भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर
Indian Institute of Information Technology, Nagpur
Institute of National Importance by the Act of Parliament
The Indian Institute of Information Technology (PPP) Act, 2017

Group 7

Presented by:
Om Varma [BT22CSE216]
Vinayak Gupta [BT22CSE222]
Kamal Chandra[BT22CSE223]
Harsh Agarwal [BT22CSE228]

Presented to:
Dr. Jitendra V. Tembhurne

Online Learning Algorithms

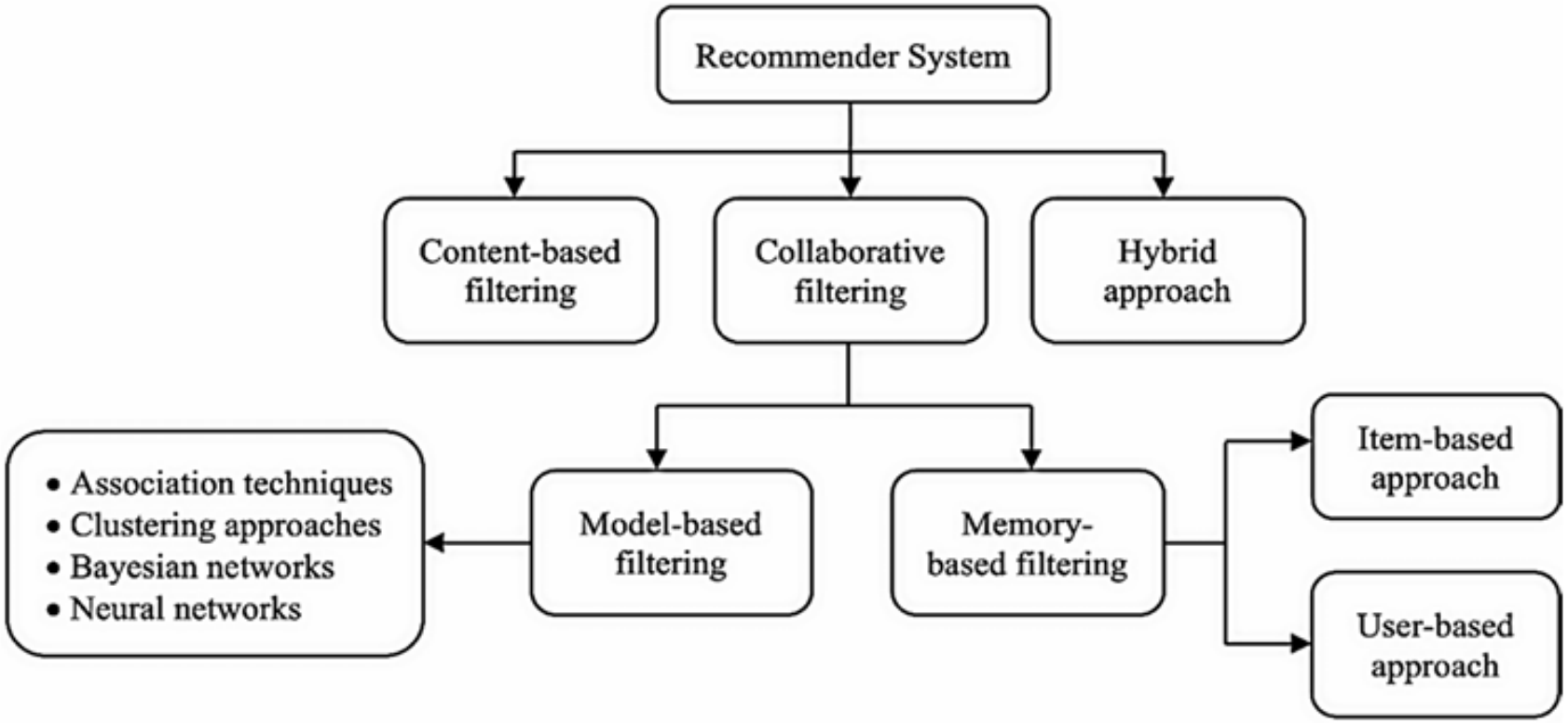
Agenda

- Introduction
- Challenges
- Literature review
- Research Gaps
- Problem Statement
- Motivation
- Dataset used
- Methodology
- Findings
- Time Complexity Analysis
- References

Introduction

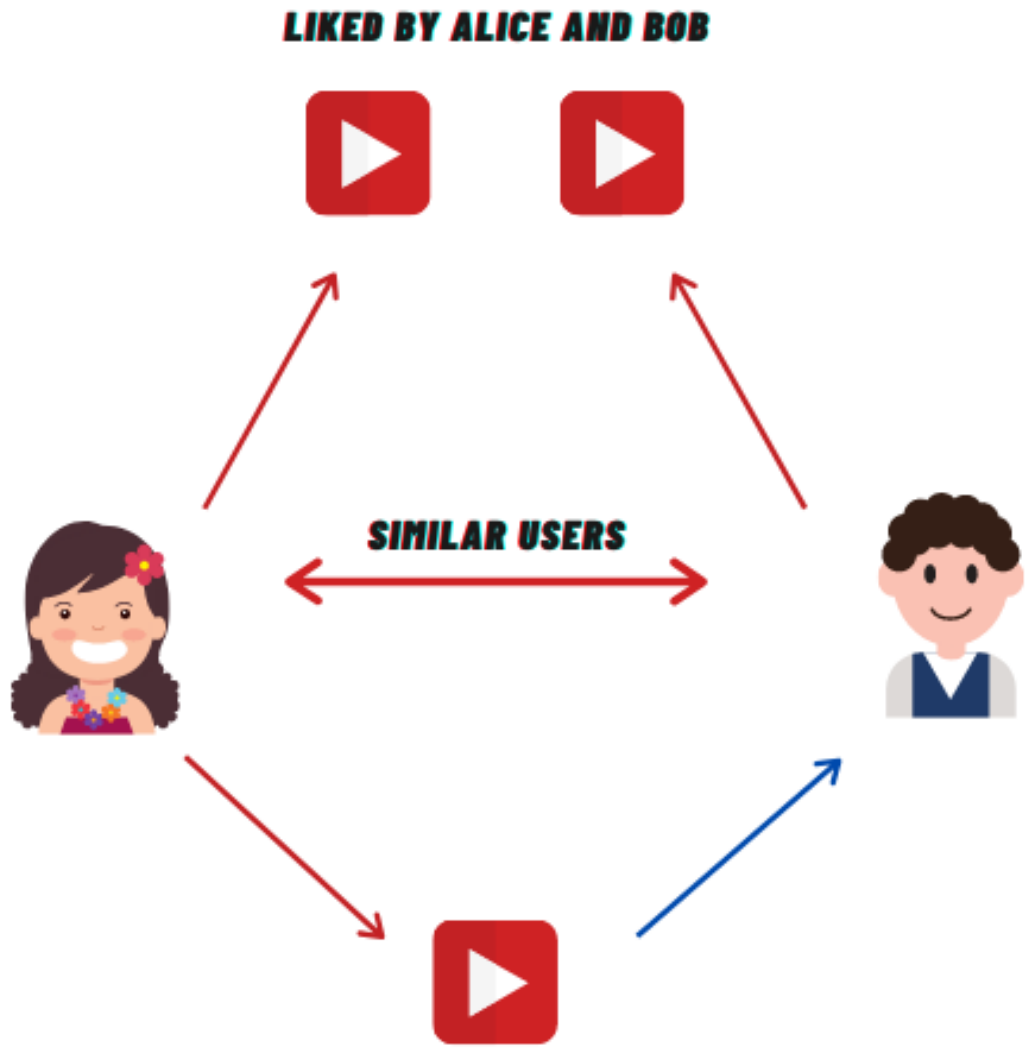
- Online learning represents an important family of machine learning algorithms, in which a learner attempts to resolve an online prediction task by learning a hypothesis from a sequence of data instances one at a time.
- Recommender systems are an extensive class of Web Applications that aim to forecast user responses to options by considering preferences and objective behaviours.
- Recommender systems leverage algorithms to analyse user preferences and behaviours, thereby facilitating personalised recommendations tailored to individual users' interests.

Types of Recommender Systems

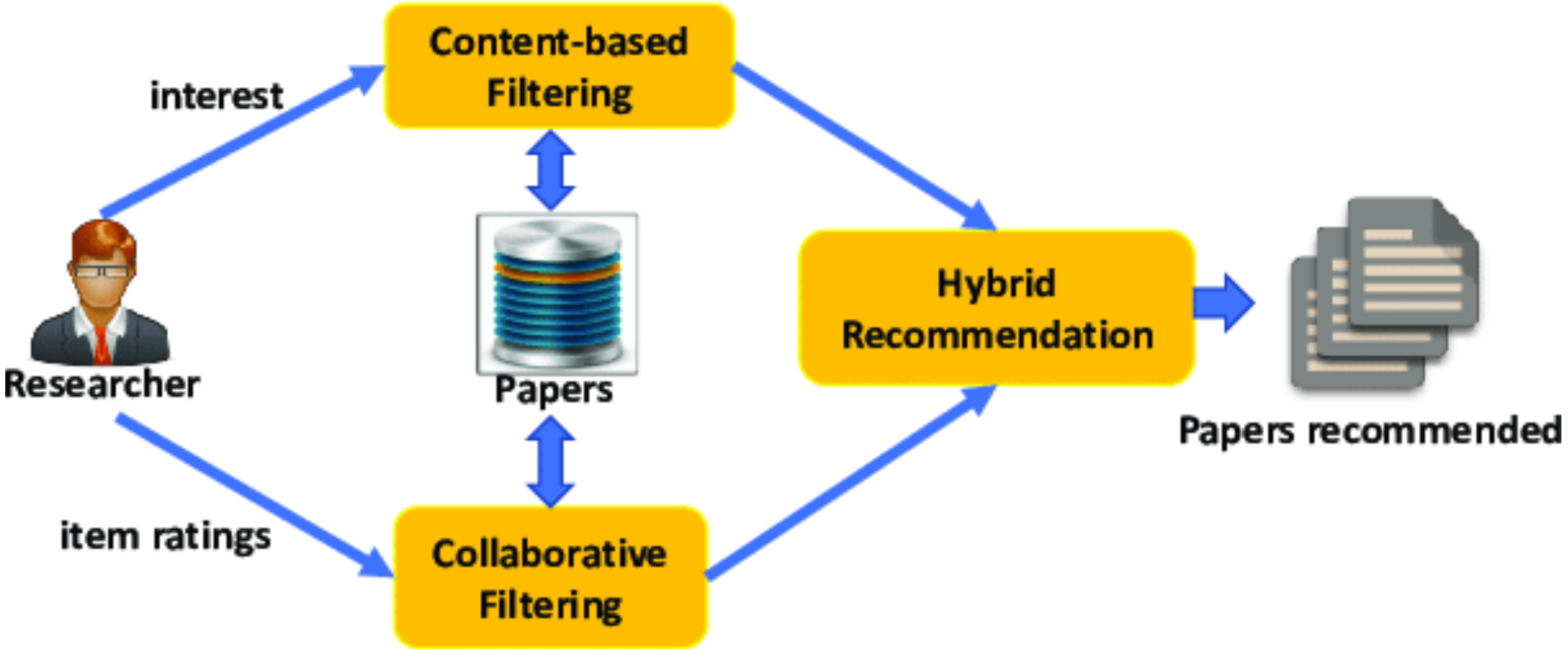


Content-based recommender systems

COLLABORATIVE FILTERING



Hybrid Recommender System



Challenges

Despite their pervasive adoption and evident utility, recommender systems grapple with multifaceted challenges that undermine their effectiveness and reliability.

- Cold Start Problem
- Sparsity Problem
- Over-specialisation Problem
- Shilling Attacks
- Latency Problem
- Synonymy

Literature Review

- FuJie Tey et al. introduced a social network-based recommender system focusing on user preferences and their friends' shopping experiences to enhance recommendations.
- YiBo Chen et al. proposed a collaborative filtering approach addressing sparsity issues by computing direct and indirect similarities between users, resulting in a similarity matrix.
- Dimensionality reduction approaches aim to directly reduce the dimensionality of the consumer-product interaction matrix, potentially forming clusters of items or users and using advanced techniques like Principal Component Analysis (PCA).
- Dimensionality reduction can lead to a denser user-item interaction matrix, improving recommendation quality in some cases but may lose potentially useful information in others
- Another approach treats the data as a bipartite graph, with nodes representing users and items and edges representing user-item interactions weighted by ratings.

- Global similarities between users or items are derived using graph theoretic measures, such as average commute time between nodes or minimal hop distance.
- Other measures include spread activation of nodes in the graph, but the main drawback is the lack of clear interpretation of similarity measures for prediction purposes.
- Matrix Factorisation results are reused to reconfigure the input matrix, enhancing recommendation performance.
- Gantner et al. addressed item-side cold-start by clustering new items based on feature data, applying content-based filtering (CBF) concepts.
- Sun et al. clustered items using attribute data and preferences, creating a decision tree for predicting preferences for new items.
- User features were constructed based on user action history to derive similarities between users and items for recommendations.

Research Gaps

- Deployment challenges such as cold start, scalability, sparsity.
- Enhancing Real-Time User Feedback Handling: Developing techniques to improve the handling of real-time user feedback in recommender systems, focusing on capturing and processing feedback efficiently to enhance recommendation quality in dynamic environments.
- Developing standardised evaluation criteria or methodologies to assess the performance of different recommender systems comprehensively, facilitating fair comparisons across systems and domains.
- Developing Evaluation criteria or metrics to assess user satisfaction in real-time, enabling the evaluation of recommender system performance based on user feedback and preferences as they interact with the system.
- Application of recommender systems in diverse contexts beyond traditional domains, such as psychology, mathematics, and computer science, including methods to adapt recommender algorithms to different contexts effectively

Problem Statement

- Our work centres around tackling the **Sparsity Problem** within the context of recommender systems. We also aim to address the **Cold Start Problem**, which is a distinct subset of the broader Sparsity Problem.
- Our work will concentrate on developing methodologies, algorithms, or frameworks to mitigate the detrimental effects of data sparsity and the Cold Start Problem, thereby enhancing the effectiveness and efficiency of recommender systems.

Motivation

- In order to alleviate the cold-start and sparsity problem many studies currently construct and propose a method of predicting users' preferences for items using information, such as the feature of items.
- In addition, various studies have been conducted to predict users' evaluation by extracting and analysing feature vectors of items using deep learning approaches.
- However, current studies are attempting research that transforms the shape of the existing system using metadata of users or items. Namely, it proposes methods that can produce users' evaluations in cold-start and sparse situations to produce more accurate results.
- More analysis is required to use the proposed techniques in real situations. The approaches currently being studied can produce more accurate results in specific situations, but have the disadvantage of not having universality, such as conventional collaborative filtering.
- In addition, the sparsity problems persist since most studies still utilise the same input form.

Dataset Used

MovieLens Dataset

- This dataset contains 100836 ratings by 610 users and 3683 tag applications across 9742 movies between March 29, 1996 and September 24, 2018. All selected users had rated at least 20 movies.
- The data was contained in the files links.csv, movies.csv, rating s.csv and tags.csv.
- For appropriate use, we merged these different CSV files into a single one.

ratings

	userId	movieId	rating	title	genres
1	1	1	4	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	1	3	4	Grumpier Old Men (1995)	Comedy|Romance
3	1	6	4	Heat (1995)	Action|Crime|Thriller
4	1	47	5	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
5	1	50	5	Usual Suspects, The (1995)	Crime|Mystery|Thriller
6	1	70	3	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
7	1	101	5	Bottle Rocket (1996)	Adventure|Comedy|Crime|Romance
8	1	110	4	Braveheart (1995)	Action|Drama|War
9	1	151	5	Rob Roy (1995)	Action|Drama|Romance|War
10	1	157	5	Canadian Bacon (1995)	Comedy|War
11	1	163	5	Desperado (1995)	Action|Romance|Western
12	1	216	5	Billy Madison (1995)	Comedy
13	1	223	3	Clerks (1994)	Comedy
14	1	231	5	Dumb & Dumber (Dumb and Dumber) (1994)	Adventure|Comedy
15	1	235	4	Ed Wood (1994)	Comedy|Drama
16	1	260	5	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi
17	1	296	3	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
18	1	316	3	Stargate (1994)	Action|Adventure|Sci-Fi
19	1	333	5	Tommy Boy (1995)	Comedy

Snapshot of the Dataframe

Methodology

- Having multiple features in a model, suppose for a movie recommender system we have different features such as action, horror, thriller, etc. Some movies have been rated by users, most aren't, i.e., we have sparsity of data.
- Some features are correlated to each other, for example a horror movie will have some component of thriller. This is useful, for instance, recommending horror and action movies to someone who likes thrillers.
- We then create a rating matrix where each **row** represents a **user** and each **column** represents a **movie**. This matrix will store the ratings given by users to movies. The NaN (Not a Number) values indicate missing ratings, as not every user may have rated every movie.
- Another matrix is initialised where each **row** represents a **movie** and each **column** represents a movie **genre**. This matrix helps in analysing the distribution of movie genres across the dataset.
- Each cell in this matrix indicates whether a particular movie belongs to a specific genre. This is represented by binary values (0 or 1), where 1 indicates the presence of the genre and 0 indicates its absence.

- Any NaN values in the rating matrix (which represent missing ratings) are filled with 0. This simplifies subsequent analysis by treating missing ratings as neutral (i.e., a rating of 0).
- These matrices serve as the foundational data structures for various types of analyses, such as collaborative filtering for recommendation systems, genre-based analysis, or user-item matrix factorisation for dimensionality reduction.
- For each user, an empty dictionary is initialised to store genre preferences for that user. The dictionary is initialised with genre names as keys and initial preference values set to 0 for each genre. We then filter the ratings dataframe to retrieve movies rated by the current user.
- Given a movie, the genre preferences for the current user based on the movie genres and ratings are updated. Genre preferences are calculated by multiplying the genre value (1 if the movie belongs to the genre, 0 otherwise) by the rating given by the user for that movie and adding it to the corresponding genre preference.
- After calculating genre preferences, we normalise them using **Min-Max normalisation**. This ensures that preferences are scaled between 0 and 1, making them comparable across users. This step prevents biases caused by users who might rate movies differently on average.

	Romance	Sci-Fi	Thriller	War	Western
1	0.287918	0.434447	0.586118	0.254499	0.077121
2	0.068182	0.234848	0.560606	0.068182	0.053030
3	0.039683	1.000000	0.460317	0.039683	0.000000
4	0.468900	0.081340	0.322967	0.059809	0.090909
5	0.357895	0.052632	0.336842	0.105263	0.063158
..	...	...	...	...	...
606	0.502269	0.106278	0.265318	0.093230	0.021936
607	0.310030	0.355623	0.762918	0.075988	0.024316
608	0.314977	0.566650	0.942872	0.069995	0.029851
609	0.250000	0.234375	0.718750	0.218750	0.062500
610	0.238517	0.493419	0.979049	0.095353	0.066344

[610 rows x 18 columns]

Snapshot of how the normalised data looks like

- After this, we identify users who have a strong preference (liking) for each movie genre, based on their normalised genre preferences derived earlier. This provides insights into user preferences, enabling targeted recommendations or genre-based analysis.
- For each genre, users who have a preference greater than 0.65 (arbitrarily chosen threshold) are filtered for the current genre. This threshold indicates users who have a strong liking for that genre.
- Then, the filtered users are sorted in decreasing order of their likeness for the current genre. Likeness is determined by the value of the genre preference for each user.

```
Users who like Action: [1, 320, 272, 277, 279, 292, 299, 313, 340, 256, 344, 354, 361, 363, 366, 368, 267, 249, 380, 212, 180, 183, 184, 203, 208, 211, 213, 244,
Users who like Adventure: [308, 106, 120, 145, 93, 148, 154, 172, 186, 205, 210, 234, 248, 264, 291, 371, 399, 459, 475, 527, 27, 529, 531, 556, 557, 586, 114, 3
Users who like Animation: [401, 252, 396, 388, 20]
Users who like Children's: []
Users who like Comedy: [307, 396, 412, 409, 406, 194, 200, 201, 388, 173, 381, 216, 217, 218, 370, 369, 174, 169, 224, 321, 141, 143, 456, 453, 147, 151, 448, 16
Users who like Crime: [433, 55, 553, 537, 295, 519, 444, 400, 362, 189, 76, 37, 39, 413, 449, 238, 585, 532, 528, 515, 61, 338, 343, 145, 588, 197, 46, 36, 353,
Users who like Documentary: [245]
Users who like Drama: [263, 570, 275, 432, 271, 434, 568, 268, 435, 265, 254, 572, 536, 262, 261, 260, 437, 258, 278, 567, 430, 429, 304, 302, 301, 300, 566, 296
Users who like Fantasy: [605, 556, 106, 371, 148, 523, 252, 205, 398, 180]
Users who like Film-Noir: []
Users who like Horror: [505, 533, 546, 571, 312]
Users who like Musical: [554, 188]
Users who like Mystery: [128]
Users who like Romance: [10, 12, 594, 358, 598, 116, 128, 100, 185, 146, 159, 113, 263, 563, 35, 214, 108, 506, 429, 224, 578, 144, 566, 593, 507, 280, 38, 188,
Users who like Sci-Fi: [3, 149, 550, 15, 303, 440, 267, 186, 377, 69, 272, 49, 210, 312, 256, 48, 95, 228, 344, 180, 277, 313, 271, 231, 371, 475, 75, 477, 580,
Users who like Thriller: [133, 386, 102, 192, 81, 80, 207, 242, 269, 297, 353, 360, 364, 126, 374, 389, 179, 402, 451, 465, 46, 485, 501, 37, 545, 26, 565, 150,
Users who like War: [463]
Users who like Western: [499]
```

Clustering users with strong preferences

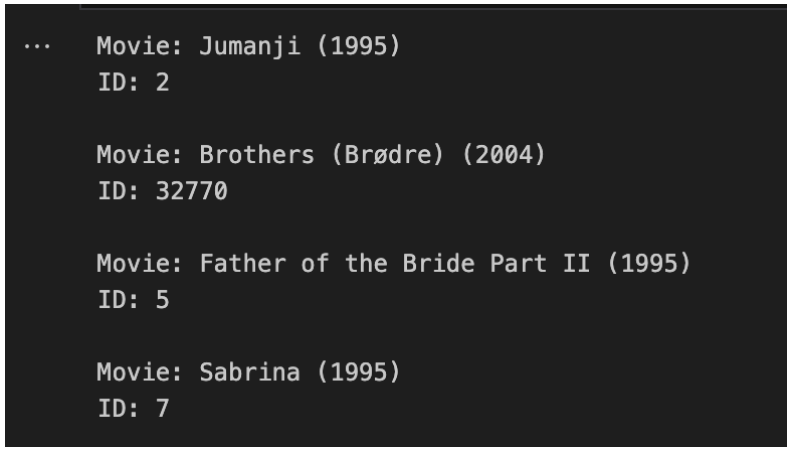
- When a movie name is passed as input, the ID associated with it is looked up and the genres associated with the specified movie are retrieved.
- Subsequently, we work on a variable called, ‘recommendation score’. The recommendation score indicates how likely a user is to enjoy the specified movie based on their genre preferences.
- For each user, it calculates a recommendation score based on the user's preferences for genres that match the genres of the specified movie. It iterates through the genres of the specified movie and checks if the user has a preference for each genre. If the user has a preference for a genre associated with the movie, it adds the preference value to the recommendation score.
- Once recommendation scores are calculated for all users, it sorts the users based on their recommendation scores in descending order. Users with higher recommendation scores are prioritised as they are deemed more likely to enjoy the specified movie.

```
Recommended users for 'Canadian Bacon (1995)': [289, 471, 194, 218, 173, 516, 310, 172, 335, 278, 431, 116, 57, 38, 565, 188, 420, 92, 1, 552, 115, 136, 347, 8, 2
```

Example : A list of users (UserIDs) for a specified movie

We also generate movie recommendations for a given user based on the preferences of similar users.

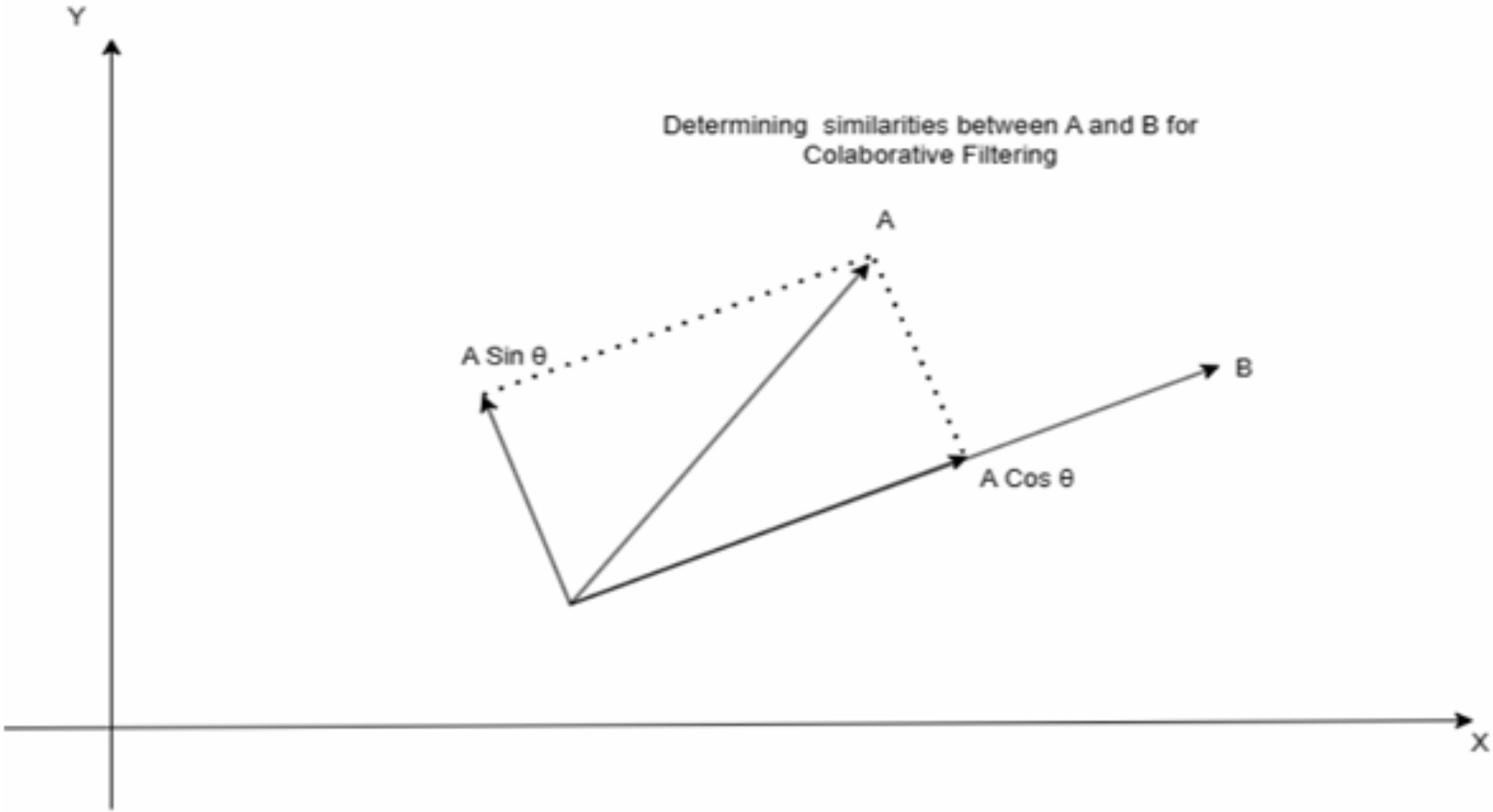
- Users similar to the specified user are identified based on their genre preferences. For each genre, if the specified user is among the users who have a strong preference for that genre, it adds all other users who also like that genre to the set of similar users.
- We then check the ratings of similar users for highly rated movies, here 3 (rated above this arbitrarily chosen threshold). The 'recommended movies' set is updated with these movies.
- The number of recommended movies are limited to the top n movies.

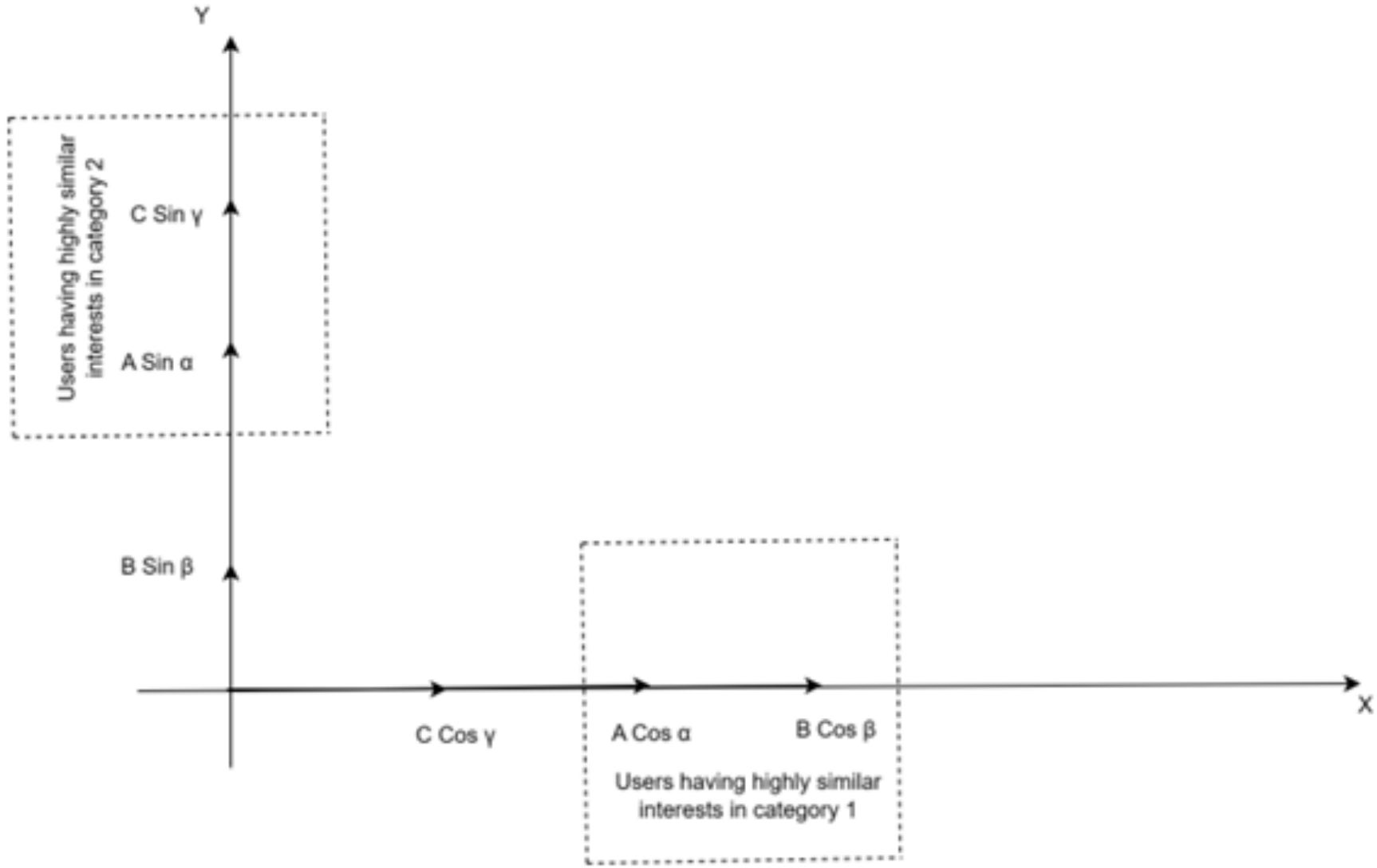


Example : A list of recommended movies for a specified user

Reconsidering the use of Dot Product

- Generally, to find similarities using **cosine similarity** between users we take dot products of their vectors. The closer two vectors are, the more similar their interests are, so we can suggest similar contents.
- However, a disadvantage is that we have to take dot products of all the pairs of users, and dot product itself is matrix multiplication. So, this entire process is quite expensive.
- On the other hand, we can just break the vector in its components and club the users with high interests on each axis to form a group of users interested in similar topics. Imagine each genre as a different axis, with the normalised score on each of them plotted for different users.
- The advantage with this method is that we don't need to compare each pair here, also if two users have similar interests we know exactly in which category and how much they have similar interests.
- For Example, if two users have similar interests but one user is more inclined to the other axis then recommending only those content which is extremely liked by the second user in the first category will be beneficial. This operation is not as expensive as the previous approach.


Determining similarities between A and B for
Colaborative Filtering
Y
A
A Sin θ
B
A Cos θ
X


Y
Users having highly similar interests in category 2
C Sin γ
A Sin α
B Sin β
C Cos γ
A Cos α
B Cos β
X
Users having highly similar interests in category 1

Time Complexity Analysis

Previous research focused on conventional similarity measurements like Cosine similarity and Pearson correlation coefficient to tackle sparsity and cold-start situations in collaborative filtering.

- In this approach, calculating similarities between users involves computing the **Pearson correlation coefficient** for each pair of users. This process has a time complexity of **O(n^2 * m)**, where n is the number of users and m is the number of movies. Additionally, generating recommendations for each user involves iterating over similar users and their ratings, which can also be computationally expensive.

By adjusting similarity measurements, the accuracy of recommendation results is enhanced compared to conventional methods in sparse and cold-start scenarios:

- The new approach involves grouping on each dimension = m (No. of features)
- Sorting time for each dimension = n*log n
- Total Time : **O(nlogn * m)**

Addressing the Cold Start Problem

- The strategies used for solving the cold start problem are, recommending the popular items to users based on their geolocation, if we have information about users' connections then recommending based on it.
- Some other approaches include collecting user context from the data available on their device and connected accounts to recommend items, such as, recommending a user based on his browsing history and collaborating with other platforms for finding users' interests. However, these methods do raise privacy concerns.
- Another approach to find out users' interests is asking users' interests upfront at time of login and providing him access to modify his interest any time by choosing topics.

Scope for future work

- This algorithm works at its best when there are users with strong preference to at least one genre. For a user with mild to moderate likings, this system falls short to address their recommendations.
- Optimisation algorithms that provide a quantitative assessment of the threshold values need to be standardised so that the recommender systems work as efficiently as possible.
- To address the Cold Start Problem, user context in the form of geolocation, search histories, and other preferences is needed. The concern of privacy needs to be addressed, in essence, there needs to be a standard protocol in place to **minimise** the amount of user's data taken and **maximise** the use of the attributes that have been collected. A qualitative study in regard with what attributes help the recommender systems the most is fundamental.

References

[1] Steven C.H. Hoi et al, Online Learning: A Comprehensive Survey, Cornell University

[2] Mining of Massive Datasets: Jure Leskovec Stanford Univ., Anand Rajaraman Milliway Labs, Je rey D. Ullman Stanford Univ.

[3] Roy, D., Dutta, M., “A systematic review and research perspective on recommender systems.” J Big Data 9, 59 (2022)

[4] Nitin Mishra et al 2021 J. Phys.: Conf. Ser. 1717 012002, “Research Problems in Recommender systems”

[5] Zan Huang, Hsinchun Chen, et al. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004, 116–142.

[6] Sarwar, B., Karypis, G., et al. Application of dimensionality reduction in recommender systems: A case study. In Proceedings of the WebKDD WorkshopattheACMSIGKKD.ACM,NewYork.2000.

[7] Good, N., Schafer, J., et al. Combining collaborative ltering with personal agents for better recommendations. In Proceedings of the 16th National Conference on Arti cial Intelligence, 1999, 439–446.

[8] Huang, Z., Chung, W., et al. A graph-based recommender system for digital libraries. In Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (Portland, Ore.). ACM, New York, 2002, 65–73.

[9] Goldberg K, Roeder T, et al. Eigentaste: A constant time collaborative ltering algorithm. Information Retrieval. 2001, 4(2):133-151

[10] Chrsistian Desrosiers, George Karypis. Solving the Sparsity Problem: Collaborative Filtering via Indirect Similarities. Technical Report. Department of Computer Science and Engineering University of Minnesota 4-192 EECS Building 200 Union Street SE Minneapolis, MN 55455-0159 USA.2008.

[11] Chen, Yibo et al. “Solving the Sparsity Problem in Recommender Systems Using Association Retrieval.” J. Comput. 6 (2011): 1896-1902.

[12] Çano, E.; Morisio, M. Hybrid Recommender Systems: A Systematic Literature Review. CoRR 2019, abs/1901.03888. Available online: https://arxiv.org/abs/1901.03888 (accessed on 12 January 2019).

[13] Rojsattarat, E.; Soonthornphisaj, N. Hybrid Recommendation: Combining Content-Based Prediction and Collaborative Filtering. In Proceedings of the Intelligent Data Engineering and Automated Learning; Springer: Berlin/Heidelberg, Germany, 2003; pp. 337–344.

[14] Lang, K. NewsWeeder: Learning to Filter Netnews. In Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, USA, 9–12 July 1995; pp. 331–339.

[15] Krulwich, B. Learning user interests across heterogeneous document databases. In Proceedings of the 1995 AAAI Spring SymposiumSeries, Palo Alto, CA, USA, 27–29 March 1995; pp. 106–110.

[16] Chughtai, M.W.; Selamat, A.; Ghani, I.; Jung, J. E-Learning Recommender Systems Based on Goal-Based Hybrid Filtering. Int. J. Distrib. Sens. Netw. 2014, 2014