TAG BASED RECOMMENDATION ENGINE USING COSINE SIMILARITY

Rohan Waghode   
*Computer Science and Engineering Data Science  
PCT’s A. P. Shah Institute of Technology*Thane, India  
rohanwaghode@ieee.org

Varad Joshi  
*Computer Science and Engineering Data Science  
PCT’s A. P. Shah Institute of Technology*Thane, India  
varadjoshi404@apsit.edu.in

*Prof. Anagha Aher   
Head of Department of CSE (Data Science),  
A.P. Shah Institute of Technology,  
Thane (M.H), India 400615  
Email:* Meet Jamsutkar  
*Computer Science and Engineering Data Science  
PCT’s A. P. Shah Institute of Technology*Thane, India  
meetjamsutkar645@apsit.edu.in*:*

Aman Dhumal  
*Computer Science and Engineering Data Science*  
*PCT’s A. P. Shah Institute of Technology*Thane, India  
amandhumal434@apsit.edu.in

***Abstract*—** **This paper introduces a revolutionary paradigm shift in recommendation systems, advocating the integration of advanced techniques to enhance the recommendation process. Traditional systems primarily rely on historical user behavior and collaborative filtering, offering valuable but often static recommendations. Our proposed approach leverages cutting-edge methods like cosine similarity calculations, tag-based recommendations, and hybrid models to achieve unparalleled recommendation precision. Key system components encompass sophisticated algorithms for tag-based recommendations, cosine similarity-driven engines, and hybrid models that seamlessly blend multiple techniques. This fusion aims to elevate the recommendation experience, providing users with highly personalized suggestions that align with both their past interactions and evolving preferences. By embracing these innovations, our system redefines recommendation systems, empowering users with dynamic recommendations that adapt in real time. This adaptability ensures that recommendations remain in sync with users' nuanced and ever-changing tastes, enriching their digital experiences across diverse domains.**

***Keywords—Tag-based recommendations, cosine similarity, collaborative filtering, content-based filtering, and personalization.***

# Introduction

In the era of information overload, recommendation engines have emerged as indispensable tools to assist users in discovering relevant content, products, and services. These systems leverage advanced algorithms and machine learning techniques to predict user preferences and provide tailored suggestions, thereby enhancing user experiences across various digital platforms. A recommendation engine, at its core, is a software system that analyzes user behavior, preferences, and interactions to generate personalized recommendations. This paper explores the pivotal role of tags in recommendation systems and how they contribute to refining the recommendation process, the services recommendation engines employ, and the importance of a feedback loop in continually improving their performance. Recommendation engines rely heavily on tags, which are descriptive labels or keywords associated with items in a database. These tags serve as metadata, providing valuable information about the characteristics, content, or attributes of

the items. Tags enable recommendation engines to categorize

and understand the items, as well as relate them to user preferences. By associating tags with user profiles and their interactions, recommendation engines can make more precise recommendations. For instance, if we consider twitter, it employs a recommendation engine to enhance user engagement and content discovery. It analyzes user behavior, preferences, and interactions to suggest tweets, accounts to follow, and trending topics, thereby curating personalized timelines, and expanding user interaction within the platform. This recommendation system aims to keep users engaged and connected by delivering content that aligns with their interests and preferences. To provide personalized recommendations, recommendation engines utilize a range of services and techniques, including collaborative filtering, content-based filtering, and hybrid methods. Collaborative filtering analyzes user behavior and interactions to find similarities between users, recommending items that similar users have liked. Content-based filtering, on the other hand, relies on the attributes and tags associated with items to recommend items with similar characteristics to those the user has shown interest in. Hybrid methods combine both approaches to provide more accurate and diverse recommendations. Crucially, recommendation engines do not operate in isolation; they thrive on feedback loops. Continuous feedback from users, such as ratings, reviews, and user interactions, is invaluable for refining recommendations over time. This iterative process ensures that the recommendation engine adapts to changing user preferences and delivers increasingly relevant suggestions.

In this paper, we delve deeper into the significance of tags in recommendation engines and their role in enhancing recommendation accuracy. Additionally, we discuss the various services that recommendation engines employ to create tailored recommendations and the pivotal role of feedback loops in maintaining and improving their performance. By understanding these key components, we can shed light on the mechanisms behind effective recommendation systems and pave the way for future enhancements in personalized content discovery.

.

# A diagram of a software flowchart Description automatically generatedLITERATURE REVIEW

In the ever-evolving landscape of recommendation systems, a multitude of research papers have emerged to advance our understanding and improve user experiences.

"Recommendation Systems: Principles, Methods, and Evaluation" [1] offers an in-depth exploration of recommendation systems, emphasizing their indispensable role in enhancing user experiences, particularly in the realms of e-commerce and content streaming. This paper delves into various aspects, including the fundamental principles, methodological approaches, and the critical evaluation criteria that underpin these systems.

"Content-Based Filtering for Recommendation Systems Using Multiattribute Networks" [2] introduces an innovative methodology leveraging multiattribute networks to alleviate the challenges posed by information overload in recommendation systems. Empirical evidence provided in the study showcases the notable improvement in recommendation accuracy achieved through this approach.

In "Learning similarity with cosine similarity ensemble" [3], the authors introduce an ensemble technique tailored to enhancing similarity assessments, with a specific focus on the widely used cosine similarity measure.

Dehak et al.'s 2010 research [4] delves into cosine similarity scoring without the need for score normalization, particularly in the context of speech and language processing.

Li, Guo, and Zhao's 2008 paper [5] explores the concept of "Tag-based Social Interest Discovery," shedding light on how this approach aids in comprehending user interests within online social environments.

"Neural Collaborative Filtering" [6] represents a significant advancement in recommendation systems, integrating neural networks into collaborative filtering to optimize personalized recommendations. This approach leverages the power of deep learning to enhance the quality of recommendations, aligning them more closely with individual user preferences and behaviors.

1. ARCHITECTURE

In this chapter, we delve into the core components and mechanisms of our recommendation system. We'll explore how user interests are dynamically generated and updated

through the Score Generator and how user feedback is incorporated to enhance accuracy. Additionally, we'll uncover the intricate process of recommendation generation, which leverages both collaborative filtering and content-based filtering for optimal results. This chapter provides a comprehensive understanding of the system's architecture, setting the stage for a detailed analysis of its performance and effectiveness in subsequent sections.

Figure (System Architecture)

**Post:**

A Post is a piece of content that will be recommended to the User based on the decision made by the Recommendation Engine.

*post\_tags={postid=post1: tag1=yes, tag2=no, tag3=no ... tagn=yes}*

While processing, all "yes" values become 10 and no become 0 for cosine similarity calculation.

**Tags:**

Tags are the attributes given to our Posts which help us establish an analogy between a post and the User. Each post can have multiple tags describing itself. Tags can either be set by reviewers, post author or automatically by using NLP based on different use cases. In our testing, the Post Author gets to set its Tags.

**User Interests:**

User interests track the interest of a user in different categories of Posts by assigning a value, hereon referred to as Score, to each Tag. This is generated by the "Score Generator" part of the recommendation engine and serves as a basis for generating Recommendations.

*user\_interests={user\_id=u1: tag1=score, tag2=score, tag3=score ... tagn=score}*

where the score ranges between 0 to 10.

User Interactions:

User Interactions refer to the various things a user might do on the platform. In our case, we are using a simple implementation which records the timestamp, post id, user id and action. Based on what action is being done, there is provision in the recommendation engine to provide more or less weightage to the interest of the user in that post. Interactions are also our main basis for calculating Analytics, both Post analytics as well as Engine analytics. User interactions are also used by the Score generator to generate User Interests. For more advanced analytics, we can add more user data such as region, language, gender, age group and get analytics based on those factors.

*user\_interactions={userid=u1, postid=post1, event:click}*

Score Generator:

The score generator is the first part of our recommendation engine which generates the User Interests which can then be compared with Post Tags in order to decide whether to recommend a post or not. Following is the example of how the score generator generates user interests.

Iterate over users:

Retrieve existing User Interests from the database and store them in array A.

Let us assume some values for array A.

*A = {0, 1, 5, 3, 10} ------- (1)*

Next, we check the database for user interactions and retrieve the tags of every post the user has interacted with since the last score generation. Let us assume this data is saved in a two-dimensional array B where the rows represent Posts and columns represent tags. Let us assume data for array B

*B = {0, 10, 0, 10, 0*

*10, 0, 10, 0, 10*

*A mathematical equation with numbers and a bar code

Description automatically generated 0, 0, 10, 0, 0*

*10, 10, 0, 0, 0*

*10, 0, 10, 0, 0} ------- (2)*

Now, the score generator takes summation of all rows and stores it in an array C.

*C = {30, 20, 30, 10, 10} ------- (3.1)*

Array C represents the latest interests of the User. The array C is now normalized to 10 by dividing throughout by 3.

*C = {10, 6.67, 10, 3.34, 3.34} ------- (3.2)*

Now, we have array A representing the user's historical interests and array C representing the user's latest interests. We average both the arrays, and the resulting data is updated into the database. In this case, let us assume the result is array D.

*D = {5, 8.34, 7.5, 3.17, 1.67} ------- (4.1)*

Rounding to nearest integer,

*D = {5, 8, 7, 3, 2} ------- (4.2)*

This way, the scores never become stagnated, and the user never gets recommended something that he is no longer interested in.

**Feedback Loop:**

Our primary way of generating latest user interests is by aggregating past user interests and their latest interactions. However, sometimes a stray tag might get too weighted while the user isn't really interested in it. This is why we introduced a Feedback Loop which the user can use for letting the system know that they are not interested in a particular post/tag. These posts/tags are negated while calculating user interests by the score generator. Doing this improves the accuracy of User Interests, thereby reducing the error factor while generating recommendations and enhancing prediction quality.

**Recommendation Generation:**

For generating recommendations, we use the cosine similarity algorithm along with clustering of Users as well as Posts, thus making it a hybrid system. A hybrid system is the one that implements both, collaborative filtering as well as content-based filtering.

Cosine Similarity: Cosine similarity is a metric used to measure the similarity between two vectors. We can use Post Tags and User Interests as the dimensions for 2 distinct vectors and compare them using cosine similarity, as follows.

Let array A represent User Interests and array B represent Post Tags. Assuming data,

*A = {0, 10, 10, 0, 10, 0, 0, 10} ------- (5)*

*B = {0, 4, 6, 1, 4, 0, 3, 10} ------- (6)*

By formula, we first need to find the dot product between A and B vectors as well as the magnitude of A and B.

*A.B = 240*

*|A| = 20*

*|B| = 13.341*

Now, *cosine similarity = 240/(20\*13.341) = 0.8988*

Based on the value of cosine similarity (0.8988), we can assume that vectors A and B are similar. The closer the value of cosine similarity is to 1, the better fit it is.

**Collaborative Filtering:** Collaborative filtering is a method which analyses the history of Users with similar interests and recommends them the same Posts. In our implementation, we use collaborative filtering to divide Users into categories by using cosine similarity to find users with similar interests.

**Tag Based Clustering:** Tag-based clustering is a data analysis technique used to group or categorize Posts on the basis of Tags. We already have categories in place and hence do not need to do any additional processing for this.

Using the methods given above, we obtain User clusters and Categories. In order to generate recommendations, the first step is to generate scores for Categories using the same score generator used for users. Once obtained, we can compare User clusters with Categories using cosine similarity and rank the outcomes on the basis of the outcome of cosine similarity. These recommendations can be served to all the users from a cluster.

1. RESULT AND ANALYSIS

In this section, we delve into the results and analysis of our recommendation engine based on the system architecture described in the previous section. We begin by presenting the outcomes of the score generation process, followed by an examination of the feedback loop's impact on user interests. Finally, we discuss the recommendation generation process, including the application of cosine similarity, collaborative filtering, and tag-based clustering to provide users with personalized recommendations. We will also discuss how the recommendation engine provides insights regarding its own accuracy for further enhancing its effectiveness.

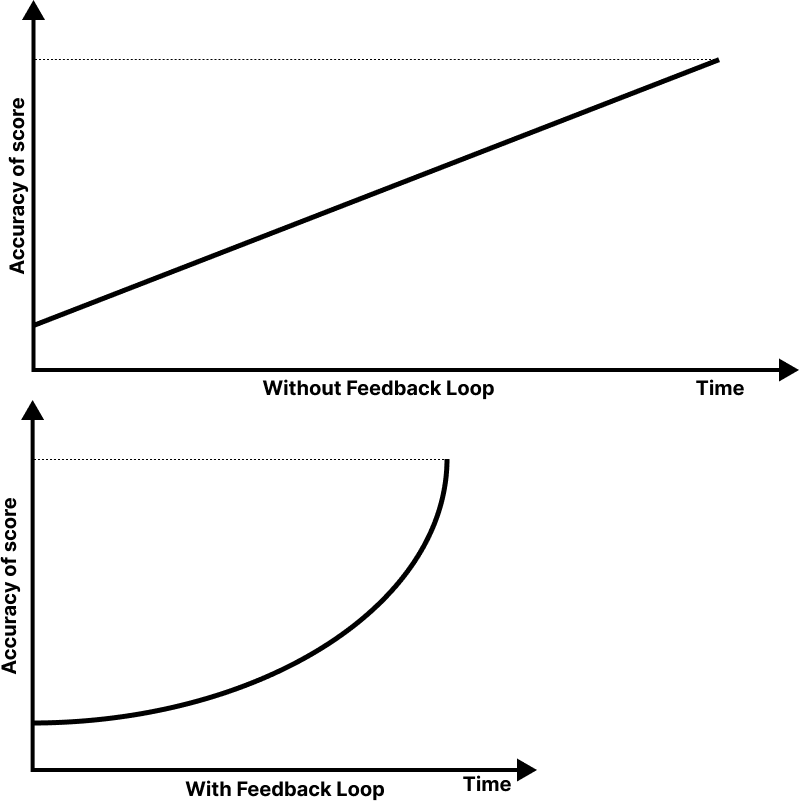


Figure (Feedback Loop)

The process of score generation incorporates User inputs to generate User Interests and feedback loop assists in reducing the factor of error that might be prevalent in such a system. Furthermore, since every interaction gets weighted differently, the chances of having error diminish even further. We can see the difference between a system without feedback loop and a system with feedback loop in figure (Feedback Loop). It can be observed that the system with a constant feedback loop will achieve the same accuracy in less time as the user has more control over their interests.

A black background with a black square

Description automatically generated with medium confidence

Figure (Cosine Accuracy)

The recommendation generation process incorporates three different levels of filtering and analysis in order to generate most accurate recommendations with least number of comparisons. If we consider it to just use cosine similarity, the number of comparisons will equal (m\*n) where m represents the number of active users and n represents the number of posts. If we consider a small platform with 100 active users and 1000 posts, the number of comparisons becomes 1,00,000. As we can observe, this will grow even more for mid-sized platforms and is literally unusable for platforms with a larger audience. Furthermore, in order to get a reasonable number of recommendations, we need to use a threshold for cosine similarity around 0.5. This makes the mean accuracy, ignoring errors in tag assignment and user interests to be around 50%. (Cosine accuracy)

In order to optimize this, we first categorize our Posts based on their Tags. This reduces the number of comparisons to (m\*n) where m represents the number of active users and n represents the number of categories. The number of categories will always will less than the total number of Posts. Along with these benefits, we also have higher accuracy in cosine similarity because our Category scores are also in a range of 0-10 as opposed to each Post tags, which are either 0 or 10. In order to get recommendations nearly the same as with just using cosine similarity, we can now use tighter thresholds for cosine similarity and hence our simulated mean accuracy, ignoring errors in tag assignment and user interests to be higher than before.

In our system, we take this a step further and consolidate Users into clusters as well. This further reduces the number of comparisons. Also, we can use much tighter thresholds now and still get a large amount of recommendations. For example, we can set a threshold of 0.9 cosine similarity to categorize Users into similar clusters and then categorize the results of comparing User cluster with Category in ranges like 0.7-0.79 cosine similarity, 0.8 to 0.89 cosine similarity and 0.9 to 1.00 cosine similarity, we get a large amount of recommendations and the simulated mean accuracy for each range will be as follows:

For 0.7 to 0.79, it will be (0.7) \*(0.9) =63%

For 0.8 to 0.89, it will be (0.8) \*(0.9) =72%

For 0.9 to 1.00, it will be (0.9) \* (0.9) =81%

A black background with a black square

Description automatically generated with medium confidenceThis can be visualized by figure (engine accuracy)

Figure (Recommendation Engine Accuracy)

Apart from the advantages discussed above, another huge advantage of our system is that there are 2 levels of processing happening with different levels of complexity. At one level, User clusters and Post categories are processed to generate recommendations in general for all clusters. This takes much less processing power but only makes sense to run after there has been significant interactions. Meanwhile, we can generate thresholds to fit even new users into these clusters and provide them with recommendations in real time as score generation and comparisons can be offloaded to client side.

The recommendation engine is equipped with the ability to analyse its own effectiveness and report it back to the system administrator every time it is run. This is reported in the form of hits/recommendations ratio which is calculated based on how many recommendations served to the user were actually clicked. Using this data, the thresholds can be reworked and experimented with till we have the most optimal results.

1. CONCLUSION

Through this research, we have learned the importance of adaptability in recommendation systems. User preferences are dynamic, and our system's ability to continuously update and refine recommendations in response to evolving interests is a crucial feature. We have also gained insights into the significance of user feedback in reducing recommendation errors and enhancing prediction accuracy. Additionally, exploring more advanced machine learning techniques and considering real-time user interactions could refine our recommendation engine even further.

In summary, our research has illuminated the path to a recommendation engine that not only adapts to users' changing preferences but also actively engages them in the improvement process. By leveraging the lessons learned and exploring future enhancements, we can continue to elevate the user experience and deliver increasingly accurate and relevant recommendations.

##### References

1. Isinkaye, Folasade Olubusola, Yetunde O. Folajimi, and Bolande Adefowoke Ojokoh. "Recommendation systems: Principles, methods and evaluation." \_Egyptian informatics journal\_ 16.3 (2015): 261-273.
2. Son, Jieun, and Seoung Bum Kim. "Content-based filtering for recommendation systems using multiattribute networks." \_Expert Systems with Applications\_ 89 (2017): 404-412
3. Xia, Peipei, Li Zhang, and Fanzhang Li. "Learning similarity with cosine similarity ensemble." \_Information sciences\_ 307 (2015): 39-52.
4. Dehak, Najim, et al. "Cosine similarity scoring without score normalization techniques." \_Odyssey\_. 2010.
5. Li, Xin, Lei Guo, and Yihong Eric Zhao. "Tag-based social interest discovery." \_Proceedings of the 17th international conference on World Wide Web\_. 2008.
6. He, Xiangnan, et al. "Neural collaborative filtering." \_Proceedings of the 26th international conference on world wide web\_. 2017
7. Koren, Yehuda, Steffen Rendle, and Robert Bell. "Advances in collaborative filtering." \_Recommender systems handbook\_ (2021): 91-142
8. Jannach, Dietmar, et al. \_Recommender systems: an introduction\_. Cambridge University Press, 2010