1. **Giới thiệu**
   1. **Nền tảng**

Google Play là chợ ứng dụng số một trên nền tảng Android, với hàng trăm nghìn ứng dụng và

trò chơi cùng với hàng trăm triệu người dùng trên khắp thế giới. Để có thể gợi ý cho từng ấy người dùng các tựa game mà họ sẽ yêu thích, giúp các nhà phát triển ứng dụng tăng số lượt tải, phải có một hệ thống gợi ý ứng dụng trò chơi hiệu quả hỗ trợ kèm theo.

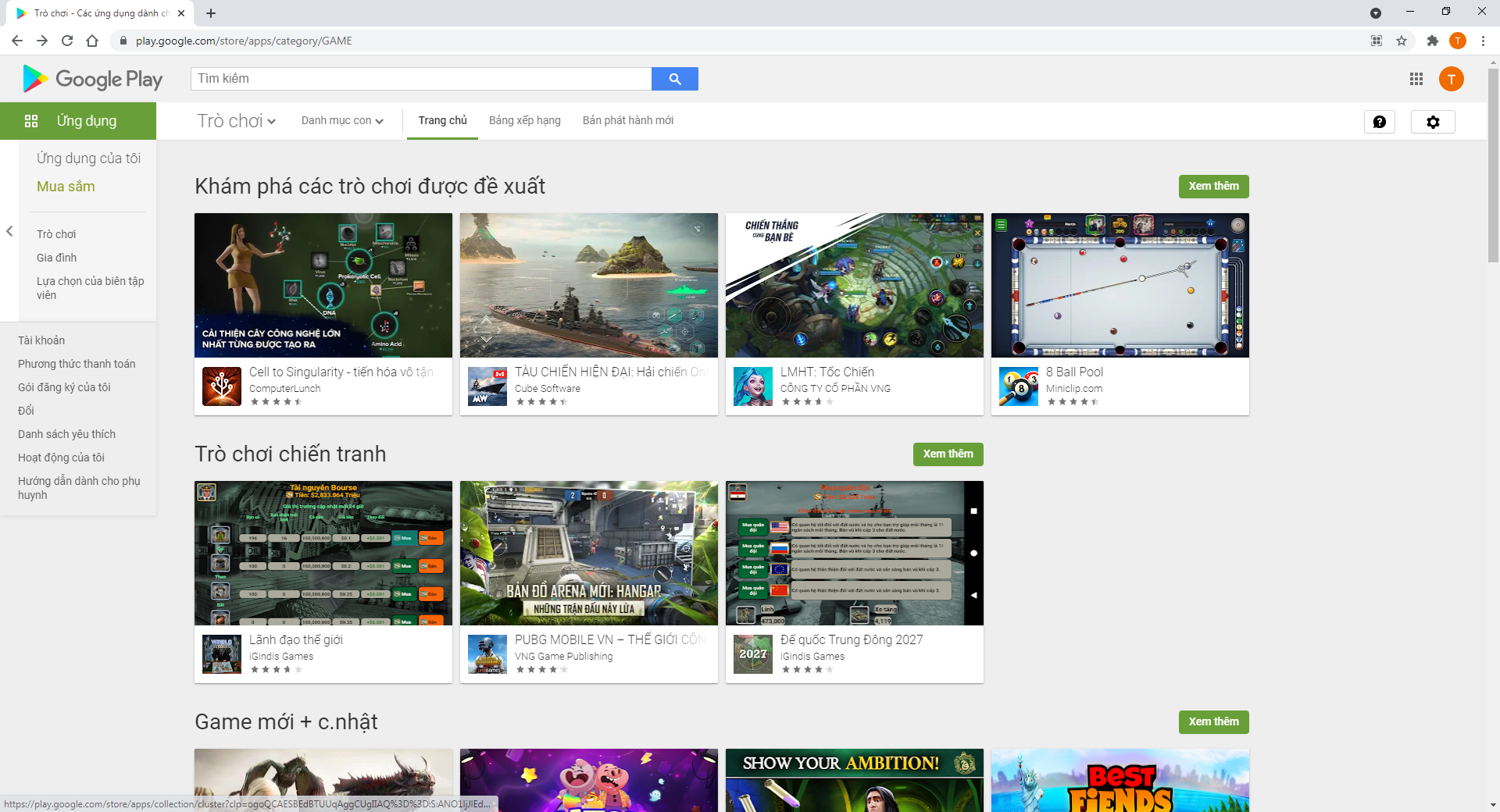
* 1. **Hướng nghiên cứu**

Trong dự án này, tôi sẽ trình bày cách thức xây dựng số mô hình gợi ý bằng dữ liệu của game và đánh giá trên Google Play, phân tích điểm mạnh và điểm yếu của từng mô hình gồm:

* Description Based Recommender: gợi ý dựa trên các thuộc tính và nội dung mô tả của trò chơi
* User-Based Collaborative Filtering: gợi ý dựa trên tương tác của người dùng
* Item-Based Collaborative Filtering: lọc cộng tác dựa trên sản phẩm

1. **Thu thập và xử lý dữ liệu**
   1. **Nguồn dữ liệu**

Toàn bộ dữ liệu bao gồm thông tin về game kèm theo một trăm bình luận mỗi game được lấy từ Google Play trong phân mục “Trò Chơi” (<https://play.google.com/store/apps/category/GAME>)



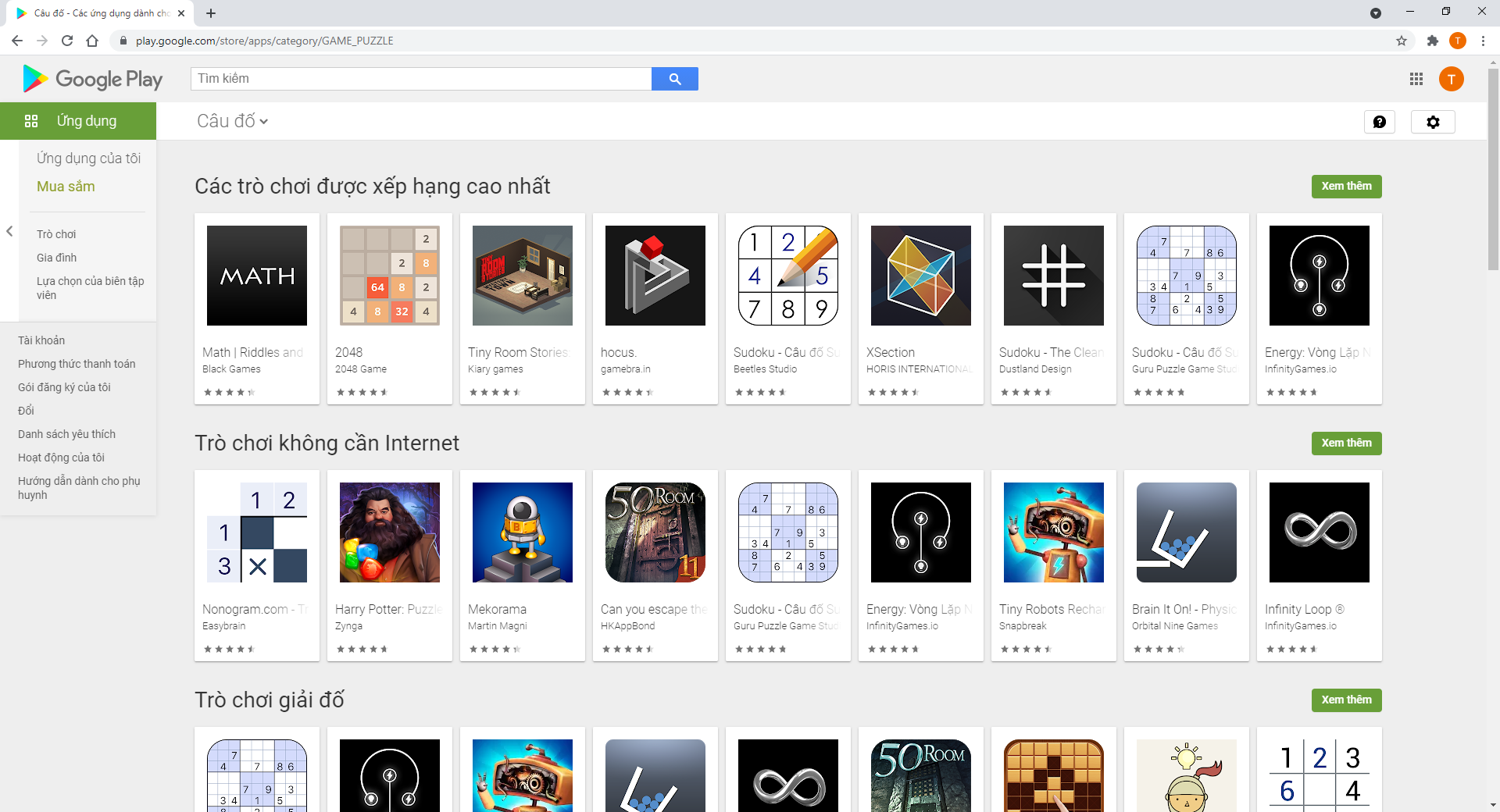
* + 1. **Quá trình cào dữ liệu trò chơi**

Thư viện đã dùng:

* scrapy
* google-play-scraper (<https://pypi.org/project/google-play-scraper/> )

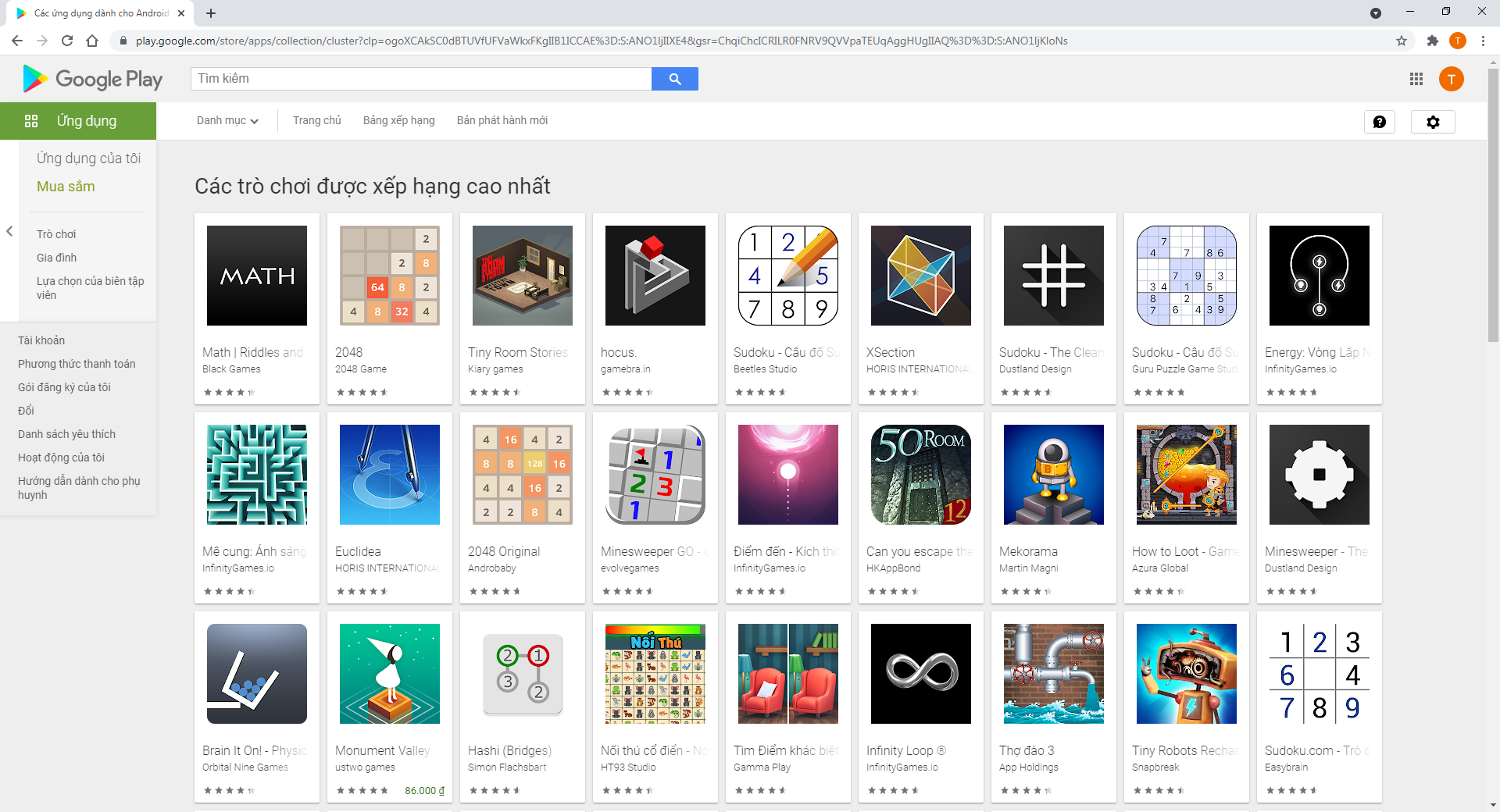
Bước 1: Lấy đường dẫn của các thư mục con bằng xpath:

categories = response.xpath('//a[@class="r2Osbf"]/@href').extract()

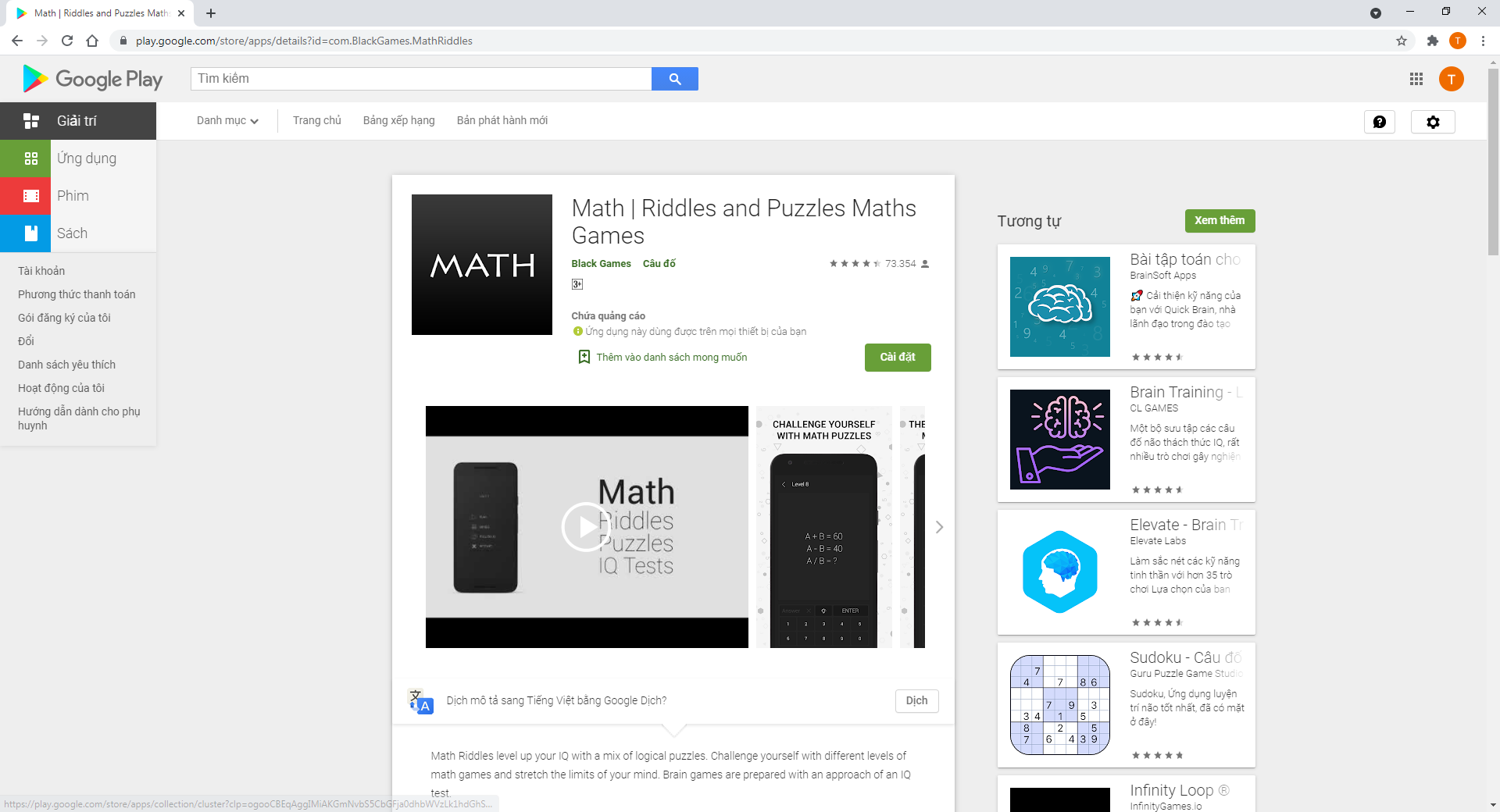


Bước 2: Lấy đường dẫn ở nút xem thêm ở từng hạng mục

sub\_category\_urls = response.xpath(  
 '//a[@class="LkLjZd ScJHi U8Ww7d xjAeve nMZKrb id-track-click "]/@href').extract()



Bước 3: Thu thập thông tin trong từng game



Dùng API google-play-scraper để thu thập các thông tin chi tiết về trò chơi.

result = app(game\_id, lang='en', country='us')

Dữ liệu đã thu thập được:

* 3509 trò chơi với 17 cột
  + Id
  + Title (tên game)
  + url (đường link của game)
  + category (thể loại)
  + avg\_rating (số điểm đánh giá trung bình)
  + rating\_count (số lượt đánh giá)
  + price (giá)
  + developer\_name (tên nhà phát triển)
  + developer\_url (trang Google Play của nhà phát triển)
  + description (mô tả về game)
  + summary (mô tả vắng tắt)
  + min\_installs (số lượt tải tối thiểu)
  + editors\_choice (game được biên tập viên lựa chọn)
  + size (kích cỡ)
  + android\_version (yêu cầu về phiên bản android tối thiểu)
  + content\_rating (phân loại nội dung theo độ tuổi)
  + ad\_supported (game có chứa quảng cáo)
  + released (ngày phát hành)
    1. **Quá trình cào dữ liệu người đánh giá (reviews)**

Sử dụng thư viện google-play-scraper lấy các đánh giá bằng id của trò chơi

results, continuation\_token = reviews(game\_id,  
 count=10,  
 lang='en',  
 country='us',  
 sort=Sort.MOST\_RELEVANT)

* 336024 dòng dữ liệu về đánh giá game với 9 cột
  + id
  + username (tên người dùng)
  + score (số điểm đánh giá)
  + content (nội dung đánh giá)
  + like\_count (số lượt thích của đánh giá)
  + date (ngày đánh giá)
  + game\_title (tên game)
  + game\_id (id game)
  + game\_category (thể loại của game)
  1. **Làm sạch**
     1. **Games**
* Chuyển đổi cột “price” từ VNĐ sang USD.
* Tạo cột “is\_free” dựa trên cột “price”.
* Thay thế các giá trị NaN của cột “size” thành “Varies with device”.
* Bỏ các dòng có “content\_rating” NaN.
* Chuyển đổi cột size thành dạng biến phân loại.
  + Tiny: Nhỏ hơn 20 Mb.
  + Small: Từ 20 Mb đến 100 Mb.
  + Medium: Từ 100 Mb đến 500 Mb.
  + Large: Trên 500 Mb.
  + Varies with device: Không xác định được kích thước cụ thể.
* Do các trò chơi có min\_installs nhỏ hơn 5000 khá ít nên tôi sẽ gom các game này vào nhóm 5000.

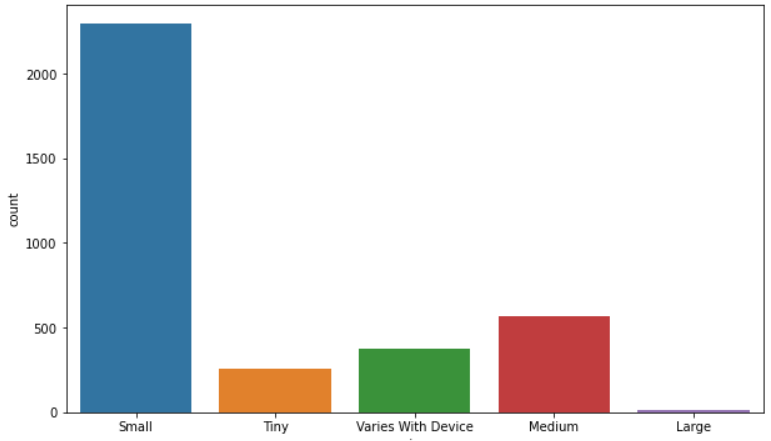
Tương tự với các game có min\_installs trên 100 triệu, sẽ gom chung vào nhóm 100 triệu

.

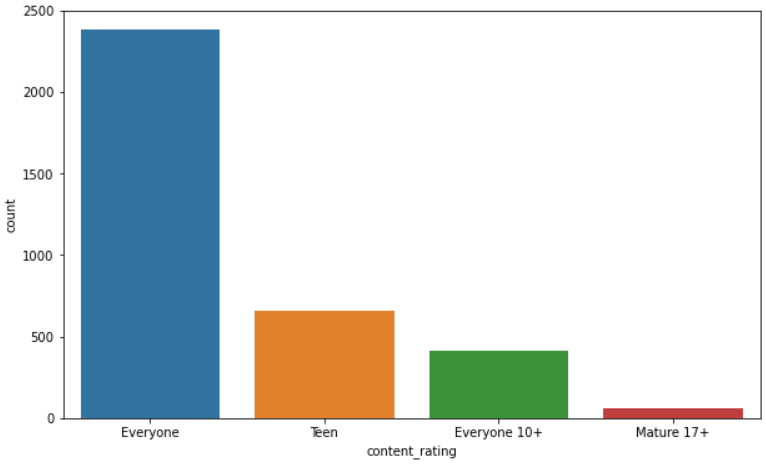
* + 1. **Reviews**

Trong số các username, có một username là “A Google user” đánh giá hơn năm nghìn lượt, thật ra đây là rất nhiều người dùng đã bình luận game bằng tài khoản Google +. Sau khi Google khai tử Google + thì ở Google Play hiển thị từ “A Google user” thay cho tên người dùng thật. Để tránh nhiễu, tôi sẽ bỏ các đánh giá có username này. Tổng số đánh giá còn lai là 330,607.

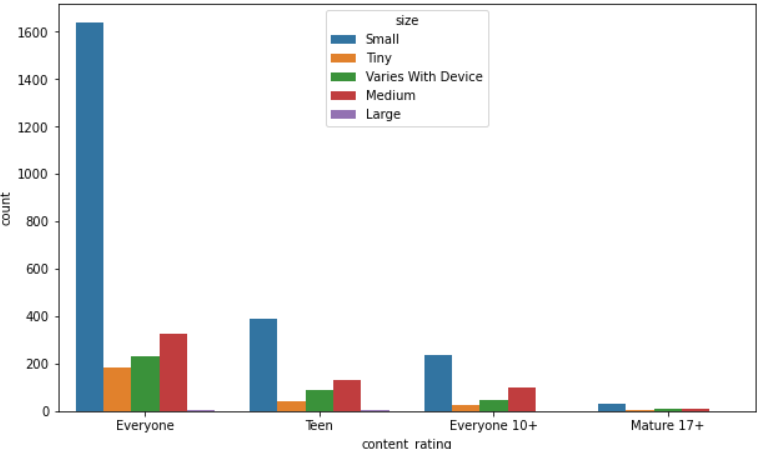
1. **Phân tích Khám phá Dữ liệu – EDA**
   1. **Mối quan hệ giữa “size” và “content\_rating”**



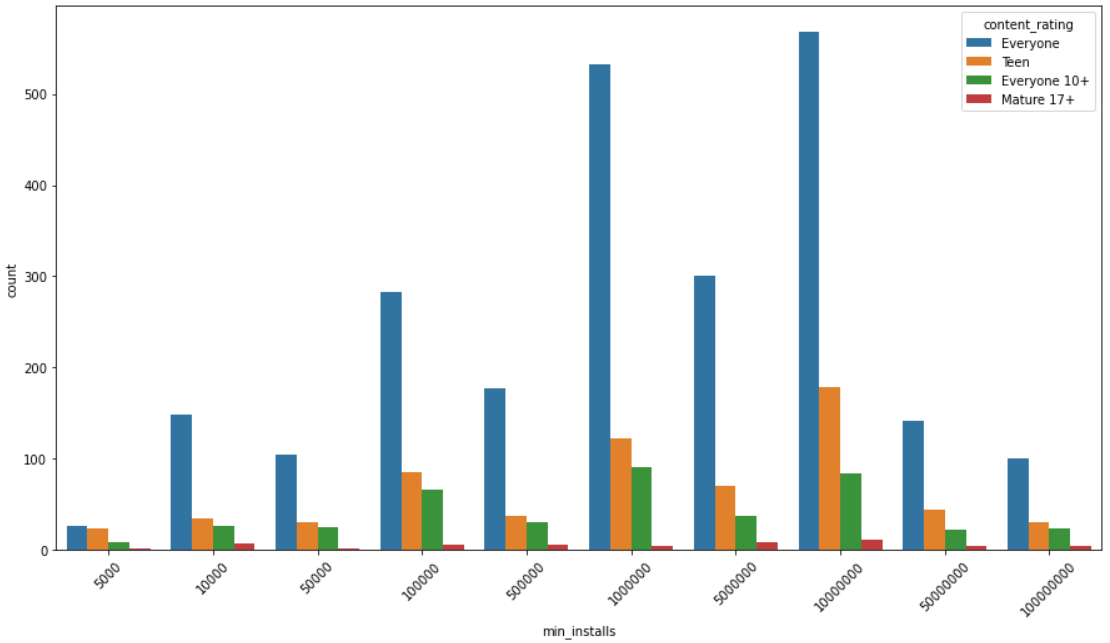
Các game có dung lượng nhỏ (từ 20-100 Mb) chiếm đa số trong dữ liệu. Game có dung lượng trung bình (từ 100-500 Mb) đứng thứ hai về số lượng. Game có dung lượng tùy thiết bị cũng chiếm khoảng 15%. Game có dung lượng dưới 20 Mb đứng thứ 3. Các game có dung lượng lớn (hơn 500 Mb) là không đáng kể.

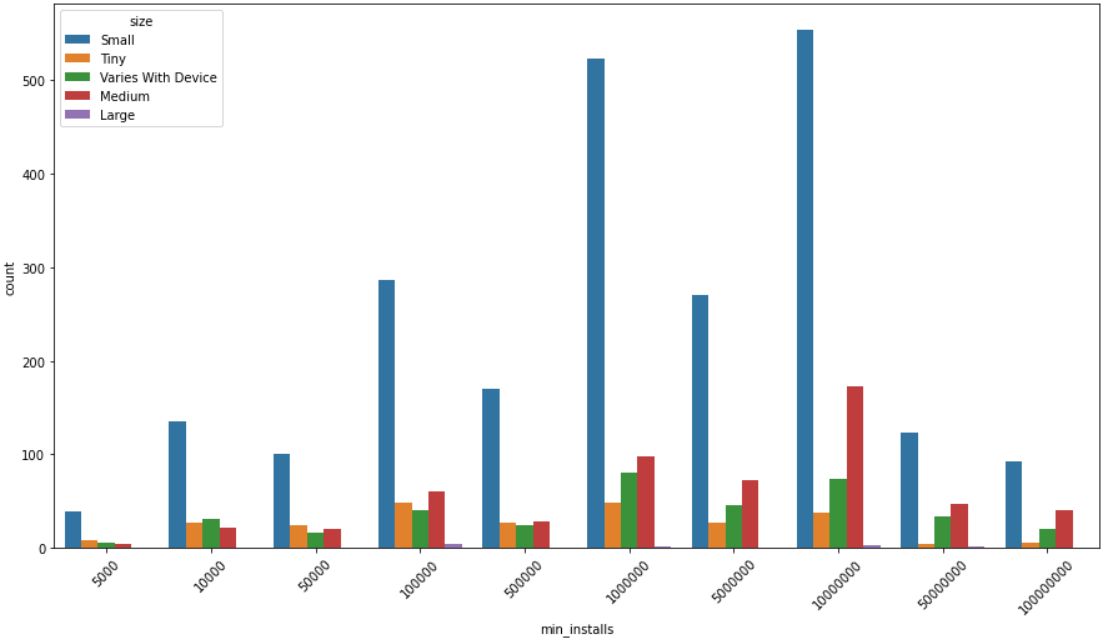


Các game có nội dung dành cho mọi lứa tuổi cũng chiếm đa số trong dữ liệu. Game dành cho chứa lứa tuổi teen và trên 10 tuổi chiếm khoảng 1/3 số game trong dữ liệu. Game có nội dung người lớn chiếm số lượng không đáng kể



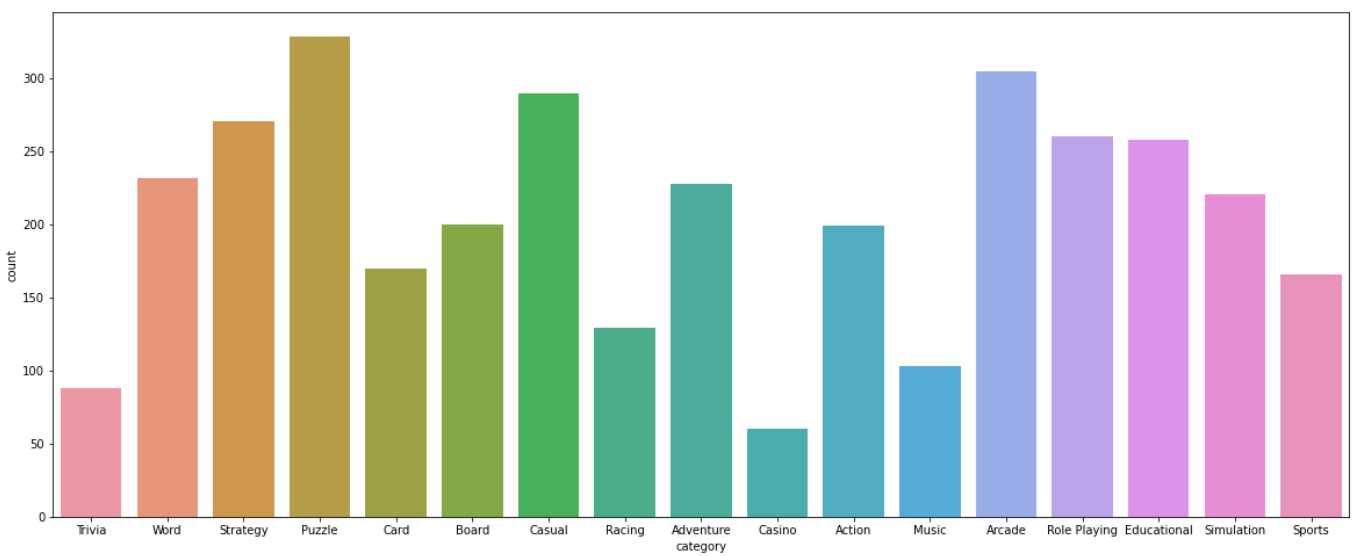
Game có dung lượng nhỏ và có nội dung phù hợp cho mọi lứa tuổi chiếm gần phân nửa trong dữ liệu (1600 trong số với 3500 game)



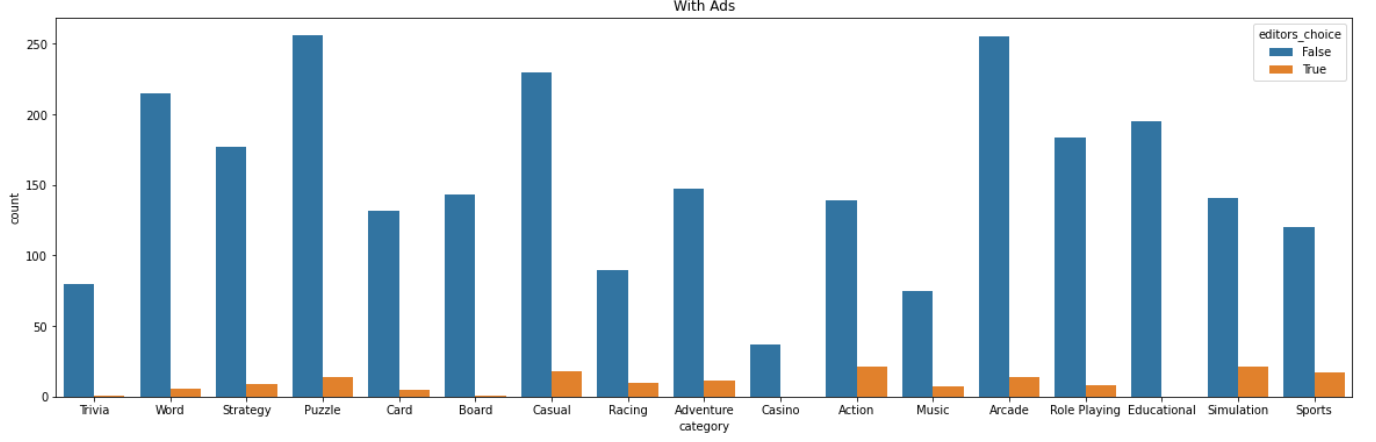


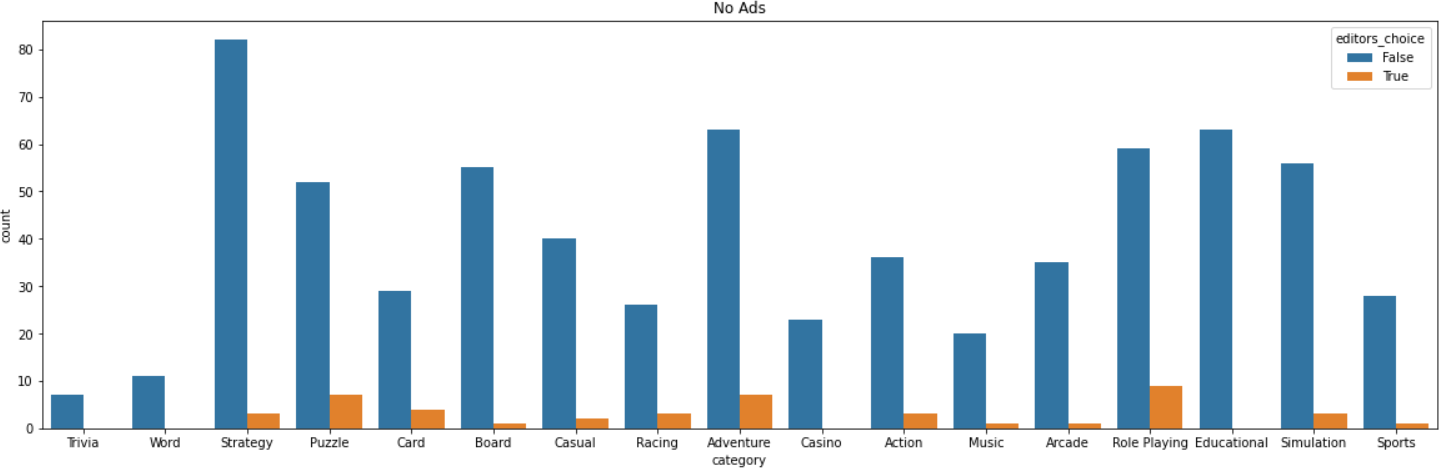
Đa số các game có lượt tải từ một trăm nghìn đến 10 triệu, content\_rating và size có phân phối tương đồng nhau phân theo lượt tải.

* 1. **Phân tích các thể loại trong game**



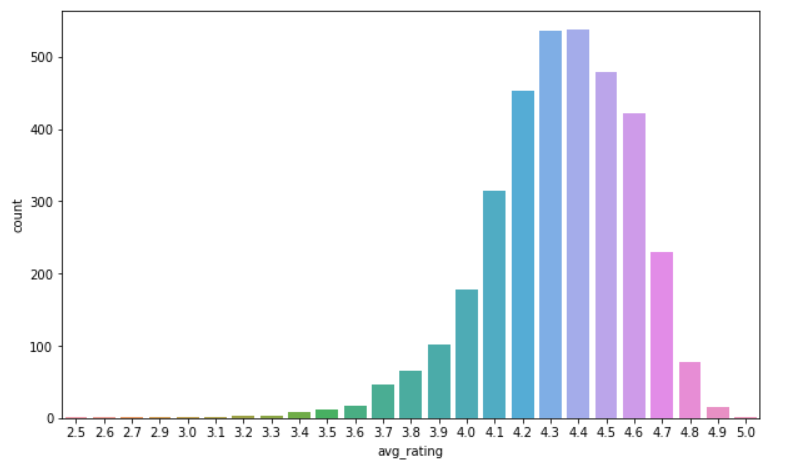
Game Puzzle, Arcade, Casual và Strategy chiếm nhiều nhất, game Casino và Trivia (đố vui) có số lượng ít hơn cả.

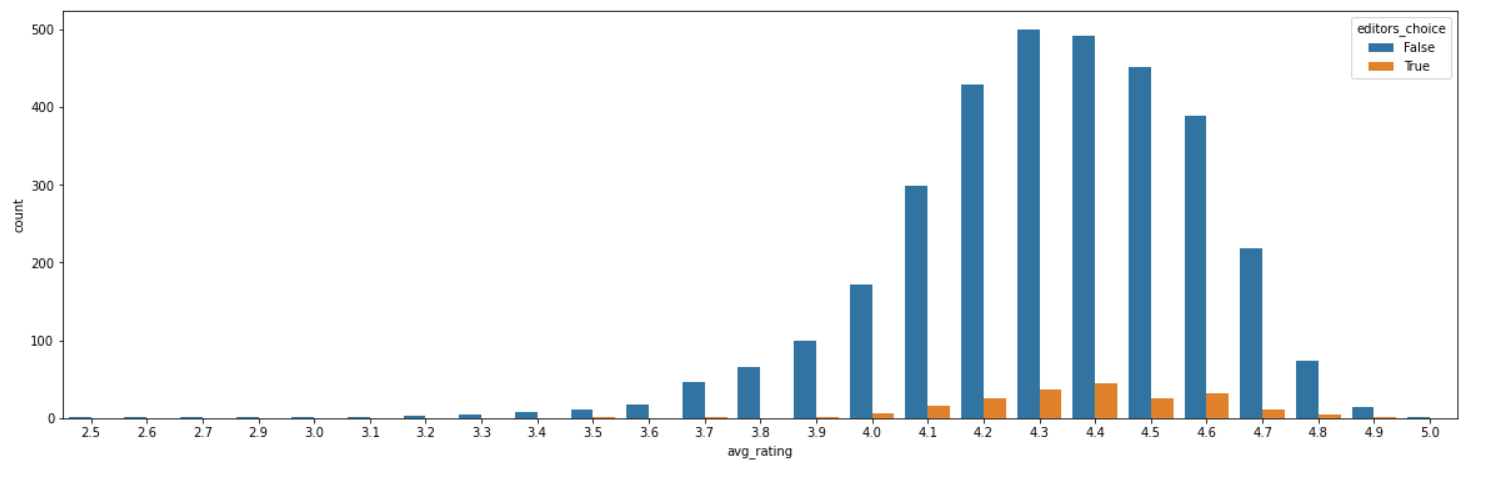




Số game có “editor\_choice” có ở hầu hết các thể loại (trừ Educational và Casino). Game thuộc thể loại “Trivia” và “Board” không được các biên tập viên đánh giá cao. Tỉ lệ phân bố của “editor\_choice” giữa game có quảng cáo và không có quảng cáo là tương đương nhau.

* 1. **Phân tích rating**





Rating phần lớn từ 4.1 đến 4.6, phần lớn các game editors\_choice cũng nằm trong khoảng này

* 1. **Lựa chọn thuộc tính**

Các thuộc tính được chọn để xây dựng mô hình Content-based filtering:

* Category
* content\_rating
* size
* avg\_rating
* rating\_count

1. **Xây dựng mô hình**
   1. **Content-based filtering**
      1. **Tf-Idf and Cosine similarity**

Ý tưởng: Dựa trên cột “description” và các cột khác trong mỗi game để tìm xem game nào có nội dung gần giống nhau để tiến hành gợi ý do người dùng.

**TF-IDF (*term frequency – inverse document frequency*)**

Là một thuật toán thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản. Trong đó:

* TF (*term frequency*): dùng để ước lượng tần xuất xuất hiện của từ trong văn bản. Nếu từ đó xuất hiện càng nhiều thì trọng số càng cao. Nó được tính bằng:

TF(t, d) = ( số lần từ t xuất hiện trong văn bản d) / (tổng số từ trong văn bản d)

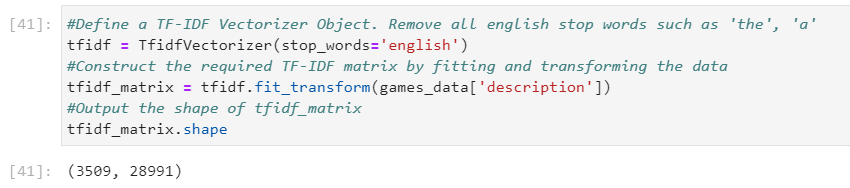
* **IDF (**inverse document frequency**) tần số nghịch đảo của từ trong tập văn bản. Tính IDF để giảm giá trị của những từ phổ biến**

*IDF(t, D) = log\_e( Tổng số văn bản trong tập mẫu D/ Số văn bản có chứa từ t )*

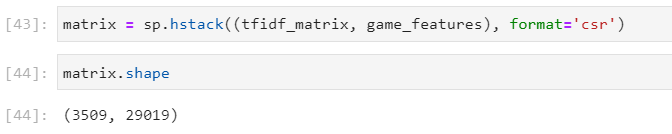
* Giá trị TF-IDF

*TF\_IDF(t, d, D) = TF(t, d) \* IDF(t, D)*

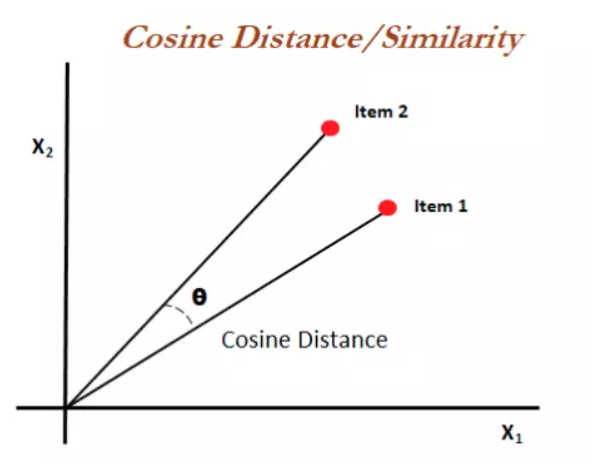
Những từ có TF\_IDF là những từ có nội dung liên quan càng lớn. Trong bài tôi tạo ma trận TF\_IDF bằng thư viện sklearn



Sau khi tạo lập ma trận TF-IDF của description, tôi có ma trận có kích thước [3509 x 28991] kết hợp thêm các thuộc tính “category”, “content\_rating”, “size”, “avg\_rating”, “rating\_count”, ma trận mở rộng có kích thước [3509 x 29019].



Tiếp theo, áp dụng cosine similarity đế tính ra ma trận độ tương đồng. Kết quả là tôi có thể chọn ra TOP các game dựa trên số điểm tương đồng.



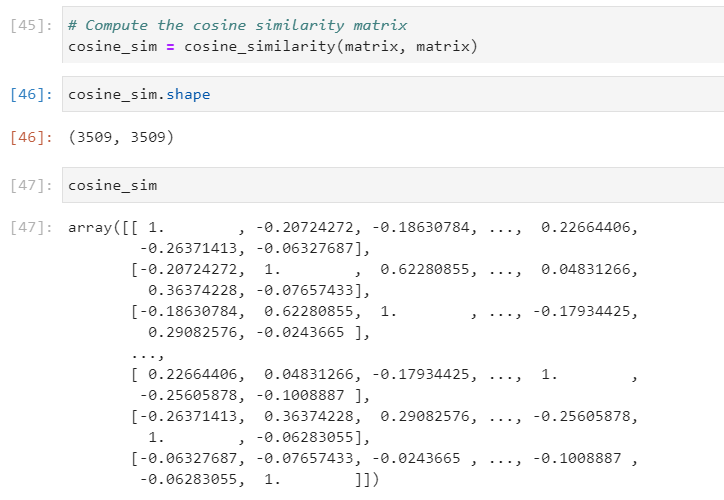
**Cosine Similarity**

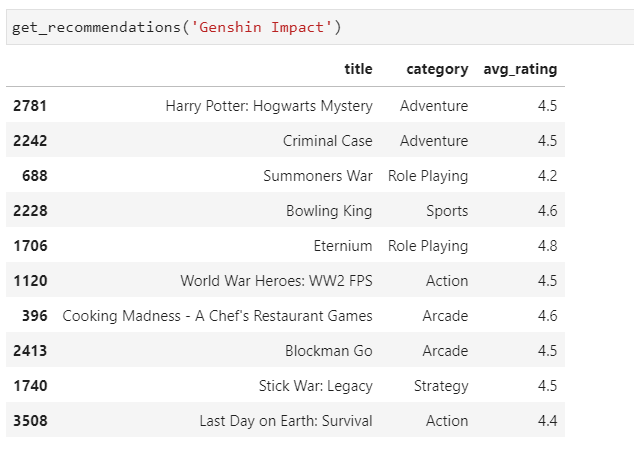
Chúng ta có một ma trận có kích thước [3509 x 29019] với mỗi game là một vector, chúng ta có tổng cộng 3509 vector. Áp dụng công thức tính khoảng cách

Trong đó , n = 29019

* Kết quả của công thức nằm trong khoảng từ -1 đến 1
* Như vậy nếu giá trị Cosine Similarity tiệm cận về 1 có nghĩa rằng 2 game có độ tương đồng lớn và ngược lại.

Sử dụng cosine\_similarity của thư viện sklearn





Vấn đề:

* Các nhà phát triển trên Google Play thường dùng “Description” để mô tả nội dụng bản cập nhật của họ chứ không phải hoàn toàn là nội dụng trò chơi. Vì vậy, dẫn đến việc gợi ý nội dung không thật sự tương đồng.
* Không áp dụng với tập dữ liệu game có ngôn ngữ khác nhau.
  1. **Collaborative filtering**

Collaborative Filtering là một phương pháp gợi ý sản phẩm với ý tưởng chính dựa trên các hành vi của các users khác (collaborative) cùng trên một item để suy ra mức độ quan tâm (filtering) của một user lên sản phẩm. Việc suy ra này được thực hiện dựa trên Similarity matrix đo độ giống nhau giữa các users.

* + 1. **Các phương pháp đánh giá mô hình**
       1. MAE

MAE là một phương pháp đo lường sự khác biệt giữa hai biến liên tục. Giả sử rằng X và Y là hai biến liên tục thể hiện kết quả dự đoán của mô hình và kết quả thực tế. tôi có độ đo MAE được tính theo công thức sau:

|  |  |  |
| --- | --- | --- |
| *Dự đoán* | *Thực tế* | *Lỗi* |
| 5 | 3 | 2 |
| 4 | 1 | 3 |
| 5 | 4 | 1 |
| 1 | 1 | 0 |

MAE = (2+3+1+0)/4 = 1.5

* + - 1. RMSE

RMSE của một phép ước lượng là căn bậc hai trung bình của bình phương của sai số, tức là sự khác biệt giữa các giá trị được mô hình dự đoán và gía trị thực. RMSE là một hàm rủi ro, tương ứng với giá trị kỳ vọng của sự mất mát căn bậc hai sai số bình phương.

|  |  |  |
| --- | --- | --- |
| *Dự đoán* | *Thực tế* | *Lỗi bậc hai* |
| 5 | 3 | 4 |
| 4 | 1 | 9 |
| 5 | 4 | 1 |
| 1 | 1 | 0 |

RMSE = = 1.87

* + - 1. Top-N Hit Rate

Khi thực hiên gợi ý cho người dùng, tôi tạo ra một danh sách n các sản phẩm. Top-N Hit Rate đo lường tỉ lệ người dùng thích, mua hoặc đánh giá sản phẩm trong danh sách. Top-N Hit Rate càng cao, mô hình càng tốt

* + - 1. Average Reciprocal Hit Rank

Average Reciprocal Hit Rank là một biến thể

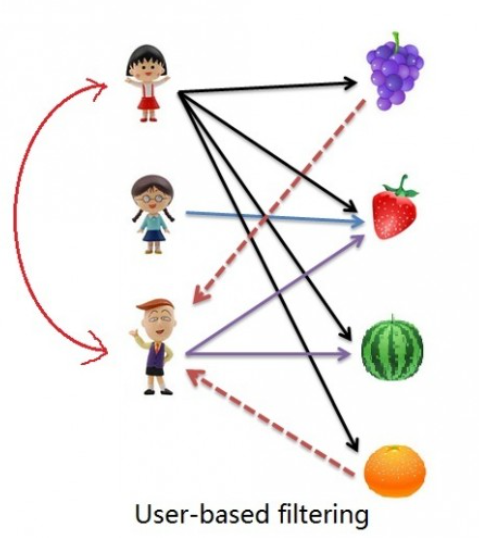
* Sự khác biệt là chúng tôi tổng hợp đối ứng của xếp hạng của mỗi lần mua hoặc đánh giá
* Nó cho thấy vị trí của các sản phẩm thật sự được đánh giá trong danh sách
* Mô hình sẽ thành công hơn nếu nó gợi ý được nhiều sản phẩm đứng đầu danh sách hơn là cuối danh sách
* Vì nó tính đến thứ hạng của danh sách gợi ý nên ARHR càng cao càng tốt

|  |  |
| --- | --- |
| Rank | Reciprocal |
| 3 | 1/3 |
| 2 | 1/2 |
| 1 | 1 |

* + 1. **User-Based Collaborative Filtering**

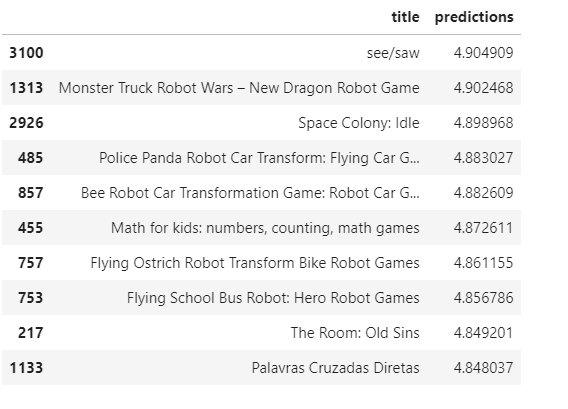
Lọc cộng tác dựa trên người dùng - User-Based Collaborative Filtering dựa trên sự tương đồng của người dùng. Nếu A và B cùng chơi một game, có khả năng người này cũng sẽ B cũng sẽ thích những game khác mà A chơi và ngược lại.

Trong thực tế số người chơi lúc nào cũng đông hơn số đầu game và đa số mọi người thường chỉ chơi một game trong một khoảng thời gian dài nên hướng tiếp cận này đòi hỏi lượng dữ liệu người dùng là rất lớn.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **NMF** | **SVD** | **SVD++** | **Co-Clustering** |
| **RMSE** | 1.5245 | 1.37 | 1.397 | 1.513 |
| **MAE** | 1.3029 | 1.37 | 1.179 | 1.278 |
| **Hit Rate** | 0.0001982 | 0.0004 | 0.00041 | 0.0002 |
| **Cumulative Hit Rate (rating >= 4)** | 0.00028 | 0.0006 | 0.000597 | 0.00025 |
| **Average Reciprocal Hit Rank** | 0.0000587 | 0.00013 | 0.00013 | 0.000062 |

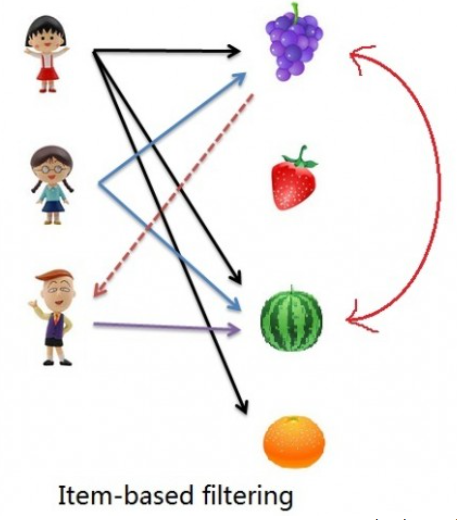
SVD và SVD++ cho kết quả tốt nhất và gần tương đương nhau, RMSE của SVD thấp hơn SVD++ nhưng còn cao. Do mỗi game chỉ khoảng 100 bình luận và ít có sự giao nhau nên kết quả chưa thật sự tốt.



* + 1. **Item-Based Collaborative Filtering**

Lọc cộng tác dựa trên sản phẩm - Item-Based Collaborative Filtering tiếp cận theo hướng tính toán sự tương đồng giữa các sản phẩm rồi gợi ý sản phẩm gần giống sản phẩm yêu thích của một ai đó. So với User-Based Collaborative Filtering có các ưu điểm:

* Vì số lượng items thường nhỏ hơn số lượng users, Similarity matrix trong trường hợp này cũng nhỏ hơn nhiều, thuận lợi cho việc lưu trữ và tính toán ở các bước sau.
* Vì số lượng phần tử đã biết trong Utility matrix là như nhau nhưng số hàng (items) ít hơn số cột (users), nên trung bình, mỗi hàng của ma trận này sẽ có nhiều phần tử đã biết hơn số phần tử đã biết trong mỗi cột. Việc này cũng dễ hiểu vì mỗi item có thể được rated bởi nhiều users. Kéo theo đó, giá trị trung bình của mỗi hàng ít bị thay đổi hơn khi có thêm một vài ratings. Như vậy, việc cập nhật ma trận Similarity Matrix có thể được thực hiện ít thường xuyên hơn.



|  |  |  |  |
| --- | --- | --- | --- |
|  | **KNN Baseline** | **KNN with Means** | **KNN with Z-Score** |
| **RMSE** | 1.377 | 1.494 | 1.494 |
| **MAE** | 1.144 | 1.287 | 1.287 |
| **Hit Rate** | 0.00162 | 0.000502 | 0.000498 |
| **Cumulative Hit Rate (rating >= 4)** | 0.00236 | 0.000708 | 0.000697 |
| **Average Reciprocal Hit Rank** | 0.000963 | 0.000135 | 0.000133 |

KNNBaseline hoàn toàn áp đảo hai thuật toán còn lại. Nhưng cũng giống như SVD, chất lượng dữ liệu khiến cho RMSE khá cao.



1. **Kết luận**

Trong dự án này, tôi đã phân tích các thuộc tính của game trên Google Play và lựa chọn một số cá thuộc tính cơ bản nhất để xây dựng mô hình “Content-based recommender”. Do những khuyết điểm về mặt thuật toán và dữ liệu của mô hình “Content-based recommender” nên tôi đã xây dựng thêm model theo hướng “Collaborative Filtering” tiếp cận theo “Item-Based” và “User-Base”. Tuy kết quả có cải thiện nhưng vẫn chưa đạt đến mức có thể triển khai thực tế.

1. **Định hướng cải tiến**

* Thu thập thêm dữ liệu để triển khai mô hình “Association Rule” dựa trên hành vi tiêu dùng của khách hàng.
* Thử triển khai các mô hình của Deep Learning như “Restricted Boltzmann Machine”