Intelligent Caching for Scalable Content Delivery

Pranali Divekar
Rochester Institute of Technology

Advisor: Prof. M. Mustafa Rafique

CONTENTS

Intelligent Caching for Scalable Content Delivery

Pranali Divekar
Rochester Institute of Technology
Rochester, New York, USA
pd1267@rit.edu

***Abstract*—Sometimes you see content on the internet that you are remotely interested in. 'Top' for the world might not be 'top' for you. Personalized recommendation systems have become the basis of user interaction with any online platform. Systems that implement this, use complex algorithms to tailor content, products, and services to individual users. These systems analyze huge amounts of user data including their behavior online, their interaction patterns, browsing habits etc. The main goal of these systems is to cater to these dynamic interests and deliver content unique to each user. This project aims at enhancing the existing methodology of making personalized recommendation systems by considering the content browsed by the users. It also aims to address the issue of latency by attempting to reduce the time taken to deliver these unique recommendations. This is achieved by caching the dynamic data as required. The project also attempts to resolve the cold start problem that most recommendation systems face today. At its core, a hybrid system is implemented that leverages the strengths of both collaborative filtering and content-based filtering techniques. The system is designed to adapt to the latest trends so that the recommendations are contextually relevant. The system will be tested on near-live data using publicly available APIs. Through all these techniques, the proposed system aims to offer a quicker and a more personalized and engaging online experience to the users.**

I. Introduction

Over the past few years, there's been a big shift in how people shop, read, and engage with content, moving from physical stores and traditional media like newspapers, magazines, and radios to online platforms. Offline market sales have significantly dropped as most people now turn to online shopping for convenience, a wider selection of options and more attractive discounts.

This shift is not limited to just shopping, people prefer reading books through platforms like Kindle for ease and accessibility. People prefer browsing an online store like Amazon or Zara to buy what they like instead of spending hours in malls. So, the online stores have to adapt to the growing needs of people and therefore tailor their feed in ways that engage the customer. This need for personalized content is not just limited to online stores. Platforms like Netflix [1] and Amazon Prime suggest movies and TV shows based on what the user has watched before. Similarly, music platforms like YouTube Music and Spotify tailor playlists and song recommendations to the user's listening habits. Nothing different happens on social media platforms like Instagram or X (formerly known as Twitter). All this, to keep the content relevant and engaging enough to attract a larger audience. The feeds are curated based on the posts liked by and commented on by users. Different engagement metrics are gauged so that the feed remains highly enjoyable. Thus, personalization makes the content more efficient and engaging as the users are presented with content that directly benefits them. Personalization is actually a win-win. This is because the company greatly benefits not just by increasing sales because of efficient marketing but also by retaining users and earning their loyalty. The users in turn get what they want, quickly.

In all these scenarios, the system in place must be fast and highly sensitive to the changing nature of the user's interests, needs and demands. It must be able to quickly learn and adapt to user interactions to offer recommendations that are increasingly accurate and tailored. This demand for personalization, calls for a sophisticated recommendation system today.

Let us touch base a little bit on the existing personalization systems and their challenges. Tech giants like Amazon for instance [2], track a user's browsing history, purchase history, items in the shopping cart, and even how long they look at a product before moving on. Similarly, Netflix [1] analyzes the shows and movies you watch, the time of day you watch them, and even whether you finish them or stop halfway through. They consider your ratings and the similarity of titles to create a complex web of connections that inform their recommendation engine.

By now, we have established that any kind of recommendation system focuses on understanding the patterns of user behavior to create a highly tailored user experience. Despite these sophisticated technologies, these systems face significant challenges. The first being accuracy of recommendations. If the system 'learns' wrong, they might earn frustration from the users. Next, there is the challenge of "filter bubble" [3] where the system's eagerness to please can end up narrowing a user's exposure to new and diverse content, limiting their experience. Next, the sheer volume of data can lead to significant scalability issues. There is also the problem of latency [4], wait times faced by users that is not welcomed by impatient users. Lastly, there is also the "cold start" problem [5]. In this, a newly added user doesn't get accurate recommendations as the system cannot draw any inferences for the user because it has not gained enough information.

My project aims to address some of the above mentioned challenges – relevancy of recommendations, latency and the cold start problem. This project focuses on making the recommendation process more unique to the user. The goal is

to generate a unique 'top' recommendation for each user. The project focuses on data gathered from the publicly available YouTube API. The system integrates both collaborative filtering and content-based filtering techniques to develop direct video profiles and indirect user profiles. The users with similar preferences are grouped into one unit. Looking at the engagement metric data, similar videos are clustered together. Ultimately, the algorithm maps the users' interests with the most relevant video, ensuring personalized and relevant content recommendations unique for each user. The sections that follow are organized as – Literature Review, Problem Statement, System Design, Algorithms Used, Implementation, Evaluation, Conclusions.

II. Literature Review

Many attempts have been made in the past, some are ongoing, in order to personalize content, we see online. Given that, recommendation systems have significantly improved by focusing on content that closely aligns with the user preferences. At their core, these systems are built either by analyzing content directly [6] or by examining user interactions [7]. [8] and [9] highlights the latest advancements performed in the domain of content-based filtering. The paper talks about a video recommendation system where videos are analyzed by identifying objects and sounds within them which is a technique that analyzes content beyond the basic metadata tags. [10] details on collaborative filtering techniques. This paper describes an approach that not only tracks user interactions but also attempts to understand deeper patterns of user behavior. It encompasses memory-based, model-based and hybrid models to refine user preference predictions. Thus, it enables the system to deliver more accurate predictions by utilizing user similarities and item similarities with past interactions. Although both these techniques seem close to perfect in recommendations, challenges persist. To begin with, systems that use collaborative filtering face a major issue of "cold start". [5] talks extensively about this problem and how it still remains in most systems. [11] explains how homophily in social networks can be used to find similarity between users. Thus, the system can successfully generate recommendations to new users with no history with the system. Secondly these systems also face issues like scalability and the gray sheep problem [12], where users with unique tastes find recommendations less relevant. This paper suggests using clustering algorithms to pick out grey-sheep users. The proposed system uses a hybrid algorithm to address this issue. In [13] challenges faced by current personal recommendation systems such as data sparsity and diversity were highlighted. On similar lines, the content-based systems face the challenge of relevancy [9]. [14] examines how click through rate (CTR) contribute in finding the relevancy of recommendations. In order to overcome challenges like the cold start and data sparsity, in came the hybrid systems. [15] provides a comprehensive overview, highlighting the evolution of recommendation systems from basic collaborative and content-based filtering to more sophisticated hybrid approaches. In [16], the working of a hybrid approach that encompasses both collaborative filtering and content-based filtering is described. It is a news application. They utilized a Bayesian framework to predict a user's current interest in news based on that individual's activities and the news trends observed across all users' activities. This paper highlighted the need for a dynamic system that adapts to the changing requirements and preferences of the user. It is important to track live or near-live activity of the users. This paper also introduced click-stream analysis which basically monitors the mouse clicks of the users. Utilizing this technique can help us draw valuable insights in user behavior. Thus, [17] developed a dynamic authentication system that continuously verified user authenticity. This contributes to the importance of understanding user preferences and behaviors. The paper discussed the utilization of click-stream data for user profiling which emphasizes on uniqueness of user interaction patterns proposing a model where server side click-stream data is analyzed to create behavioral profiles. These profiles are very important to determine relevant user data. In [18], the authors focus on immediacy of news. They propose an efficient system architecture that generates immediate personalized news in a practical environment. The system adapts well to the new trends and user interests are reflected. The algorithm proposed works on a decay score that rates the 'staleness' of the news article. Current recommendation systems that are in place for giants like Netflix [1] and [19] explain currently employed complex algorithms that analyze viewing patterns, search histories and even the time of the day of content consumption. [20] explains a new algorithm called Tunes Recommendation System (T-RECSYS) that uses a hybrid approach as input to a deep learning classification model that can be applied to different platforms like YouTube, Netflix, Amazon etc. Similarly, YouTube [21] employs a mix of collaborative filtering and machine learning. They use core concepts of association rule mining in order to calculate the similarity count between videos. Depending on the content data and the user activity data the system calculates a relatedness score. This score is then mapped to users' activity and accordingly recommended to the users.

III. Problem Statement

As the project aims to effectively use a caching solution to store and fetch frequently used data, the problem statement is as follows –

"Design a dynamic personalized feed recommendation system that integrates user trends and behavior to make contextually relevant, unique suggestions with low latency."

Before the recommendations are made to the user, the system first checks the cache for a match. If a hit occurs, the recommendation is made from the cache, if a miss occurs, it falls back to the database, updates the cache and makes the recommendations from the next cache hit. We will discuss this implementation in the following sections.

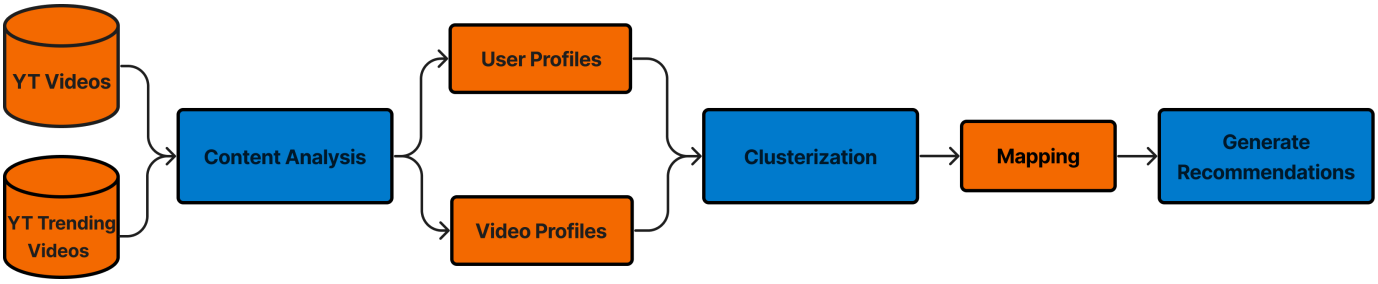

Fig. 1. Architecture of the system.

IV. System Design

Fig. 1. describes the architecture of the proposed recommendation system. It is an attempt to optimize the personalization of content delivery, ensuring that each user gets feed tailored to their unique preferences and behaviors. The heart of the system comprises of data collection, content processing units, clustering mechanisms and a mapping block. The system processes voluminous data and uses content recognition to create indirect user profiles due to lack of direct user interaction data. Subsequently, it also creates direct video profiles based on the engagement metrics. After the content-based profile creations, the system clusters them using two main strategies - inner clusters (content-specific) and outer clusters (domain-specific). It then maps the relevant videos falling in either one or both of the clusters to the respective user profiles. The recommendations are made in a way such that first, the system searches for a direct inner cluster match. If it succeeds, recommendations tailored to a specific user are generated. If it fails, then the outer clusters are looked up to find and provide relevant recommendations. This is done in case the system lacks user history. Thus, tailored content is delivered by the system regardless of whether the user is new or returning.
Now let us look at each component of the architecture one by one.

A. Data Collection

Data is sourced from two primary repositories: YouTube Trending Videos and another collection of YouTube Videos. Finding datasets with detailed engagement metrics was challenging due to user privacy concerns, making the choice of these specific repositories crucial. After an exhaustive analysis of publicly available data, these two datasets are identified as the optimal fit because of detailed data on the engagement metrics. These metrics include likes, dislikes, variations in the number of likes/dislikes over time, average views, and average watch time per video. The datasets hold intricate details such as click-through rate which serve as a direct reference to how well the video performs with users. Furthermore, these datasets provide more details such as geographical distribution data, publication time frames, and subscriber trends over time. These factors are crucial in developing direct video profiles and calculating a 'popularity factor' which is a measure of changing popularity and appeal of a video to the audience over time. This factor plays a crucial role in generating recommendations as the 'most popular' recommendations appear at the top of the list. All these factors contribute significantly to understand what is currently trending and what the users are preferring. This in turn helps to fine-tune recommendations in order to deliver content that is not only interesting to the users but is also timely and relevant.

B. Content Analysis

Once the datasets are finalized, a comprehensive analysis is performed on the collected data. The content analysis serves as a bridge between the video profiles and the user profiles. The datasets provide direct information concerning the videos. Dominant keywords picked out from the title of the video along with the engagement metrics together contribute to the creation of video profiles. The popularity factor is also included in the profile creation. In a similar way, dominant keywords are picked from the user comments to create user profiles. LDA (Latent Dirichlet Allocation) is implemented to pick out the dominant keywords from the titles and comments. These dominant keywords are mapped and matched in order to detect and group users with overlapping interests into clusters, based on the similarity of their language patterns.
As a result of this analysis, both users and videos have an associated set of dominant keywords, which is utilized to cluster them according to shared content interests, laying the groundwork for the next step - clusterization.

C. Clustering Mechanism

Following the content analysis, the videos and users are categorized into clusters based on content similarity and relevance. To accomplish this, the system employs a hierarchical clustering mechanism that relies heavily on the k-means clustering algorithm, a well-established method known for its efficiency in grouping data into k distinct clusters. As stated earlier, the clustering follows two main aspects - (i) Inner clustering (this includes content highly relevant to the user) (ii) Outer clustering (this is mainly concerned with the overall domain of the pool of users associated with the cluster.

Let us go ahead and discuss how exactly this works. For user clustering, the system analyzes the comment data associated with each video. An outer user cluster is formed around a specific video based on the user engagement — specifically,

the users who have commented on that video. These users are clustered together. So, the outer clusters are basically a group of users interested in the same video. This implies a shared interest among these users in the content of the video they engaged with. Once the video is pin-pointed, a cosine similarity is calculated between this particular video and the rest of the videos by matching the dominant keywords associated with the video titles. Similar videos are then added to the cluster. Thus, the outer clusters, on a high level look like a pool of users mapped to a pool of videos. The pool of videos imply shared interest amongst the pool of users. These outer clusters are further analyzed for each user to create inner clusters. The users themselves form the inner clusters of this system. The inner clusters are a cluster of videos that are specific to the user's interest. They are formed by mapping of dominant keywords associated with the users with the dominant keywords associated with the videos already belonging to the cluster. This potentially increases the relevance of the content to be recommended.

By applying k-means clustering through user engagement and content similarity, the system ensures that the resulting clusters are robust and multidimensional, leading to highly targeted and contextually relevant recommendations. The final aim is to map these clusters to deliver a personalized feed that resonates with the user's preferences and viewing patterns.

D. Mapping Block

Let us discuss the inner and outer clusters in more detail. The primary function of the block is the delivery of video recommendations, achieved through two distinct mapping strategies:

1) *Inner cluster recommendations*: These are the recommendations that happen per user. On analyzing the user comments and picking out the dominant keywords, these keywords are matched against they keywords of the video titles that are a part of the outer clusters. This matching encourages fine tuning thereby making the recommendations unique for each user.

2) *Outer cluster recommendations*: These are recommendations that are made for a cluster of users who have commented on the same video. These videos will be similar to each other as there will be matching made between the dominant keywords extracted from the title of the video on which this cluster of user has commented on. These keywords are matched with the top 'n' most popular video title keywords. Thus, the mapping block undergoes two stages of recommendation filters thereby making the recommendations unique to the user.

Once a list of top n videos is created, centroids of the clusters are calculated. The centroids along with the mapped videos to each cluster are pushed to the cache for an easier fetch. The centroids represent the average or mean position of all the data points within their respective clusters based on the features considered during clustering. By caching centroids, a snapshot of the core characteristics of the clusters which includes typical values and behaviors of data points (videos and users here) grouped within the clusters. This serves as a reference point for new data points (new users here) by simply measuring the cosine similarity to the centroids. Thus, the system can determine the most appropriate cluster and can make relevant recommendations even to a new user without having to recompute and reprocess the entire dataset.

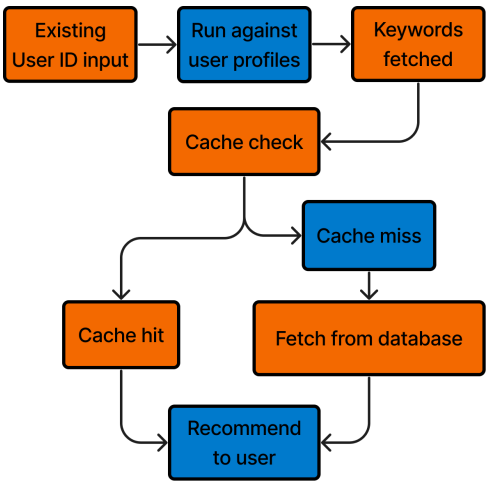

Fig. 2. Flow of system for a new/returning user

V. Algorithms Used

The system uses two main algorithms K-Means clustering and topic modelling. Each play an important role in filtering out personalized content.

A. K-Means

K-Means is a popular partitioning method that segregates a given dataset into 'K' distinct, non-overlapping subsets (or clusters) based on feature similarity. The algorithm iteratively assigns each data point to one of K groups based on the nearest mean. The system incorporated K-means twice while creating the outer and inner clusters.

B. LDA - Latent Dirichlet Allocation

Topic modeling is a type of statistical model used for discovering abstract topics within a collection of documents. It is commonly implemented using algorithms like Latent Dirichlet Allocation (LDA). LDA is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For videos, we perform topic modeling on video titles. The process involves preprocessing the text to lower case, removing punctuation and numbers, tokenizing the text into words, eliminating stopwords, and lemmatizing the remaining tokens. A dictionary and corpus are created from the preprocessed text, which LDA then uses to identify the dominant topics across the videos. Thus, the dominant topics and their keywords are extracted for each video.

Next, a similar process is carried out for creating user profiles. LDA is applied to user comments. The process begins by merging user comments with video titles to create a composite textual dataset. First the data is preprocessed as explained in the previous step, and then structured, collected and then

mapped to each user. With the corpus and dictionary in place, the LDA algorithm is employed to discern the dominant topics present in the user comments. It identifies patterns in word usage across the datasets and thereby effectively groups words into topics. These topics are associated with each user profile therefore identifying most significant topic keywords for that particular user. This process categorizes users into 'thematic buckets' and hence increases he fine tuning effort of my model.

VI. Implementation

Now that the system design is in place, let us pick a user flow to understand how recommendations are made by the system. Fig. 2 explains how the systems works when it is fed with a new user input. Let us go through it block by block. First, the cache block. The cache block forms an important component in the recommendation system as it is responsible to keep check on the system performance. The cache is initially warmed up by loading it with some data, particularly centroids of the clusters formed which represent groups of videos with similar content along with the actual associated videos. When a user ID is fed to the system, the system first implements LDA on the engagement metrics associated with this user to extract key content features. The engagement metrics can be in any form - likes/dislikes associated with video titles or comments of the user. The system functions as long as the user has some associated engagement data that can be picked for dominant keywords. Once the keywords are picked, the system first attempts to run them against the cached centroids. If there is a match between the list of dominant keywords and one of the pre-computed clusters (outer or inner), it indicates that the user's preferences align with one of the available clusters. The system retrieves the associated videos with the matched cluster and recommends them to the user. However, if the match fails, the words are sent to the database where they are added to the clusters as a new entry. This triggers an update to centroid recalculation and assignment. The new centroids are cached again ensuring that future requests including these keywords can be efficiently matched without needing to recalculate the centroids. Thus the system attempts to adapt to changing user preferences and also updates the centroids based on new user data. This ensures that the system's recommendations are relevant and engaging.

VII. Evaluation

The goal of this project was to improve the current personalized recommendation strategies by considering user interactions thereby providing accurately tailored content. Next, the goal was also to address prevalent challenges such as reducing the latency of responses and attempting to resolve the cold start problem [5].
The system is evaluated on all the three factors. Let us go through them one by one -

A. YouTube API

Till now, we have established how my system is making recommendations through *Sections IV* and *VI*. It is a hybrid system that employs both content-based and clustering algorithms to achieve these goals. As explained in section VI, we need a set of keywords as input to the system. Instead of using synthetic data, the YouTube Data API is used to procure video titles. The API acts as a representation of near real-time dynamic data and as a fresh input. It fetches data that is up-to-date and continuously evolving as against pre-recorded, historic data already present in the database. Due to the limitations imposed on gathering direct user data from this public API, the video titles are gathered from the API and are fed to the system as input. Raw titles are processed and dominant keywords are extracted from them. Thus, we have a brand new set of dominant keywords (which can be treated or looked at as new users) ready.

B. Relevancy of Recommendations

As described in *Section VI*, this new set of keywords is fed to the system. The first thing to evaluate should be the relevancy of the recommendations made. For the keywords ['project', 'knn'], the system is able to recommend titles as shown in Fig 4. Now, how relevant are these titles? To answer this question, the precision-recall framework is used. Refer Fig 3 for the plot. Precision indicates the proportion of relevant recommendations out of all recommendations provided while recall suggests out of all relevant recommendations how many were actually suggested. Precision measures quality - how much of the content delivered is likely to be of interest to the user thus maintaining user engagement and satisfaction. High value of precision implies that the user is receiving content that is highly relevant to him. On the other hand, recall measures the quantity - how much of the content that the user is exposed to is relevant. Basically, is the system is able to recommend a wide array of content that is relevant to the user? You can look at it as a high level view of the recommendations made. In most systems, as this, there is a considerable trade off between precision and recall. Increase in one value leads to decrease in the other. Taking a look Fig 3, it is clear that this system has a high precision and a low recall. The graph just shows results for 5 users, but upon testing for 15 users, it was clear that the system has a relatively higher precision rate - 0.60 (60%) and a lower recall rate. There can be many reasons for this result including but not limited to the nature of the dataset used, recommendation logic, etc. However, looking at the result, we can determine that the system is relatively good at recommending content that users find relevant. This is an important observation as it encourages user retention and engagement. The low recall score indicates that while the system is selective in its recommendations, it may seem to be restrictive. However, given the recommendation logic as described in the earlier section, the low recall score is heavily dependant on the nature of the data. The mechanism of utilizing outer clusters when there isn't any match for the inner clusters is in place to handle a situation like this. But, given the nature of the dataset which mainly revolves around videos pertaining to data science projects, the system restricts itself while recommending. If trained on a diverse

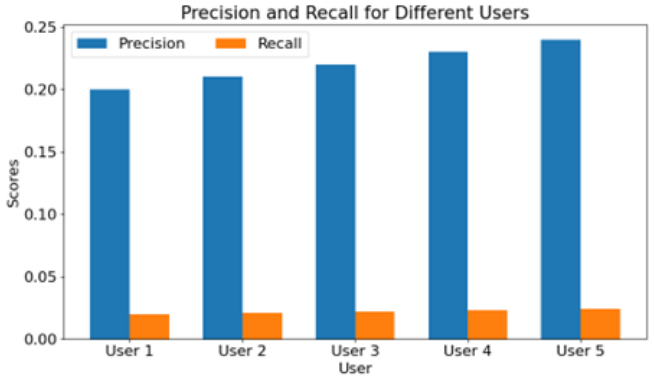

Fig. 3. Plot of Precision-Recall identifying the relevancy of recommendations.

```
PS C:\Users\prana\Desktop\Capstone> python test_with_cache.py
Link: 742LQ38OioU, Title: Data Science Productivity, Motivatic
Link: Ip50cXvpWY4, Title: The Best Free Data Science Courses M
Link: Xgg7dIKys9E, Title: Interview with the Director of AI Re
Link: 4qZINLzwYyk, Title: The State of Data Science with Krish
Link: -3d1NctSv0c, Title: Ken Jee Q & A Live Stream (50,000 Su
Recommendation generation time: 5.026550292968750 seconds
```

Fig. 4. Recommendations generated with cache

```
PS C:\Users\prana\Desktop\Capstone> python test_without_cache.py
No matching video found for Video ID: #NAME?
['-3d1NctSv0c', 'Ip50cXvpWY4', 'Xgg7dIKys9E', '742LQ38OioU', '4qZ
Time taken: 64.13247013092041 seconds
```

Fig. 5. Recommendations generated without cache

dataset, the recommendation logic should be able to function with a balance between the precision-recall scores. Fig 7 represents F1 scores for five users. The purpose of this score is to represent both precision and recall as a single unit of measurement. It measures both the accuracy and the system's ability to retrieve all relevant items. F1 score is the mean of precision and recall values. Thus, it is beneficial in calculating the trade-offs that can be made by the system between retrieving as many relevant videos as possible while maintaining the relevancy of recommendations. The increasing graph of Fig 7 shows a steady improvement in the recommendation strategies of the system. The users fed to the system here are new users. A low reading for the first user indicates that the systems recommendation was not very precise. However, the increasing trend in the graph and a high reading for the last user indicates that the system learns from the new users and improves its recommendation strategies thereby giving relevant recommendations.

C. Impact of Caching on Response Latency

Addressing the next goal, latency. In recommendation systems, caching is a critical optimization that drastically reduces the time required to process and make recommendations. Caching is a technique where subsets of data are stored in a temporary storage for quicker retrieval. The next important question to address is - what to cache? Given this system, there are many go-to options for data that can be cached however,

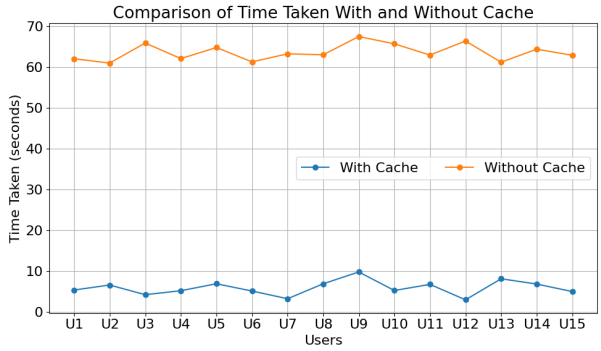

Fig. 6. Comparison of time (in seconds) taken with and without cache.

the system caches the cluster centroids of both the inner and outer clusters. The centroids represent an aggregation of user preferences and behaviors. To review the clustering mechanism used, the outer clusters hold a broader domain while the inner clusters hold a more user-centric topic. By definition, centroids are central vectors that represent the location of the clusters center in the datasets feature space. In this case, it represents a collective preference of a group of users. By caching these centroids, quick matching of the users profile with a corresponding cluster can be achieved which accelerates the recommendation process.

In the implementation of cache, when a recommendation request is initiated, the system first checks the cache for matching centroids. If a relevant centroid is found, the system recommends from the cache without recalculating all the clusters. This direct access to pre-computed data reduces the processing time drastically. Fig 4 and Fig 5 show how the system performs with and without caching respectively. Without caching the entire process of calculating and assigning clusters takes approximately a minute (60 seconds). After caching the centroids, the system took just 5 seconds to recommend thereby making significant improvement in terms of latency.

Fig 6 shows a plot of the time taken to generate recommendations for 15 users with and without cache. The users here are different combinations of keywords. The same users were tested with and without cache. The large gap between the two lines shows how effective caching of the centroids is. The blue line indicates the time taken with the cache while the orange line represents time without cache. The plot clearly indicates how crucial caching is in reducing the latency of the system. The idea of caching the centroids can be expanded further to cache individual user profiles or frequently accessed metadata (video titles in this case). It can also be extended to hold just the trending videos which dynamically change over time and can act as another filtering layer before making recommendations.

D. Addressing the Cold Start Problem

The Cold Start problem as described in [5] has been a significant challenge in most recommendation systems. There are

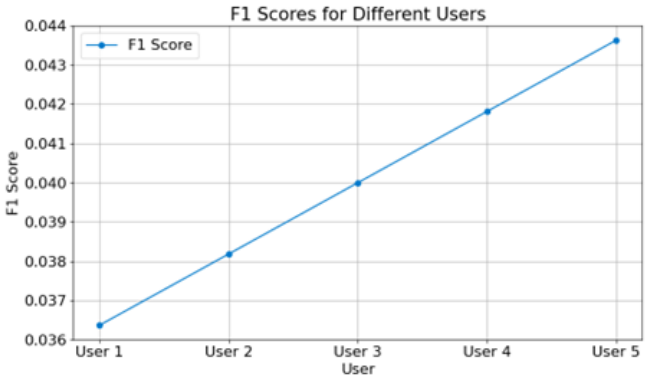

Fig. 7. F1 score plot that combines precision and recall scores to evaluate accuracy of recommendations.

many ongoing attempts made to address it. Cold start basically means that the system is unable to make recommendations to new users because it cannot draw inferences of users for whom it hasn't gained enough information. This paper presents an approach to potentially resolve this issue. The solution to this problem is a hybrid system [15] which employs the strengths of both cluster and content-based filtering techniques. Let us look at how this system implements this.

Firstly, the system employs a content-based filtering approach where the video titles are analyzed for dominant keywords so that it is able to generate recommendations for new users based on the keywords they show interest in through initial interactions. The initial interaction for the system is the set of new keywords (assumed to come from the users as stated in *Section A*) that the system is fed with. Secondly, the system uses hierarchical clustering for a broader clusterization. The outer clusters are based on general topics which can be utilized in recommending a broader content for new users, typically those that have limited historical data. As the system learns from the user, the system recommends from the inner clusters which focus on more nuanced user preferences. You can look at it as fine tuning of the broader recommendations.

Thus from the first interaction of the user, the system begins building a profile, much like the existing recommendation systems. However, unlike the other systems, this system is able to recommend even with minimal data because it picks up the dominant keywords extracted from any form of user interaction.

VIII. Conclusions and Future Scope

The goal of the project was to enhance the relevancy of recommendations in a personalized recommendation system. Given the results the system successfully recommends top relevant recommendations to the user. This project has also demonstrated the potential of caching in enhancing the performance of content delivery systems. By integrating both collaborative and content-based filtering techniques, the system not only addresses the latency issues prevalent in traditional recommendation systems but also effectively tackles the cold start problem. The implementation of a caching solution to store frequently accessed data ensures rapid retrieval and response, significantly reducing wait times and improving user satisfaction. Moreover, the system's ability to dynamically adapt to user preferences and the integration of real-time data using the YouTube API highlights its practical applicability and potential for scalability. The project thus marks a significant step towards more responsive and personalized recommendation systems, offering a wider application in various content delivery platforms. This idea can be expanded in terms of creating a scalable middleware solution as a future scope. The middleware can not only act as an interface to multiple application by standardizing the personalization process but by decentralizing the cache storage, the system can reduce the load on a single server increasing fault tolerance of the system in addition to quicker data retrieval. The middleware could implement load balancing algorithms to dynamically distribute user requests to the least occupied cache servers, increasing the scalability and reliability of the system.

The underlying algorithms that process user data can be enhanced to learn and adapt in real-time to provide more accurate and contextually relevant suggestions. This requires utilizing deep learning to understand user patterns and behaviors more effectively.

References

[1] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, dec 2016. [Online]. Available: https://doi.org/10.1145/2843948

[2] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[3] T. T. Nguyen, P.-M. Hui, F. M. Harper, L. Terveen, and J. A. Konstan, "Exploring the filter bubble: the effect of using recommender systems on content diversity," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 677–686. [Online]. Available: https://doi.org/10.1145/2566486.2568012

[4] M. Kim and S. Lee, "Reducing tail latency of dnn-based recommender systems using in-storage processing," in *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*, ser. APSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 90–97. [Online]. Available: https://doi.org/10.1145/3409963.3410501

[5] L. Li, Z. Zhang, and S. Zhang, "Hybrid algorithm based on content and collaborative filtering in recommendation system optimization and simulation," *Scientific Programming*, vol. 2021, pp. 1–11, 05 2021.

[6] S. G. Devi, D. Joseph, and P. G. Student, "Various methods of using content-based filtering algorithm for recommender systems," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:214687141

[7] N. Hazrati and F. Ricci, "Simulating users' interactions with recommender systems," in *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '22 Adjunct. New York, NY, USA: Association for Computing Machinery, 2022, p. 95–98. [Online]. Available: https://doi.org/10.1145/3511047.3536402

[8] M. Nisa, D. Mahmood, G. Ahmed, S. Khan, M. Mohammed, and R. Damaševičius, "Optimizing prediction of youtube video popularity using xgboost," *Electronics*, vol. 10, p. 2962, 11 2021.

[9] M. A. Hassan, "An efficient content-based video recommendation," *Journal of Computing and Communication*, 2022.

[10] N. Mustafa, A. O. Ibrahim, A. Ahmed, and A. Abdullah, "Collaborative filtering: Techniques and applications," in *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, 2017, pp. 1–6.

[11] S. Sahebi and W. W. Cohen, "Community-based recommendations: a solution to the cold start problem," in *Workshop on recommender systems and the social web, RSWEB*, vol. 60, 2011.

[12] M. A. Ghazanfar and A. Prügel-Bennett, "Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3261–3275, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413009214

[13] F. Ortega, J. L. Sanchez, J. Bobadilla, and A. Gutierrez, "Improving collaborative filtering-based recommender systems results using pareto dominance," *Information Sciences*, vol. 239, p. 50–61, 08 2013.

[14] H. Zheng, D. Wang, Q. Zhang, H. Li, and T. Yang, "Do clicks measure recommendation relevancy? an empirical user study," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 249–252. [Online]. Available: https://doi.org/10.1145/1864708.1864759

[15] P. B.Thorat, R. Goudar, and S. Barve, "Survey on collaborative filtering, content-based filtering and hybrid recommendation system," *International Journal of Computer Applications*, vol. 110, pp. 31–36, 01 2015.

[16] J. Liu, P. Dolan, and E. Pedersen, "Personalized news recommendation based on click behavior," 02 2010, pp. 31–40.

[17] W. Alswiti, J. Alqatawna, B. Al-Shboul, H. Faris, and H. Hakh, "Users profiling using clickstream data analysis and classification," in *2016 Cybersecurity and Cyberforensics Conference (CCC)*, 2016, pp. 96–99.

[18] T. Yoneda, S. Kozawa, K. Osone, Y. Koide, Y. Abe, and Y. Seki, "Algorithms and system architecture for immediate personalized news recommendations," in *IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 124–131. [Online]. Available: https://doi.org/10.1145/3350546.3352509

[19] H. Steck, L. Baltrunas, E. Elahi, D. Liang, Y. Raimond, and J. Basilico, "Deep learning for recommender systems: A netflix case study," *AI Magazine*, vol. 42, no. 3, pp. 7–18, Nov. 2021. [Online]. Available: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/18140

[20] F. Fessahaye, L. Perez, T. Zhan, R. Zhang, C. Fossier, R. Markarian, C. Chiu, J. Zhan, L. Gewali, and P. Oh, "T-recsys: A novel music recommendation system using deep learning," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1–6.

[21] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The youtube video recommendation system," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 293–296. [Online]. Available: https://doi.org/10.1145/1864708.1864770