Movie Recommender System using Deep Autoencoders

Rudrani Bhadra
Department of Mathematics
University of Waterloo
Waterloo, Canada
r2bhadra@uwaterloo.ca

Lipsa Mishra
Department of Mathematics
University of Waterloo
Waterloo, Canada
lmishra@uwaterloo.ca

Lavanya Sharma
Department of Mathematics
University of Waterloo
Waterloo, Canada
l9sharma@uwaterloo.ca

***Abstract*—A recommender system is a filtering system that helps to filter out the contents based on the user's preferences and choices. Collaborative and content based recommendation systems are the two most popularly used filtering systems. In this paper, we are building a recommendation system using collaborative filtering, content based filtering and hybrid filtering using deep autoencoders in neural networks to recommend movies to the user. A hybrid filtering system uses the features of both collaborative and content filtering system to provide better recommendations. To create the hybrid model, we accumulated the features of content and collaborative filtering and build a recommendation model. The experiments demonstrate the behavior of the three recommendation systems on the MovieLens data set. In this paper, different activation functions like ReLUs (rectified linear Units), SELU (scaled exponential linear unit) and Tanh (hyperbolic tangent) on MSE (mean squared loss) of autoencoders are investigated. Also, effect of changing batch size and learning rate is studied. We also compare the three models and define the hyper-parameters to get the best recommendation.**

***Index Terms*—Content Filtering, Collaborative Filtering, Hybrid Filtering, Recommender System, Deep Autoencoders**

I. Introduction

Huge content of information over the internet has made it difficult for users to choose the content of interest. This problem is often known as Information Overload. The main objective of this paper is to overcome Information Overload and narrow down the content for the user in accordance with user's taste with the help of a Recommendation System (RS) [13] [14] [15] [16]. The two main approaches of building the recommendation system are content-based filtering (CB) and collaborative filtering (CF). However, many researchers [17] [18] [19] [20] indicate limitations in both approaches. Hence, we will also be building a hybrid recommendation system that has properties of both content and collaborative filtering and help us to overcome the limitations of both.

Most of the recommendation systems use machine learning techniques like matrix factorisation. The matrix factorization approach is not that efficient. This is an example of a cold-start problem as the recommender cannot deal efficiently with new users or items. Deep learning has lately been widely proposed for recommendation systems. Due to the high performance, it provides better understanding of user demands. We will be using deep learning approach for developing an efficient recommender system that provides a wide filtering criteria.

II. Types of Recommendation Systems

A. Content-Based Recommendation system

This filtering method is based on the description of an item and a profile of the user's preferred choices. In this system, keywords are used to describe the items. Besides, a user profile is built to state the type of item this user likes. In other words, the algorithms try to recommend products which are similar to the ones that a user has liked in the past as shown in Fig. 1. The idea of content based filtering is that if you like an item you will also like a 'similar' item. For example, when we are recommending the same kind of item like a movie or song recommendation. This approach has its roots in information retrieval and information filtering research.

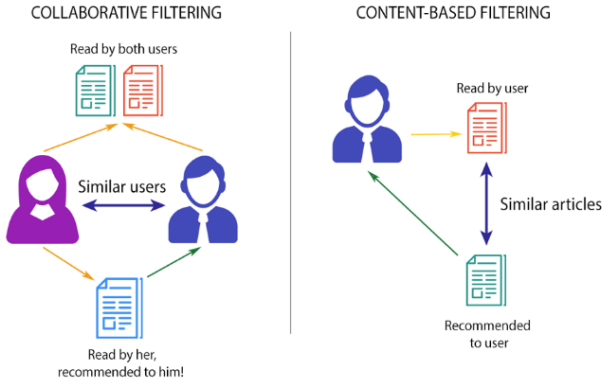

Fig. 1. Collaborative and Content Filtering

B. Collaborative Recommendation system

This filtering method is usually based on collecting and analyzing information based on user's behaviors, their activities or preferences. It then predicts items based on the similarity with other users as shown in Fig. 1. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and thus, it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Collaborative

filtering is based on the assumption that people who liked an item in the past will like similar items in the future.

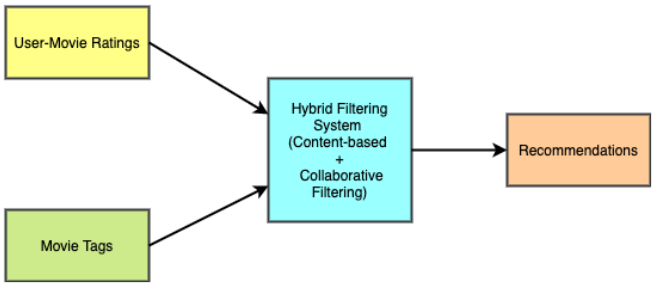

Fig. 2. Hybrid Recommendation System

It is of two types:

1) User-Based Approach: In this approach, the system recommends items that are positively rated by the neighbourhood of a user. Thus, even if the user has not explored a certain category, he might get recommendations if some item of that category is highly rated by the user's neighbourhood. The user-based approach has some drawbacks, the major one being that user's preferences change over time. To overcome the drawbacks, item-based approach is used.

2) Item-Based Approach: In item-based filtering, the similarities between the items instead of between the users is calculated, as the former is always constant. Focusing on similarities between unchanging objects can produce better results than looking at similarities between people who may have liked something last week and something totally different this week. Another advantage is that we usually have far fewer items to deal with than people. Thus, the similarity between items-based on the ratings of users is used for making recommendations.

C. Hybrid Recommendation system

Hybrid recommendation system is a combination of the features of both content and collaborative filtering. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model. For example, Netflix is a good example of the use of hybrid recommender systems. Netflix makes recommendations by comparing the watching and searching habits of similar users (i.e., collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering). We took the user-movie matrix with respect to ratings, combined it with movie-tags data set and then, the recommendations are made. Our approach is shown in Fig. 2.

III. Related Work

A. Recommendation system

Some of the recent research works on recommendation system not based on deep learning are described here. Features of item-item collaborative filtering as well as user-user collaborative filtering to make efficient group recommendation was suggested [7]. An overview of recommender systems that are usually classified as content-based, collaborative and hybrid was carried out [9]. A cascading hybrid recommendation was also proposed to overcome scalability, data sparsity and cold-start issues [11]. A switching hybrid recommendation approach was proposed by combining a Naive Bayes classification approach with the collaborative filtering [11]. A Bayesian network model to deal with the hybrid recommendation by combining the features of both content-based and collaborative was also suggested [13].

B. Deep learning based recommendation system

In this section, deep learning techniques that have been applied for recommendation system are described. A make-up recommendation system using rule guided deep neural network was proposed [1]. A deep learning framework was used to address the challenges and perform accurate long-tail recommendations to tackle the problem of unsatisfactory quality of description content using stacked autoencoders [2]. Different deep learning approaches were studied including content based, collaborative and hybrid recommenders system [3]. This approach enables the model to learn different features of users and items automatically to improve accuracy of recommendation. Performance improvements brought by deep learning for youtube recommendation system were discussed [4]. Practical lessons and insights from designing, iterating and maintaining a massive recommendation were also described. A deep learning based model was presented for contextual recommendation using denoising autoencoder neural network architecture [5]. An approach based on deep learning-based augmentation of the collaborative filtering approach for predicting user ratings was suggested [6]. This makes it possible to capture the hidden interactions and lead to better recommender system. A content-based recommender system based on deep neural networks leveraging user and item embeddings was proposed for producing top recommendations [8]. A comprehensive review of recent research on deep learning based recommender systems is also presented [10].

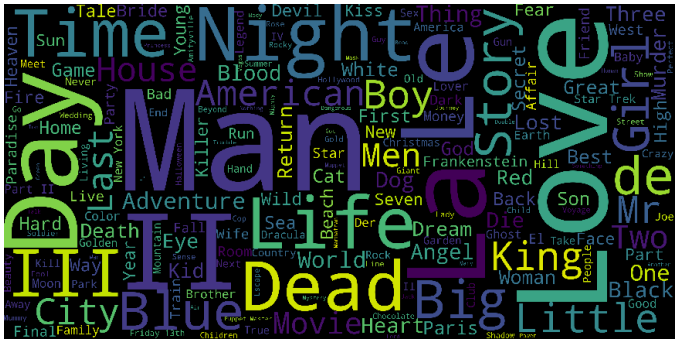

Fig. 3. WordCloud of the most common movie titles in the MovieLens data set

Fig. 4. WordCloud of the genres in the MovieLens data set

IV. Methods

We propose to develop a content, collborative and hybrid movie recommender system using autoencoders.

A. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyze data sets to summarize their main characteristics, often with visual methods. We have analyzed our three different data sets namely movies, ratings and tags to get more insight of how these data sets can be used for recommendations. We have normalized the data sets before the processing. Fig. 3 shows the wordcloud visualization of the most common movie titles. This helps us to know that we have a lot of movie franchises in this data set and the most commonly occuring titles. Fig. 4 shows the wordcloud visualization of the genres in the movies file. The top 5 genres seems to be as: Drama, Comedy, Action, Thriller, and Romance. Fig. 5 shows the distribution of the user's ratings. It seems that users are quite generous while rating the movies. The mean rating is 3.58 on a scale of 5. Half the movies have a rating of 4 and 5.

B. Deep autoencoders for recommendation

Autoencoders are the most common building block of the Deep Learning architecture. It is mainly used for representation learning. It is an artificial neural network used to learn a representation (encoding) for a set of input data, usually to a achieve dimensionality reduction. The form of an autoencoder is a feedforward neural network having an input layer, a hidden layer and an output layer. The first half of the network is called as the "encoder" and the second half of the network is called the "decoder". The output layer has the same number of neurons as the input layer for the purpose of reconstructing it's own inputs. This makes an autoencoder a form of unsupervised learning, which means no labelled data are necessary. It only needs a set of input data instead of input-output pairs. It is useful that an autoencoder has a smaller hidden layer than the input layer. This effect forces the model to create a compressed representation of the data in the hidden layer by learning correlations in the data. They have been successfully employed in Collaborative Filtering (CF) recommender systems and achieves state of the art performance.

We have used deep autoencoders in which both the encoder and decoder have two layers, to create a compressed representation for content, collaborative and hybrid recommendations.

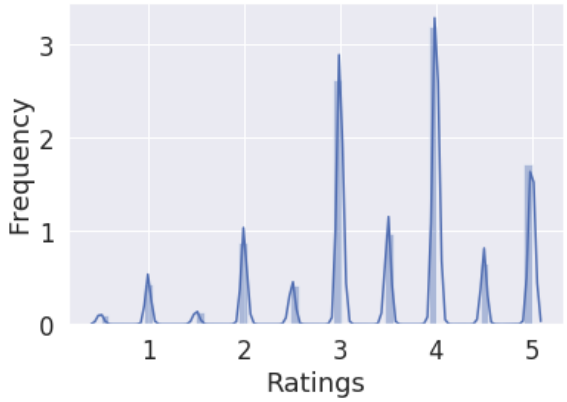

Fig. 5. Distribution of ratings

We have used dropouts of value 0.2 in the inner layers to avoid overfitting or underfitting. We have also used L2 regularization to reduce the likelihood of model overfitting. Our implemented model is explained in detail in the sections below.

1) Content Filtering: In the MovieLens data set, the tag data set has around 100k user-generated movie tags. The tag data is grouped by movie and the tags are concatenated to produce a corpus of documents. In this corpus, documents are the combination of all tags for a particular movie. The tag documents of the tag data set is transformed into a Term Frequency — Inverse Document Frequency (TF-IDF) representation. TF-IDF tokenizes unstructured text data into numeric features that can be more easily processed by machine learning algorithms. The frequency of each term in a document is scaled by the number of documents containing that term. As a result, words that differentiate movies (e.g., "alien", "fantasy") will be weighted higher than words used to describe all movies (e.g. "movie", "cast").

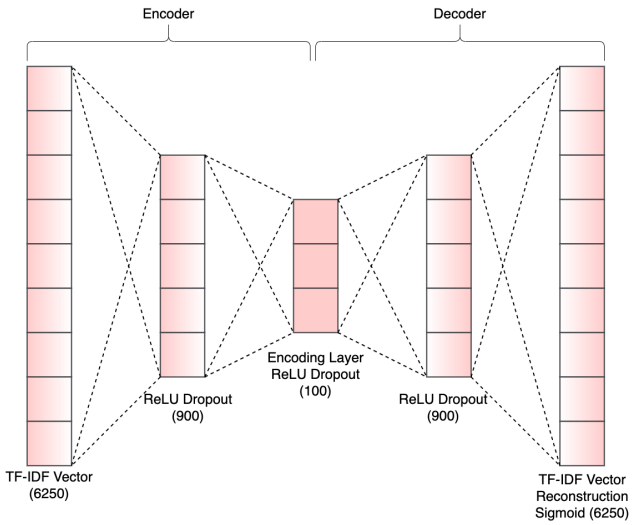

Fig. 6. Autoencoder architecture for Content Filtering

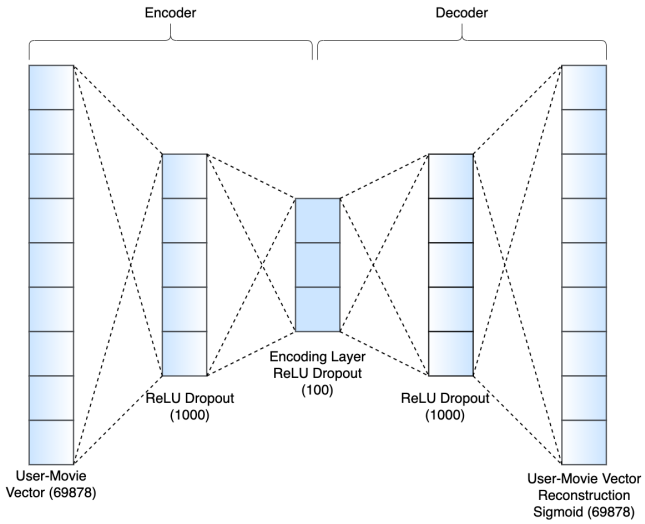

Fig. 7. Autoencoder architecture for Collaborative Filtering

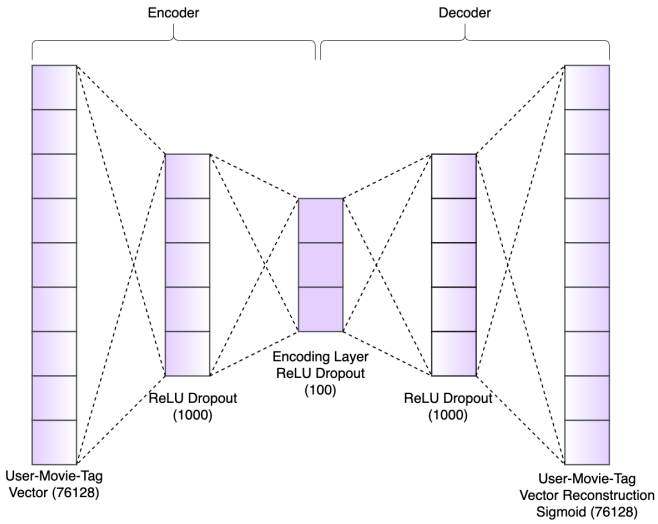

Fig. 8. Autoencoder architecture for Hybrid Filtering

In TF-IDF space, each dimension represents the importance of a certain word to a movie. The representation is less ideal as the encoding of a single concept (e.g. fantasy/magic) is fragmented and scattered across multiple dimensions. The autoencoder is used to compress the TF-IDF data into a lower dimensional space where concepts are consolidated into shared dimensions. As shown in the Fig. 6, the high-dimensional TF-IDF input is compressed gradually into a 100 dimensional central hidden layer. The decoder attempts to reconstruct the original input.

A mean-squared error loss function is applied and back-propagation is performed. The network (particularly the encoder) learns a function which maps data into lower dimensional space in a way that most of the information is preserved. We used different learning rates, batch sizes and activation functions to find the optimum parameters for training as shown in Fig. 9. Using the content-based movie embeddings learned by the autoencoder, cosine similarity is then used to calculate the similar movies and recommend to the user.

2) Collaborative Filtering: We have used autoencoders for item and user based collaborative filtering. In case of item-based filtering, we transform the data set into a movie-user data set and then train the autoencoder on this data. The data set consists of all the ratings with respect to each movie that users have provided. This data set is sparse by nature which is then normalized and passed through the autoencoder. The architecture of autoencoders for collaborative filtering is shown in Fig. 7. A mean-squared error loss function is applied and back-propagation is performed. The autoencoder learns a function which maps data into lower dimensional space. We used different learning rates, batch sizes and activation functions to find the optimum parameters for training as shown in Fig. 10. Using the collaborative-based movie embeddings learned by the autoencoder, cosine similarity is then used to calculate the similar movies and recommend to the user.

For user-based filtering, we transform the data set into a user-movie data set. The data set consists of all the ratings with respect to each user that they have provided for a movie. This sparse data set is normalized and then used to train the autoencoder. Based on the past movie ratings given by the users, the autoencoder predicts the future ratings for movies not seen yet and then we recommended movies to the user based on those ratings [22].

3) Hybrid Filtering: We build a hybrid recommendation system using the features of both content and collaborative based filtering. We combine the normalized movie ratings and the tags processed using TF-IDF vectorization into a single data set. This data set is then passed through the autoencoder. The architecture of autoencoders for hybrid filtering is shown in Fig. 8. A mean-squared error loss function is applied and back-propagation is performed. We used the same learning rate, batch size and activation function that was found optimal in case of content and collaborative filtering as shown in Fig. 11. Using the hybrid-based movie embeddings learned by the autoencoder, cosine similarity is then used to calculate the similar movies and recommend to the user.

V. Evaluation

A. MovieLens Data set

One of the most common data sets that is available on the internet for building a recommender System is the MovieLens data set. This version of the data set that we are working with 10 million contains 10 million movie ratings and 100,000 tag applications to 10,000 movies by 72,000 users [23].

The data was collected by GroupLens researchers over various periods of time, depending on the size of the set. This 10M version was released on February 2009. Users were selected at random for inclusion. All users selected had rated at least 20 movies. Each user is represented by an id, and no other information was provided. The original data are contained in three files, movies.dat, ratings.dat, and tags.dat.

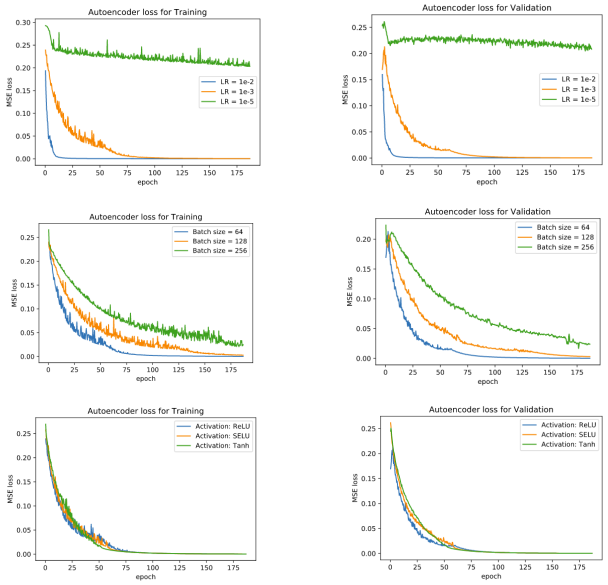

Fig. 9. MSE Loss for Training and Validation with respect to batch size, learning rate and activation function for Content Based Filtering

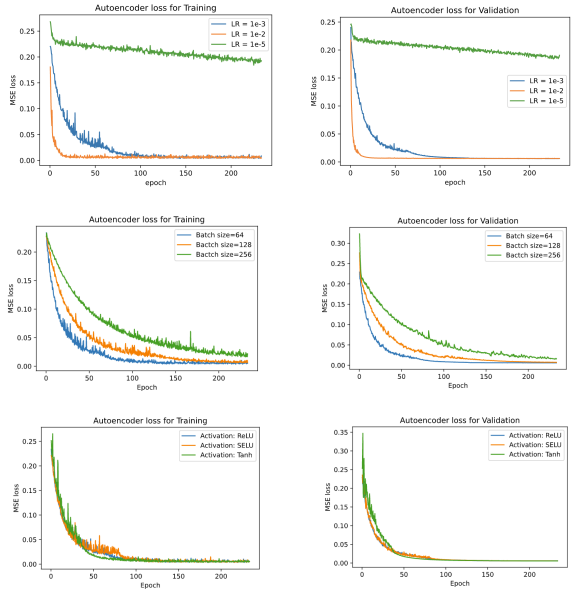

Fig. 10. MSE Loss for Training and Validation with respect to batch size, learning rate and activation function for Collaborative Filtering

B. Effect of the activation function

Activation functions are very important in Neural Network in order to learn and understand the complicated and non-linear complex functional mappings between the inputs and response variable. They introduce non-linear properties to the Network. Their main purpose is to convert a input signal of a node in a neural network to an output signal. That output signal now is used as a input in the next layer in the stack. We explore the effect of using different activation functions. We tested some of the most popular choices in deep learning: Rectified Linear Units (ReLUs), Scaled Exponential Linear Units (SELUs) and hyperbolic tangent (Tanh). For all the models, we used the respective activation functions for all the layers. We have used sigmoid function in the last layer. Though in case of content and collaborative filtering, the performance of all the activation functions used are almost the same as shown in Fig. 9 and Fig. 10. We chose to use the ReLU activation function for all the models.

C. Effect of the batch size

Batch size is one of the most important hyper-parameters to tune in modern deep learning systems. We often want to use a larger batch size to train the model as it allows computational speedups from the parallelism of GPUs. However, it is well known that too large of a batch size will lead to poor generalization. In case of smaller batch size, the model will convergence faster to "good" solutions without having to see all the data.

We explore the effect of using different batch sizes. We tested with 64, 128 and 256 batch sizes for content and collaborative filtering. We learn that the batch sizes of 64 and 128 are giving good results compared to a batch size of 256. But we consider using the batch size of 64 over 128 as it performs better for both content and collaborative filtering as shown in Fig. 9 and Fig. 10.

D. Effect of the learning rate

The learning rate is a hyper-parameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. The learning rate is one of the most important hyper-parameter when configuring a neural network.

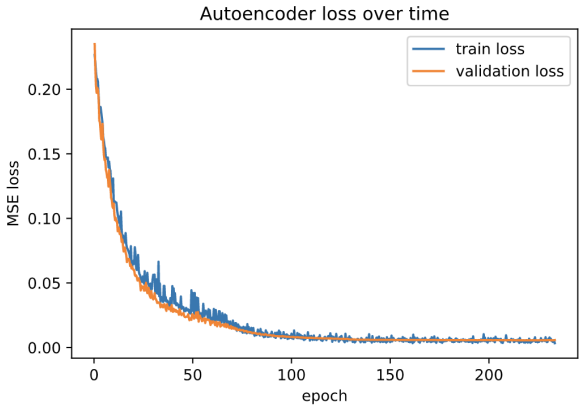

Fig. 11. Autoencoder MSE Loss for training and validation for Hybrid filtering

We explore the effect of using different learning rates. We tested with learning rates of 1e-2 and, 1e-3 and 1e-5 for content and collaborative filtering. We learn that the learning

TABLE I
TOP 5 MOVIES SEEN BY USER 10070

MovieID	Rating	Movie Title	Genres
1450	1.0	Cinderella (1950)	Animation|Children| Fantasy|Musical
1331	1.0	Little Mermaid, The (1989)	Animation|Children| Fantasy|Musical
196	1.0	Star Wars: Episode I- The Phantom Menace (1999)	Action|Adventure|Sci-Fi
24	1.0	Star Wars: Episode IV A New Hope (1977)	Action|Adventure|Sci-Fi
381	1.0	Sound of Music, The (1965)	Musical|Romance

TABLE II
TOP 5 MOVIES RECOMMENDED FOR USER 10070 USING USER-BASED CF

MovieID	Rating	Movie Title	Genres
1466	0.435913	Fantasia (1940)	Animation|Children| Fantasy|Musical
3116	0.427960	Sleeping Beauty (1959)	Animation|Children| Musical
156	0.411429	E.T. the Extra-Terrestrial (1982)	Children|Drama|Sci-Fi
177	0.336002	Star Wars: Episode V - The Empire Strikes Back (1980)	Action|Adventure|Sci-Fi
193	0.331580	Shakespeare in Love (1998)	Comedy|Drama| Romance

rates of 1e-5 is not giving good results as the value is too small for the loss to converge.

The learning rates 1e-2 and 1e-3 give good results. But we considered the learning rate of 1e-3 for our recommendation system as it gives the smaller loss value out of the two. Hence, the learning rate of 1e-3 is used as it is giving the minimum MSE loss for both content and collaborative filtering as shown in Fig. 9 and Fig. 10.

E. Mean Squared Error

The mean squared error (MSE) or mean squared deviation (MSD) of an estimator helps to measure the average of the squares of the errors, i.e, the average squared difference between the estimated values and the actual value. MSE is a loss function, corresponding to the expected value of the squared error loss. The MSE is a measure of the quality of an estimator, non-negative, and values closer to zero are better.

The maximum number of training epochs is selected as a stopping criteria on the basis of mean squared error (MSE). Fig. 9, Fig. 10, and Fig. 11 show the variation of the MSE versus the number of epochs for training and validation data. The gradual decrease of the values of MSE proves that the training and the validation data are accurate. The gradual decrease is happening because the weights update after each epoch.

F. Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. We used cosine similarity to quantify the similarities between the movies for content, collaborative and hybrid recommendation systems. Cosine similarity ranges from -1 to 1 and is calculated as the dot product between two vectors divided by their magnitudes.

Equation for calculating cosine similarity between two vectors:

$$similarity(A, B) = cos(\theta) = \frac{A.B}{|A||B|} \tag{1}$$

In this model, the movie vectors pointing in the same direction will receive high cosine similarity scores. The main idea is that the directions through the latent concept space capture the essence of movies.

VI. RESULTS

Collaborative Filtering. In Table I, for user-based collaborative filtering, the movies that are previously seen by a user whose user ID is 10070 are shown. Table II shows the list of movies that are recommended for the user based on his/her past preferences. Table III shows the list of movies recommended by using the item-based collaborative filtering. The highlighted row shows the movie with respect to which the recommendation is made. The movies with similar genres and higher cosine similarity scores are selected. The similarity scores are in the range from 0.91 to 0.92. The genres of the recommendations are also very similar to "The Evil Dead" movie.

Content Filtering. The top 5 recommendations for "The Evil Dead" movie for content based filtering are given in Table IV. The highlighted row shows the movie with respect to which the recommendation is made. In this table, we observe that the recommendations from content filtering model are selected with similar genres and high cosine similarity. First recommendation, we get is of "Evil Dead II" which has a similarity score of 0.84 and the genre of "Evil Dead II" resembles the genre of "The Evil Dead" movie.

Hybrid Filtering. Table V shows us the list of movies recommended by hybrid filtering model. The highlighted row shows the movie with respect to which the recommendation is made. The movies with similar genres and higher cosine similarity scores are selected. The similarity scores are in the range from 0.87 to 0.90. The genres of the recommendations are also very similar to the "The Evil Dead" movie.

In our model, we observe that the collaborative filter is good at spanning gaps across genres and recommended movies with high ratings. The content filter is good at recommending similar items and lesser-viewed, potentially hidden unknown movies. The aim of the hybrid approach was to combine the strengths of both the systems to overcome the drawbacks faced by using these filtering methods independently. The hybrid recommender system using autoencoders can ensure good scalability and strength of recommendation systems.

TABLE III
TOP 5 MOVIES RECOMMENDED BY CONTENT FILTERING MODEL

MovieID	Movie Title	Genres	Ratings Count	Avg Rating	Similarity Score
724	Evil Dead, The (1981)	Fantasy|Horror	2532	3.661137	1.000000
409	Evil Dead II (Dead by Dawn) (1987)	Action|Comedy|Fantasy|Horror	3952	3.792637	0.846004
1610	Night of the Living Dead (1968)	Horror|Sci-Fi|Thriller Comedy|Musical	4202	3.628986	0.836847
3034	Mummy, The (1959)	Horror	373	3.126005	0.831568
7298	Class of Nuke 'Em High (1986)	Comedy|Horror	177	2.276836	0.830757
7774	Citizen Toxie: The Toxic Avenger IV (2000)	Action|Comedy|Horror	36	3.027778	0.813691

TABLE IV
TOP 5 MOVIES RECOMMENDED BY COLLABORATIVE FILTERING MODEL

MovieID	Movie Title	Genres	Ratings Count	Avg Rating	Similarity Score
724	Evil Dead, The (1981)	Fantasy|Horror	2532	3.661137	1.000000
409	Evil Dead II (Dead by Dawn) (1987)	Action|Comedy|Fantasy|Horror	3952	3.792637	0.922419
3343	Dawn of the Dead (1978)	Action|Drama|Horror	1223	3.800899	0.916275
2307	Ring (Ringu) (1998)	Horror|Mystery|Thriller	1301	3.617602	0.914258
2072	Hellraiser (1987)	Horror	1309	3.294500	0.913229
3666	Dead Alive (Braindead) (1992)	Comedy|Fantasy|Horror	1284	3.671729	0.911045

TABLE V
TOP 5 MOVIES RECOMMENDED BY HYBRID MODEL

MovieID	Movie Title	Genres	Ratings Count	Avg Rating	Similarity Score
724	Evil Dead, The (1981)	Fantasy|Horror	2532	3.661137	1.000000
3718	Bubba Ho-tep (2002)	Comedy|Horror	1004	3.555279	0.902878
2072	Hellraiser (1987)	Horror	1309	3.294500	0.893269
409	Evil Dead II (Dead by Dawn) (1987)	Action|Comedy|Fantasy|Horror	3952	3.792637	0.881886
5035	Return of the Living Dead, The (1985)	Comedy|Horror|Sci-Fi	560	3.289286	0.878855
3713	Tremors (1990)	Comedy|Horror|Sci-Fi	1615	3.328793	0.876206

Comparison between Content, Collaborative and Hybrid Filtering. The collaborative recommenders mostly rely on the data generated by the users as they interact with the items. In a movie recommendation model, the collaborative filters find trends in how similar users rate movies based on the ratings. In the context of a movie-to-movie recommender, the collaborative filter gives us the list of movies which have similar user-rating profiles.

The advantage of collaborative model is that the users create the data naturally as they interact with items. It helps the users discover new items that are outside the subspace defined by their historical profile. However, there are some drawbacks such as the cold start problem. It is also difficult to accurately recommend novel items because these items typically do not have enough user-item interaction data.

Content recommenders rely on item features to make recommendations. The content based filter gives us the list of movies which have similar features. For example, in this model, we have features like user-generated tags on movies. Content filters tend to be more robust against popularity bias and the cold start problem. They can easily recommend new or novel items based on niche tastes. However, in a movie-to-movie recommender, content filters can only recommend movies with features similar to the original movie. This limits the scope of recommendations, and can also result in surfacing items with low ratings.

By creating a hybrid recommender, we have attempted to create a system that recommends movies that other users rated in a similar manner, while still making on-topic recommendations based on the features of that movie.

VII. CONCLUSION

In this paper, we have developed a movie-to-movie recommendation system using content, collaborative and hybrid filtering systems. We have also built a user-based collaborative filtering system. We have analyzed the pros and cons of content and collaborative-based methods. We also showed how recommendation systems are built using deep autoencoders instead of traditional matrix factorization methods.

For future work, the hybrid model can be extended by taking into account additional contextual information regarding users in order to find out similarity among users. Especially, in the integration of data of different natures, such as demographic information. The information could help to improve the performance of our approach. This model could also be implemented using different kinds of autoencoders like variational autoencoders for implicit feedback.

REFERENCES

[1] T. Alashkar, S. Jiang, S. Wang, and Y. Fu, "Examples-Rules Guided Deep Neural Network for Makeup Recommendation".In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17) pp. 941–947, 2017.

[2] B. Bai, Y. Fan, W. Tan, and J. Zhang, “A Deep Learning Framework for Recommendation of Long-tail Web”, IEEE Transactions on Services Computing , doi: 10.1109/TSC.2017.2681666, 2017.

[3] B. T. Betru, C. A. Onana, and B. Batchakui, “Deep Learning Methods on Recommender System: A Survey of State-of-the-art”, International Journal of Computer Applications, Vol 162, pp. 17-22, 2017.

[4] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations” Proceedings of the 10th ACM Conference on Recommender Systems, pp 191-198, 2016.

[5] Y. Jhamb, T. Ebesu, and Y. Fang, “Attentive Contextual Denoising Autoencoder for Recommendation”, ACM SIGIR International Conference, doi:10.1145/3234944.3234956, pp. 27-34, 2018.

[6] M. Almaghrabi, G. Chetty, “A Deep Learning Based Collaborative Neural Network Framework for Recommender System”, International Conference on Machine Learning and Data Engineering, pp.121-127, 2018.

[7] A. Pujahari and V. Padmanabhan, “Group Recommender Systems: Combining user-user and item-item Collaborative filtering techniques”, 14th International Conference on Information Technology, 2015.

[8] C. Musto, C. Greco, A. Suglia, and G. Semeraro, “A Content-based Recommender System based on Recurrent Neural Networks”, Italian Information Retrieval Workshop, 2016.

[9] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”, IEEE transactions on knowledge and data engineering,Vol 17, pp. 743-749, 2005.

[10] Z. Shuai, Y. Lina, S. Aixin and T. Yi, “Deep learning based recommender system: A survey and new perspectives”, Journal of ACM Computing Surveys, Vol 52, pp.1-38, 2019.

[11] M. A. Ghazanfar and A. Prugel-Bennett, “A Scalable, Accurate Hybrid Recommender System,” Third International Conference on Knowledge Discovery and Data Mining, pp. 94-98, 2010.

[12] L. CamposJuan, M. Fernández, F. HueteMiguel and A. Morales, “Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks”, A International Journal of Approximate Reasoning, Vol 51, pp. 785-799, 2010.

[13] B. Fernandes, J. Sacenti, and R. Willrich, “Using Implicit Feedback for Neighbors Selection: Alleviating the Sparsity Problem in Collaborative Recommendation Systems”. In Proceedings of WEBMEDIA. ACM, pp. 341–348, 2017.

[14] L. Laique, B. Santana, A. Souza, D. Santana, W. Dourado, and F. Durão, “Evaluating Ensemble Strategies for Recommender Systems under Metadata Reduction”. In Proceedings of WEBMEDIA. ACM, 125–132, 2017.

[15] D. Silva, R.Silva, and F. Durão, “Recommending Stores for Shopping Mall Customers”. In Proceedings of WEBMEDIA. ACM, pp. 117–124, 2017.

[16] N. Silva, D. Carvalho, A. Pereira, F. Mourão, and L. Rocha, “Evaluating Different Strategies to Mitigate the Ramp-up Problem in Recommendation Domains”, In Proceedings of WEBMEDIA. ACM, pp. 333–340, 2017.

[17] A. Das, M. Datar, A. Garg, and S. Rajaram, “Google news personalization: scalable online collaborative filtering”. In Proceedings of WWW. ACM, pp. 271–280, 2007.

[18] D. Goldberg, D. Nichols, B. Oki, and D. Terry. 1992, “Using collaborative filtering to weave an information tapestry”, ACM 35, pp. 61–70, 1992.

[19] J. Konstan and J. Riedl, “Recommender systems: from algorithms to user experience”, User Modeling and User-Adapted Interaction 22 , pp. 101–123. 2012

[20] U. Shardanand and P. Maes, “Social information filtering: algorithms for automating “word of mouth””. In Proceedings of SIGCHI. ACM Press/Addison-Wesley Publishing Co., 210–217, 1995.

[21] A. Rajaraman, J.D. Ullman, “Data Mining (PDF). Mining of Massive Datasets”, doi:10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2, pp. 1–17, 2011.

[22] O. Kuchaiev and B. Ginsburg, “Training Deep AutoEncoders for Collaborative Filtering”, arXiv:1708.01715 (2017)

[23] MovieLens-10M-Dataset,Available:https://grouplens.org/datasets/movielens/10m, Accessed: March 26, 2020.