iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** IX **Month of publication:** September 2021

DOI: https://doi.org/10.22214/ijraset.2021.38246

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IX Sep 2021- Available at www.ijraset.com

A Hybrid Recommendation System for Video Games: Combining Content-based & Collaborative Filtering

Aditya Manikantan
Department of Computer Science and IT, Ramnarain Ruia Autonomous College, Mumbai - 400019, India

Abstract: ***Recommending video games can be trickier than movies. When it comes to selecting a video game, many factors are involved such as its genre, platform on which it's played, duration of main and side quests, and more. However, recommending games based on just these features won't suffice as a person who, for example, enjoys a certain genre of game can equally enjoy a vastly different genre. Therefore, a scoring mechanism is required which takes into account both, features of a game (content-based filtering) and also studies the buying patterns of people playing a particular game (collaborative filtering). In this paper I have proposed a way to take into account both content-based and collaborative filtering into the final recommendation. I have used cosine similarity to quantify the similarity between the features of games. Along with this, I have employed a Deep fully-connected AutoEncoder (DAE) to generalize the implicit data representation of an user's buying patterns. Finally, I present a novel approach to combine the scores of these filtering techniques in such a way that it gives equal weightage to both. In other words, they both have equal influence over the final list of the top 10 games recommended to the user.***
Keywords: Hybrid Recommender, Collaborative filtering, Content-based filtering, Cosine similarity, AutoEncoder.

I. INTRODUCTION

The video games industry is one of the biggest entertainment industries in the world. In 2018 alone it generated more revenue than both the music and movie industries combined [1]. This can be attributed to the fact that there is a huge selection of games, multiple platforms to play on, social functionalities such as ability to talk and play with players anywhere in the world. Even gaming on smartphones has become the favorite pastime of many.All this has led video games to be attracted to a larger demographic than ever before. There is something for everyone. Along with this, video games share many similarities with traditional games in the sense that they encourage competitiveness, problem solving and cooperating with others [2]. Studies have shown that video games can improve cognitive and social skills in adults and also enhance the overall mental well-being of an individual [3]. Due to this many individuals choose video games as a recurrent pass time. This has encouraged many studios and developers to produce a more diverse variety of games which will not only appeal to a larger demographic ranging from men, women, children, and adults but also persuade them to keep coming back to play more.

Thousands of games are released each year by big AAA studios or small indie developers across multiple genres and platforms. The rate of games produced each year is increasing at a staggering rate [4]. Also, unlike movies the length of games can vary drastically ranging from short games which could last about 2-3 hours to even games that take around 200-300 hours to complete. Currently there are nine genres of video games each having their sub-genres [5]. The wide selection of games coupled by the time it takes to complete one can prove to be a daunting task for a buyer. Currently most users select games by first checking their reviews on blogs, forums, articles or YouTube. However, due to the astounding number of games released each year it becomes impossible for a reviewer to objectively review every single game while on the other hand it is impractical for us to scour the internet for reviews of every single game that comes out. This is where a recommender system comes into play.

One of the most important things when it comes to recommending a game is its features such as genre, platform, developer and so on. Recommending games based on it's features is called content-based filtering. Content-based filtering is most popularly used to recommend books, websites, movies, products etc. Informative descriptors of the media should exist for this type of filtering. However, this alone does not capture every aspect of what a person may enjoy in a game. A pair of people who enjoy the same genre of games on the same platform may prefer a vastly different selection of games. There could be some implicit indicators which influence what's enjoyable about a game to the user. Content-based filtering alone would fail to identify these latent features. So, the recommendations can be improved if it is used in conjunction with collaborative filtering [6]. Collaborative filtering takes into account the opinions and buying patterns of users who like content similar to yours. On that account, the recommendations can be of higher quality and have higher coverage of games if they are both used in complement to each other into a single learning framework.

In this paper, for content-based filtering I have used cosine similarity [7], which is a similarity metric, to measure the similarity between games between its features. This sort of provides a baseline recommendation based on the games the user likes. For collaborative filtering, I have used a Deep full-connected AutoEncoder [8] which is one of the architectures of an AutoEncoder (AE) [9] to study the buying patterns of users. Finally, I have presented a way to combine the content-based scores and collaborative scores in such a way that it gives equal weightage to both. In other words they both have equal influence over the final list of top ten games recommended to the user. So in this framework, the user enters a list of games he/she enjoys and based on these games the user is recommended the top ten games based on the above framework. Furthermore, there could be millions of different combinations of games different players could like and it is impractical to train the AutoEncoder on all these combinations. Also, these combinations could keep expanding as new games are introduced every year. So, to make this system more robust and future-proof, the neural network trains in real-time each time a new user inputs a list of games they like.

II. RELATED WORK

Several papers have been written presenting different solutions to address the challenges regarding recommendation systems. The translation of these systems when it comes to recommending video games is still in its infancy as compared to the systems developed for movies. Despite this, there are some noteworthy papers which I have come across which have good applicable solutions to this problem.

Cheuque et al. [10], for example, have put emphasis on online games. The authors have explored the potential of receiving multiple and different types of input by implementing state-of-the-art recommender models based respectively on Factorization Machines (FM), deep neural networks (DeepNN) and one derived from the mixture of both (DeepFM). Syed Anwar et al. [11], have suggested a collaborative filtering technique which uses individual ratings given by the members of community, along-side rating of the games that a particular gamer likes, in order to predict and recommend new games to that gamer.An ensemble-based model and a deep neural network was evaluated and compared by Paul Bertens et al. [12] in their paper to help users automatically get personalized recommendations while playing. They have employed machine learning algorithms which are able to predict ratings for a particular game by a user. Certainly, I am not the first to think of a hybrid solution for recommendation systems. Nicholas Crawford [13] in his paper has proposed a solution to the cold start problem by using a hybrid of neural network and keyword ranking. Gong J. et al. [14] have taken advantage of social facilities emerging in online game platforms. In their paper, the authors have suggested an hybrid approach where in addition to producing suggestions based on the descriptors of games played by the user such as play time, price, release date, it also takes the game preferences of their friends into account. In this paper, I have presented a way to recommend games which does not require games to have descriptors for collaborative filtering and also the users are not required to rate the games.

III. MATERIALS AND METHODS

A. *Data*

The data containing the features/descriptors of the games was collected from VGChartz [15].

List of features were:

1) Name
2) Genre
3) Platform
4) ESRB rating
5) Publisher
6) Developer
7) Rating
8) All styles average playtime
9) Main quest average playtime

These features encompass most of the common indicators based on which users buy games such as genre, platform, rating even their favorite publisher or developer. It is common practise among users to buy games from their favorite developer or publisher. Moreover, to take into account the different play styles of users (some users like to play the game to completion which includes all the side quests and challenges while some just prefer to play the main quest to enjoy the story), the list of features include the average playtimes of these different styles. The dataset containing the purchase records of users was obtained from Kaggle [16] which was scrapped from Steam, an online platform to play, purchase and discuss video games. This dataset contains the record of over a thousand users.

B. Content-based filtering

1) Cosine similarity: For content-based filtering I used cosine similarity to measure the similarity between games based on its features. Cosine similarity is mainly used in natural language processing, more commonly to measure similarity between two documents. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space [17]. So, smaller the angle, higher the similarity. Cosine similarity is a more robust approach as in euclidean distance the distance between two vectors can be large if the document is large even though they are actually similar. It is represented by the formula:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where A and B are the vectors. This metric returns a value between 1 and -1, with 1 denoting that the two vectors are perfectly similar and -1 denoting that they are perfectly dissimilar. So to begin, the data had to be preprocessed accordingly where each row is a vector of features of a video game.

TABLE I
Sample of How Features of Different Data Types were Encoded

Name	Playtime (hours)	Rating	Genre1	Genre2	Publisher1	Publisher2	Developer1
Game1	14	4.5	0	1	0	1	1
Game2	5	9.3	1	1	1	0	0
Game3	64	7.8	1	0	1	0	0

This features a dataset containing numerical as well as categorical data. As shown in Table 1, the numerical columns such as playtime and rating were left as it was. However, as cosine similarity can only take vectors containing numerical values as input, the categorical features such as genre, publisher, developer, and ESRB rating were one-hot encoded. This creates an array of integers where the value of 1 denotes the respective category it represents.

	Psychonauts	Silverfall	Remember Me	Borderlands 2
Psychonauts	1.000000	0.067970	0.998268	0.985335
Silverfall	0.067970	1.000000	0.080187	0.021432
Remember Me	0.998268	0.080187	1.000000	0.973617
Borderlands 2	0.985335	0.021432	0.973617	1.000000

Fig. 1 A sample of the final similarity matrix with similarity scores

After computing the similarity scores for each game I formed a square matrix with the number of rows and columns equal to the number of games. Each row is a vector containing similarity scores of a game with the corresponding game in the column. Fig. 1 shows a sample of the final similarity matrix containing the cosine similarity scores of all the games with each other. The final vector of scores is obtained by taking similarity scores vectors of each game input by the user and taking the mean over all these vectors.

C. Collaborative filtering

1) Vectorizing: The data containing the buying records consisted of two columns, one denoting the user id of a user and second denoting the game the user purchased. This data had to be preprocessed as the AutoEncoder could only take vectors as input. So, the dataset had to be pivoted with each row representing the games the respective user bought and each column representing a game. Each row was one-hot encoded. As shown in Fig. 2, the value 1 and its corresponding position in the vector indicated the game a user bought.

game	Psychonauts	Silverfall	Remember Me	Borderlands 2
user_id				
5250	0.0	0.0	0.0	0.0
76767	0.0	0.0	0.0	0.0
86540	0.0	0.0	0.0	1.0

Fig. 2 Dataset after one-hot encoding. Each row represents the games a user bought.

2) Deep fully-connected AutoEncoder (DAE): There can be some latent factors other than the basic descriptors of games (such as genre, platform etc.) which can influence a user to buy a certain game. To learn the latent data mappings between the users and the games they bought I decided to use an unsupervised learning algorithm. For this I implemented a Deep fully-connected AutoEncoder. The model aims to find these implicit data representations during training. The model specifications are shown in Table 2. There are several other unsupervised algorithms such as the Restricted Boltzmann Machines (RBM) [18] which can be used for collaborative filtering. However, RBMs can be tricky to train as they have to approximate the log-likelihood gradient [19] i.e they are probabilistic whereas AEs are deterministic. So, I went with AEs as they are easier to train and can learn more consistent latent representations.

TABLE II
Parameters of the Deep Fully-connected AutoEncoder

Parameter	Value
Number of layers	8
Number of neurons in input layer	457
Number of neurons in hidden layers	Between 32 and 128
Number of neurons in output layer	457
Total trainable parameters	154,857
Activation function	ReLU
Activation function in output layer	Softmax
Optimizer	Adam
Learning rate	0.001
Decay rate	0.00001
Loss metric	Mean squared error (MSE)

The AE architecture consists of two parts, an encoder, and a decoder. The encoder takes the input vector and compresses it into a latent-space representation. This representation can correspond to the latent mappings between a user and their games. The output is then reconstructed from this representation. Fig. 3 illustrates a simple delineation of this architecture.

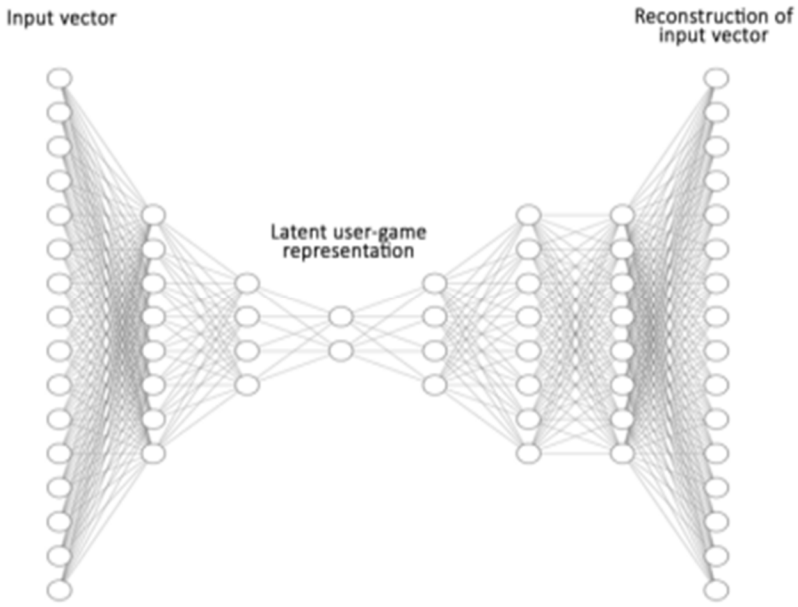


Fig. 3 Simple representation of the Deep fully-connected AutoEncoder. (The number of neurons depicted is different from the actual model.)

This model was trained for 200 epochs with a batch size of 128 on 9044 samples and tested on a 1000 samples. The best model achieved a test loss of 1.9206e-04.
Furthermore, there are over a 1000 games released each year and over a million different combinations of games different users could like. It becomes an impractical and almost an impossible task to train the model on these millions of different combinations of games. So, to help simplify this process, the DAE not only recommends games based on the user input but also trains itself in real-time on the given input.

D. Combining content-based and collaborative filtering
Both the content-based filtering and collaborative filtering returns a vector of length equal to the number of games with the index in the vector corresponding to the same game in both. The cosine similarity scores on average score between 0.7 and 0.99. On the other hand, as the softmax function in the DAE calculates the score based on the number of neurons in the output, which causes the scores to be miniscule in comparison. This leads to the cosine scores severely overpowering the DAE scores which causes the final recommendation to incline towards the content-based scoring. So, to counter this, the collaborative scores were scaled up by using min-max scaler. This new score values were calculated using the formula:

$$\text{Scaled up value} = \frac{\text{Initial Value} - \text{Minimum value}}{\text{Maximum value} - \text{Minimum value}}$$

This helps both types of filtering have equal influence over the final list of recommended games. In practise, this drastically improved the quality of recommendations as compared to before min-max scaling.
The scores of these two vectors are then added and divided by 2 to scale the values between 0 and 1. Finally, all the scores are then sorted in descending order and the corresponding top 10 games are selected as a recommendation to the user. Fig. 4 below gives a basic overview of this entire recommendation framework.

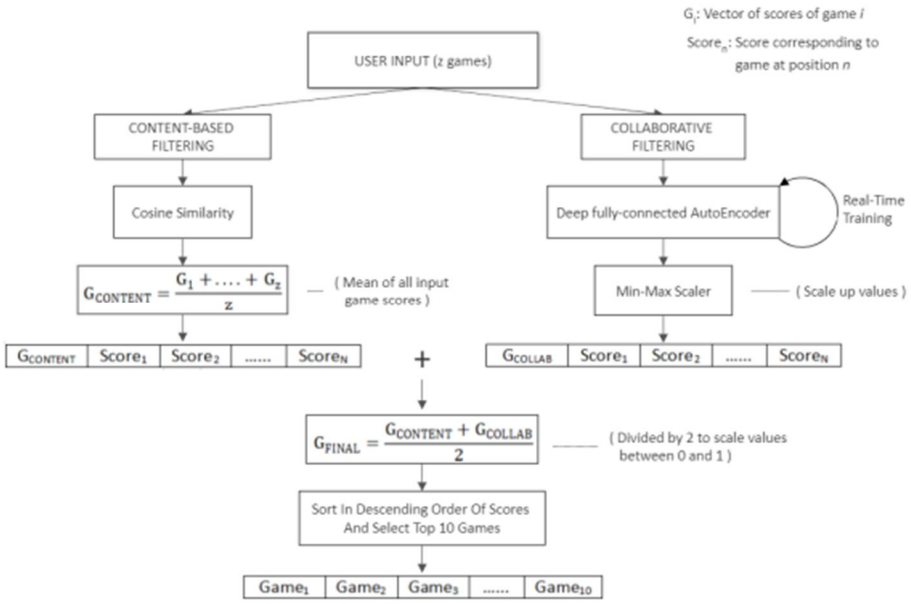


Fig. 4 Process flow of the recommendation framework

IV. RESULTS

The final hybrid recommender was tested along with its constituent components (collaborative filtering and content-based filtering) individually on 30 users. A segment of games bought by a user was taken as input and the three recommenders had to predict the remaining games he/she bought based on the input. The mean squared error (MSE) was calculated between the actual games and the recommended games. Finally, the aggregate MSE for each system was calculated. Fig. 5 displays the comparison of the average MSE between the models discussed above.

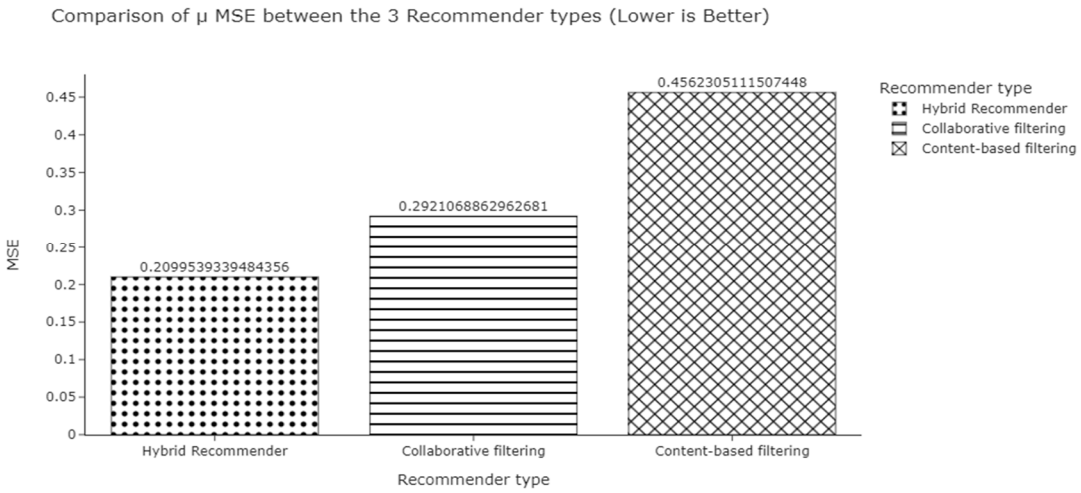


Fig. 5 Comparison of μ MSE between the 3 recommender types

Firstly, we see that content-based filtering with cosine similarity has performed the worst with the highest average mean squared error of 0.4562. Collaborative filtering on the other hand has performed comparatively better, with an average MSE of 0.2921, indicating that learning the buying patterns of users can help give more accurate recommendations when compared to recommending games based on just its features. Lastly, the hybrid recommendation system performed the best with an average MSE of 0.2099, which is 54% lower than content-based filtering and 28% lower than collaborative filtering. This suggests that the hybrid recommender leads to recommendations better tailored to that particular user.

Upon further inspection regarding the game's descriptors, i.e upon testing with games across different genres and platforms such as adventure, role-playing, sports, and puzzle on PS4 and Xbox One, led to recommendation of games from genres such fantasy, sci-fi, horror on smartphones and PC in addition to ones from the input. This indicates that the hybrid recommender has a good coverage of different features of the games and is not limited to the descriptors of the input.

V. CONCLUSION AND FUTURE WORK

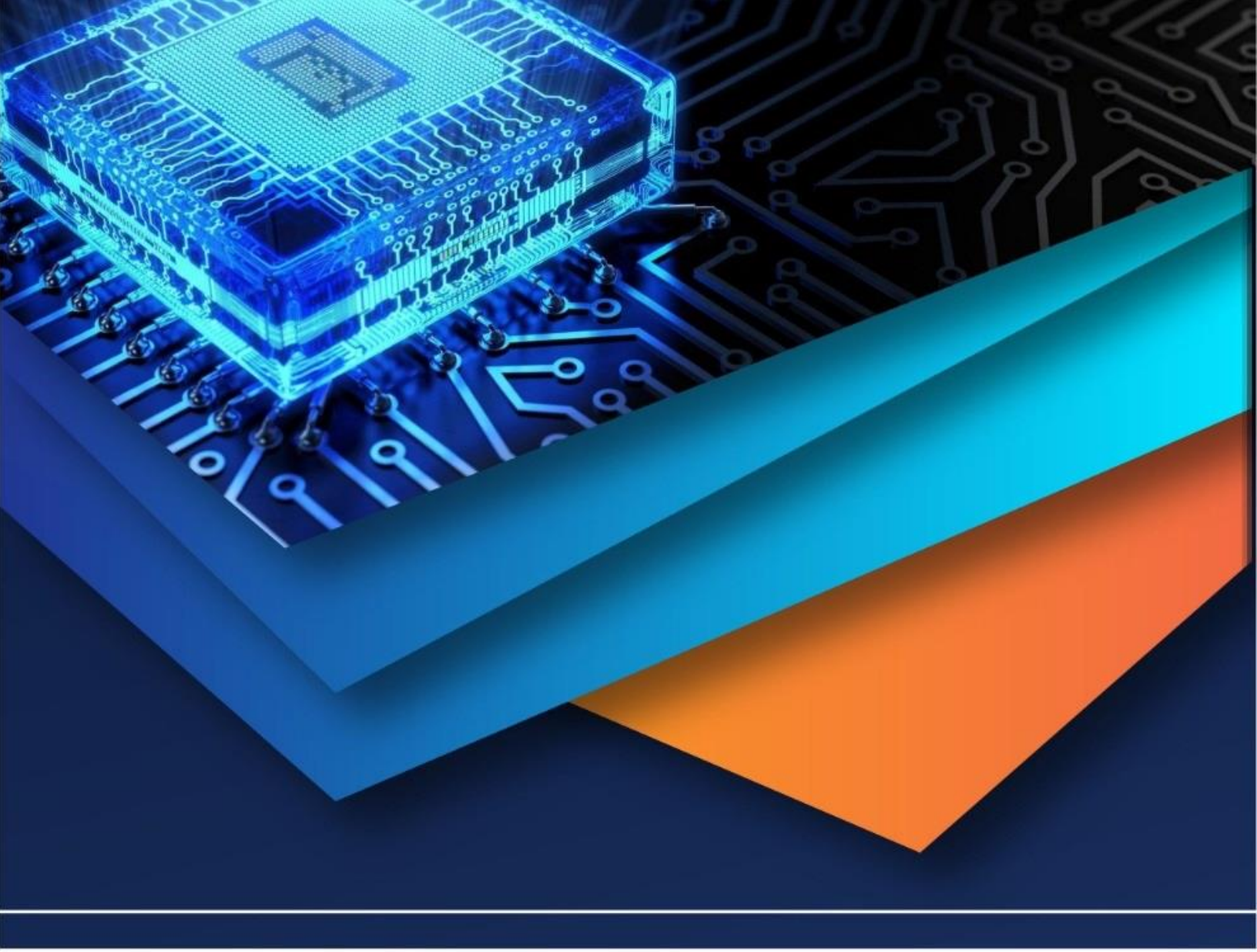
In this paper, I have presented a novel approach on how content-based filtering using cosine similarity and collaborative filtering using a Deep fully-connected AutoEncoder can be combined into a single hybrid recommendation system. The implementation presented ensures that both types of filtering have equal influence over the final list of the top 10 games recommended to the user. Moreover, I have also stated how this hybrid system demonstrates a good coverage regarding the features of the games such as its genre, platform, developer as so forth.

The hybrid system and its constituent filtering methods was trained on 2 datasets, one containing informative descriptors of games and other containing the buying records of different users. Upon evaluation, the hybrid recommender was found to outperform both content-based filtering and collaborative filtering by attaining the lowest average mean squared error of the three. Additionally, the AutoEncoder model trains in real-time whenever a user inputs a new list of games he/she likes. Unlike traditional collaborative systems, this does not require the features of the games or for users to rate games. The Deep fully-connected AutoEncoder gives more consistent results than a Restricted Boltzmann Machine as it is deterministic in nature and not probabilistic. Finally, using cosine similarity for content-based filtering helped in further improvement as it does not negatively impact the similarity score based on the size of the vectors unlike euclidean distance.

In future, I plan to expand on this approach by including the demographics of users. By analyzing users across different age gaps, gender, income group and location and mapping this to features of a game can lead to more precise recommendations. Furthermore, understanding the sentiments of users towards a game through their tweets or reviews can also prove to be a useful inclusion in recommending games.

REFERENCES

[1] Samuel Stewart. Video game industry silently taking over entertainment world. 2019
[2] Isabela Granic, Adam Lobel, and Rutger C. M. E. Engels. The Benefits of Playing Video Games. 2014. DOI: 10.1037/a0034857
[3] Federica Pallavicini, Ambra Ferrari and Fabrizia Mantovani. Video Games for Well-Being: A Systematic Review on the Application of Computer Games for Cognitive and Emotional Training in the Adult Population. 2018. https://doi.org/10.3389/fpsyg.2018.02127
[4] Number of games released on Steam worldwide from 2004 to 2021. At: https://www.statista.com/statistics/552623/number-games-released-steam/
[5] Vince. The Many Different Types of Video Games & Their Subgenres. 2018
[6] Schafer, Ben & J, Ben & Frankowski, Dan & Dan, & Herlocker, & Jon, & Shilad, & Sen, Shilad. Collaborative Filtering Recommender Systems. 2007
[7] D Gunawan et al 2018 J. Phys.: Conf. Ser. 978 012120
[8] Francois Chollet. Building AutoEncoders in Keras. 2016.
[9] G. E. Hinton, R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. 2006 : 504-507
[10] Cheuque, Germán & Guzman Gomez, Jose Antonio & Parra, Denis. Recommender Systems for Online Video Game Platforms: the Case of STEAM. 2019. 63-771. DOI: 10.1145/3308560.3316457.
[11] Anwar, Syed & Shahzad, Talha & Sattar, Zunaira & Majid, Muhammad. A game recommender system using collaborative filtering (GAMBIT). 2017. DOI: 328-332. 10.1109/IBCAST.2017.7868073.
[12] P. Bertens, A. Guitart, P. P. Chen and A. Perianez. A Machine-Learning Item Recommendation System for Video Games. IEEE Conference on Computational Intelligence and Games (CIG). 2018. pp. 1-4, DOI: 10.1109/CIG.2018.8490456.
[13] Crawford, Nicholas. Improving Video Game Recommendations Using a Hybrid, Neural Network and Keyword Ranking Approach. 2019.
[14] Gong J., Ye Y., Stefanidis K. A Hybrid Recommender System for Steam Games. In: Flouris G., Laurent D., Plexousakis D., Spyratos N., Tanaka Y. (eds) Information Search, Integration, and Personalization. ISIP 2019. Communications in Computer and Information Science, vol 1197. Springer, Cham. 2020. https://doi.org/10.1007/978-3-030-44900-1_9
[15] VGChartz. Retrieved from: https://www.vgchartz.com/gamedb/. 2020
[16] Tamber. Steam Video Games. Retrieved from: https://www.kaggle.com/tamber/steam-video-games. 2020
[17] Selva Prabhakaran. Cosine Similarity – Understanding the math and how it works. October 22, 2018
[18] Ruslan Salakhutdinov, Andriy Mnih, Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. 2007
[19] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. 2011



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY