia')  plt.title('Biểu đồ Elbow (KMeans)')  plt.show() |



**Hình 5.1.** Biểu đồ Elbow (TF-IDF)

* Word2Vec:

|  |
| --- |
| df\_final\_w2v = pd.read\_csv('final\_w2v.csv')  w2v\_pca = df\_final\_w2v.values  inertias\_w2v = []  K\_w2v = range(2, 11)  for k in K\_w2v:  km = KMeans(n\_clusters=k, random\_state=42)  km.fit(w2v\_pca)  inertias\_w2v.append(km.inertia\_)  plt.plot(K\_w2v, inertias\_w2v)  plt.xlabel('Số cụm')  plt.ylabel('Inertia')  plt.title('Biểu đồ Elbow (KMeans - Word2Vec)')  plt.show() |

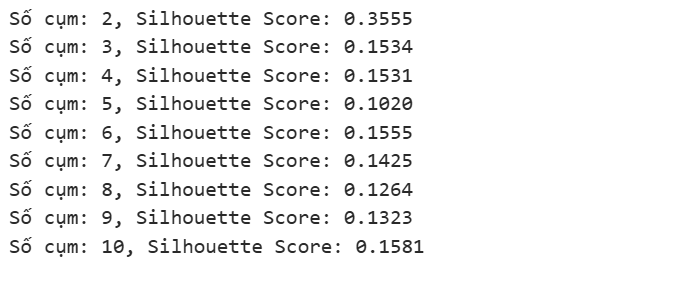


**Hình 5.2.** Biểu đồ Elbow (Word2Vec)

**Silhouette:** Sử dụng Silhouette để kiểm tra lại kết quả tìm cụm ở phía trên, đưa ra lựa chọn tốt hơn.

|  |
| --- |
| for k in range(2, 11): #Dựa vào cái này để trình bày là chính, tại sao gtri nó lại xếp lộn xộn  km = KMeans(n\_clusters=k)  labels = km.fit\_predict(final\_tfidf)  sil\_score = silhouette\_score(final\_tfidf, labels)  print(f"Số cụm: {k}, Silhouette Score: {sil\_score:.4f}") |

Kết quả thu được như sau:



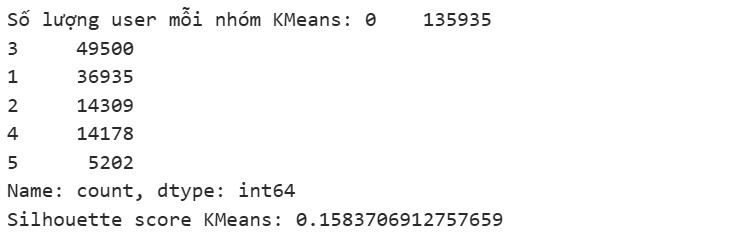
**Hình 5.3.** Kết quả Silhouette

Từ kết quả trên, chúng ta có thể dễ dàng thấy được với số cụm là 6 thì giá trị của Silhouette là gần như ổn định nhất.

* **Kết quả phân cụm:** Phân cụm trên Final User Vector (kết hợp của Textual + Quantitative Features).
* TF-IDF:

|  |
| --- |
| from sklearn.decomposition import PCA  import matplotlib.pyplot as plt  k\_best = 6 # Có thể chọn dựa vào Elbow hoặc Silhouette Score tốt nhất  kmeans = KMeans(n\_clusters=k\_best, ramdom\_state=42).fit(final\_tfidf)  labels\_kmeans = kmeans.labels\_  print("Số lượng user mỗi nhóm KMeans:", pd.Series(labels\_kmeans).value\_counts())  print("Silhouette score KMeans:", silhouette\_score(final\_tfidf, labels\_kmeans))  # Trực quan hóa sau PCA 2 chiều  pca2 = PCA(n\_components=2)  red\_2d = pca2.fit\_transform(final\_tfidf)  plt.scatter(red\_2d[:,0], red\_2d[:,1], c=labels\_kmeans)  plt.title('Scatter plot KMeans')  plt.show() |

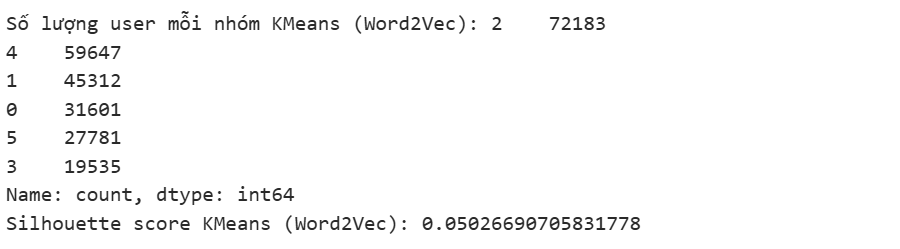
Thu được kết quả như sau:



**Hình 5.4.** Kết quả phân cụm (TF-IDF) với k = 6

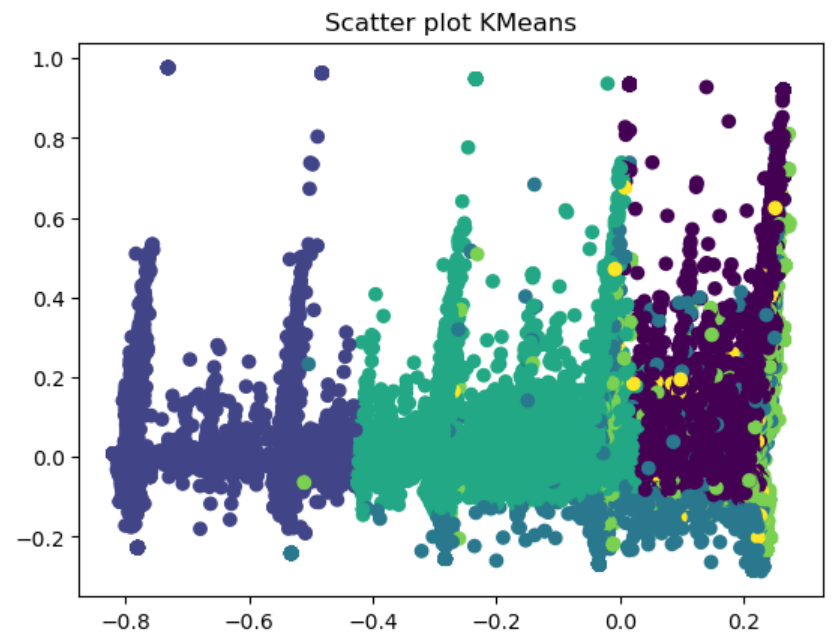
* Word2Vec:

|  |
| --- |
| # chạy với w2v  k\_best = 6 # Có thể chọn dựa vào Elbow hoặc Silhouette Score tốt nhất  kmeans = KMeans(n\_clusters=k\_best, ramdom\_state=42).fit(w2v\_pca)  labels\_kmeans = kmeans.labels\_  print("Số lượng user mỗi nhóm KMeans (Word2Vec):", pd.Series(labels\_kmeans).value\_counts())  print("Silhouette score KMeans (Word2Vec):", silhouette\_score(w2v\_pca, labels\_kmeans))  # Trực quan hóa sau PCA 2 chiều (chỉ cần giảm tiếp còn 2 chiều với dữ liệu này)  from sklearn.decomposition import PCA  import matplotlib.pyplot as plt  pca2 = PCA(n\_components=2)  red\_2d = pca2.fit\_transform(w2v\_pca)  plt.scatter(red\_2d[:,0], red\_2d[:,1], c=labels\_kmeans)  plt.title('Scatter plot KMeans (Word2Vec)')  plt.show() |

Thu được kết quả như sau:

**Hình 5.5.** Kết quả phân cụm (Word2Vec) với k = 6

* **Trực quan hóa các cụm thu được:** Ví dụ như biểu đồ Scatter Plot 2D sau khi giảm chiều.
* **TF-IDF:**



**Hình 5.6.** Kết quả phân cụm sử dụng phương pháp TF-IDF

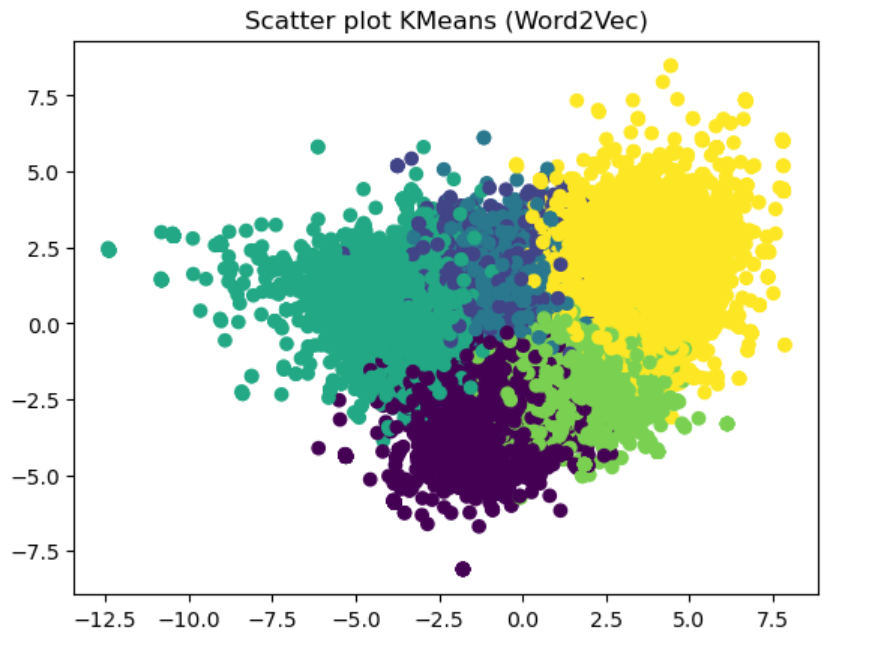
Với kết quả như trên, chúng ta thấy được:

**Nhận xét:**

* Các cụm dữ liệu có xu hướng phân bố dọc, xen kẽ nhau - tức là ranh giới giữa các cụm không tách biệt rõ ràng.
* Điều này là do TF-IDF chỉ phản ánh tần suất xuất hiện của từ, chứ không hiểu được ngữ nghĩa giữa các từ.
* Hai câu có ý nghĩa gần nhau nhưng dùng từ khác nhau sẽ dễ bị tách vào hai cụm khác nhau.

**Ý nghĩa với hành vi người dùng:**

* Mô hình này chủ yếu phân biệt dựa trên cách diễn đạt, chứ chưa phản ánh thực sự ý định hay cảm xúc.
* Phù hợp khi bạn muốn nhóm các phản hồi hoặc bình luận có từ khóa tương đồng (ví dụ: “ngon”, “tốt”, “đẹp”) - tức là mức độ liên quan theo từ khóa.
* **Word2Vec:**



**Hình 5.7.** Kết quả phân cụm sử dụng phương pháp Word2Vec

Với kết quả phân cụm như trên, chúng ta thấy được rằng:

**Nhận xét:**

* Các cụm phân bố tách biệt rõ ràng hơn, hình thành thành từng vùng có ranh giới tự nhiên.
* Word2Vec biến từ thành vector theo ngữ cảnh, nên các câu có ý nghĩa tương tự sẽ nằm gần nhau dù khác từ vựng.
* Điều này cho thấy mô hình hiểu được ngữ nghĩa và mối quan hệ giữa các từ tốt hơn TF-IDF.

**Ý nghĩa với hành vi người dùng:**

* Kết quả này phản ánh mức độ tương đồng về ý định, cảm xúc hoặc trải nghiệm người dùng.
* Giúp ta phân nhóm người dùng hoặc phản hồi theo chủ đề thực tế, ví dụ:
  + Cụm 1: người dùng hài lòng (từ ngữ tích cực)
  + Cụm 2: người dùng phàn nàn (từ ngữ tiêu cực)
  + Cụm 3: người dùng trung lập hoặc hỏi thông tin

Từ đó, doanh nghiệp có thể hiểu hành vi người dùng sâu hơn và đưa ra chiến lược cá nhân hóa hoặc cải thiện sản phẩm/dịch vụ phù hợp.

=> Với kết quả phân cụm sử dụng TF-IDF, các cụm chưa tách biệt rõ ràng, cho thấy mô hình chỉ phản ánh sự khác biệt về mặt từ vựng.

**Tổng kết:**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **TF-IDF** | **Word2Vec** |
| Cơ sở | Tần suất từ | Nghĩa của từ |
| Cấu trúc cụm | Rời rạc, xen kẽ | Rõ ràng, tách biệt |
| Mức độ hiểu ngữ cảnh | Thấp | Cao |
| Phù hợp với | Tìm từ khóa, lọc văn bản | Hiểu chủ đề, cảm xúc, hành vi |
| Ý nghĩa hành vi | Phân loại theo cách viết | Phân loại theo ý định/ngữ nghĩa |

**Bảng 5.1.** Bảng đánh giá 2 phương pháp TF-IDF và Word2Vec

Trong khi đó, kết quả phân cụm bằng Word2Vec cho thấy sự phân nhóm rõ ràng hơn, thể hiện khả năng nắm bắt ngữ nghĩa và mối quan hệ giữa các từ.

Do đó, Word2Vec giúp mô hình hiểu rõ hơn về ý định và hành vi của người dùng, phù hợp cho việc phân tích cảm xúc hoặc chủ đề trong phản hồi khách hàng.

### 5.2. Phân cụm bằng DBSCAN

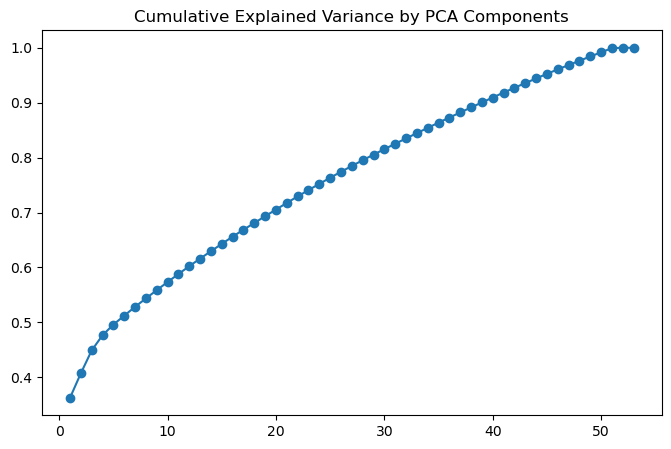
Đối với việc sử dụng DBSCAN để phân cụm cho dữ liệu có tới hơn 256 nghìn dòng sẽ khiến cho máy tính bình thường dễ gặp phải vấn đề về OOM, khiến máy tính đột ngột chương trình và tệ hơn là ngừng hẳn máy tính. Vì vậy mà trong bài này, chúng em đã thực hiện chọn lấy 10 nghìn mẫu để làm dữ liệu. Điều này sẽ khiến kết quả thu được không chính xác như mong đợi.

Tuy nhiên chúng ta vẫn sẽ xác định đâu là số chiều nên giảm để giữ lại được độ chính xác nhất cho dữ liệu:

* Đối với TF-IDF:

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.decomposition import PCA  import matplotlib.pyplot as plt  # Tìm số thành phần chính để giữ lại 90% phương sai  df\_tfidf = pd.read\_csv('final\_tfidf.csv')  X\_scaled = df\_tfidf.values  pca = PCA()  pca.fit(X\_scaled)  # Lấy cumulative variance  cumulative\_variance = np.cumsum(pca.explained\_variance\_ratio\_)  plt.figure(figsize=(8, 5))  plt.plot(range(1, len(cumulative\_variance) + 1), cumulative\_variance, marker='o')  plt.title('Cumulative Explained Variance by PCA Components') |

Kết quả thu được như sau:

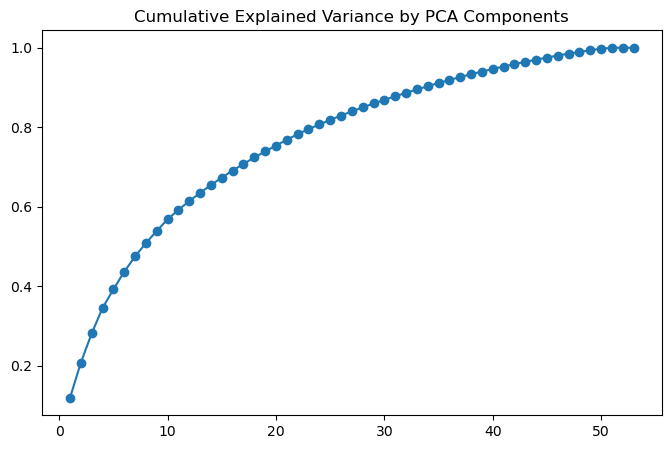


**Hình 5.8.** Kết quả tìm phương sai và số chiều tương ứng

* Đối với Word2Vec:

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.decomposition import PCA  import matplotlib.pyplot as plt  # Tìm số thành phần chính để giữ lại 90% phương sai  df\_w2v = pd.read\_csv('final\_w2v.csv')  X\_scaled = df\_w2v.values  pca = PCA()  pca.fit(X\_scaled)  # Lấy cumulative variance  cumulative\_variance = np.cumsum(pca.explained\_variance\_ratio\_)  plt.figure(figsize=(8, 5))  plt.plot(range(1, len(cumulative\_variance) + 1), cumulative\_variance, marker='o')  plt.title('Cumulative Explained Variance by PCA Components') |

Kết quả thu được như sau:



**Hình 5.9.** Kết quả tìm phương sai và số chiều tương ứng

Lấy ngẫu nhiên 10 nghìn dữ liệu và xuất file:

|  |
| --- |
| import numpy as np  import pandas as pd  np.random.seed(42)  sample\_size = 10000  indices = np.random.choice(X\_scaled.shape[0], size=sample\_size, replace=False)  X\_sample = X\_scaled[indices]  # 1. Chuyển mảng NumPy thành DataFrame của pandas  # Thêm 'header=True' nếu bạn muốn tiêu đề cột (column names)  df\_sample = pd.DataFrame(X\_sample)  # 2. Xuất DataFrame ra file CSV  # 'X\_sample.csv' là tên file  # 'index=False' để không xuất chỉ mục (index) của DataFrame vào file  df\_sample.to\_csv('X\_sample.csv', index=False)  print("Đã xuất X\_sample ra file 'X\_sample.csv'") |

#### 5.2.1. Phân cụm DBSCAN kết hợp dữ liệu xử lý bằng TF-IDF

* **Xác định tham số:** Phân tích cách chọn epsilon và MinPts.

Chọn MinPts:

|  |
| --- |
| import pandas as pd  # Tính min\_simple (MinPts)  n\_samples, n\_dims = X\_reduced.shape  min\_simple = 2 \* n\_dims # theo quy tắc: MinPts ≈ 2 \* số chiều  print(f"\nSố mẫu: {n\_samples}, Số chiều (PCA): {n\_dims}, min\_simple (MinPts đề xuất): {min\_simple}") |

Kết quả in ra như sau:

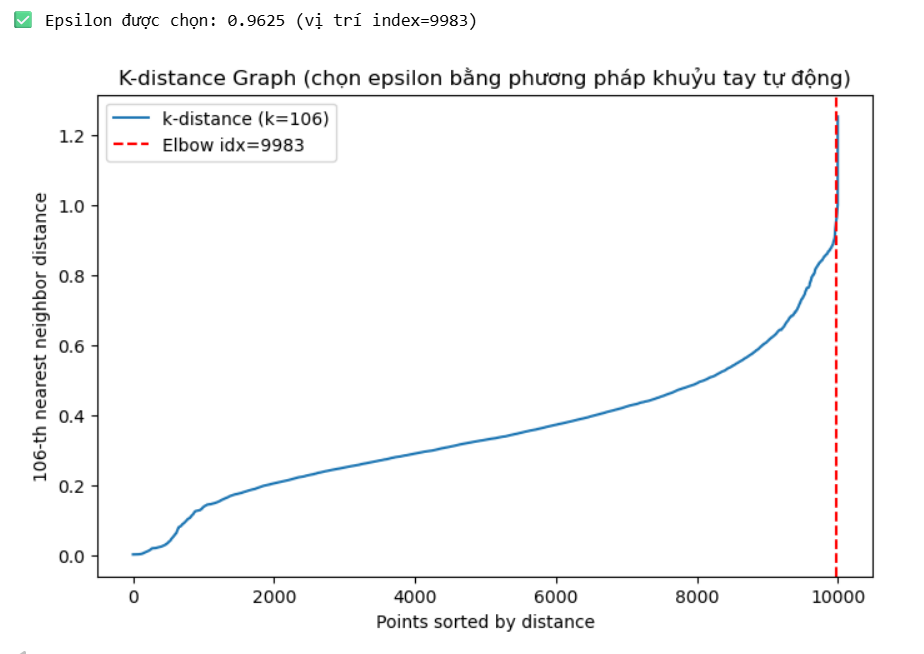


**Hình 5.10.** Kết quả chạy tìm kết quả tìm min\_simple

Chọn epsilon:

|  |
| --- |
| from kneed import KneeLocator  import numpy as np  import matplotlib.pyplot as plt  from sklearn.neighbors import NearestNeighbors # Khai báo lớp KneeLocator từ thư viện kneed, công cụ chuyên dụng để tự động xác định điểm uốn (elbow point) trên đường cong.  K = min\_simple  # Bước 1: Tính khoảng cách k-lân cận  neigh = NearestNeighbors(n\_neighbors=K) # Khởi tạo mô hình Tìm kiếm Láng giềng Gần nhất (Nearest Neighbors) với K=8.  nbrs = neigh.fit(df\_sample) # Huấn luyện mô hình trên dữ liệu đã giảm chiều (X\_reduced - giả sử đây là X\_pca). Mô hình học cách tính khoảng cách giữa các điểm.  distances, indices = nbrs.kneighbors(df\_sample) #Thực hiện truy vấn: Tính toán và trả về khoảng cách (distances) và chỉ số (indices) của K láng giềng gần nhất cho mọi điểm trong tập dữ liệu.  k\_distances = distances[:, K-1] #Trích xuất: Lấy cột cuối cùng trong mảng distances (cột thứ K−1, tức cột thứ 7) để thu được khoảng cách đến láng giềng gần nhất thứ K cho mỗi điểm.  k\_distances = np.sort(k\_distances, axis=0) # Sắp xếp tất cả các khoảng cách K-distance theo thứ tự tăng dần. Đây là bước chuẩn bị để vẽ biểu đồ K-distance.  # Bước 2: Tìm điểm khuỷu tay tự động bằng KneeLocator  x = np.arange(len(k\_distances)) # Tạo mảng chỉ số cho các điểm  knee = KneeLocator(x, k\_distances, curve='convex', direction='increasing') # Khởi tạo đối tượng KneeLocator với dữ liệu đã sắp xếp. Thiết lập curve='convex' và direction='increasing' để phù hợp với hình dạng của đồ thị K-distance.  # Tự động xác định vị trí điểm khuỷu tay (elbow point) trên đồ thị K-distance.  epsilon = k\_distances[knee.knee]  print(f"✅ Epsilon được chọn: {epsilon:.4f} (vị trí index={knee.knee})")  # Bước 3: Vẽ biểu đồ minh họa  plt.figure(figsize=(8,5))  plt.plot(k\_distances, label=f'k-distance (k={K})')  plt.axvline(x=knee.knee, color='r', linestyle='--', label=f'Elbow idx={knee.knee}')  plt.title("K-distance Graph (chọn epsilon bằng phương pháp khuỷu tay tự động)")  plt.xlabel("Points sorted by distance")  plt.ylabel(f"{K}-th nearest neighbor distance")  plt.legend()  plt.show() |

Kết quả in ra như sau:



**Hình 5.11.** Kết quả chạy tìm epsilon tối ưu

Vậy là chúng ta thấy được rằng giá trị epsilon tối ưu sẽ là 0.9625.

* **Kết quả phân cụm:** Phân cụm trên Final User Vector.

|  |
| --- |
| import pandas as pd  from sklearn.cluster import DBSCAN  from sklearn.metrics import silhouette\_score  import numpy as np  from sklearn.decomposition import PCA  # Lấy mẫu ngẫu nhiên 10,000 điểm để phân cụm  eps\_candidates = np.arange(epsilon\*0.2, epsilon\*1.5, 0.05) # quét quanh giá trị epsilon tìm được  for eps in eps\_candidates:  dbscan = DBSCAN(eps=eps, min\_samples=10)  labels = dbscan.fit\_predict(df\_sample)  n\_clusters = len(set(labels)) - (1 if -1 in labels else 0)  n\_noise = list(labels).count(-1)  print(f"eps={eps:.3f}: số cụm={n\_clusters}, nhiễu={n\_noise}")  if n\_clusters > 1:  silhouette\_avg = silhouette\_score(df\_sample, labels)  print(f" => Silhouette Score: {silhouette\_avg:.4f}") |

Kết quả thu được như sau:

|  |
| --- |
| eps=0.192: số cụm=5, nhiễu=5653  => Silhouette Score: -0.0428  eps=0.242: số cụm=5, nhiễu=3939  => Silhouette Score: 0.0179  eps=0.292: số cụm=3, nhiễu=2558  => Silhouette Score: 0.1629  eps=0.342: số cụm=1, nhiễu=1641  eps=0.392: số cụm=1, nhiễu=1032  eps=0.442: số cụm=1, nhiễu=655  eps=0.492: số cụm=1, nhiễu=399  eps=0.542: số cụm=1, nhiễu=220  eps=0.592: số cụm=1, nhiễu=131  eps=0.642: số cụm=1, nhiễu=60  eps=0.692: số cụm=1, nhiễu=27  eps=0.742: số cụm=1, nhiễu=7  eps=0.792: số cụm=1, nhiễu=2  eps=0.842: số cụm=1, nhiễu=1  eps=0.892: số cụm=1, nhiễu=1  eps=0.942: số cụm=1, nhiễu=0  eps=0.992: số cụm=1, nhiễu=0  eps=1.042: số cụm=1, nhiễu=0  eps=1.092: số cụm=1, nhiễu=0  eps=1.142: số cụm=1, nhiễu=0  eps=1.192: số cụm=1, nhiễu=0  eps=1.242: số cụm=1, nhiễu=0  eps=1.292: số cụm=1, nhiễu=0  eps=1.342: số cụm=1, nhiễu=0  eps=1.392: số cụm=1, nhiễu=0  eps=1.442: số cụm=1, nhiễu=0 |

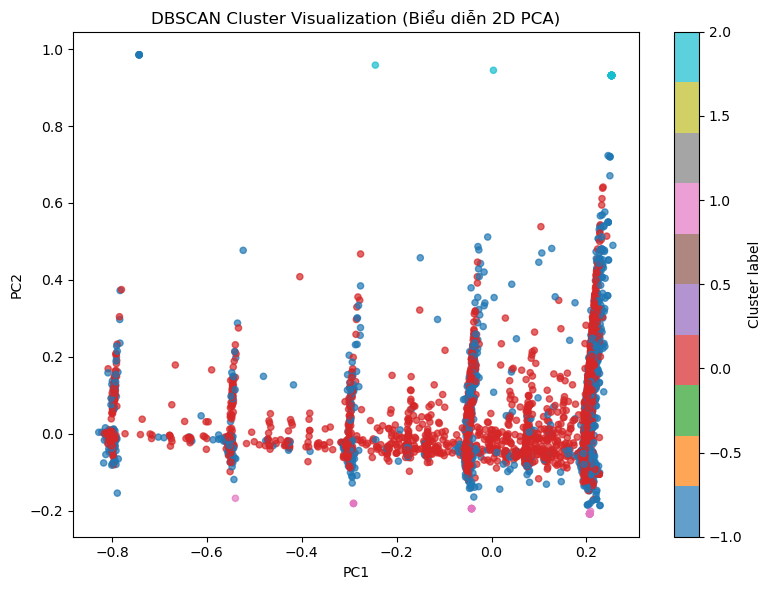
Thấy được rằng mô hình tốt nhất là eps = 0.292.

|  |
| --- |
| import pandas as pd  dbscan = DBSCAN(eps=0.292, min\_samples=10)  labels = dbscan.fit\_predict(X\_sample)  # Xuất kết quả nhãn phân cụm ra file CSV  df\_output = pd.DataFrame(X\_sample, columns=[f'PC{i+1}' for i in range(X\_sample.shape[1])])  df\_output['DBSCAN\_label'] = labels  df\_output.to\_csv('dbscan\_labels\_tfidf.csv', index=False)  print("Đã xuất file dbscan\_labels\_tfidf.csv chứa dữ liệu sample và nhãn phân cụm!") |

**Nhận xét:** Từ kết quả, tìm ra được mô hình DBSCAN tốt nhất với giá trị epsilon là 0.2 và MinPts là 10. Số điểm nhiễu mà DBSCAN tìm được là 63, tức là có khoảng 63 người có hành vi khác biệt.

Minh họa kết quả:

|  |
| --- |
| from sklearn.decomposition import PCA  import matplotlib.pyplot as plt  # Ép kiểu nhãn về số nguyên  labels = df\_vis['DBSCAN\_label'].astype(int)  # PCA giảm chiều  pca\_vis = PCA(n\_components=2)  X\_vis2d = pca\_vis.fit\_transform(df\_vis.drop(columns=['DBSCAN\_label']))  # Vẽ biểu đồ  plt.figure(figsize=(8,6))  scatter = plt.scatter(X\_vis2d[:,0], X\_vis2d[:,1], c=labels, cmap='tab10', s=20, alpha=0.7)  plt.xlabel('PC1')  plt.ylabel('PC2')  plt.title('DBSCAN Cluster Visualization (Biểu diễn 2D PCA)')  plt.colorbar(scatter, label='Cluster label')  plt.tight\_layout()  plt.show() |

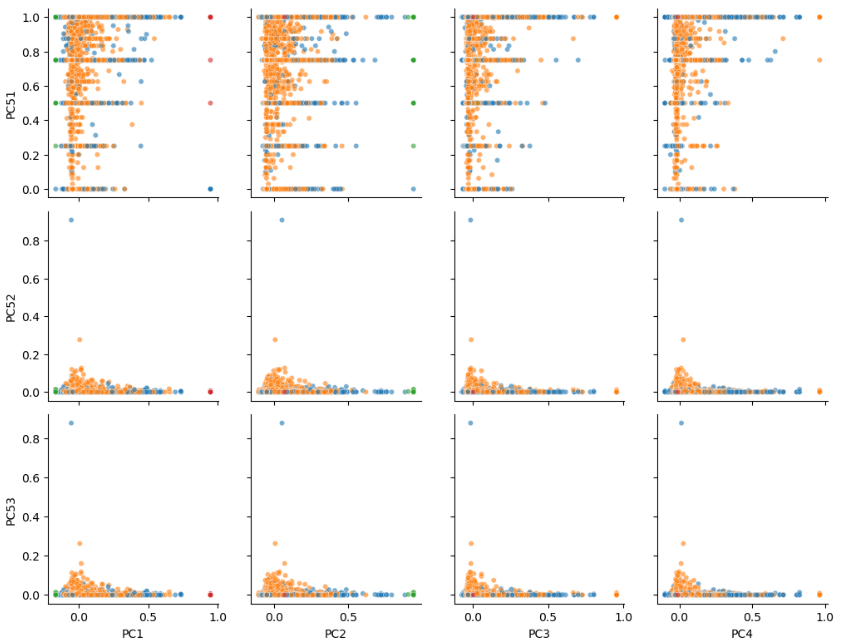


**Hình 5.12.** Biểu diễn với PCA (DBSCAN + TF-IDF)

Có 3 cụm chính (label 0, 1, 2) và một nhóm nhiễu (label -1).

Các cụm được phân tách khá rõ => cho thấy người dùng trong mỗi cụm có cách dùng từ khóa khác nhau.

Nhóm nhiễu nằm rải rác => có thể là người dùng ít nội dung, dùng từ không phổ biến, hoặc không khớp với cụm nào.



**Hình 5.13.** Biểu diễn cụm theo các cặp chiều

#### 5.2.2. Phân cụm DBSCAN kết hợp với dữ liệu xử lý bằng Word2Vec

* **Xác định tham số:** Phân tích cách chọn epsilon và MinPts.

Chọn MinPts:

|  |
| --- |
| import pandas as pd  # Tính min\_simple (MinPts)  n\_samples, n\_dims = X\_reduced.shape  min\_simple = 2 \* n\_dims # theo quy tắc: MinPts ≈ 2 \* số chiều  print(f"\nSố mẫu: {n\_samples}, Số chiều (PCA): {n\_dims}, min\_simple (MinPts đề xuất): {min\_simple}") |

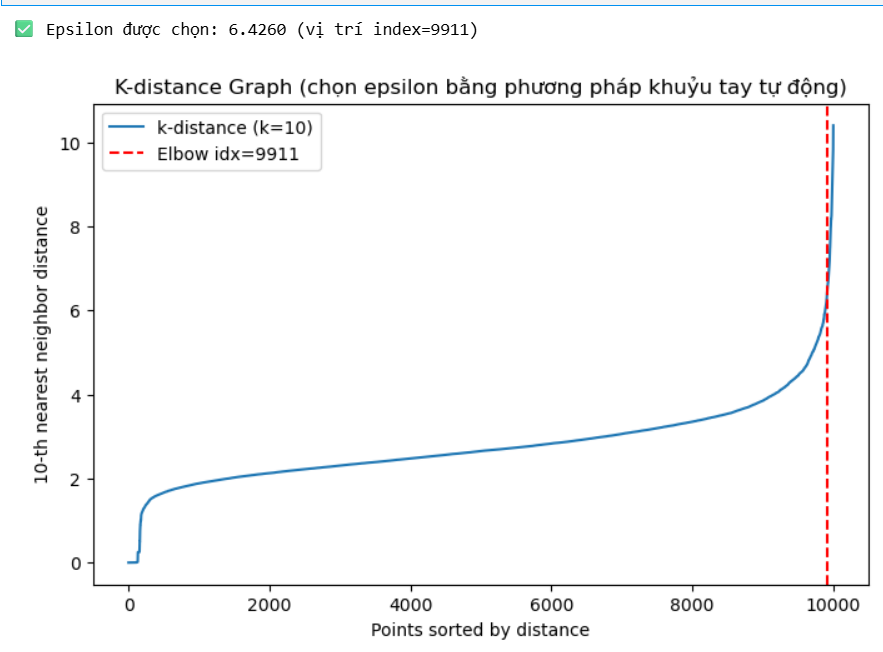
Kết quả in ra như sau:



Chọn epsilon:

|  |
| --- |
| from kneed import KneeLocator  import numpy as np  import matplotlib.pyplot as plt  from sklearn.neighbors import NearestNeighbors # Khai báo lớp KneeLocator  import pandas as pd  df\_sample = pd.read\_csv('X\_sample\_w2v.csv')  K = min\_simple  # Bước 1: Tính khoảng cách k-lân cận  neigh = NearestNeighbors(n\_neighbors=K) # Khởi tạo mô hình KNN  nbrs = neigh.fit(df\_sample) # Huấn luyện mô hình  distances, indices = nbrs.kneighbors(df\_sample) # Tính toán và trả về khoảng cách (distances) và chỉ số (indices) của K láng giềng gần nhất cho mọi điểm trong tập dữ liệu.  k\_distances = distances[:, K-1] #Trích xuất: Lấy cột cuối cùng trong mảng distances (cột thứ K−1, tức cột thứ 9) để thu được khoảng cách đến láng giềng gần nhất thứ K cho mỗi điểm.  k\_distances = np.sort(k\_distances, axis=0) # Sắp xếp tăng dần  # Bước 2: Tìm điểm khuỷu tay tự động bằng KneeLocator  x = np.arange(len(k\_distances)) # Tạo mảng chỉ số cho các điểm  knee = KneeLocator(x, k\_distances, curve='convex', direction='increasing') # Khởi tạo đối tượng KneeLocator với dữ liệu đã sắp xếp  epsilon = k\_distances[knee.knee]  print(f"✅ Epsilon được chọn: {epsilon:.4f} (vị trí index={knee.knee})")  # Bước 3: Vẽ biểu đồ minh họa  plt.figure(figsize=(8,5))  plt.plot(k\_distances, label=f'k-distance (k={K})')  plt.axvline(x=knee.knee, color='r', linestyle='--', label=f'Elbow idx={knee.knee}')  plt.title("K-distance Graph (chọn epsilon bằng phương pháp khuỷu tay tự động)")  plt.xlabel("Points sorted by distance")  plt.ylabel(f"{K}-th nearest neighbor distance")  plt.legend()  plt.show() |

Kết quả in ra như sau:



**Hình 5.14.** Biểu đồ K-distance

* **Kết quả phân cụm:** Phân cụm trên Final User Vector.

|  |
| --- |
| import pandas as pd  from sklearn.cluster import DBSCAN  from sklearn.metrics import silhouette\_score  import numpy as np  from sklearn.decomposition import PCA  eps\_candidates = np.arange(epsilon\*0.2, epsilon\*1.5, 0.05) # quét quanh giá trị epsilon tìm được  # Khởi tạo biến để lưu trữ kết quả tốt nhất  best\_eps = None  best\_silhouette\_score = -1  for eps in eps\_candidates:  dbscan = DBSCAN(eps=eps, min\_samples=10)  labels = dbscan.fit\_predict(df\_sample)  n\_clusters = len(set(labels)) - (1 if -1 in labels else 0)  n\_noise = list(labels).count(-1)  print(f"eps={eps:.3f}: số cụm={n\_clusters}, nhiễu={n\_noise}")  if n\_clusters > 1:  silhouette\_avg = silhouette\_score(df\_sample, labels)  print(f" => Silhouette Score: {silhouette\_avg:.4f}")  # Cập nhật kết quả tốt nhất  if silhouette\_avg > best\_silhouette\_score:  best\_silhouette\_score = silhouette\_avg  best\_eps = eps |

Kết quả thu được như sau:

|  |
| --- |
| eps=1.285: số cụm=7, nhiễu=9710  => Silhouette Score: -0.2851  eps=1.335: số cụm=7, nhiễu=9666  => Silhouette Score: -0.2806  eps=1.385: số cụm=5, nhiễu=9624  => Silhouette Score: -0.2527  eps=1.435: số cụm=5, nhiễu=9559  => Silhouette Score: -0.2468  eps=1.485: số cụm=7, nhiễu=9466  => Silhouette Score: -0.2486  eps=1.535: số cụm=5, nhiễu=9378  => Silhouette Score: -0.2221  eps=1.585: số cụm=4, nhiễu=9256  => Silhouette Score: -0.2109  eps=1.635: số cụm=2, nhiễu=9139  => Silhouette Score: -0.1712  eps=1.685: số cụm=2, nhiễu=8998  => Silhouette Score: -0.1638  eps=1.735: số cụm=2, nhiễu=8816  => Silhouette Score: -0.1528  eps=1.785: số cụm=2, nhiễu=8604  => Silhouette Score: -0.1397  eps=1.835: số cụm=2, nhiễu=8346  => Silhouette Score: -0.1244  eps=1.885: số cụm=2, nhiễu=8083  ...  eps=9.485: số cụm=1, nhiễu=0  eps=9.535: số cụm=1, nhiễu=0  eps=9.585: số cụm=1, nhiễu=0  eps=9.635: số cụm=1, nhiễu=0  *Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...* |

Tìm được mô hình tốt nhất như sau:

|  |
| --- |
| if best\_eps is not None:  print(f"Giá trị epsilon tốt nhất (best eps) là: {best\_eps:.3f}")  print(f"Với Silhouette Score tốt nhất là: {best\_silhouette\_score:.4f}")  else:  print(f"Không tìm thấy giá trị epsilon nào tạo ra hơn 1 cụm.")  print("="\*50) |

Và kết quả như sau:

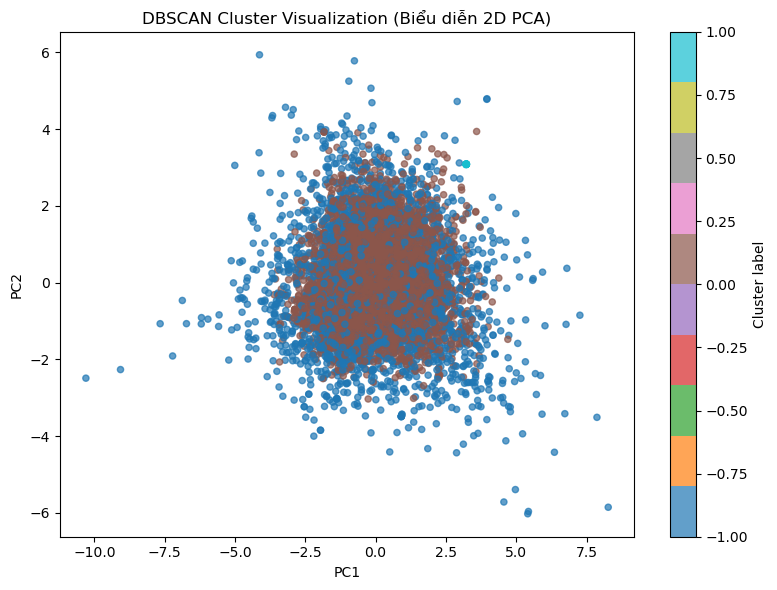
|  |
| --- |
| Giá trị epsilon tốt nhất (best eps) là: 2.685  Với Silhouette Score tốt nhất là: 0.1375 |

**Nhận xét:** Từ kết quả, tìm ra được mô hình DBSCAN tốt nhất với giá trị epsilon là 0.298 và MinPts là 10. Số điểm nhiễu mà DBSCAN tìm được là 880, tức là có khoảng 880 người có hành vi khác biệt, nhiễu ở phần này có nhiều hơn so với việc dữ liệu được xử lý bởi TF-IDF.

Minh họa kết quả:

|  |
| --- |
| from sklearn.decomposition import PCA  import matplotlib.pyplot as plt  df = pd.read\_csv("dbscan\_labels\_w2v.csv")  # Ép kiểu nhãn về số nguyên  labels = df['DBSCAN\_label'].astype(int)  # PCA giảm chiều  pca\_vis = PCA(n\_components=2)  X\_vis2d = pca\_vis.fit\_transform(df.drop(columns=['DBSCAN\_label']))  # Vẽ biểu đồ  plt.figure(figsize=(8,6))  scatter = plt.scatter(X\_vis2d[:,0], X\_vis2d[:,1], c=labels, cmap='tab10', s=20, alpha=0.7)  plt.xlabel('PC1')  plt.ylabel('PC2')  plt.title('DBSCAN Cluster Visualization (Biểu diễn 2D PCA)')  plt.colorbar(scatter, label='Cluster label')  plt.tight\_layout()  plt.show() |

Thu được kết quả như sau:



**Hình 5.15.** Hình minh họa các cụm

Vector Word2Vec biểu diễn ngữ nghĩa của nội dung người dùng => các điểm gần nhau có thể dùng từ khác nhau nhưng cùng chủ đề. PCA giữ được cấu trúc tuyến tính, nhưng có thể bỏ sót mối quan hệ phi tuyến giữa các chiều ngữ nghĩa. Phân cụm DBSCAN tạo ra các nhóm mềm hơn, phản ánh sự tương đồng ngữ nghĩa hơn là từ khóa cụ thể. Ứng dụng phù hợp: phân tích hành vi, nhóm người dùng theo mục đích sử dụng.

Do trong bài tập này, dữ liệu trên tập vẫn là quá lớn đối với DBSCAN, nên việc trích lấy 10 nghìn dòng dữ liệu thôi cũng khiến cho dữ liệu bị sai lệch đi.

### 5.3. Đánh giá và So sánh Hiệu quả Phân cụm

* **Sử dụng chỉ số Silhouette Score để đánh giá và so sánh 4 mô hình:**
  + TF-IDF (Combined) + KMeans
  + Word2Vec (Combined) + KMeans
  + TF-IDF (Combined) + DBSCAN
  + Word2Vec (Combined) + DBSCAN

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Phương pháp** | |
| **TF-IDF** | **Word2Vec** |
| K-Means | Silhouette score KMeans (TF-IDF): 0.1583706912757659 | Silhouette score KMeans (Word2Vec): 0.05026690705831778 |
| DBSCAN | eps=0.292  => Silhouette Score: 0.1629 | eps=2.685  => Silhouette Score: 0.1375 |

**Bảng 5.2.** Bảng so sánh chỉ số Silhouette

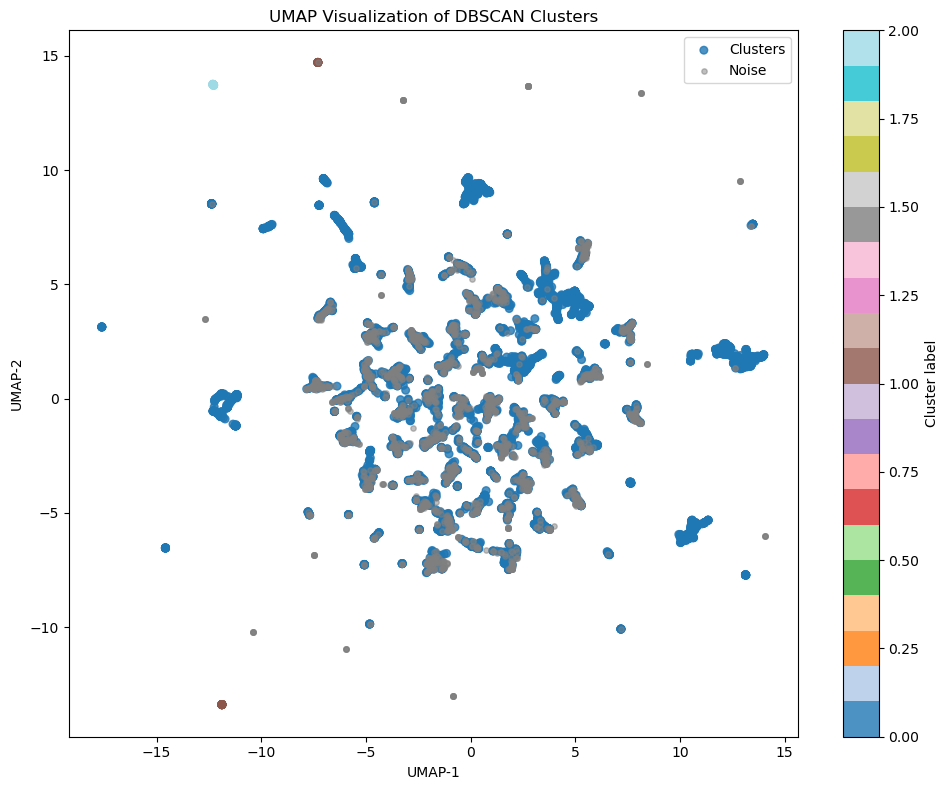
**Nhận xét về phương pháp:** Như kết quả cho thấy thì chỉ số Silhouette của DBSCAN ổn định hơn so với phương pháp truyền thống như K-Means. Vậy nên việc sử dụng DBSCAN sẽ cho thấy kết quả phân cụm tốt hơn.

### 5.4. Diễn giải và mô tả các phân khúc khách hàng

* **Phân tích trọng tâm cụm (Centroids):**
  + Đối với các cụm:

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import umap  from sklearn.preprocessing import StandardScaler  # Bước 1: Đọc dữ liệu  # Giả sử file CSV chứa TF-IDF và nhãn DBSCAN  df = pd.read\_csv("dbscan\_labels\_tfidf.csv")  # Tách dữ liệu và nhãn  X = df.drop(columns=["DBSCAN\_label"]) # TF-IDF features  labels = df["DBSCAN\_label"].values # Nhãn DBSCAN (-1 là noise)  # Bước 2: Chuẩn hóa dữ liệu  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)  # Bước 3: Giảm chiều bằng UMAP  reducer = umap.UMAP(n\_neighbors=5, min\_dist=0.0, n\_components=2, random\_state=42)  X\_umap = reducer.fit\_transform(X\_scaled)  # Bước 4: Trực quan hóa kết quả  plt.figure(figsize=(10, 8))  # Vẽ điểm thuộc cụm  mask = labels != -1  scatter = plt.scatter(X\_umap[mask, 0], X\_umap[mask, 1],  c=labels[mask], cmap='tab20', s=30, alpha=0.8, label='Clusters')  # Vẽ điểm nhiễu  plt.scatter(X\_umap[~mask, 0], X\_umap[~mask, 1],  c='gray', s=15, alpha=0.5, label='Noise')  # Bước 5: Tùy chỉnh biểu đồ  plt.xlabel("UMAP-1")  plt.ylabel("UMAP-2")  plt.title("UMAP Visualization of DBSCAN Clusters")  plt.legend()  plt.colorbar(scatter, label='Cluster label')  plt.tight\_layout()  plt.show() |

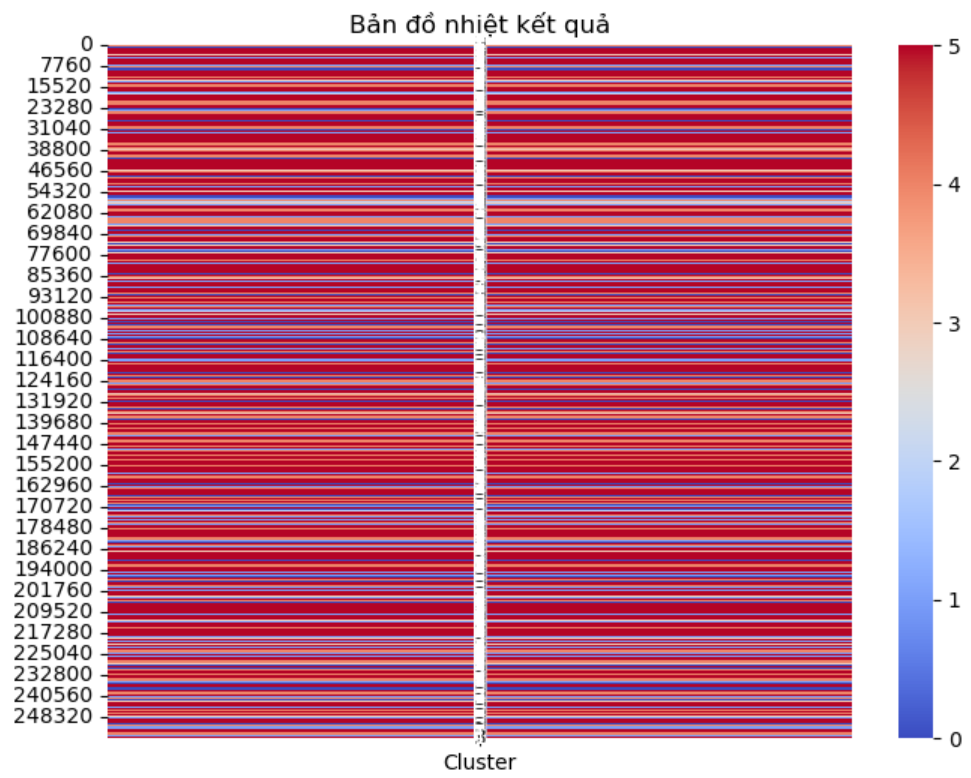
Kết quả thu được như sau:



**Hình 5.16.** Sử dụng UMAP (DBSCAN + TF-IDF)

* **Mô tả phân khúc:**

**K-Means + TF-IDF**

****

**Hình 5.17.** Biểu đồ nhiệt kết quả khi dùng phương pháp TF-IDF

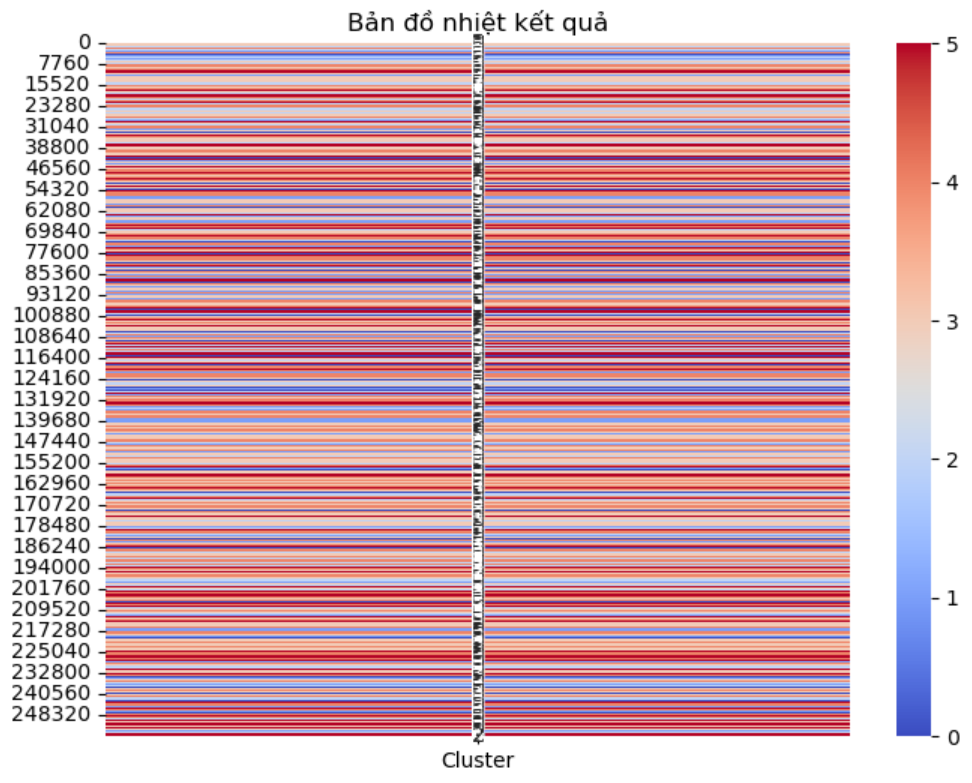
**Nhận xét biểu đồ nhiệt TF-IDF:**

* Ở biểu đồ TF-IDF, các dải màu đỏ và xanh phân bố khá dày và không đồng đều, thể hiện rằng:
  + Mức độ tương đồng giữa các tài liệu hoặc cụm từ khá rời rạc.
  + TF-IDF chủ yếu phản ánh tần suất xuất hiện của từ, nên các cụm có xu hướng bị ảnh hưởng bởi những từ phổ biến hơn là ý nghĩa ngữ nghĩa.
* Có những vùng xen kẽ đỏ – xanh rõ rệt → cho thấy rào cản giữa các cụm chưa mạnh, các cụm có thể chồng lấn nhau.

=> **Tóm lại:**

* TF-IDF thể hiện sự khác biệt về mặt “từ khóa” giữa các cụm, nhưng chưa phản ánh được mối quan hệ ngữ nghĩa sâu hơn giữa các từ hay ý kiến người dùng.
* TF-IDF phản ánh các cụm dựa trên tần suất và từ khóa nổi bật, nên hữu ích nếu bạn muốn biết người dùng đang nhắc tới từ khóa cụ thể nào nhiều nhất (ví dụ: “giá”, “chất lượng”, “dịch vụ”…).

**K-Means + Word2Vec:**



**Hình 5.18.** Biểu đồ nhiệt kết quả khi dùng phương pháp Word2Vec

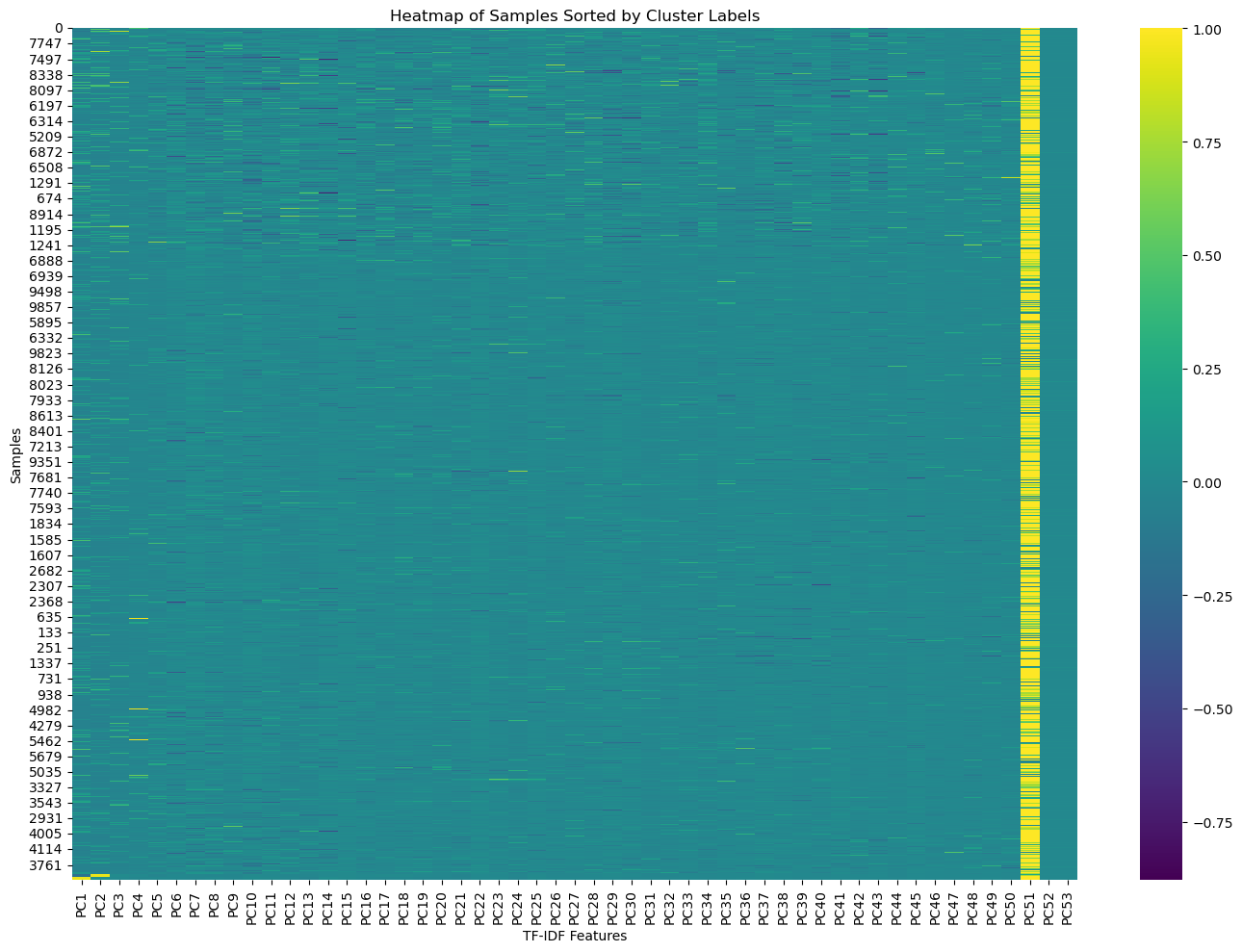
**Nhận xét biểu đồ nhiệt Word2Vec:**

* Biểu đồ Word2Vec nhìn mềm hơn, các vùng màu chuyển tiếp đều đặn hơn, không còn sự thay đổi đột ngột giữa các cụm như TF-IDF. Điều này thể hiện Word2Vec đã biểu diễn ngữ nghĩa của từ, giúp các câu hoặc đánh giá có nghĩa tương tự được đưa vào gần nhau.
* Dải màu xen kẽ xanh – đỏ ít hơn → các cụm thể hiện sự tương quan nội tại tốt hơn, phản ánh cấu trúc ngữ nghĩa ổn định.
* Mức độ “trơn tru” của bản đồ nhiệt cho thấy các vector Word2Vec nằm gần nhau hơn trong không gian liên tục.

=> **Tóm lại:**

* Word2Vec cho kết quả phân cụm ổn định và có tính ngữ nghĩa cao hơn so với TF-IDF, giúp phát hiện được những nhóm đánh giá mang nội dung tương tự, dù cách dùng từ khác nhau.
* Word2Vec nhóm các đánh giá dựa trên ngữ nghĩa, nên thích hợp để phân tích cảm xúc hoặc chủ đề ngầm. Ví dụ những người dùng nói tích cực/tiêu cực về cùng một khía cạnh sản phẩm, dù dùng từ khác nhau.

**DBCAN + TF-IDF:**



**Hình 5.19.** Biểu đồ nhiệt DBSCAN + TF-IDF

**Về phân khúc khách hàng (Clustering):**

* **Trục y (Samples) đã được sắp xếp:** Các hàng (mẫu/khách hàng) đã được sắp xếp theo nhãn cụm DBSCAN. Trong một heatmap được sắp xếp thành công, các cụm sẽ xuất hiện dưới dạng các khối màu đồng nhất dọc theo trục y (từ trên xuống dưới).
* **Sự đồng nhất của cụm (Cluster Homogeneity):**
  + Phần lớn dữ liệu (Màu tím/xanh): Phần lớn các mẫu (từ trên xuống gần cuối) có vẻ như thuộc về một cụm lớn hoặc vùng nhiễu (-1), hoặc các cụm con trong đó có hành vi rất tương đồng.
  + Cụm nổi bật (Màu vàng): Quan sát thấy một dải màu vàng dọc ở phía bên phải cùng (PC52 và PC53). Đây là dấu hiệu của một hoặc nhiều cụm khách hàng nổi bật/tách biệt (được thể hiện ở cuối trục y). Những khách hàng này có giá trị cao bất thường (gần 1.0, màu vàng) trên các thành phần chính PC52 và PC53.

**Về hoạt động và hành vi khách hàng:**

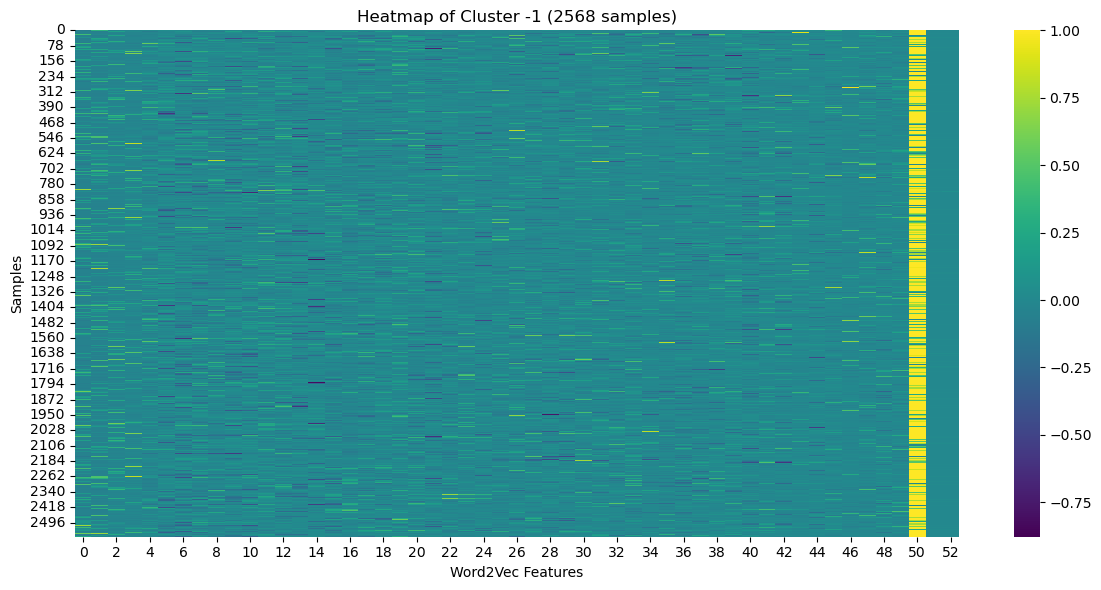
Vì dữ liệu ban đầu là TF-IDF Features (thường dùng trong xử lý ngôn ngữ tự nhiên để đánh giá mức độ quan trọng của từ khóa/thuật ngữ), các thành phần chính (PC) đại diện cho các tổ hợp tuyến tính của các từ khóa/thuật ngữ mà khách hàng tương tác.

* **Hành vi đồng nhất (Phần lớn khách hàng):**
  + Phần lớn khách hàng thể hiện mức độ tương tác thấp (màu tím/xanh đậm, giá trị gần -0.75 đến 0) với phần lớn các thành phần chính (PC1 đến PC51).
  + Điều này có thể chỉ ra rằng phần lớn khách hàng có hành vi tương tác tiêu chuẩn, phổ thông hoặc không tương tác mạnh với các chủ đề đặc biệt.
* **Phân khúc khách hàng đặc biệt (Cụm vàng):**
  + Các khách hàng ở dưới cùng (những người tạo ra dải màu vàng) là phân khúc đặc biệt.
  + Họ có hoạt động rất cao (giá trị PC gần 1.0) trên PC52 và PC53.
  + Kết luận về hoạt động: Hành vi của nhóm khách hàng này được xác định bởi các tổ hợp từ khóa được biểu diễn bởi PC52 và PC53. Để hiểu rõ hơn, bạn cần quay lại ma trận PCA để xem PC52 và PC53 có trọng số cao nhất với các từ khóa/chủ đề nào trong tập dữ liệu TF-IDF gốc. Đây có thể là:
    - Khách hàng quan tâm đến một số sản phẩm/chủ đề ngách hoặc mới nổi.
    - Khách hàng có lượng tương tác cực kỳ lớn với nội dung liên quan đến các từ khóa đó.

**Ý nghĩa đối với phân khúc và tiếp thị:**

* **Phân khúc lớn (Cụm chính/Nhiễu):** Đây là phân khúc khách hàng mục tiêu chung, cần chiến lược tiếp thị đại trà hoặc tìm hiểu thêm tại sao họ không có tương tác nổi bật trên các PC.
* **Phân khúc ngách/ưu tiên (Cụm vàng):** Đây là nhóm rất quan trọng cần được cô lập.
  + **Chiến lược:** Phát triển các chiến dịch tiếp thị cá nhân hóa/chuyên sâu tập trung vào các chủ đề/sản phẩm được đại diện bởi PC52 và PC53.
  + **Giá trị:** Nếu đây là cụm nhỏ, nó có thể đại diện cho nhóm khách hàng đóng góp giá trị cao hoặc nhóm thử nghiệm sản phẩm mới.

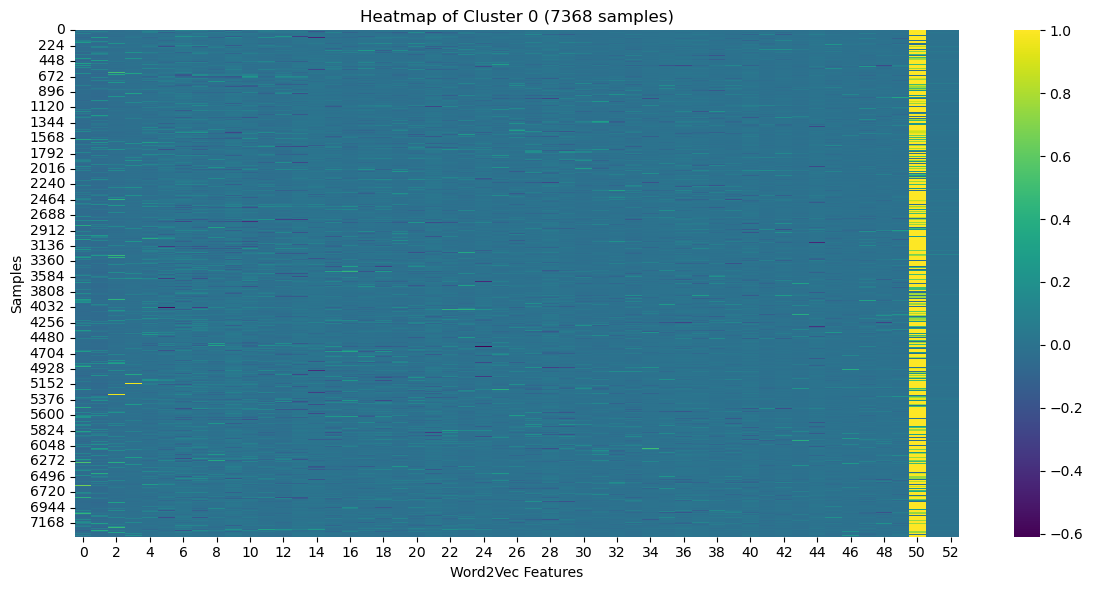
**DBSCAN + Word2Vec**



**Hình 5.20.** Biểu đồ nhiệt cho cụm -1 (DBSCAN + Word2Vec)

**Phân tích:**

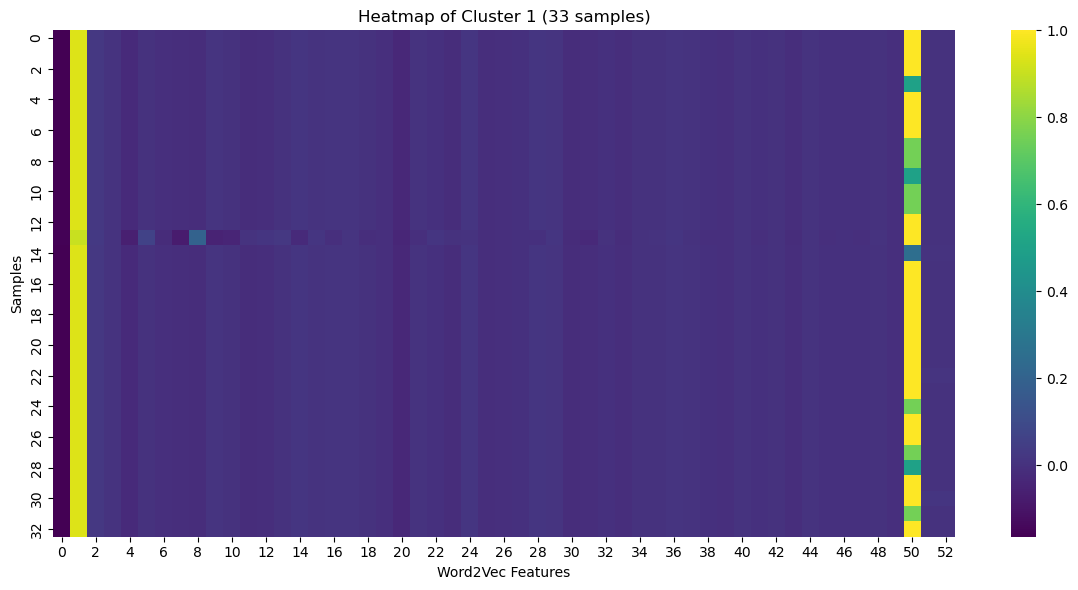
* Biểu hiện ngữ nghĩa phân tán => nhóm người dùng có hành vi không rõ ràng, hoặc khác biệt so với số đông.
* Có thể là người dùng mới, spam, hoặc sử dụng ngôn ngữ không phổ biến.



**Hình 5.21.** Biểu đồ nhiệt cho cụm 0 (DBSCAN + Word2Vec)

**Phân tích**

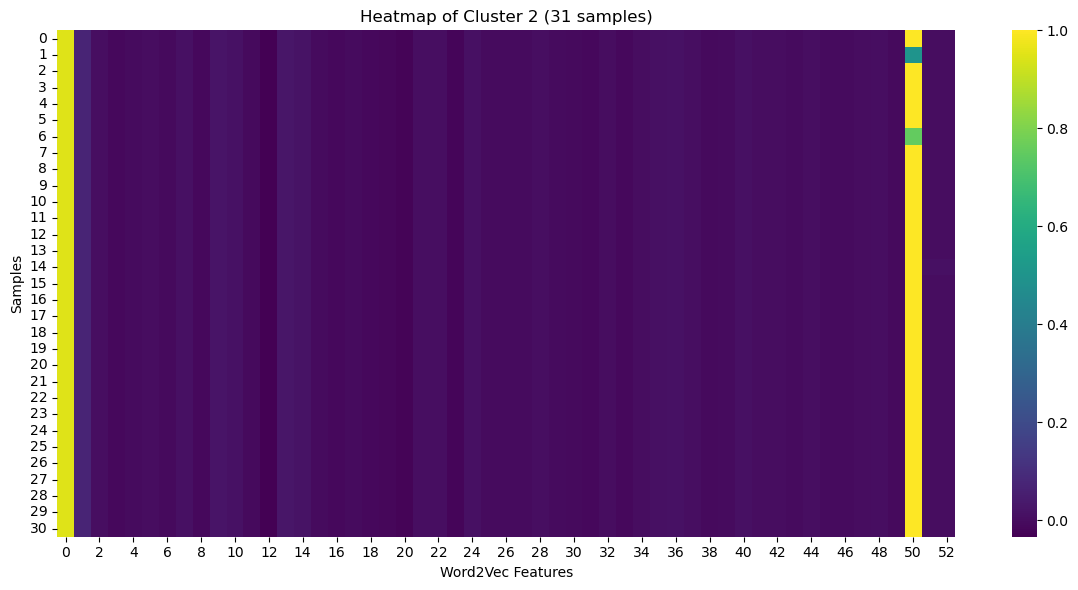
* Có biểu hiện ngữ nghĩa đồng nhất => nhóm người dùng có mục đích sử dụng tương tự, ví dụ cùng tìm kiếm sản phẩm, cùng thảo luận chủ đề.
* Một số chiều vector có giá trị cao => có thể là ngữ nghĩa đặc trưng như “mua hàng”, “đánh giá”, “tư vấn”.



**Hình 5.22.** Biểu đồ nhiệt cho cụm 1 (DBSCAN + Word2Vec)

**Phân tích:**

* Các chiều vector có giá trị thấp > nhóm này có biểu hiện ngữ nghĩa mờ nhạt hoặc phân tán.
* Có thể là người dùng ít tương tác, hoặc nội dung của họ không mang tính ngữ nghĩa rõ ràng (ví dụ: câu ngắn, từ chung chung).
* Một vài chiều sáng hơn => có thể là ngữ nghĩa đặc trưng, ví dụ liên quan đến cảm xúc, câu hỏi, hoặc từ khóa chuyên biệt.



**Hình 5.23.** Biểu đồ nhiệt cho cụm 2 (DBSCAN + Word2Vec)

**Phân tích:**

* Có một số chiều vector (cuối biểu đồ) sáng rõ => nhóm này có ngữ nghĩa tập trung, có thể cùng nói về một chủ đề cụ thể.
* Dù số lượng ít, cụm này có tính đồng nhất cao hơn Cluster 1 => có thể là nhóm người dùng cùng mục đích (ví dụ: hỏi về sản phẩm, khiếu nại, đánh giá).

## **CHƯƠNG 6: KẾT LUẬN VÀ KIẾN NGHỊ**

### 6.1. Kết luận

Thu thập và tiền xử lý dữ liệu:

* Dữ liệu đánh giá từ Amazon đã được thu thập đầy đủ với các trường thông tin quan trọng như UserId, ProductId, Score, Text, v.v.
* Văn bản được làm sạch, chuẩn hóa và mã hóa bằng hai phương pháp: TF-IDF và Word2Vec, đảm bảo sẵn sàng cho phân tích.

Phân cụm người dùng bằng các thuật toán học không giám sát:

* Đã áp dụng DBSCAN để phân cụm người dùng dựa trên biểu diễn văn bản.
* Kết quả phân cụm được trực quan hóa bằng PCA và UMAP, cho thấy sự phân tách rõ ràng giữa các nhóm người dùng.

So sánh hai phương pháp biểu diễn văn bản:

* TF-IDF giúp phân cụm người dùng theo cách sử dụng từ khóa, tạo ra các cụm sắc nét, dễ phân biệt.
* Word2Vec giúp nhóm người dùng theo ngữ nghĩa, phản ánh hành vi và mục đích sử dụng sâu hơn, dù cụm mềm hơn.

Đánh giá chất lượng phân cụm:

* Đã sử dụng Silhouette Score để chọn giá trị epsilon tối ưu cho DBSCAN.
* Các cụm có độ đồng nhất cao, đặc biệt là cụm lớn (Cluster 0) và các cụm nhỏ chuyên biệt (Cluster 1, 2).

### 6.2. Hạn chế của đề tài

* Các vấn đề chưa giải quyết được: chưa thể chạy thực nghiệm trên toàn bộ tập dữ liệu, phải giảm chiều nhiều lần khiến mất độ tin cậy của dữ liệu.
* Chỉ tập trung vào văn bản review là chính, chưa kết hợp triệt để dữ liệu giao dịch.

### 6.3. Kiến nghị và hướng phát triển

**Đề xuất ứng dụng thực tiễn cho doanh nghiệp Amazon:**

* **Cá nhân hóa gợi ý sản phẩm:** Các cụm người dùng được phân nhóm theo đặc điểm và sở thích tương đồng có thể được sử dụng để xây dựng hệ thống gợi ý sản phẩm theo nhóm hành vi, giúp tăng tỷ lệ chuyển đổi và mức độ hài lòng của khách hàng.
* **Tối ưu hóa chiến dịch quảng cáo:** Việc hiểu rõ từng phân khúc khách hàng cho phép Amazon triển khai các chiến dịch quảng cáo nhắm mục tiêu chính xác hơn, giảm chi phí marketing và tăng hiệu quả truyền thông.
* **Phân tích phản hồi và cải thiện sản phẩm:** Nhóm người dùng có hành vi đánh giá tương đồng có thể phản ánh những điểm mạnh/yếu của sản phẩm. Doanh nghiệp có thể khai thác thông tin này để cải tiến sản phẩm hoặc dịch vụ phù hợp với từng nhóm khách hàng.
* **Phát hiện người dùng tiềm năng hoặc bất thường:** Các cụm nhỏ hoặc nhóm nhiễu có thể chứa người dùng mới, người dùng chuyên biệt hoặc hành vi bất thường (spam, thử nghiệm). Việc theo dõi các nhóm này giúp Amazon điều chỉnh chính sách hỗ trợ và kiểm soát chất lượng.

**Hướng nghiên cứu mở rộng:**

Sử dụng các mô hình ngôn ngữ tiên tiến hơn (BERT) để phân tích sắc thái cảm xúc chi tiết hơn:

* **Kết hợp dữ liệu hành vi mua sắm:** Tích hợp thêm dữ liệu như lịch sử mua hàng, thời gian truy cập, lượt xem sản phẩm để tăng độ chính xác của phân cụm.
* **Áp dụng mô hình phân cụm động theo thời gian**: Hành vi người dùng có thể thay đổi theo mùa, xu hướng hoặc sự kiện. Việc cập nhật mô hình phân cụm định kỳ sẽ giúp doanh nghiệp phản ứng linh hoạt hơn.
* **Triển khai mô hình trong hệ thống thực tế:** Đề xuất tích hợp mô hình phân cụm vào hệ thống gợi ý sản phẩm, dashboard quản trị khách hàng hoặc công cụ phân tích nội bộ của Amazon.
* **Mở rộng sang các nền tảng thương mại điện tử khác:** Mô hình có thể được điều chỉnh để áp dụng cho các sàn như eBay, Shopee, Lazada… nhằm kiểm chứng tính tổng quát và khả năng mở rộng.

## **TÀI LIỆU THAM KHẢO**

[1] “Amazon Fine Food Reviews.” *Kaggle*. [Trực tuyến]. Có sẵn tại: <https://www.kaggle.com/snap/amazon-fine-food-reviews>

[2] "DBSCAN Clustering in ML - Density based clustering." *GeeksforGeeks*. [Trực tuyến]. Có sẵn tại: [https://www.geeksforgeeks.org/dbscan-clustering-ml-density-based-clustering/](https://www.google.com/search?q=https://www.geeksforgeeks.org/dbscan-clustering-ml-density-based-clustering/)

[3] “DBSCAN — scikit-learn 1.7.2 documentation.” *scikit-learn*. [Trực tuyến]. Có sẵn tại: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>

[4] "How To Build Customer Segmentation Models In Python." *Medium*. [Trực tuyến]. Có sẵn tại: [https://towardsdatascience.com/how-to-build-customer-segmentation-models-in-python-8e27c7370860](https://www.google.com/search?q=https://towardsdatascience.com/how-to-build-customer-segmentation-models-in-python-8e27c7370860)

[5] “15.1. Phương pháp phân cụm dựa trên mật độ (Density-Based Clustering).” *Deep AI KhanhBlog*. [Trực tuyến]. Có sẵn tại: [https://deepa.io/density-based-clustering-dbscan/](https://www.google.com/search?q=https://deepa.io/density-based-clustering-dbscan/)

[6] "Project Tutorial: Customer Segmentation Using K-Means Clustering." *Dataquest*. [Trực tuyến]. Có sẵn tại: [https://www.dataquest.io/blog/customer-segmentation-using-k-means/](https://www.google.com/search?q=https://www.dataquest.io/blog/customer-segmentation-using-k-means/)

[7] "Ultimate Guide to Heatmap Grids for Segmentation." *growth-onomics*. [Trực tuyến]. Có sẵn tại: [https://growth-onomics.com/heatmap-grids-for-segmentation/](https://www.google.com/search?q=https://growth-onomics.com/heatmap-grids-for-segmentation/)

[8] N. H. Dien, “Phân cụm dữ liệu K-Means, DBSCAN và AP.” *Kaggle*. [Trực tuyến]. Có sẵn tại: <https://www.kaggle.com/code/nguyenhuudien/ph-n-c-m-d-li-u-k-means-dbscan-v-ap/input>