TRƯỜNG ĐẠI HỌC

**SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**





**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

NGÀNH CÔNG NGHỆ THÔNG TIN

**BÁO CÁO ĐỒ ÁN 3**

**TÌM HIỂU THUẬT TOÁN RECOMMENDATION CHO NGƯỜI THUÊ TRỌ**

**SVTH: CHU MINH HOÀNG 17110139**

**PHẠM HUỲNH THANH LÂM 17110168**

**Khóa: 2017 – 2021**

 **Ngành: CÔNG NGHỆ THÔNG TIN**

 **` GVHD: TS. Huỳnh Xuân Phụng**

TP. Hồ Chí Minh, tháng 12 năm 2020

# **LỜI CẢM ƠN**

Lời đầu tiên, nhóm thực hiện xin được phép gửi lời cảm ơn chân thành đến khoa Đào tạo Chất Lượng Cao – Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh đã tạo mọi điều kiện thuận lợi nhất cho nhóm thực hiện có cơ hội được tự do tiếp cận, tham khảo, mở rộng thêm kiến thức trong lĩnh vực Công nghệ thông tin nói chung và đồ án 3.

Lời cảm ơn trân trọng nhất nhóm thực hiện xin chân thành gửi đến Thầy **Huỳnh Xuân Phụng** – người đã dùng mọi tâm huyết và tri thức của người Thầy, cùng đồng hành và trực tiếp hướng dẫn và tạo mọi điều kiện thuận lợi giúp đỡ cho nhóm phát huy hết khả năng cũng như nâng cao kiến thức trong suốt quá trình học tập và tìm hiểu, đặc biệt là trong quá trình chuẩn bị và thực hiện đồ án 3. Cảm ơn sự nhiệt tình của Thầy, là động lực vô cùng to lớn giúp nhóm thực hiện thực hiện, kiên trì trong suốt quá trình thực hiện đề tài và khám phá ra những kiến thức mới đầy thú vị và bổ ích liên quan đến đề tài mà cụ thể là đề tài **Tìm hiểu thuật toán Recommendation cho người thuê trọ**

Cuối cùng nhóm thực hiện kính chúc quý thầy cô dồi dào sức khỏe, niềm tin để tiếp tục sự nghiệp cao quý của mình là truyền đạt kiến thức cho thế hệ mai sau.

Nhóm thực hiện xin chân thành cám ơn.

TPHCM, ngày 11 tháng 12 năm 2020

Sinh viên thực hiện

# **LỜI MỞ ĐẦU**

Những năm gần đây, AI - Artificial Intelligence (Trí Tuệ Nhân Tạo), và cụ thể hơn là Machine Learning (Học Máy hoặc Máy Học) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư (1 - động cơ hơi nước, 2 - năng lượng điện, 3 - công nghệ thông tin). Trí Tuệ Nhân Tạo đang len lỏi vào mọi lĩnh vực trong đời sống mà có thể chúng ta không nhận ra. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, …, chỉ là một vài trong vô vàn những ứng dụng của AI/Machine Learning. Mọi công việc phức tạp hiện nay hầu hết đều có thể giải quyết được và ở những trang về tin tức hoặc phim đều có chức năng gợi ý để tìm kiếm nhanh hơn. Thú vị về gợi ý của các trang nên nhóm quyết định tìm hiểu về gợi ý về nhà trọ bằng thuật toán sử dụng hệ thống gợi ý recommendation.

# **MỤC LỤC**

[**LỜI CẢM ƠN** ii](#_Toc60744917)

[**LỜI MỞ ĐẦU** iii](#_Toc60744918)

[**MỤC LỤC** iv](#_Toc60744919)

[**DANH MỤC CÁC BẢNG BIỂU** v](#_Toc60744920)

[**DANH MỤC CÁC BIỂU ĐỒ VÀ HÌNH ẢNH** vi](#_Toc60744921)

[**CHƯƠNG 1 GIỚI THIỆU** 1](#_Toc60744922)

[**1.1.** **Tìm hiểu về ngôn ngữ python** 1](#_Toc60744923)

[**1.2.** **8 bước xây dựng một machine learning** 1](#_Toc60744924)

[**1.3.** **Recommendation systerm là gì?** 3](#_Toc60744925)

[**1.4.** **Các thành phần cơ bản** 3](#_Toc60744926)

[**1.5.** **Phân loại** 4](#_Toc60744927)

[1.5.1 Collaborative filtering 5](#_Toc60744928)

[1.5.2 Content-based filtering 11](#_Toc60744929)

[1.5.3 Các phương pháp lai 14](#_Toc60744930)

[**1.6.** **Vậy chúng ta nên sử dụng phương pháp nào?** 15](#_Toc60744931)

[**CHƯƠNG 2 CHƯƠNG TRÌNH DEMO** 16](#_Toc60744932)

[**2.1** **Các thư viện sử dụng** 16](#_Toc60744933)

[**2.2** **Thực hiện bài toán** 17](#_Toc60744934)

[**CHƯƠNG 3 KẾT LUẬN** 28](#_Toc60744935)

[**3.1** **Kết quả đạt được** 28](#_Toc60744936)

[**3.2** **Khuyết điểm** 28](#_Toc60744937)

[**TÀI LIỆU THAM KHẢO** 29](#_Toc60744938)

# **DANH MỤC CÁC BIỂU ĐỒ VÀ HÌNH ẢNH**

[**Hình 1.1:** Hàm mất mát 10](#_Toc60744972)

[**Hình 1.2:** Công thức tối ưa hàm mất mát của w 11](#_Toc60744973)

[**Hình 1.3:** Công thức tối ưa hàm mất mát của x 11](#_Toc60744974)

[**Hình 1.4:** Ma trận user-item 13](#_Toc60744975)

[**Hình 1.5:** Ví dụ item profile 14](#_Toc60744976)

[**Hình 2.1:** Đọc dữ liệu file data 19](#_Toc60744977)

[**Hình 2.2:** Dữ liệu kinh độ 20](#_Toc60744978)

[**Hình 2.3:** Mô tả dữ liệu 20](#_Toc60744979)

[**Hình 2.4:** Gộp title và mô tả 22](#_Toc60744980)

[**Hình 2.5:** Danh sách 10 keyword đầu 22](#_Toc60744981)

[**Hình 2.6:** Danh sách keyword qua stopword 24](#_Toc60744982)

[**Hình 2.7:** Kết quả thu được 28](#_Toc60744983)

# **CHƯƠNG 1 GIỚI THIỆU**

## **Tìm hiểu về ngôn ngữ python**

Python là một ngôn ngữ lập trình thông dịch, tương tác, hướng đối tượng và cấp cao có mục đích chung. Nó được tạo ra bởi Guido van Rossum trong thời gian 1985-1990. Giống như Perl, mã nguồn Python cũng có sẵn theo Giấy phép Công cộng GNU (GPL).

* Python được thông dịch - Python được trình thông dịch xử lý trong thời gian chạy. Bạn không cần phải biên dịch chương trình của mình trước khi thực thi nó. Điều này tương tự với PERL và PHP.
* Python là tương tác - Bạn thực sự có thể ngồi tại lời nhắc Python và tương tác trực tiếp với trình thông dịch để viết chương trình của mình.
* Python hướng đối tượng - Python hỗ trợ phong cách hướng đối tượng hoặc kỹ thuật lập trình đóng gói mã bên trong các đối tượng.
* Python là ngôn ngữ dành cho người mới bắt đầu - Python là một ngôn ngữ tuyệt vời cho các lập trình viên mới bắt đầu và hỗ trợ phát triển một loạt các ứng dụng từ xử lý văn bản đơn giản đến trình duyệt WWW đến trò chơi.

## **8 bước xây dựng một machine learning**

* Look at the big picture.

Ta đi trả lời cho các câu hỏi:

* Các giải pháp hiện có (nếu có) hoạt động như thế nào?
* Sử dụng những thuật toán nào?
* Biện pháp hiệu suất nào có liên quan?
* Dữ liệu nào chúng ta cần thu thập?

Xác định các yếu tố trước khi bắt đầu một thuật toán. Bước thu thập dữ liệu là nền tảng của quá trình học máy. Những sai lầm như chọn các tính năng không chính xác hoặc tập trung vào các loại mục nhập hạn chế cho tập dữ liệu có thể làm cho mô hình hoàn toàn không hiệu quả

* Get the data.

Lấy dữ liệu được xác định trong bước 1, lấy dữ liệu ở bước 1 có thể giúp giảm các điểm mù của mô hình, mang lại độ chính xác cao hơn cho các dự đoán

Một số kho dữ liệu: kho lưu trữ Máy học UCI, bộ dữ liệu Kaggle, bộ dữ liệu công khai, AWS của Amazon, ...

* Discover and visualize the data to gain insights.

Khám phá dữ liệu, phân tích để hình dung thuật toán được rõ hơn. Có nhiều mô hình hiện có khác nhau được phát triển bởi các nhà khoa học dữ liệu có thể được sử dụng cho các mục đích khác nhau nên việc khám phá và phân tích thuật toán nào cho hợp lý là điều cần thiết

* Prepare the data for machine learning algorithms.

Chuẩn bị dữ liệu cho thuật toán train. Sử lý các dữ liệu khiến cho máy có thể đọc và hiểu được

* Train and evaluate models.

Huấn luyện và đánh giá các mô hình. Ở bước 5 kiểm tra mô hình có hoạt động tốt với thực tế hay không, máy có thể ngoại suy được dữ liệu truyền vô hay không

* Fine-tune your models.

Tinh chỉnh mô hình điều chỉnh siêu tham số. Bước này cố gắng cải thiện kết quả tích cực đạt được trong bước đánh giá

* Analyze and test your solution.

Phân tích và thử nghiệm các giải pháp

* Launch, monitor, and maintain your system.

Khởi chạy, giám sát và bảo trì hệ thống

## **Recommendation systerm là gì?**

Hệ thống đề xuất hoặc hệ thống khuyến nghị (đôi khi thay thế 'hệ thống' bằng một từ đồng nghĩa như nền tảng hoặc động cơ), là một lớp con của hệ thống lọc thông tin nhằm tìm cách dự đoán "xếp hạng" hoặc "sở thích" mà người dùng sẽ đưa ra cho một mặt hàng. Chúng chủ yếu được sử dụng trong các ứng dụng thương mại.

Hệ thống đề xuất được sử dụng trong nhiều lĩnh vực và được công nhận phổ biến nhất là bộ tạo danh sách phát cho các dịch vụ video và âm nhạc như Netflix, YouTube và Spotify, bộ giới thiệu sản phẩm cho các dịch vụ như Amazon hoặc bộ giới thiệu nội dung cho các nền tảng truyền thông xã hội như Facebook và Twitter. Các hệ thống này có thể hoạt động bằng cách sử dụng một đầu vào duy nhất, như âm nhạc hoặc nhiều đầu vào trong và trên các nền tảng như tin tức, sách và truy vấn tìm kiếm. Ngoài ra còn có các hệ thống giới thiệu phổ biến cho các chủ đề cụ thể như nhà hàng và hẹn hò trực tuyến. Hệ thống khuyến nghị cũng đã được phát triển để khám phá các bài báo và chuyên gia nghiên cứu, cộng tác viên, và các dịch vụ tài chính.

* Recommendation là khuyến nghị với đúng sản phẩm đến với người dủng mà họ đang cần. Có hai thực thể chính trong Recommendation Systems là users và items. Users là người dùng. Items là sản phẩm

## **Các thành phần cơ bản**

* Thứ nhất: Điều đầu tiên cần phải quan tâm đó chính là người dùng (user) (hiển nhiên nếu không có user thì chúng ta biết gợi ý cho ai)
* Thứ hai: Chúng ta cần phải quan tâm đến mục tin (items) các mục tin này có thể là sản phẩm trên các trang bán hàng, bài hát trên các trang nghe nhạc, một user khác như trên mạng xã hội hay một bài viết như trên Viblo cuả chúng ta chẳng hạn. Tại sao cần phải quan tâm đến mục tin bởi vì nếu không có mục tin thì chúng ta lấy cái gì mà gợi ý cho người dùng.
* Thứ ba: Chúng ta cần phải quan tâm đến phản hồi (feedback) của mỗi user lên mục tin đó. Nó có thể là điểm đánh giá, có thể là một chỉ số thể hiện sự quan tâm của user lên item đó.... Đơn giản là vì chúng ta phải định lượng được các đại lượng này thì mới có thể có cơ sở gợi ý cho người dùng phải không nào

## **Phân loại**

* Collaborative filtering: xây dựng mô hình từ hành vi trong quá khứ của người dùng cũng như các quyết định tương tự do những người dùng khác đưa ra (các mặt hàng đã mua hoặc đã chọn trước đó và / hoặc xếp hạng bằng số cho các mặt hàng đó). Mô hình được sử dụng để dự đoán các mặt hàng (hoặc xếp hạng cho các mặt hàng) mà người dùng có thể quan tâm. Các phương pháp lọc dựa trên nội dung sử dụng một loạt các đặc điểm rời rạc, được gắn thẻ trước của một mục để đề xuất các mục bổ sung có đặc tính tương tự. *Dựa vào những hành vi của những người dùng đối với sản phẩm từ đó suy ra được sở thích của một người dùng đối với sản phẩm đó*
* Content-based filtering: là một dạng của hệ thống lọc thông tin (information filtering), nó được sử dụng để dự đoán sở thích (preferences) hay xếp hạng (rating) mà người dùng có thể dành cho một mục thông tin (item) nào đó mà họ chưa xem xét tới trong quá khứ (item có thể là bài hát, bộ phim, đoạn video clip, sách, bài báo, ...). *Hay nói cách khác nó sẽ khuyến nghị sản phẩm cho người dùng dựa vào những nội dung liên quan đến sản phẩm.*

Ngoài ra trong deep learning có những phương pháp được ứng dụng trong recommandation

* Thuật toán LSTM dự đoán sản phẩm có khả năng mua tiếp theo của khách hàng dựa vào lịch sử mua sắm.
* Sử dụng các thuật toán NLP (Natural language processing - Xử lý ngôn ngữ tự nhiên) để phân tích các thông tin như phần tên sản phẩm, mô tả sản phẩm, comment khách hàng về sản phẩm để tìm ra sản phẩm tương đồng.
* Sử dụng AutoEncoder để tìm kiếm sản phẩm tương đồng, khách hàng tương đồng. Về auto encoder có thể xem thêm tại
* Các hệ thống search engine dựa trên hình ảnh của sản phẩm.
* Sử dụng reignforcement learning để recommend sản phẩm dựa trên các dự báo về hành vi tiếp theo của khách hàng.
* Sử dụng LDA để clustering các nhóm sản phẩm có chung đặc điểm và có thể thay thế được cho nhau.
* Thuật toán association để tìm các sản phẩm hoặc nhóm khách hàng có mối liên hệ trong hành vi mua sắm thông qua một chỉ số là levarage.

### **Collaborative filtering**

Tìm ra sản phẩm mà khách hàng có khả năng ưa thích nhất dựa vào những sản phẩm mà những khách hàng khác có hành vi tương tự đã lựa chọn. Thuật toán sẽ không cần sử dụng thông tin sản phẩm là đầu vào cho dự báo rating. Đầu vào của thuật toán là một **ma trận tiện ích** (*ultility matrix*) chứa giá trị rating của các cặp (user, item)

* Ưu điểm không dựa vào nội dung và do đó có khả năng đề xuất chính xác các mục phức tạp như phim mà không yêu cầu "hiểu biết" về mục đó.
* Vấn đề thường hặp phải: cold start, scalability và sparsity, dữ liệu thưa.
* Cold start: Đối với người dùng hoặc mặt hàng mới, không có đủ dữ liệu để đưa ra đề xuất chính xác
* Vấn đề người dùng mới (new user problem) chưa đánh giá sản phẩm nào, chưa có các dữ liệu về các hành vi, vấn đề sản phẩm mới (new item problem), chưa được người dùng nào đánh giá, chưa được ai xem, mua, tìm kiếm, ...
* Giải pháp:
  + Tư vấn các sản phẩm phổ biến, ngẫu nhiên cho người dùng mới; các sản phẩm mới được xuất hiện ở đầu trang
  + Content-boosted CF: tích hợp thêm hồ sơ (profile) người dùng mới hoặc sử dụng thêm các đặc tính của sản phẩm.
* Scalability: Trong nhiều môi trường mà các hệ thống này đưa ra đề xuất, có hàng triệu người dùng và sản phẩm. Do đó, một lượng lớn sức mạnh tính toán thường cần thiết để tính toán các khuyến nghị. Khi ma trận đánh giá lớn, tức số người dùng lẫn sản phẩm lớn thì thời gian tính toán sẽ tăng cao, khó đáp ứng tư vấn thời gian thực hoặc gần thời gian thực.
* Giải pháp:
  + Áp dụng các kỹ thuật giảm số chiều như SVD, PCA.
  + Item-based CF có khả năng mở rộng cao hơn so với user-based CF.
* Sparsity: Số lượng mặt hàng được rao bán trên các trang thương mại điện tử lớn là vô cùng lớn. Những người dùng tích cực nhất sẽ chỉ đánh giá một tập hợp con nhỏ của cơ sở dữ liệu tổng thể. Vì vậy, ngay cả những mặt hàng phổ biến nhất cũng có rất ít xếp hạng.
* Dữ liệu thừa (data sparsity): Ma trận đánh giá có thể rất thừa. Dữ liệu thưa ảnh hưởng rất nhiều đến hiệu quả hệ tư vấn bởi rất khó tính toán sự tương tự giữa các người dùng (users) hoặc giữa các sản phẩm (items)
  + Hai sản phẩm có thể rất giống nhau nhưng có ít người cùng đánh giá đồng thời hai sản phẩm.
  + Hai người dùng có thể giống nhau về sở thích nhưng chưa đánh giá cùng sản phẩm.
* Giải pháp: Áp dụng các kỹ thuật giảm số chiều (dimensionality reduction).
* Có 2 dạng phương pháp sử dụng:
* Neighborhood-based collaborative Filtering: Tìm ra hệ số tương quan giữa các user, hệ số càng lớn sở thích càng giống nhau và ngược lại. Gồm hai lại chính memory-based CF và model-based CF
  + Memory-based CF
  + Sử dụng ma trận đánh giá để thực hiện dự đoán và tư vấn.
  + Giả sử mỗi người dùng thuộc ít nhất một nhóm những người có chung sở thích, mối quan tâm.
  + Người cần được tư vấn được gọi là active user.
  + Những người dùng có sở thích tương tự với active user được gọi là neighbors.
  + Cách tiếp cận:
    - * User-based: dựa trên người dùng để dự đoán.
      * Items-based: dựa trên sản phẩm để dự đoán.
  + Model-based CF
* Thực hiện tư vấn dựa trên các mô hình học máy.
* Các mô hình được xây dựng dựa trên dữ liệu huấn luyện.
* Các phương pháp để xây dựng mô hình lọc cộng tác thường dùng:
  + - * Bayesian models
      * Clustering model
* Matrix Fractorization: được chia làm hai loại là ma trận tường mình và ma trận đánh giá suy diễn.
  + Ma trận tường minh (explicit rating matrix)   
    Người dùng đánh giá trực tiếp đối với các sản phẩm, dịch vụ, nội dung. Thang điểm thường là:
    - Nhị phân: Like, Dislike.
    - Liên tục trong đoạn [0,1]
    - Năm mức rời rạc: 1, 2, 3, 4, 5 (với 5 là mức đánh giá tốt nhất)
* Ma trận đánh giá suy diễn (implicit rating matrix)  
  Ma trận được suy diễn từ thông tin thu thập được về hành vi người dùng như:
* Tìm kiếm (browsing)
* Đọc (reading)
* Xem (watching)
* Chia sẻ (sharing)
* Mua (buying)

Sau đó ánh xạ hành vi người dùng vào các mức điểm.

* Hạn chế: Thường phải lưu một ma trận hệ số tương quan với kích thước rất lớn. Việc này dẫn tới tốn tài nguyên lưu trữ và thời gian tính toán.

#### **Neighborhood-based collaborative Filtering**

Từ những của Content-based Recommendation Systems:

* Thứ nhất, khi xây dựng mô hình cho một user, các hệ thống Content-based không tận dụng được thông tin từ các users khác. Những thông tin này thường rất hữu ích vì hành vi mua hàng của các users thường được nhóm thành một vài nhóm đơn giản; nếu biết hành vi mua hàng của một vài users trong nhóm, hệ thống nên suy luận ra hành vi của những users còn lại.
* Thứ hai, không phải lúc nào chúng ta cũng có bản mô tả cho mỗi item. Việc yêu cầu users gắn tags còn khó khăn hơn vì không phải ai cũng sẵn sàng làm việc đó; hoặc có làm nhưng sẽ mang xu hướng cá nhân.

Ý tưởng cơ bản của Neighborhood-based collaborative Filtering là: xác định *mức độ quan tâm* của một *user* tới một *item* dựa trên các *users* khác *gần giống* với *user* này. Việc *gần giống nhau* giữa các *users* có thể được xác định thông qua *mức độ quan tâm* của các *users* này tới các *items* khác mà hệ thống đã biết.

Ví dụ, *A, B* đều thích phim *Cảnh sát hình sự*, tức đều *rate* bộ phim này 5 sao. Ta đã biết *A* cũng thích *Người phán xử*, vậy nhiều khả năng *B* cũng thích bộ phim này.

Vậy thì ta phải trả lời được 2 câu hỏi:

* Làm thế nào xác định được *sự giống nhau* giữa hai *users*?
* Khi đã xác định được các *users* *gần giống nhau* (*similar users*) rồi, làm thế nào dự đoán được *mức độ quan tâm* của một *user* lên một *item*?

Giải pháp:

* User-user Collaborative Filtering: xác định mức độ quan tâm của mỗi user tới một item dựa trên mức độ quan tâm của similar users tới item đó
* Similarity functions: là phải xác định được sự giống nhau (similarity) giữa hai users.
* Rating prediction: là trung bình có trọng số của các ratings đã chuẩn hoá.
* Item-item Collaborative Filtering cách khắc phục của user-user

Một số hạn chế của User-user CF:

* Số lượng user lớn
* Một khi *user* đó thay đổi *rating* hoặc rate thêm *items.* Kéo theo đó, việc tính toán ma trận Similarity, vốn tốn nhiều bộ nhớ và thời gian, cũng cần được thực hiện lại.

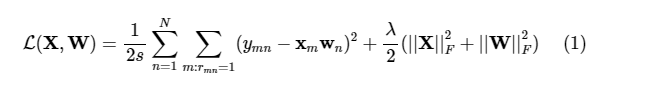
Về mặt tính toán, Item-item CF có thể nhận được từ User-user CF bằng cách chuyển vị (transpose) ma trận utility, và coi như *items* đang *rate* *users*. Sau khi tính ra kết quả cuối cùng, ta lại chuyển vị một lần nữa để thu được kết quả.

#### **Matrix Fractorization**

Ý tưởng chính đằng sau Matrix Factorization cho Recommendation Systems là tồn tại các *latent features* (tính chất ẩn) mô tả sự liên quan giữa các *items* và *users*. Ví dụ với hệ thống gợi ý các bộ phim, tính chất ẩn có thể là *hình sự*, *chính trị*, *hành động*, *hài*, …; cũng có thể là một sự kết hợp nào đó của các thể loại này; hoặc cũng có thể là bất cứ điều gì mà chúng ta không thực sự cần đặt tên.

Việc *learning* có thể hơi phức tạp một chút vì phải lặp đi lặp lại việc tối ưu một ma trận khi cố định ma trận còn lại, nhưng quá trình dự đoán (*inference)* đơn giản hơn [Neighborhood-based Collaborative Filtering](https://machinelearningcoban.com/2017/05/24/collaborativefiltering/)

* Hàm mất mát

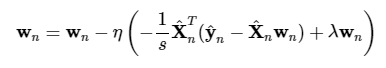


**Hình 1.1:** Hàm mất mát

Trong đó rmn=1 nếu item thứ m đã được đánh giá bởi *user* thứ nn,  ||X||2F là [Frobineous norm](https://machinelearningcoban.com/math/" \l "chuan-cua-ma-tran), tức căn bậc hai của tổng bình phương tất cả các phần tử của ma trận (giống với norm 2 trong vector), ss là toàn bộ số *ratings* đã có. Thành phần thứ nhất chính là trung bình sai số của mô hình. Thành phần thứ hai trong hàm mất mát phía trên là [l2 regularization](https://machinelearningcoban.com/2017/03/04/overfitting/#-\\l\\-regularization), giúp tránh [overfitting](https://machinelearningcoban.com/2017/03/04/overfitting/).

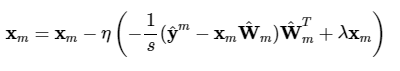
* **Lưu ý:** Giá trị ratings thường là các giá trị đã được chuẩn hoá, bằng cách trừ mỗi hàng của Utility Matrix đi trung bình cộng của các giá trị đã biết của hàng đó (item-based) hoặc trừ mỗi cột đi trung bình cộng của các giá trị đã biết trong cột đó (user\_based). Trong một số trường hợp nhất định, ta không cần chuẩn hoá ma trận này, nhưng kèm theo đó phải có thêm các kỹ thuật khác để giải quyết vấn đề thiên lệch trong khi rating.
* Tối ưa hàm mất mát

**Công thức cập nhật cho mỗi cột của W là:**



**Hình 1.2:** Công thức tối ưa hàm mất mát của w

**Công thức cập nhật cho mỗi cột của X là:**



**Hình 1.3:** Công thức tối ưa hàm mất mát của x

### **Content-based filtering**

Đặc điểm của phương pháp này là việc xây dựng mô hình cho mỗi *user* không phụ thuộc vào các *users* khác. Được coi như bài toán Regression hoặc Classsification với training data là cặp dữ liệu (item profile, rating) mà user đó đã rated

Các bước:

* Item profiles
* Xây dựng hàm mất mát

Hạn chế:

* Không tận dụng được thông tin của user
* Không phải lúc nào cũng có bản mô tả các items

Ví dụ: Bài toán với cơ sở dữ liệu MovieLens 100k

Phân loại: Trong các phương pháp dựa trên nội dung, vấn đề được đưa vào phân lớp (dự đoán xem người dùng thích hay không thích một mặt hàng) hoặc vấn đề hồi quy (dự đoán mức độ đánh giá mà người dùng đưa ra cho một mặt hàng). Trong cả hai trường hợp, chúng ta sẽ thiết lập một mô hình sẽ dựa trên các đặc điểm của người dùng hoặc sản phẩm theo ý của chúng ta.

* Item-centred

Nếu việc phân lớp (hoặc phân cụm) dựa trên các đặc điểm của người dùng (users features), việc mô hình hóa, tối ưu hóa và tính toán được thực hiện trên sản phẩm.

Xây dựng và tìm hiểu một mô hình theo item-based trên users features cố gắng trả lời câu hỏi, "Xác suất của mỗi người dùng thích mặt hàng này là bao nhiêu?" (Hoặc Mức đánh giá của mỗi người dùng về sản phẩm là bao nhiêu?). Mô hình xây dựng theo phương pháp này ít cá nhân hóa so với phương pháp lấy người dùng làm trung tâm.

* User centred

Nếu chúng ta đang làm việc với các đặc điểm của sản phẩm, việc đào tạo mô hình hóa, tối ưu hóa và tính toán có thể được thực hiện dựa trên người dùng.

Mô hình theo user based trên items features cố gắng trả lời câu hỏi. "Xác suất người dùng này thích từng mặt hàng là bao nhiêu?" (Hay mức đánh giá mà người dùng này đưa ra cho mỗi mặt hàng là gì?).

Sau đó, chúng ta có thể đính kèm một mô hình cho mỗi người dùng được đào tạo về dữ liệu của mình do đó mô hình được cá nhân hóa hơn so với phương pháp trung vào item vì nó chỉ tính đến các tương tác từ người dùng được xem xét. Tuy nhiên, người dùng tương tác với tương đối ít sản phẩm và do đó, mô hình chúng ta thu được sẽ kém mạnh mẽ hơn so với mô hình tập trung vào sản phẩm.

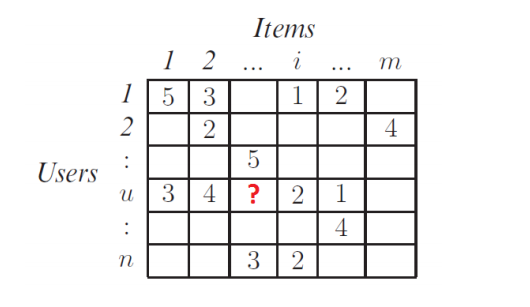
* **Cách làm của một mô hình gợi ý:**

***Xác định thành phần của một hệ thống gợi ý****:*

Dữ liệu: Đầu tiên chúng ta cần có dữ liệu về users, items, feedback

* Users là danh sách người dùng
* Items là danh sách sản phẩm, đối tượng của hệ thống. Ví dụ như các bài viết trên trang viblo, các video trên youtube, ... Và mỗi item có thể kèm theo thông tin mô tả.
* Feedback là lịch sử tương tác của user với mỗi item, có thể là đánh giá của mỗi user với một item, số ratings, hoặc comment, việc user click, view hoặc mua sản phẩm, ...

***Ma trận user-item: Utility matrix***



**Hình 1.4:** Ma trận user-item

Đây là ma trận biểu diễn mức độ quan tâm (rating) của user với mỗi item. Ma trận này được xây dựng từ dữ liệu (1). Nhưng ma trận này có rất nhiều các giá trị miss. Nhiệm vụ của Hệ gợi ý chính là dựa vào các ô đã có giá trị trong ma trận trên (dữ liệu thu được từ trong quá khứ), thông qua mô hình đã được xây dựng, dự đoán các ô còn trống (của user hiện hành), sau đó sắp xếp kết quả dự đoán (ví dụ, từ cao xuống thấp) và chọn ra Top-N items theo thứ tự rating giảm dần, từ đó gợi ý chúng cho người dùng.

***Phương pháp gợi ý:***

Content-based Filtering: Gợi ý các item dựa vào hồ sơ (profiles) của người dùng hoặc dựa vào nội dung/thuộc tính (attributes) của những item tương tự như item mà người dùng đã chọn trong quá khứ.

* **Phương pháp thực hiện Content-based Filtering:**

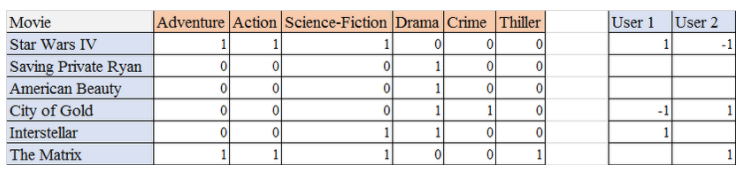
Thuật toán content-based gồm 2 bước:

* Bước 1: Biểu diễn items dưới dạng vec-tơ thuộc tính - item profile
* Bước 2: Học mô hình của mỗi user

***Xây dựng Items Profile***

Trong các hệ thống content-based, chúng ta cần xây dựng một bộ hồ sơ (profile) cho mỗi item. Profile này được biểu diễn dưới dạng toán học là một "feature vector" n chiều. Trong những trường hợp đơn giản (ví dụ như item là dữ liệu dạng văn bản), feature vector được trực tiếp trích xuất từ item. Từ đó chúng ta có thể xác định các item có nội dung tương tự bằng cách tính độ tương đồng giữa các feature vector của chúng.

Sử dụng biểu diễn nhị phân: Ví dụ:



**Hình 1.5:** Ví dụ item profile

Trên đây là danh sách 6 bộ phim. Mỗi giá trị 0/1 thể hiện bộ phim đó không/có thuộc thể loại ở cột tương ứng. Bên cạnh đó, một hồ sơ người dùng cũng được tạo ra, với 1 là quan tâm, -1 là không, và null là chưa đánh giá. Như trong ví dụ trên, User 1 có quan tâm bộ phim Star Wars IV, còn User 2 thì không.

### **Các phương pháp lai**

Các phương pháp này là sự kết hợp các phương pháp tiếp cận dựa trên nội dung và lọc cộng tác, chúng cho kết quả tốt trong nhiều trường hợp và do đó, được sử dụng trong nhiều hệ thống đề xuất quy mô lớn hiện nay. Sự kết hợp được thực hiện trong các phương pháp lai có thể chủ yếu có hai dạng:

* Huấn luyện hai mô hình một cách độc lập (một mô hình lọc cộng tác và một mô hình dựa trên nội dung) và kết hợp các đề xuất của chúng
* Trực tiếp xây dựng một mô hình để thống nhất cả hai cách tiếp cận bằng cách sử dụng làm thông tin trước khi nhập (về người dùng và / hoặc vật phẩm) cũng như thông tin tương tác của cộng tác trực tuyến.

## **Vậy chúng ta nên sử dụng phương pháp nào?**

Có một điều dễ nhận thấy thì phương pháp gợi ý dựa trên nội dung đòi hỏi chúng ta phải thu thập rất nhiều thông tin về các mục tin tương tự. Chính việc xác định xem một mục tin nào là tương tự với mục tin hiện tại đòi hỏi chúng ta phải thu thập và phần tích, xử lý toàn bộ các mục tin trong cơ sở dữ liệu. Tuy nhiên với phương pháp lọc công tác chúng ta không cần quá nhiều thông tin. Đơn giản chỉ là item\_id của item hiện tại, các user\_id và các feedback trên item đó mà thôi nên thực tế thì phương pháp lọc cộng tác được sử dụng phổ biến hơn để xây dựng các hệ thống gợi ý.

# **CHƯƠNG 2 CHƯƠNG TRÌNH DEMO**

## **Các thư viện sử dụng**

* **import pandas as pd**

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Tên thư viện được bắt nguồn từ panel data (bảng dữ liệu). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có tiềm năng không đồng nhất) và dữ liệu chuỗi thời gian.

* **import numpy as np**

Numpy (Numeric Python): là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.

* **import matplotlib.pyplot as plt**

Matplotlib là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và NumPy. Module được sử dụng nhiều nhất của Matplotib là Pyplot cung cấp giao diện như MATLAB nhưng thay vào đó, nó sử dụng Python và matplotlib là nguồn mở.

* **import re**

Biểu thức chính quy (Regular Expressions) hay Regex trong Python có thể được định nghĩa là chuỗi các ký tự được sử dụng để tìm kiếm một mẫu trong chuỗi

* **import nltk**

Bộ công cụ ngôn ngữ tự nhiên (NLTK) là thư viện phổ biến nhất để xử lý ngôn ngữ tự nhiên (NLP). Nó được viết bằng Python và có một cộng đồng lớn đằng sau Python.

* **import seaborn as sns**

Trực quan hóa dữ liệu thống kê là một thư viện Python phổ biến để thực hiện EDA. Phân tích dữ liệu thăm dò (EDA) là một cách tiếp cận để phân tích các tập dữ liệu để tóm tắt các đặc điểm chính của chúng, thường bằng các phương pháp trực quan.

* **from sklearn.metrics.pairwise import cosine\_similarity**

Độ tương tự cosine là một phép đo độ giống nhau giữa hai vectơ khác không của một không gian tích bên trong . Nó được định nghĩa là bằng cosin của góc giữa chúng, cũng giống như tích trong của các vectơ giống nhau được chuẩn hóa để cả hai đều có độ dài 1. Tính cosin của 0 ° là 1 và nó nhỏ hơn 1 đối với bất kỳ góc nào trong khoảng (0, π] radian. Do đó, nó là một phán đoán về định hướng chứ không phải độ lớn: hai vectơ cùng hướng có độ tương tự cosin là 1, hai vectơ hướng 90 ° so với nhau có độ giống nhau là 0, và hai vectơ đối nhau theo đường kính có độ giống nhau là – 1

* **from sklearn.feature\_extraction.text import CountVectorizer**

Chuyển đổi bộ sưu tập tài liệu văn bản thành ma trận số lượng mã thông báo

* **from sklearn.feature\_extraction.text import TfidfVectorizer**

Là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản.

* **from nltk.corpus import stopwords**

Là thư viện chứa các từ không liên quan còn gọi là “từ dừng”

* from scipy import spatial

Các cấu trúc dữ liệu không gian và thuật toán của thư viện Scipy

## **Thực hiện bài toán**

# In[0]: thêm thư viện

Thêm các thư viện để sử dụng các lệnh từ thư viện đã khai báo

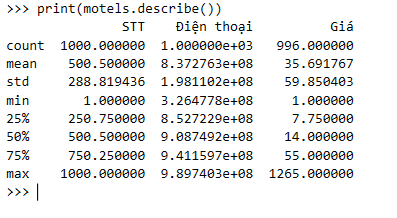
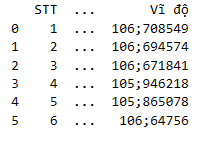
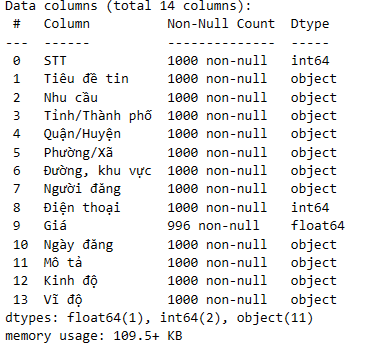
|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import re  import nltk  import seaborn as sns  from sklearn.metrics.pairwise import cosine\_similarity  from sklearn.feature\_extraction.text import CountVectorizer  from sklearn.feature\_extraction.text import TfidfVectorizer |

# In[1]: Xem nhanh dữ liệu

Sau khi thực hiện khai báo các thư viện cần sử dụng, tiếp tục đến bước xem nhanh dữ liệu.

|  |
| --- |
| motels = pd.read\_csv(r"C:\Users\hoang\Desktop\DOAN\_3\DULIEU\dulieu.csv")  moteltest = pd.read\_csv(r"C:\Users\hoang\Desktop\DOAN\_3\DULIEU\dulieutest.csv")  print('\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dataset info  print(motels.info())  print('\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Some first data examples  print(motels.head(6))  print('\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Statistics of numeric features  print(motels.describe()) |

Đọc dữ liệu từ file csv dữ liệu đã chuẩn bị sẵn. Dữ liệu(998x14) gồm 14 cột (bao gồm các mục: STT, Đường, khu vực, Người đăng, Điện thoại, Giá, Ngày đăng, Mô tả, Kinh độ, Vĩ độ) và 998 dòng.

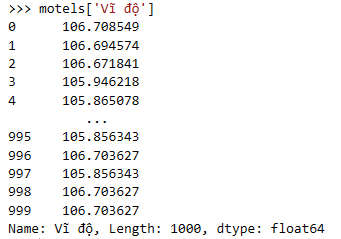


**Hình 2.1:** Đọc dữ liệu file data

Vấn đề khi load dữ liệu từ cột “Kinh độ” và “Vĩ độ” ở dạng dấu chấm phẩy sẽ chuyển thành dấu chấm do khi sử lý dữ liệu từ excel sang csv dấu chấm sẽ bị lỗi (106.708549 sẽ chuyển thành 106.708.549). Để sử lý dữ liệu ta sử dụng hàm Clean để chuyển dữ liệu chuyển dấu “;” (dữ liệu ban đầu: 10;731739) thành dấu chấm khi đó dữ liệu được định dạng lại đúng là kiểu float64.

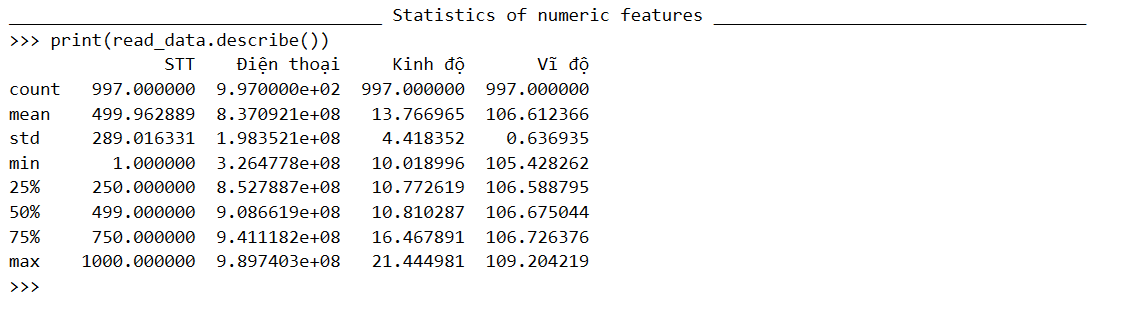
|  |
| --- |
| def clean(text):  text = text.replace(";", ".")  return float(text)  read\_data['Vĩ độ'] = read\_data['Vĩ độ'].apply(lambda x: clean(x))  read\_data['Kinh độ'] = read\_data['Kinh độ'].apply(lambda x: clean(x)) |

Dữ liệu sau khi được chuẩn hóa



**Hình 2.2:** Dữ liệu kinh độ

Đánh giá dữ liệu: Số lượng các cột(count), mean là trung bình cộng của tất cả các giá trị, min là giá trị nhỏ nhất, 25%-50%-75% là phần trăm giá trị bé hơn trong 100% giá trị, max là giá trị lớn nhất.



**Hình 2.3:** Mô tả dữ liệu

# In[2]: Tìm key word

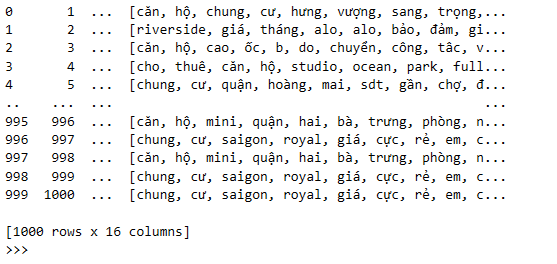
Khi sử lý dữ liệu ở phần “Mô tả”, chứa các thông tin về nhà trọ và được viết như một văn bản, từ văn bản này để cho máy tính hiểu được thì ta sẽ chuyển văn bản về dạng mảng để cho máy tính đọc hiểu. Để giải quyết vấn đề đó thì ta sẽ khởi tạo một hàm clean\_text(text): ở dạng chuỗi là một chuỗi dài gồm các ký tự có dấu nên để sử lý nhóm em sử dụng thư viện “re” để tách chuỗi bao gồm các chữ có dấu.

|  |
| --- |
| def clean\_text(text):  # remove backslash-apostrophe  text = re.sub("\'", "", text)  # remove whitespaces  text = ' '.join(text.split())  # remove everything except alphabets  # text = re.sub("[^a-zA-Z]"," ",text)  # convert text to lowercase  text = text.lower()  text = re.findall(r'(?i)\b[a-záàảãạăắằẳẵặâấầẩẫậéèẻẽẹêếềểễệóòỏõọôốồổỗộơớờởỡợíìỉĩịúùủũụưứừửữựýỳỷỹỵđ]+\b', text)  return text |

Hàm clean\_text(text) lấy ý tưởng: khi gặp khoảng trắng hoặc ký tự đặc biệt sẽ tách ký tự ra thành một keyword và một đoạn văn sẽ được lưu thành một mảng và các mảng sẽ được lưu vào một cột có tên là “keyword”

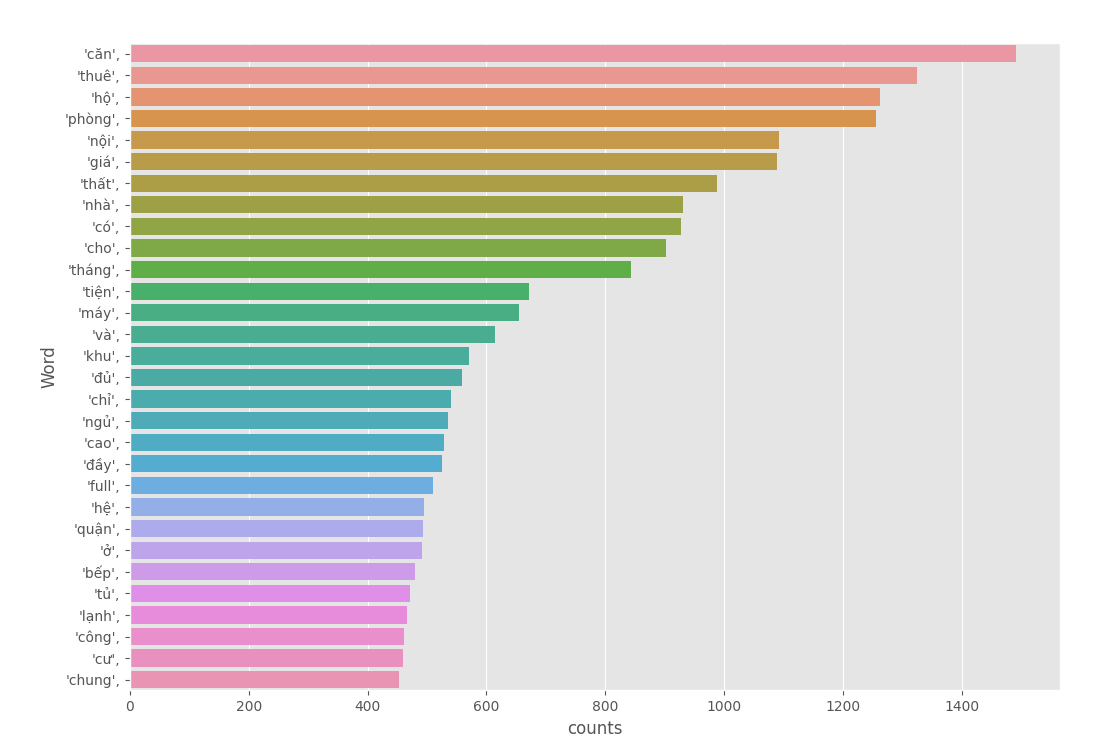
|  |
| --- |
| motels['merger table'] = motels['Tiêu đề tin']+' '+motels['Mô tả']  motels['keyword'] = motels['merger table'].apply(lambda x: clean\_text(x)) |

Trong dữ liệu nhóm em tham khảo có 2 yếu tố ảnh hưởng đến bài toán gợi ý là “Tiêu đề tin” và “Mô tả”, “Tiêu để tin” chứa tiêu đề sẽ được gợi ý ra và “Mô tả” sẽ là phần ghi chú thông tin của “Tiêu đề tin” đó. Bước tiếp theo để thực hiện đưa các chuỗi về mảng, ở bước này sẽ gộp hai cột lại với nhau tạo ra cột “merger table” và tiếp sau đó là thực hiện tách các chữ trong các hàng của cột “merger table”.



**Hình 2.4:** Gộp title và mô tả

Sau khi đã có dữ liệu sẽ tiến hành lọc ra xem keyword nào được sử dụng nhiều nhất



**Hình 2.5:** Danh sách 10 keyword đầu

Ở bước này chạy khởi tạo một hàm freq\_words (x, terms = 30) để thực hiện đếm dữ liệu và vẽ dữ liệu ra trên hình

|  |
| --- |
| def freq\_words(x, terms = 30):  #all\_words = ' '.join([text for text in x])  all\_words = ' '.join(str(text) for text in x)  all\_words = all\_words.split()  fdist = nltk.FreqDist(all\_words)  words\_df = pd.DataFrame({'word':list(fdist.keys()), 'counts':list(fdist.values())})    # selecting top 20 most frequent words  d = words\_df.nlargest(columns="counts", n = terms)    # visualize words and frequencies  plt.figure(figsize=(12,15))  ax = sns.barplot(data=d, x= "counts", y = "word")  ax.set(ylabel = 'Word')  plt.show() |

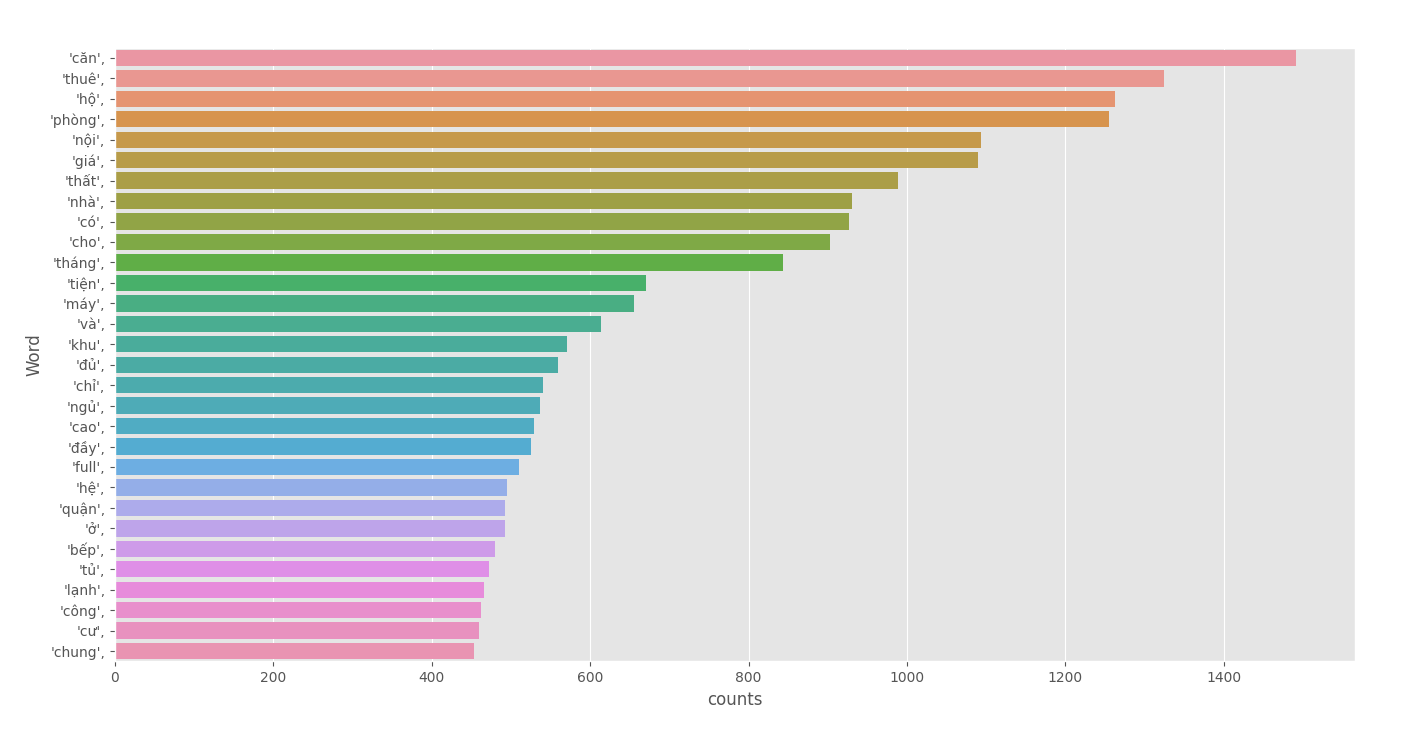
Tiếp theo sau khi thực hiện tìm ra được keyword, tiếp tục thực hiện xóa các từ không liên quan (ở đây các từ không liên quan được gọi là stopword những từ này sẽ làm cho bài toán bị nhiễu). Nhóm em tiếp tục xóa các từ dừng(stopword). Để sử lý vấn đề này thì nltk đã cung cấp một thư viện để thực hiện xóa các từ dừng này.

|  |
| --- |
| nltk.download('stopwords')  from nltk.corpus import stopwords |

Sau đó thực hiện khởi tạo một hàm remove\_stopwords(text):

|  |
| --- |
| def remove\_stopwords(text):  no\_stopword\_text = [w for w in str(text).split() if not w in stop\_words]  return ' '.join(no\_stopword\_text) |

Sau khi xóa từ dừng ta sẽ đưuọc dữ liệu mới và sẽ in ra 30 dữ liệu đầu tiên



**Hình 2.6:** Danh sách keyword qua stopword

Để học được keyword, ở đây đã có sẵn hàm để hỗ trợ trong sklearn là cosine\_similarity, CountVectorizer, TfidfVectorizer:

* CountVectorizer thực hiện: Chuyển đổi bộ sưu tập tài liệu văn bản thành ma trận số lượng mã thông báo
* TfidfVectorizer thực hiện: Chuyển đổi một bộ sưu tập các tài liệu thô sang một ma trận các tính năng TF-IDF
* Cosine\_similarity thực hiện: tính tích số chấm chuẩn hóa L2 của vectơ. Đó là, nếu x và y là các vectơ hàng, tính tương tự cosin của chúng

|  |
| --- |
| # Đánh số label  # Training  train\_data = count\_matrix.toarray()  label = np.random.randint(5, size=(1000))  # Call MultinomialNB  clf = BernoulliNB()  clf.fit(train\_data, label)  array = clf.predict(train\_data)  motels['labelrank'] = pd.DataFrame(array, columns=['labelrank']) |

Thực hiện đánh số lable cho từng giá trị train, sau đó ta dùng BernoulliNB() (bộ phân loại này phù hợp với dữ liệu rời rạc. Sự khác biệt là trong khi MultinomialNB hoạt động với số lần xuất hiện, BernoulliNB được thiết kế cho các tính năng nhị phân / boolean.) dữ liệu truyền vào là một countdown vector train\_data và lable của dữ liệu. Kết quả trả về là một mảng gồm các lable của từng tiêu đề tin

# In[3]: Similarity between motels

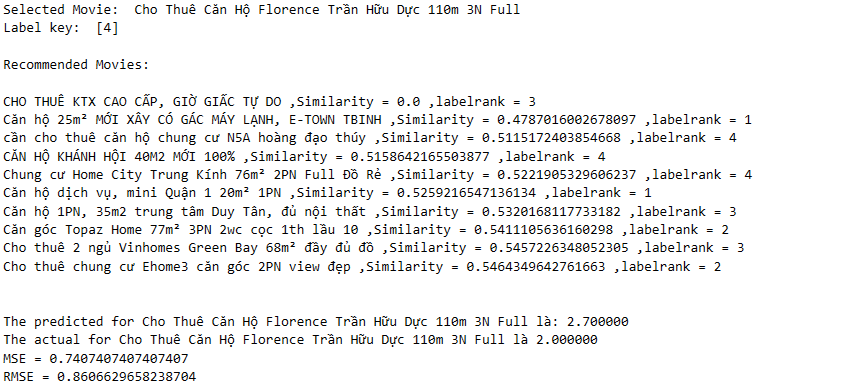
|  |
| --- |
| from scipy import spatial  def Similarity(motelid1, motelid2):  wordsA = count\_matrix.toarray()[motelid1]  wordsB = count\_matrix.toarray()[motelid2]  wordsDistance = spatial.distance.cosine(wordsA, wordsB)  return wordsDistance |

Hàm tính độ giống nhau giữa hai mảng.

# In45]: Score Predictor

|  |
| --- |
| def predict\_score(title, id):  #name = input('Nhập title của nhà trọ: ')  #title = "THIẾT KẾ MỚI - NỘI THẤT ĐẸP - 2PN 65m2 - SAFIRA Q9"  name = title;  try:  new\_motel = motels[motels['Tiêu đề tin'].str.contains(name)].iloc[0].to\_frame().T    original\_title = 'Tiêu đề tin';  print('Selected Movie: ',new\_motel['Tiêu đề tin'].values[0])  array = np.array([count\_matrix\_test.toarray()[id]])  print('Label key: ',clf.predict(array))  def getNeighbors(baseMovie, K):  distances = []    for index, motel in motels.iterrows():  if motel['STT'] != baseMovie['STT'].values[0]:  dist = Similarity(baseMovie['STT'].values[0], motel['STT'] - 1)  distances.append((motel['STT'], dist))    distances.sort(key=operator.itemgetter(1))  neighbors = []    for x in range(K):  neighbors.append(distances[x])  return neighbors  # Lấy 10 motel  K = 10  labelkey = 0  neighbors = getNeighbors(new\_motel, K)  print('\nRecommended Movies: \n')  for neighbor in neighbors:  labelkey = labelkey + motels.iloc[neighbor[0]][16]  print("{} ,Similarity = {} ,labelrank = {}".format(motels.iloc[neighbor[0]][1], neighbor[1], motels.iloc[neighbor[0]][16]))    print('\n')  labelkey = labelkey/K  print('The predicted for %s là: %f' %(new\_motel['Tiêu đề tin'].values[0],labelkey))  print('The actual for %s là %f' %(new\_motel['Tiêu đề tin'].values[0],new\_motel['labelrank']))  mse = float(new\_motel['labelrank'].values[0])/labelkey  rmse = np.sqrt(mse)  print('MSE = {}'.format(mse))  print('RMSE = {}'.format(rmse))  except:  print("Không tìm thể tìm thấy") |

Ở trong hàm predict\_score sẽ thực hiện tìm ra được giá tiền mà căn nhà bạn muốn gợi ý và giá tiền thật của căn nhà đó là bao nhiêu: getNeighbors là tìm kiếm những tiêu đề có độ giống nhau(similar) cao hoặc là gần nhau



**Hình 2.7:** Kết quả thu được

Kết quả thu được sau khi chạy.

# **CHƯƠNG 3 KẾT LUẬN**

## **Kết quả đạt được**

Với dữ liệu đầu vào là một file dữ liệu thực tế, thuật toán có thể dự đoán được những kết quả gần như tương đồng dựa vào điểm tương đồng giữa các feature có ảnh hưởng đến kết quả. Người dùng sau khi nhập yêu cầu ngôi nhà cần tìm kết quả gợi ý trả ra là 10 tiêu đề tin. Thuật toán có thể áp dụng tốt mới trang web có nhiều dữ liệu.

## **Khuyết điểm**

Với dự đoán ra có phần còn sai sót chưa hoàn toàn chính xác .

# **TÀI LIỆU THAM KHẢO**

[1] Tham khảo lý thuyết về recommendation

<https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/>

[2] Tham khảo demo code

<https://viblo.asia/p/gioi-thieu-ve-he-thong-goi-y-recommender-systems-hoac-recommendation-systems-maGK78yOZj2>

[3] Tham khảo source code

<https://www.kaggle.com/rounakbanik/movie-recommender-systems>

[4] Tham khảo demo code

https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/