Session-Based Recommendation System for E-commerce: A Comparative Study of Algorithmic Approaches to Product Matching

Student: Andrea Battaglia
Link to the code: “https://colab.research.google.com/drive/1_prPg6lbwCJyityleWEhFxmhSjXf7zMl”

Abstract

This project analyzes e-commerce session data to understand user behavior and enhance product recommendation systems. After cleaning and formatting the data, an exploratory analysis is conducted to examine page visit durations and assign an interest score based on user interactions (viewing, adding to cart, purchasing) and time spent on each page.
To determine product similarity, two distinct approaches are employed. The first method leverages Jaccard Similarity, approximated via MinHashing, to analyze correlations between product categories. Additionally, Cosine Similarity is applied to normalized interest scores to identify closely related products. The second approach utilizes Locality-Sensitive Hashing (LSH) with SimHash, leveraging random hyperplanes to generate binary signatures and efficiently estimate product similarity in large-scale datasets.
The performance of both algorithms is compared to evaluating the trade-off between accuracy and computational efficiency. The results indicate that the first method provides more precise recommendations, while the LSH-based approach, though slightly less accurate, proves highly scalable and well-suited for massive datasets.

1 Introduction

Recommendation systems play a crucial role in e-commerce platforms, directly impacting user experience and sales. Providing relevant product suggestions not only increases conversion rates but also enhances user engagement and retention. However, with ever-growing and dynamic datasets, it is essential to adopt strategies that balance recommendation accuracy with computational efficiency.
This project aims to develop a product’s page recommendation system using two distinct approaches: one focused on precision, based on Cosine Similarity, and another optimized for scalability, leveraging Locality-Sensitive Hashing (LSH) combined with SimHash.
The analysis begins with data preparation and cleaning, focusing on user interactions with products during browsing sessions. A key aspect of this process is assigning an interest score to each viewed product, determined by the type of interaction (viewing, adding to cart, purchasing) and the time spent on the page. This score provides a more effective representation of user interest in a given product.
To enhance recommendation quality, both approaches incorporate product category similarity, estimated using Jaccard Similarity approximated via MinHashing. This technique helps identify categories frequently explored together by users, narrowing the scope of product comparisons and improving the relevance of recommendations.
The two proposed approaches are as follows:

1. **Cosine Similarity-Based approach:** compares products within the same or related categories by leveraging normalized interest scores to estimate similarity. While this method provides highly accurate recommendations, it becomes less scalable with large datasets.
2. **Locality-Sensitive Hashing (LSH) with SimHash:** reduces data dimensionality by using random hyperplanes to generate binary signatures representing user-product interactions. The Hamming Similarity between these signatures enables the efficient identification of similar products, even in very large datasets.

Finally, the two approaches are compared to evaluate the trade-off between accuracy and scalability.
This study demonstrates how integrating similarity techniques with advanced hashing methods can enhance recommendation system performance, achieving an optimal balance between recommendation quality and computational efficiency.

2 Literature Review

2.1 Recommendation Models

Recommendation systems are essential tools for personalizing user experiences on digital platforms, suggesting products or content based on past preferences. There are several approaches to building an effective recommendation system, including Content-Based and Collaborative Filtering.

Content-Based system

Content-Based system is one of the most common techniques used in building recommendation systems. This approach relies on analyzing the attributes of items and comparing them with user preferences. For example, in an e-commerce context, content-based system might recommend products similar to those a user has already interacted with, based on their features such as product type, brand, price, or color. The strength of this model lies in its ability to personalize recommendations based on the user's past behavior. However, content-based methods have limitations in terms of diversity, as they tend to suggest items similar to those the user has already shown interest in, potentially overlooking different products that might also appeal to them.

Collaborative Filtering

Collaborative filtering takes a different approach, focusing on user behavior. Rather than relying on the attributes of items, this model makes recommendations based on patterns of user interactions with products. Collaborative filtering can be divided into two types:

- **User-based collaborative filtering**, which suggests items liked by similar users. This approach assumes that if User A and User B have shown interest in the same products in the past, they are likely to appreciate similar products in the future.

- **Item-based collaborative filtering**, which suggests products similar to those the user has already interacted with. If a user has purchased or viewed a specific product, the system will recommend other items that have been liked or viewed by other users who interacted with the same product.

While collaborative filtering is powerful in detecting latent relationships between users and products, it presents some challenges. One of the main difficulties is scalability, especially when the volume of users and items increases. As the number of users and products grows, the computational cost of generating recommendations can become prohibitive. Additionally, collaborative filtering suffers from the cold start problem: when a new user or product enters the system, there is not enough data on past interactions to generate accurate recommendations. This limits the model's effectiveness in the initial stages, when there isn't sufficient information to make reliable predictions.

Hybrid approaches

To overcome the limitations of content-based and collaborative filtering models, hybrid approaches have emerged. These models combine different recommendation techniques to improve overall performance. For example, a hybrid system might integrate content-based filtering with collaborative filtering, leveraging the strengths of both approaches while reducing their individual weaknesses. The goal of hybrid models is to provide more accurate and efficient recommendations by combining different sources of information, thereby improving personalization and scalability.

2.2 Similarity algorithms

The foundation of many recommendation systems lies in the ability to measure similarity between items (or users) effectively. These measurements help to identify products or content that are relevant to each other and allow the system to make accurate suggestions. In this chapter, we explore several similarity algorithms used in recommendation systems, with a particular focus on how they are applied in large datasets. We will cover Jaccard Similarity, MinHashing, Cosine Similarity, Locality-Sensitive Hashing (LSH) using hyperplanes, and SimHash with Hamming Similarity.

Jaccard Similarity

Jaccard Similarity is one of the simplest and most widely used metrics for measuring similarity between two sets. It calculates the ratio of the intersection of two sets to their union. The value of Jaccard Similarity ranges from 0 to 1, where 0 indicates no similarity (no common elements) and 1 indicates identical sets (all elements are shared).
In the context of recommendation systems, Jaccard Similarity is particularly useful for measuring the similarity between two sets of items, such as products purchased by two users or categories of items viewed by different users. For example, if two users view the same products, they are considered more similar based on Jaccard's formula. The higher the number of shared items between the sets, the higher the similarity score.
One limitation of Jaccard Similarity is that it doesn't account for the frequency of interactions. For instance, if two users purchase the same items but at different frequencies, the similarity score will be the same regardless of the number of purchases. Despite this, Jaccard is a straightforward and effective method for identifying shared interests.

MinHashing

MinHashing is a technique used to efficiently approximate Jaccard Similarity. While Jaccard computes the exact overlap between sets, MinHashing reduces the computational cost by generating a compact representation of sets, called "signatures". This is particularly useful for working with large datasets, where computing Jaccard Similarity directly would be computationally expensive.
The key idea behind MinHashing is to hash the elements of the sets using multiple hash functions, then compare the signatures to estimate the similarity between the sets. By using these hash functions, MinHashing can provide an approximation of the Jaccard Similarity with much lower computational effort. This technique is widely used in scenarios where scalability and efficiency are important, such as in systems handling large numbers of products or users.
MinHashing is especially beneficial in large-scale recommendation systems because it allows for quick comparisons between large sets of data. However, the trade-off is that the similarity computed via MinHashing is an approximation, and its accuracy depends on the number of hash functions used.

Cosine Similarity

Cosine Similarity is a metric used to measure the cosine of the angle between two vectors in a multi-dimensional space. It is widely used in recommendation systems to measure the similarity between items based on user interactions, especially when data is represented in a vectorized form. Unlike Jaccard, which only compares the presence or absence of items, Cosine Similarity can takes into account the frequency or magnitude of interactions (e.g., how many times a product was viewed or purchased).
The formula for Cosine Similarity calculates the cosine of the angle between two vectors, which can be interpreted as a measure of how aligned the vectors are. A value closer to 1 means the vectors are highly similar (more aligned), while a value closer to 0 means they are dissimilar. In the context of recommendation systems, Cosine Similarity can be applied to user-item interaction matrices, where the vectors represent the interaction frequency of a user with various items.
Cosine Similarity has the advantage of normalizing the length of vectors, making it insensitive to the overall magnitude of the interactions (e.g., a user who views a product 100 times is treated the same as a user who views it once). This makes it especially useful when comparing users or items with varying levels of activity.

Locality-Sensitive Hashing (LSH) using random hyperplanes

Locality-Sensitive Hashing (LSH) is a technique used to perform approximate nearest neighbor search in high-dimensional data. LSH uses hash functions that preserve the similarity between data points: similar items are more likely to be mapped to the same hash bucket. This reduces the computational cost of finding similar items, which is especially beneficial when working with large datasets.
In the case of LSH using hyperplanes, the data points are projected into a lower-dimensional space using randomly generated hyperplanes. A hyperplane is a subspace that divides the space into two regions, and each data point is assigned to a region based on which side of the hyperplane it lies. By using multiple hyperplanes, the technique can create a compact binary signature for each data point, which can then be compared efficiently to determine similarity.
LSH with hyperplanes is highly effective for large-scale datasets because it enables quick comparison between data points, even in high-dimensional spaces. The trade-off is that LSH only provides approximate results, meaning it may not always find the most similar items but will find those that are similar with high probability.

SimHash and Hamming Similarity

SimHash is a technique that leverages the generation of compact binary representations (signatures) for high-dimensional data, similar to LSH. However, SimHash uses a different approach, based on the concept of "hashing" to map high-dimensional vectors to low-dimensional binary codes. The key advantage of SimHash is its ability to reduce the dimensionality of the data while preserving the pairwise similarity between items.
Once the binary signatures are generated, similarity between items can be measured using Hamming Distance, which counts the number of differing bits between two binary strings. The fewer the differing bits, the more similar the items are. SimHash with Hamming Similarity is highly efficient because binary strings are easy to compare. It is especially useful in scenarios where large datasets need to be processed quickly, and exact similarity is not as crucial as finding near-duplicates or similar items.
SimHash can be used in large-scale recommendation systems because it can offers a good balance between computational efficiency and accuracy. However, the quality of the recommendations depends on the length of the binary signatures and the effectiveness of the hash functions used.

3 In-depth Dataset Exploration

The analyzed dataset contains 605110 records, each representing an interaction between a user and a product page on an eCommerce platform. The data comes from an eCommerce site located in an Asian country and was collected on October 1, 2019, between 00:00 and 12:00 UTC+0 (thus covering only a 12-hour interval).

3.1 Dataset structure

Each row in the dataset corresponds to an action performed by a user on the platform, recorded with the following key information:

- **event_time**: timestamp of the event. There are 35,519 unique values, suggesting that many events occur simultaneously on the site.
- **event_type**: the type of recorded interaction. There are three possible values:
    - *view*: The product was viewed by the user.
    - *cart*: The product was added to the cart.
    - *purchase*: The product was purchased.
- **product_id**: unique identifier for the product. The dataset contains 50,998 distinct products.
- **category_id**: unique identifier for the product category. There are 541 different categories.
- **category_code**: textual name of the category. However, this field is highly incomplete, with 189703 missing values, indicating that many products lack a textual categorization.
- **brand**: the product's brand. A total of 2085 different brands were recorded, but the field has 86790 missing values, meaning many products do not have an associated brand.
- **price**: the product's price converted in USD from the local currency. The dataset shows 14835 unique values, and it was observed that the price of a single product can fluctuate slightly throughout the day, indicating possible dynamic pricing strategies.
- **user_id**: unique identifier for the user. There are 105014 distinct users.
- **user_session**: unique identifier for the user session. A total of 140635 unique sessions were recorded, indicating that some users initiated multiple sessions within the observed time frame.

Field	Unique Count	Missing Count
event_time	35822	0
event_type	3	0
product_id	51188	0
category_id	541	0
category_code	123	191292
brand	2093	87492
price	14876	0
user_id	105887	0
user_session	141891	0

Table 1: Summary of fields with counts of unique and Missing values

An analysis of user activity during different hours reveals a particular pattern. Users appear to be more active during nighttime hours than expected. This phenomenon is due to the data being collected from a country located in a time zone between GMT+4 and GMT+7, which explains the apparent anomaly.

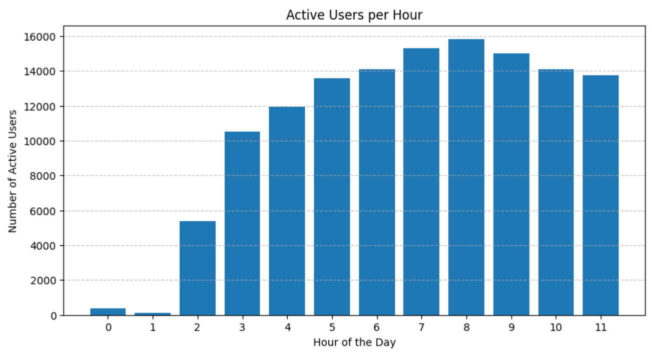

Figure 1: Number of active users per hour of the day

Analysis of User Session characteristics

Each user can initiate multiple sessions and, within each session, can view multiple products. Analyzing the distribution of the number of pages visited per session and per user, it is observed that a significant portion of users explores only one page, with an exponential decline in the number of products viewed. Specifically, it is naturally more common to find at least two distinct pages visited by the same user rather than within the same session.

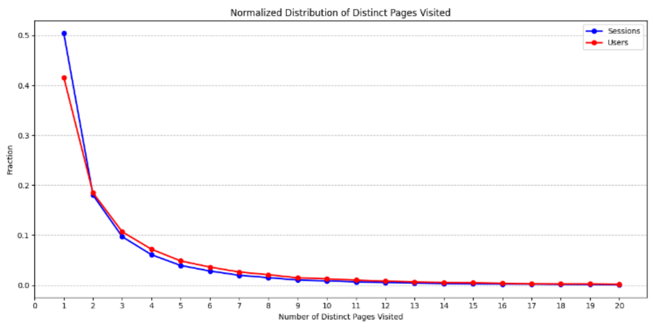

Figure 2: Normalized distribution of the number of pages visited per Session and per User

A key aspect of the analysis concerns the user's dwell time on a page. Viewing duration is calculated using the timestamp between one event and the next within the same session. However, the time spent on the last page of the session cannot be determined.
The analysis of the page dwell time distribution shows a long right tail, indicating that, in some cases, users remain on a page for an unusually long time. This may be due to external distractions that cause the page to stay open longer than necessary. Although it is impossible to identify pages where dwell time was influenced by external distractions, views with a dwell time beyond the 99th percentile of the distributions were excluded to avoid data distortions.

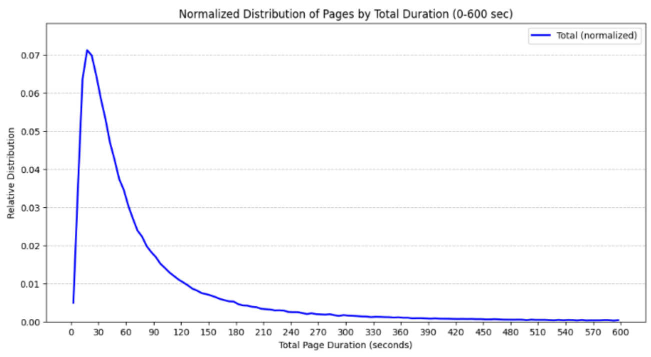

Figure 3: Normalized distribution of pages total View duration

Relationship between Interaction, Dwell time and Interest

Another step was grouping repeated views of the same page by the same user. In these cases, grouping was done by retaining only the most significant recorded event for the user-product pair, according to the hierarchy: *purchase* > *cart* > *view*.
It makes sense to hypothesize that a user more interested in a product tends to stay on its page longer. This hypothesis is confirmed by comparing the distributions of dwell time based on interaction type:

- Simple view
- Add to cart
- Purchase

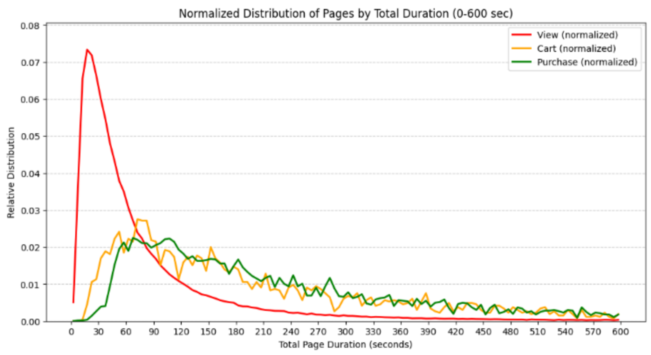

Figure 4: Normalized distribution of View duration by Interaction type

As expected, "cart" and "purchase" events show similar distributions, as they both indicate strong user interest in the product page. Both distributions are significantly different from simple views. For this reason, they have been grouped into a single distribution representing interactions from users who have shown definite interest.

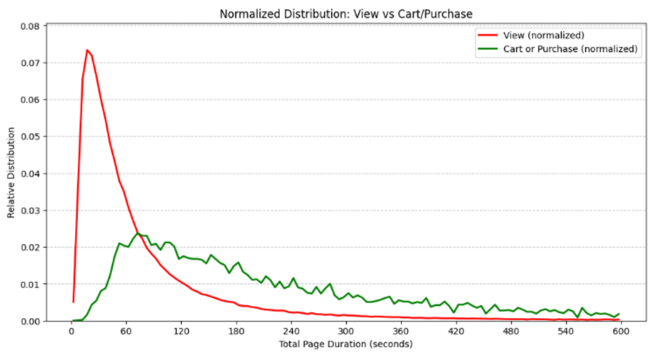

Figure 5: Comparison of View and Cart/Purchase Distributions

One might assume that users tend to spend more time evaluating expensive products. However, when analyzing the distribution of dwell time based on price ranges, no significant differences emerge between budget and premium products. This suggests that browsing behavior is not strongly influenced by price and that this variable can therefore be disregarded when evaluating dwell time.

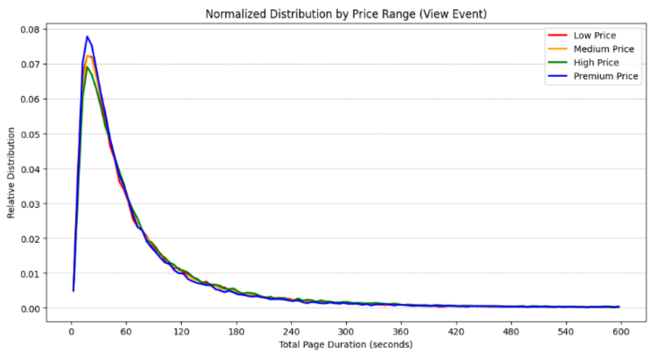

Figure 6: Distribution of View duration by Price range (View event)

Distribution model and Interest score assignment

According to existing literature, product view time follows a lognormal distribution, which the data in this case appears to confirm. The approximation to a normal distribution after a logarithmic transformation is quite good, although the right tail remains slightly influenced by anomalous sessions.

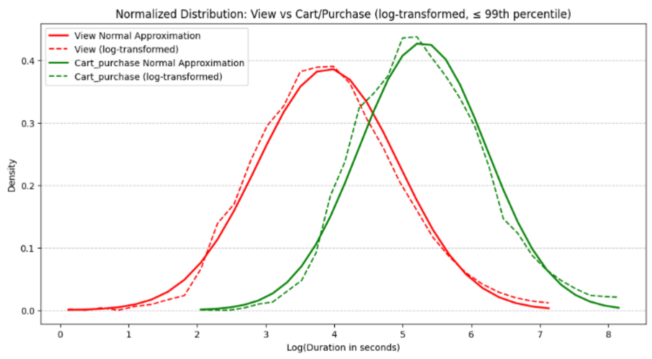

Figure 7: Normal approximation of Log-transformed distributions for View and Cart/Purchase

Based on these distributions, a model for assigning an interest score to each user and product has been developed. The core concept behind score assignment relies on inference, specifically quantiles:

Rating 1 – No Interest (below 0.01 quantile of View): the user exited almost immediately, likely landing on the page by mistake or realizing instantly that it wasn't relevant. Their dwell time was too short to indicate any real interest.

Rating 2 – Neutral Interest (between 0.01 and 0.05 quantile of View): the user briefly checked the page but left quickly after glancing at key details. This suggests they realized the product wasn't what they were looking for without fully engaging.

Rating 3 – Moderate Interest (between 0.05 quantile of View and 0.02 quantile of Cart-Purchase): the user showed clear interest, staying on the page long enough to explore key features. However, something - perhaps price or product details - prevented them from fully committing.

Rating 4 – Strong Interest (above 0.02 quantile of Cart-Purchase): the user spent a significant amount of time on the product page, similar to those who added it to the cart or purchased it. This suggests they carefully evaluated the product and were highly engaged, even if they didn't take action.

Rating 5 – Definite Interest (Cart/Purchase): the product was either added to the cart or purchased. This is the strongest signal of interest, as the user took a concrete action confirming their intent.

These scores will be fundamental for the next phase of constructing the recommendation system based on product similarity. Additionally, for all user-product pairs where dwell time could not be calculated because the product pages were the last visited in the session, a Rating of 4 is assigned. This is because there is no evidence suggesting that these views were brief.

Interest Showed	Event	Range Time (Log)	Range Time (Seconds)
1	View	(0, 1.53)	(0, 4.61)
2	View	(1.53, 2.23)	(4.61, 9.27)
3	View	(2.23, 3.38)	(9.27, 29.38)
4	View	(3.38, ∞)	(29.38, ∞)
5	Cart/Purchase	All	All

Table 2: Criteria for assigning Interest scores based on Duration and Event

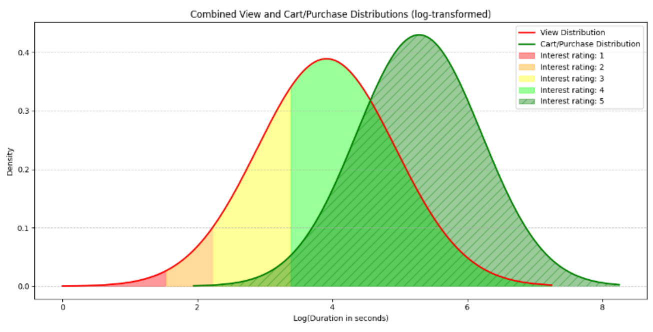

Figure 8: Visualization of Interest Scores based on Duration and Event

4. Application of the algorithms to the dataset

Once interest scores were assigned to user-product pairs, the recommendation system aimed to suggest products similar to those the user was currently viewing. The underlying assumption is that the users are looking at a product they are interested in. Therefore, the goal is to identify similar products based on users' interests.

4.1 Considerations and Dataset preparation before applying the algorithms

To determine product similarity, Cosine Similarity has to be applied after normalizing the data. However, due to the nature of the interest scores (Figure 8), where most ratings are above 3, a different normalization strategy was required. The high frequency of scores above 3 aligns with the assumption that opening a product page already indicates a degree of interest.

Centering Interest Scores

Instead of scaling values to a predefined range, the scores were centered around the neutral interest level (rating 2). So, the transformation was performed by subtracting 2 from all ratings. By centering the scores this way, positive values indicate real interest, while negative or zero values suggest no or low engagement. This normalization allows for more meaningful similarity calculations when comparing user interactions with different products.

Limiting computation: identifying candidate pairs

Performing Cosine Similarity across all possible product combinations would be computationally infeasible due to the vast number of comparisons required. Instead, an intelligent filtering strategy was implemented to restrict similarity calculations to meaningful candidate pairs.
A common approach would be to cluster products based on attributes such as brand, type, price, and color. However, in this dataset, these attributes were not consistently available for all products. Instead, the predefined product categories provided in the dataset were used as an initial basis for clustering.
Although product categories can serve as a useful grouping mechanism, they are not always perfectly distinct. There can be overlaps between different categories, making it necessary to first validate their reliability. This was achieved by measuring category similarity, based on how frequently users viewed products from different categories within the same session.

4.2 Measuring Category Similarity with Jaccard Similarity and MinHashing

To verify category reliability, Jaccard Similarity was used to determine how often users explored products from two different categories within the same session.

Defining positive Category interactions

To compute category similarity, the interest ratings were aggregated at the session level. A category was considered positively interacted with in a session if at least one product from that category received an interest score of 3 or higher.
Once category interactions were defined, Jaccard Similarity was used to measure the proportion of sessions where a user showed interest in both categories out of the total sessions where interest was shown in at least one of the two.
To efficiently compute Jaccard Similarity at scale, the MinHashing algorithm was implemented. This allowed for an approximate, but more efficient, similarity computation.

Steps in MinHashing implementation

1. **Session mapping**: for each session ID was assigned a unique numeric identifier, ranging from 0 to the total number of unique sessions minus 1.
2. **Permutation generation**: 200 random permutations of the new unique numeric identifier for the sessions were generated using appropriate hash functions, ensuring an efficient randomized transformation of session IDs.
3. **MinHash Signature computation**: for each category, the smallest session ID across all permutations was recorded. This effectively reduced the category interaction set into a compact signature vector.
4. **Similarity calculation**: category pairs were compared based on their MinHash signatures. The percentage of matching minimum values across all permutations was used as an approximation of the Jaccard Similarity.

Results: Category Similarity scores

Some examples of the most similar category pairs identified using MinHashing and Jaccard Similarity are shown in Table 3. These pairs highlight how often users viewed products from these categories together within the same session.

Category_1_Code	Category_2_Code	Similarity
furniture.kitchen.table	**sport.tennis**	**0.25**
apparel.jacket	**apparel.trousers**	**0.23**
accessories.bag	**accessories.wallet**	**0.22**
computers.cpu	**computers.motherboard**	**0.2**
apparel.shoes.slipons	**apparel.shoes.moccasins**	**0.18**
auto.accessories.player	**electronics.audio.subwoofer**	**0.17**
country_yard.hammok	**kids.swing**	**0.16**

Table 3: Some of the most similar Product categories based on User behavior

Several of these category pairs make intuitive sense:

- "furniture kitchen table" & "sport tennis": a seemingly unexpected pairing, but it makes sense when considering ping pong tables. While the tables themselves may be categorized under “furniture kitchen table”, the ping pong paddles and accessories are listed under “sport tennis”. Users searching for a ping pong table may also browse for paddles, balls, or nets, leading to this observed similarity;
- "apparel jacket" & "apparel trousers”: users browsing jackets may also be looking for trousers, as both are often considered part of a complete outfit;
- "accessories bag" & "accessories wallet": these are frequently purchased together, especially in fashion-related shopping;
- "computers cpu" & "computers motherboard": when building or upgrading a PC, buyers frequently consider both CPUs and motherboards, since compatibility is essential;
- "apparel shoes slipons" & "apparel shoes moccasins": slip-ons and moccasins are both casual shoes, often made of similar materials and used for similar purposes. Users searching for one type of shoe might also consider the other as an alternative;
- "auto accessories player" & "electronics audio subwoofer": these items are often searched together as part of car audio system upgrades. A user looking for a car stereo (player) may also be interested in a subwoofer to enhance the audio experience;
- "country yard hammock" & "kids swing": both categories contain outdoor seating and relaxation items. A garden hammock and a children’s swing serve similar recreational purposes, often placed in yards or outdoor spaces. Users interested in one may naturally consider the other.

These results suggest that product categories are not always clearly distinct in user browsing behavior. Some categories frequently overlap, and incorporating these relationships into the recommendation system can improve the relevance of suggested products.

4.3 Applying Cosine Similarity to find similar products

After validating the category similarity using Jaccard Similarity and MinHashing, the next step involved applying Cosine Similarity to determine product relationships. Instead of comparing all possible product pairs, which would be computationally infeasible, the analysis was restricted to products within the same category or belonging to two previously identified similar categories.
By following this approach, the number of required pairwise comparisons was drastically reduced, maintaining efficiency while ensuring that only relevant product pairs were compared.

Data filtering before applying Cosine Similarity

Before computing Cosine Similarity, some data preprocessing steps were performed to ensure meaningful and reliable results:

1. Users who viewed only one product were removed, as they do not provide valuable information for computing product similarities. These users do not generate any cross-product relationships, making their data irrelevant for similarity computations.
2. Products that were viewed by fewer than three distinct users were also filtered out. This ensures that recommendations are based on a minimum level of user interaction, avoiding unreliable similarity calculations due to extremely sparse data.

Applying these filters ensured that only consistent and meaningful product comparisons were included in the similarity calculations.

Reducing computation with a structured approach

A naïve approach to computing product similarity would involve comparing every product with every other product. Given the large number of products in the dataset, this would result in an extremely high number of comparisons, making the process computationally expensive.
Instead, by leveraging category information and the previously computed category similarity scores, the cosine similarity calculations were restricted to:

1. Products within the same category (assumed to be inherently similar).
2. Products from two different categories that were found to be similar in the previous analysis (Table 3).

This optimization significantly reduced the number of comparisons, focusing computational resources only on high-probability candidate pairs rather than exhaustively evaluating all possible product relationships.

Analysis of Product Recommendations

Below there are few examples of Cosine Similarity based recommendations for three selected random products. Each table provides the most similar products based on user interactions, showing the category, brand, price, and similarity score.

Example 1: Laptop recommendations

Selected Product:

- Product ID: 1307112
- Category: computers.notebook
- Brand: HP
- Price: €1146.14

product_id	category_code	brand	price	Similarity
1305843	computers.notebook	dell	1183.81	0.50
1306751	computers.notebook	lenovo	1415.71	0.27
1306361	computers.notebook	acer	1152.93	0.26
1307329	computers.notebook	hp	1145.20	0.23
1307211	computers.notebook	hp	1201.83	0.23
1306424	computers.notebook	hp	1003.63	0.23
1304106	computers.notebook	lenovo	978.07	0.22
1306249	computers.notebook	hp	1029.37	0.22
1306422	computers.notebook	hp	1019.07	0.22
1307080	computers.notebook	asus	957.81	0.19

Table 4: Recommended laptops based on Cosine Similarity

The recommended products are all notebooks from well-known brands, including Dell, Lenovo, Acer, and HP. The price range remains relatively consistent, with all products falling within a similar cost bracket. A notable aspect of these recommendations is that all the suggested laptops run Windows, ensuring relevance for users looking for an alternative device within the same operating system ecosystem. The highest similarity score (0.50) is associated with a Dell notebook.

Example 2: Sneakers recommendations

Selected Product:

- Product ID: 28713737
- Category: apparel.shoes.keds
- Brand: Nike
- Price: €67.75

product_id	category_code	brand	price	Similarity
28703801	apparel.shoes.keds	nike	61.57	0.32
28715975	apparel.shoes.keds	adidas	111.84	0.24
28703920	apparel.shoes.keds	nike	55.39	0.21
28714060	apparel.shoes.keds	nike	55.39	0.21
28714586	apparel.shoes.keds	adidas	58.38	0.18
28703972	apparel.shoes.keds	nike	47.16	0.18
28704388	apparel.shoes.keds	asics	51.48	0.18
28705312	apparel.shoes	respect	23.17	0.17
28704107	apparel.shoes.keds	nike	61.57	0.16
28715700	apparel.shoes.keds	puma	94.98	0.16

Table 5: Recommended sneakers based on Cosine Similarity

The recommended products are sneakers from globally recognized brands, including Nike, Adidas, Puma, and Asics. This pattern suggests that users tend to explore options within premium and widely available sportswear brands. The top-ranked recommendation is another Nike sneaker, which aligns with the selected product's brand. Adidas shoes also appear frequently among the suggestions, reinforcing the likelihood that users browsing a Nike product may also be interested in alternatives from another major competitor in the athletic footwear market.

Example 3: Baby stroller recommendations

Selected Product:

- Product ID: 7004810
- Category: kids.carriage
- Brand: Chicco
- Price: €205.90

product_id	category_code	brand	price	Similarity
7005618	kids.carriage	chicco	148.76	0.58
7004045	kids.carriage	chicco	272.84	0.38
7005065	kids.carriage	joie	199.23	0.37
7005810	kids.carriage	chicco	489.05	0.31
7005774	kids.carriage	alis	128.68	0.26
7004807	kids.carriage	coballe	63.99	0.22
7001153	kids.carriage	quinny	566.30	0.18
7004521	kids.carriage	alis	102.94	0.18
7005980	kids.carriage	easywalker	283.15	0.18
7005417	kids.carriage	NaN	391.26	0.17

Table 6: Recommended baby strollers based on Cosine Similarity

The recommended products include baby strollers from Chicco, Joie, and Alis, all falling within the same category. The list is dominated by Chicco models, which aligns with the original product's brand and suggests that users tend to browse multiple options within the same manufacturer. The similarity scores indicate a strong relationship between these products, likely due to shared browsing behavior. However, the price range among recommendations varies significantly, with some models priced well below or above the selected product, suggesting that users may explore a wider range of budget options when purchasing baby strollers.

4.4 Applying SimHash after using LSH with random hyperplanes

After testing a Cosine Similarity-based approach, the next step was to develop a more scalable solution capable of handling much larger datasets with significantly more users than more products. This was achieved by using SimHash and the implementation of Locality-Sensitive Hashing (LSH) with random hyperplanes.

Generating binary Signatures for Products

Instead of directly computing Cosine Similarity between high-dimensional vectors, the algorithm projects product vectors onto random hyperplanes. Each hyperplane acts as a decision boundary, determining which side of the space a product falls into. This technique transforms each product into a binary signature, reducing the complexity of similarity computations.
The process of generating signatures in this case follows these steps:

1. **Random Hyperplane generation:** 200 hyperplanes were generated, each defined by a randomly chosen normal vector. Each of these hyperplanes divide the high-dimensional rating space into two parts.
2. **Projection of Product vectors:** for each product vector, the dot product with each hyperplane's normal vector was computed. If the result was positive, a 1 was assigned in the signature; otherwise, a 0 was assigned.
3. **Binary Signature construction:** each product received a 200-bit binary signature, with each bit representing which side of the hyperplane the product vector landed on. If two products share the same bit at the same position, it means they fall into the same region of space for that hyperplane.

For example, if the third position of two product signatures is identical, it means both products fell on the same side of the space defined by the third random hyperplane chosen. The probability that two products share the same bit for any given hyperplane is directly proportional to their Cosine Similarity, so if their interest vectors are highly similar, they are more likely to fall into the same partitions across multiple hyperplanes.

Applying SimHash with Hamming Similarity

Once the binary signatures were generated, the next step was to apply SimHash to efficiently estimate the similarity between products. Instead of directly comparing long vectors of user-product interactions, the system now worked with fixed-length binary signatures, making the computation significantly more efficient.
To measure similarity, Hamming Distance was used, which counts how many bits differ between two product signatures. A lower Hamming Distance indicates that two products share more matching bits, meaning they are more likely to have similar interest patterns. Since the way these binary signatures were created is directly linked to Cosine Similarity, products with highly similar signatures are also likely to have high Cosine Similarity. This method reduces computational complexity while still preserving meaningful product relationships, making it a more scalable approach for large datasets.

Efficiently reducing comparisons

As in the previous Cosine Similarity approach, comparisons were performed in two structured loops:

1. Within the same Category: products were compared only with other products in the same category, leveraging the assumption that items within a category are naturally more similar.
2. Between similar Categories**:** products from two different but similar categories (identified in Table 3) were also compared. This step ensured that related products, such as jackets and trousers, were considered for recommendations.

By following this structured approach, computational costs were significantly reduced compared to an exhaustive pairwise comparison across all possible product pairs.

Trade-offs and Benefits of the SimHash approach

The key advantage of using SimHash with LSH is that vector comparisons are now performed on fixed-length binary signatures, rather than on vectors proportional to the number of users. This makes the algorithm highly scalable, allowing for efficient similarity computations across more products and more users.
However, there is a trade-off between precision and efficiency:

- Increasing the number of hyperplanes improves precision, as more partitions allow for a better distinction between products.
- Generating more hyperplanes increases computational cost, as the dot product must be calculated for each hyperplane and each product during the signature generation phase.

By selecting 200 hyperplanes, a balance was struck between efficiency and similarity approximation accuracy. This allows the recommendation system to rapidly retrieve similar products while minimizing computational overhead.

Analysis of Product Recommendations with SimHash

Below are examples of product recommendations based on SimHash, applied to the same three selected products from the previous Cosine Similarity analysis. Each table presents the most similar products determined through this method, including their category, brand, price, and similarity score. Unlike the previous approach, where similarity was based on Cosine Similarity, here the similarity score is derived from Hamming Similarity. Its scale is different from that of Cosine Similarity, meaning the numerical values of similarity scores are not directly comparable between the two methods. Despite this difference, the comparisons allow us to observe patterns in the recommended products and assess how well SimHash captures meaningful relationships.

Example 1: Laptop Recommendations

Selected Product:

- Product ID: 1307112
- Category: computers.notebook
- Brand: HP
- Price: €1146.14

product_id	category_code	brand	price	Similarity
1306424	computers.notebook	hp	1003.63	0.66
1307244	computers.notebook	hp	1317.67	0.61
1306296	computers.notebook	asus	746.22	0.61
1307420	computers.notebook	lenovo	2470.72	0.61
1305843	computers.notebook	dell	1183.81	0.60
1307211	computers.notebook	hp	1201.83	0.60
1306484	computers.notebook	hp	1106.59	0.59
1307107	computers.notebook	asus	1282.14	0.59
1307080	computers.notebook	asus	957.81	0.59
1305840	computers.notebook	dell	1209.55	0.59

Table 7: Recommended laptops based on SimHash Similarity

Compared to the previous Cosine Similarity recommendations, 6 out of the 10 recommended laptops appear in both lists. Once again, all suggested laptops are Windows-based, reinforcing the pattern that users tend to compare devices within the same operating system. However, in this case, the price range is broader.

Example 2: Sneakers Recommendations

Selected Product:

- Product ID: 28713737
- Category: apparel.shoes.keds
- Brand: Nike
- Price: €67.75

product_id	category_code	brand	price	Similarity
28712684	apparel.shoes	marcomen	100.65	0.64
28715975	apparel.shoes.keds	adidas	111.84	0.63
28716591	apparel.shoes.keds	baden	59.46	0.61
28703920	apparel.shoes.keds	nike	55.39	0.61
28703785	apparel.shoes.keds	nike	61.57	0.60
28704107	apparel.shoes.keds	nike	61.57	0.60
28716943	apparel.shoes	nexpero	110.43	0.60
28712000	apparel.shoes	NaN	87.52	0.60
28714355	apparel.shoes.keds	nike	102.94	0.59
28718366	apparel.shoes.keds	respect	51.22	0.59

Table 8: Recommended sneakers based on SimHash Similarity

In this case 5 out of the 10 recommended sneakers are shared with the Cosine Similarity result. While Nike remains the dominant brand, some lesser-known brands such as Marcomen, Baden, and Nexpero appear in the recommendations. Despite these differences, the price range remains consistent with the selected product.

Example 3: Baby stroller Recommendations

Selected Product:

- Product ID: 7004810
- Category: kids.carriage
- Brand: Chicco
- Price: €205.90

product_id	category_code	brand	price	Similarity
7004045	kids.carriage	chicco	272.84	0.71
7005618	kids.carriage	chicco	148.76	0.67
7005810	kids.carriage	chicco	489.05	0.64
7005065	kids.carriage	joie	199.23	0.63
7005980	kids.carriage	easywalker	283.15	0.60
7004807	kids.carriage	coballe	63.99	0.60
7005417	kids.carriage	NaN	391.26	0.59
10502145	NaN	edufun	154.19	0.59
7005189	kids.carriage	belecoo	171.84	0.58
10503038	NaN	tools	5.12	0.58

Table 9: Recommended Baby Strollers Based on SimHash Similarity

For the third product, 7 out of the 10 recommendations appear in both Cosine Similarity and SimHash Similarity results lists. As in the previous case, Chicco strollers dominate the list, suggesting that brand loyalty plays a significant role in user browsing behavior.

5. Comparison and evaluation of the algorithms

After implementing both similarity approaches, Cosine Similarity with category-based grouping and SimHash after applying LSH with random hyperplanes, the next step is to evaluate their effectiveness. The primary goal of this comparison is to analyze the trade-off between computational efficiency and accuracy, assessing how much precision is lost when prioritizing scalability.

5.1 Computational Cost Comparison

One of the most significant advantages of the SimHash-based approach is its reduced computational cost compared to the first approach. The first chart (Figure 9) provides a clear visualization of computation time as a function of the number of comparisons performed. The Cosine Similarity algorithm (blue points) requires significantly more processing time, especially as the number of comparisons increases. In contrast, the SimHash-based approach (orange points) consistently achieves faster computations for the same number of comparisons.
This efficiency gain is further quantified in Table 10, which compares total computation time across both methods. Despite performing the same number of comparisons, SimHash required approximately five times less time than the Cosine Similarity approach. Such a reduction in computational time makes SimHash a highly scalable solution, particularly for large datasets where efficiency is a critical concern.

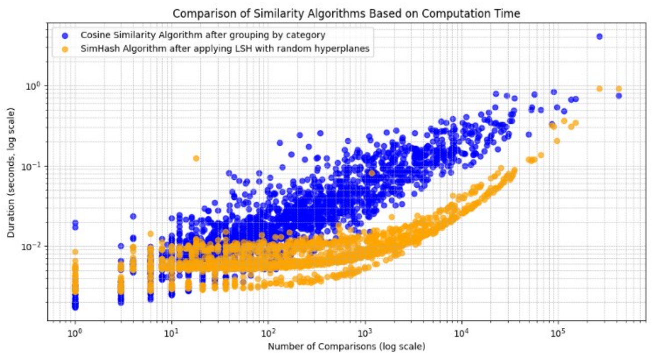

Figure 9: Computation time vs. Number of comparisons for both algorithms

Alghoritm	Total Comparisons	Total Duration (seconds)
Cosine Similarity Algorithm after grouping by category	4371878	102.17
SimHash Algorithm after applying LSH with random hyperplanes	4371878	22.60

Table 10: Total Computation time comparison for both algorithms

Accuracy of Recommended Products by True Cosine Similarity

In this section, we assess the precision of the recommended products by comparing them by the true Cosine Similarity computed using the brute-force approach described before. The goal is to determine how well each algorithm retrieves highly similar products based on their true similarity scores.
From Figure 10, we observe that:

- Both algorithms effectively identify highly similar products (those with a Cosine Similarity > 0.4). This means that when a product has a strong true similarity with another product, both algorithms reliably include it in their recommendations.
- The Cosine Similarity approach (left chart, blue) consistently provides better precision across all similarity levels. It can successfully retrieve moderately similar products, even when their actual similarity is lower.
- The SimHash-based approach (right chart, orange) struggles to recognize products with lower similarity values. It tends to focus only on the most strongly related items, leading to a drop in precision when the true similarity score is below 0.4.

This analysis suggests that while both methods perform well for highly similar products, the first algorithm has a clear advantage in identifying meaningful relationships even among products with more moderate similarity scores. The second algorithm, while computationally efficient, tends to be more selective, capturing only the strongest similarities.

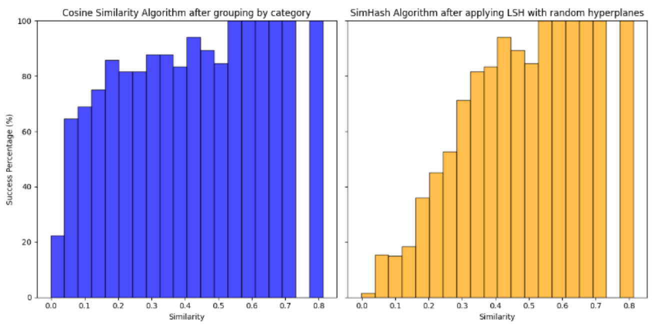

Figure 10: Precision of recommended products by true Cosine Similarity, comparison between the two algorithms

Accuracy of recommended products by True position on ranking

The evaluation of recommendation accuracy reveals that while both algorithms perform well in identifying highly similar products, the majority of product pairs exhibit lower similarity scores. This is evident from the scatter plot in Figure 11, where most data points are concentrated on the left side of the graph, indicating that many product relationships have a true Cosine Similarity below 0.4. Since the goal of an effective recommendation system is to accurately capture these weaker similarities, an algorithm's ability to distinguish moderately similar products becomes crucial for overall performance.
The bar chart in Figure 12 quantifies the precision of each algorithm to recommend the product by its true position on ranking. The Cosine Similarity approach after category grouping achieves approximately 80% precision, meaning that 8 out of 10 recommendations correctly appear in the top 20 most similar products when compared to the brute-force method. On the other hand, the SimHash-based approach achieves around 45% precision, indicating that it struggles to maintain accuracy when similarity scores are lower.
These results emphasize the key trade-off: while the second algorithm is computationally efficient, it does not generalize well to moderate similarity levels, leading to weaker recommendations overall. Meanwhile, the first algorithm performs consistently better across all similarity levels, making it the more reliable choice for high-quality recommendations.

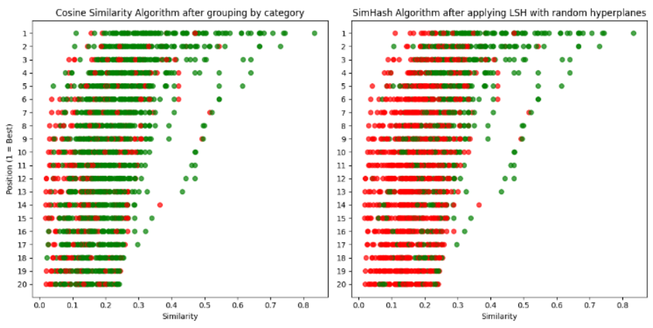

Figure 11: Scatter plot of similarity distribution between algorithms by true similarity and by true position on ranking

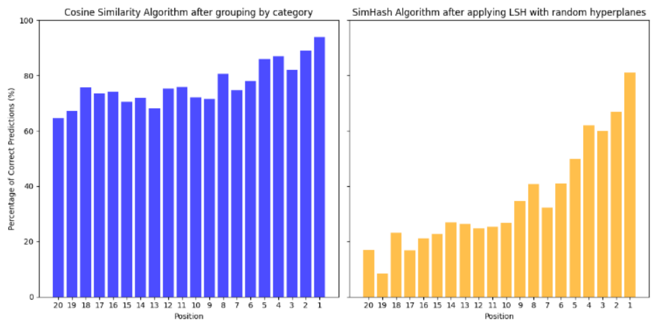

Figure 12: Precision of recommended products by true position on ranking, comparison between the two algorithms

6. Conclusion and final considerations

The goal of this study was to develop and evaluate two different product recommendation approaches using e-commerce session data. By analyzing user interactions, two distinct similarity-based recommendation methods were implemented: one leveraging Cosine Similarity after category grouping and the other using SimHash with Locality-Sensitive Hashing (LSH). Both methods were designed to provide relevant product suggestions based on user interest, while balancing accuracy and computational efficiency.

6.1 Summary of findings

The study began with an exploratory data analysis, where user engagement was quantified through an interest score based on viewing duration and interaction type (view, add to cart, purchase). The dataset was then processed to determine product relationships using category similarity derived from Jaccard Similarity and MinHashing. This allowed for a structured approach to limit similarity computations only to meaningful candidate pairs, significantly reducing the number of comparisons. It was also crucial to select an appropriate similarity threshold between categories to avoid having too many candidates and ensure only the most relevant ones were considered.
The first recommendation approach used Cosine Similarity on normalized interest scores to identify product relationships. The second approach implemented SimHash using LSH with random hyperplanes, where product vectors were transformed into binary signatures, allowing for efficient similarity approximation using Hamming Distance.
The evaluation of both methods focused on two key aspects: computational efficiency and accuracy. The results revealed a clear trade-off between the two approaches.
In terms of computational efficiency, SimHash proved to be significantly faster, requiring about five times less computation time than Cosine Similarity while performing the same number of comparisons. This was achieved through the use of binary signatures instead of full interaction vectors, making SimHash highly scalable for large datasets.
However, when considering recommendation accuracy, Cosine Similarity consistently outperformed SimHash. While both methods performed well for highly similar products (true Cosine Similarity > 0.4), SimHash struggled to maintain accuracy for moderately similar products.

6.2 Final remarks and future work

This project highlights how well-structured similarity-based methods can significantly influence the effectiveness of product recommendation systems. While Cosine Similarity remains the most precise method, SimHash presents a scalable alternative for handling massive datasets. The comparison underscores the importance of balancing computational feasibility with recommendation quality, a critical consideration for modern e-commerce platforms managing millions of users and products.

A promising approach could be to combine both methods, using SimHash to quickly filter candidate recommendations and then applying Cosine Similarity for a refined ranking within a smaller subset of products. This hybrid strategy would balance computational efficiency with recommendation precision.

As recommendation systems continue to evolve, future advancements will likely refine this balance, integrating innovative techniques to further improve both efficiency and relevance in large-scale applications.