A Movie Recommendation System Leveraging Restricted Boltzmann Machines or Kmeans for Collaborative Filtering.

Praburam Krishnaswamy Varadharajan
praburam@usc.edu

Raksheka Rajakumar
rrajakum@usc.edu

ABSTRACT

In this comprehensive study, we meticulously investigate movie recommendation systems, employing advanced models—Restricted Boltzmann Machines and KMeans Clustering.

Our research is grounded in expansive datasets encompassing detailed information on approximately 46,000 movies and a vast repository of 26 million user ratings. The principal objective is to discern the optimal model capable of providing sophisticated and tailored recommendations. Our rigorous analysis, characterized by statistical rigor, scrutinizes each model's efficacy, scalability, and capacity to furnish recommendations.

This analysis holds substantial value for several stakeholders. For academic researchers, it contributes a nuanced understanding of the comparative strengths and limitations of Boltzmann Machines and KMeans Clustering in the context of movie recommendations. This study can be used to leverage the findings to inform decision-making processes in the development and deployment of recommender systems, aiming for enhanced user satisfaction and engagement.

Another pivotal aspect of this project is the development of an interactive web interface through which users can input a movie name to get similar movie recommendations.

1. INTRODUCTION

As the volume of available content continues to grow, users face the challenge of discovering movies that align with their preferences. Recommendation systems alleviate this challenge by analyzing user behaviors, preferences, and movie characteristics to suggest content tailored to individual tastes. By employing Boltzmann Machines[4] and KMeans Clustering[6], we aim to explore diverse approaches to recommendation, combining collaborative filtering and clustering techniques to achieve superior accuracy and relevance in our recommendations.

In the vast ocean of cinematic offerings, finding the perfect movie tailored to one's preferences can be a daunting task. As our appetite for diverse and engaging content continues to grow, the need for a sophisticated and personalized movie recommendation system becomes increasingly apparent. The significance

of this project lies in its ambition to push the boundaries of conventional recommendation systems. By adopting innovative methodologies, the system aims to provide users with more accurate, diverse, and personalized movie recommendations. The outcome has the potential not only to enhance user satisfaction but also to contribute to the broader landscape of recommendation system research and development. 1 The project's impact spans both industry and academic realms. In the industry, streaming giants like Netflix can leverage advanced recommendation systems to heighten user engagement and satisfaction, ultimately increasing subscriber retention. E-commerce platforms, exemplified by Amazon, stand to benefit through optimized product suggestions, driving sales, and fostering customer loyalty. On the academic front, the project contributes to the evolution of recommendation system algorithms, offering potential inspiration for new models or hybrid approaches. Insights into user behavior foster academic research in human-computer interaction, psychology, and behavioral sciences.

2. MODELS IN FOCUS:

The first model in focus in Restricted Boltzmann Machines. Boltzmann Machines are a type of artificial neural network known for their effectiveness in collaborative filtering tasks. These models capture complex patterns in user-item interactions and are particularly adept at handling sparse data, making them well-suited for collaborative recommendation scenarios. The Boltzmann Machine will analyze user ratings and movie metadata to learn latent features and generate personalized recommendations.

The second one is a K Means based Recommender system. KMeans Clustering, a classic unsupervised learning algorithm, will be employed to group movies based on features such as vote count, vote average, and genres. This also helps in avoiding the sparcity problem[5]. By clustering movies with similar characteristics, the KMeans model aims to provide recommendations by identifying movies within the same cluster as a user-specified film. This approach offers an alternative perspective on similarity-based recommendations.

3. PROBLEM STATEMENT

The focal point of this endeavor lies in the user's ability to input a movie of interest, triggering the generation of personalized movie recommendations. Leveraging the extensive "movies" and "ratings" datasets, the objective is to implement a multi-faceted approach that seamlessly combines collaborative filtering using Boltzmann Machines and content-based clustering through KMeans. The core challenge involves

creating a system that not only deciphers user preferences from past ratings but also identifies movies with similar characteristics to the user-entered choice.

4.DATA USED

Two major datasets were used for this project. The first dataset encompasses a comprehensive collection of 45,466 entries, providing detailed information about various films. This dataset spans 24 columns, offering a wealth of insights into the world of cinema. These columns cover a range of attributes, including the film's adult classification, collection affiliations, budget, genres, homepage availability, IMDb ID, original language, title, overview, popularity, poster path, production companies, production countries, release date, revenue, runtime, spoken languages, status, tagline, and critical metrics such as vote average and vote count. It's worth noting that certain columns, such as "belongs_to_collection," "homepage," and "tagline," exhibit non-null counts that suggest variability in the dataset's completeness.

The other dataset offers a more user-centric perspective with 26,024,289 entries. Comprising four columns, this dataset includes user-specific information such as the user ID, movie ID, rating assigned by the user, and a timestamp capturing the moment of the rating. The timestamp column, represented as an integer, records the time at which each rating was given. The extensive nature of the "ratings" dataset, with over 26 million entries, reflects a rich collection of user interactions with movies, forming the basis for collaborative filtering models. Together, these datasets provide a robust foundation for the implementation of recommender systems, enabling the exploration of user behaviors, movie characteristics, and the synthesis of collaborative and clustering techniques for optimal movie recommendations.

5.MINI LITERATURE REVIEW

[1] discusses movie recommender systems and their concepts, methods, challenges, and future directions. It highlights the different filtering techniques used in recommender systems, such as collaborative filtering, content-based filtering, context-based filtering, and hybrid filtering.

The document also mentions various machine learning algorithms used in movie recommender systems, including K-means clustering, principal component analysis, and self-organizing maps with principal component analysis. The research aims to address the challenges in implementing feasible solutions and provides recommendations for future research.

[2] discusses the limitations of traditional collaborative filtering algorithms and introduces the concept of item-based collaborative filtering. Generally, the first step of the Previous recommender system is to collect the preferences of the users. Our Collaborative Filtering (CF) implementation stores the data in two 2D matrices. So for each user in a row, we have columns for each item that he or she has rated. The matrix is quite sparse since a lot of users only buy one item.

After getting the 2D dataset matrix, we implement two kinds of recommendation system models: item-based and user-based correlative filtering. For both models, we need to compute the similarity and prediction score. In the following part, we will explicitly show how we build our item-based and user-based recommendation system, and compare different models in the following subsection.

[3] gives a summary of restricted Boltzmann Machines for Collaborative Filtering, specifically for modeling tabular data such as user ratings of movies. RBMs are two-layer undirected graphical models that can efficiently handle large datasets. The paper presents efficient learning and inference procedures for RBMs and demonstrates their successful application to the Netflix dataset, which contains over 100 million user/movie ratings. We are trying to implement this model in our dataset.

[8] Ralambondrainy's 1995 paper, "A conceptual version of the k-means algorithm" in Pattern Recognition Letters (16(11), 1147-1157), introduces an innovative perspective on the traditional k-means algorithm. Focused on conceptual enhancements, the paper contributes valuable insights to the field of clustering and pattern recognition. This work is a concise yet pivotal addition to the literature, offering researchers and practitioners a unique viewpoint on the k-means algorithm and its potential applications.

6. PHASE-1

Restricted Boltzman Machines

Data Preprocessing:

The data has been cleaned, removed of duplicates and null values and the data has been sorted according to movie ID and put into a dataframe. We use the ratings and movies_metadata in our model and the model gives recommendation based on the reviews given by other users.

Model Overview:

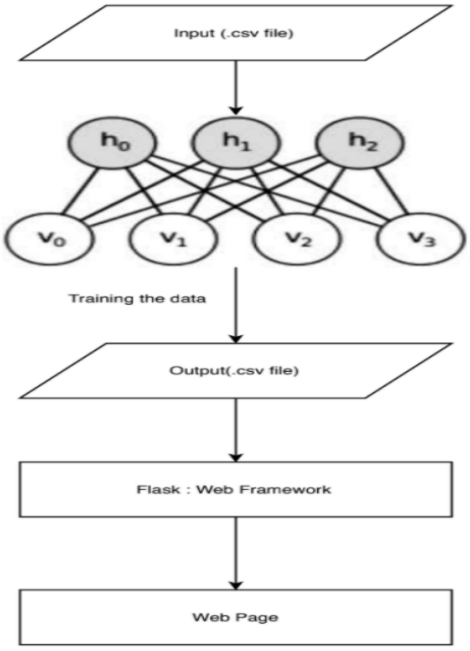

Fig 1: Overview of RBMs

The dataset is split into training and test set. The data is trained using Restricted Boltzmann Machine model . After the training the output is generated and given as input to flask which is a web framework.

Model:

1. The visible layer consists of x units where x is the number of movies a particular user has rated.
2. Each unit in the visible layer is a vector of length 5 (since ratings are from 1 to 5), and the ith index is one corresponding to the rating the user has given, the rest are zeros. The hidden layer consists of 100 units which are binary.
3. The activation function we have used is the sigmoid function both for forward propagation and backward propagation.
4. After that we have used contrastive divergence algorithm to train our model to generate an output file.
5. We tested our approach on the Movies dataset. The ratings data which consists of 26,024,289 ratings, on a scale of 1 to 5, for 45,466 movies. The data is split into training and test sets such that 10% of the latest ratings from each user are selected for the test set and the remaining are used in the training set. Ratings are normalized between 0 and 1, to be used as RBM input. We compare our results with RBM, userbased top-n and non-personalized most popular items. Each experiment is run 10 times and the average results are reported.
6. To assess the rating prediction accuracy, we used the Mean Squared Error (MSE) metric:

$$MSE = \sum_{\forall dataset} (r - r') ** 2$$

Training:

Once the training dataset is presented to the model, the model identifies common parameters among the movies based on which the prediction will be made. This is one of the major advantages using Restricted Boltzmann Machines that feature/parameter identification occurs automatically. Once the model is trained it will try and look for the same features that it identified during the training in the movies which are presented in the test set. In this case the features will be Awards, Year, Category, Duration and Genre of the movie.

The model is trained on the dataset for 100 epochs with 0.0001 as the learning rate as we found the loss is comparatively less than other learning rates.

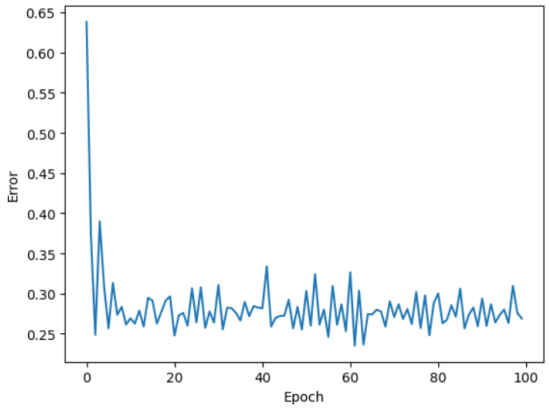

Fig.2. Training loss

Contrastive Divergence:

Algorithm 1. k-step contrastive divergence

Input: RBM $(V_1,\dots,V_m,H_1,\dots,H_n)$, training batch S
Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i=1,\dots,n$, $j=1,\dots,m$

init $\Delta w_{ij}=\Delta b_j=\Delta c_i=0$ for $i=1,\dots,n$, $j=1,\dots,m$
forall the $v\in S$ **do**
  $v^{(0)}\leftarrow v$
  for $t=0,\dots,k-1$ **do**
    for $i=1,\dots,n$ **do** sample $h_i^{(t)}\sim p(h_i\,|\,v^{(t)})$
    for $j=1,\dots,m$ **do** sample $v_j^{(t+1)}\sim p(v_j\,|\,h^{(t)})$
  for $i=1,\dots,n$, $j=1,\dots,m$ **do**
    $\Delta w_{ij}\leftarrow \Delta w_{ij}+p(H_i=1\,|\,v^{(0)})\cdot v_j^{(0)}-p(H_i=1\,|\,v^{(k)})\cdot v_j^{(k)}$
    $\Delta b_j\leftarrow \Delta b_j+v_j^{(0)}-v_j^{(k)}$
    $\Delta c_i\leftarrow \Delta c_i+p(H_i=1\,|\,v^{(0)})-p(H_i=1\,|\,v^{(k)})$

Fig:3. Algorithm for contrastive Divergence

Contrastive Divergence (CD) was introduced to RBM training by Hinton (2002) and is a commonly deployed method to overcome the computational complexity of the computation of the log derivative by using Gibbs sampling to generate an approximation. Gibbs sampling is an iterative process, where in each step the joined probability distributions of the visible and hidden layers are used to sample a state + of the hidden layer and from this sample a sample − of the visible layer is computed. This process is repeated in a loop or so called Markov chain, starting with an initial seed + picked randomly from the training data .

Fortunately, a single Gibbs step is sufficient to produce an approximation of the gradient which is good enough to let the gradient ascent method learn a good representation.

Experimental Results:

The model was tested on the testdata and we got a test loss of 0.2144. The output was stored in a file called output.csv and this was later used in flask. To generate real time recommendations we made a similar model and tested it on movies dataset and the official MovieLens Data to make the model more genralized. In here the results are the indices of the corresponding movie name in our dataset. The recommendations by this model is shown below.

```
tensor([466, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 455, 467, 468,
        469, 470, 471, 472, 473, 474, 475, 476, 444, 434, 435, 436, 437, 438,
        439, 440, 441, 442, 443, 477, 445, 446, 447, 448, 449, 450, 451, 452,
        453, 454, 509, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 498,
        510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 488, 478, 479, 480,
        481, 482, 483, 484, 485, 486, 487, 433, 489, 490, 491, 492, 493, 494,
        495, 496, 497, 379, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378,
        368, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 357, 347, 348,
        349, 350, 351, 352, 353, 354, 355, 356, 390, 358, 359, 360, 361, 362,
        363, 364, 365, 366, 367, 422, 412, 413, 414, 415, 416, 417, 418, 419,
        420, 421, 411, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 401,
        391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 520, 402, 403, 404,
        405, 406, 407, 408, 409, 410, 639, 629, 630, 631, 632, 633, 634, 635,
        636, 637, 638, 628, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649,
        617, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 650, 618, 619,
        620, 621, 622, 623, 624, 625, 626, 627, 682, 672, 673, 674, 675, 676,
        677, 678, 679, 680, 681, 671, 683, 684, 685, 686, 687, 688, 689, 690,
        691, 692, 661, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 606,
        662, 663, 664, 665, 666, 667, 668, 669, 670, 552, 542, 543, 544, 545,
        546, 547, 548, 549, 550, 551, 541, 553, 554, 555, 556, 557, 558, 559,
        560, 561, 562, 531, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530,
        563, 532, 533, 534, 535, 536, 537, 538, 539, 540, 595, 585, 586, 587,
        588, 589, 590, 591, 592, 593, 594, 584, 596, 597, 598, 599, 600, 601,
        602, 603, 604, 605, 574, 564, 565, 566, 567, 568, 569, 570, 571, 572,
        573, 346, 575, 576, 577, 578, 579, 580, 581, 582, 583, 119, 109, 110,
        111, 112, 113, 114, 115, 116, 117, 118, 108, 120, 121, 122, 123, 124,
        125, 126, 127, 128, 129,  97,  87,  88,  89,  90,  91,  92,  93,  94,
         95,  96, 130,  98,  99, 100, 101, 102, 103, 104, 105, 106, 107, 162,
        152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 151, 163, 164, 165,
        166, 167, 168, 169, 170, 171, 172, 141, 131, 132, 133, 134, 135, 136,
        137, 138, 139, 140,  86, 142, 143, 144, 145, 146, 147, 148, 149, 150,
         32,  22,  23,  24,  25,  26,  27,  28,  29,  30,  31,  21,  33,  34,
         35,  36,  37,  38,  39,  40,  41,  42,  11,   1,   2,   3,   4,   5,
          6,   7,   8,   9,  10,  43,  12,  13,  14,  15,  16,  17,  18,  19,
         20,  75,  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  64,  76,
```

Fig.4. Recommendation of movies indices in a ranked order as per preference of user.

In assessing the performance of Restricted Boltzmann Machines (RBMs), we employed the Mean Squared Error (MSE) metric as a quantifiable measure of rating prediction accuracy. Across ten experimental runs, the average MSE for RBMs was computed, resulting in a value of 0.2765. This low MSE indicates a strong capability of RBMs in accurately predicting user ratings, showcasing the model's proficiency in capturing latent features and intricate patterns in user-movie interactions.

Flask:

We used flask to generate a webpage that signs in for a particular user and generates recommendations.

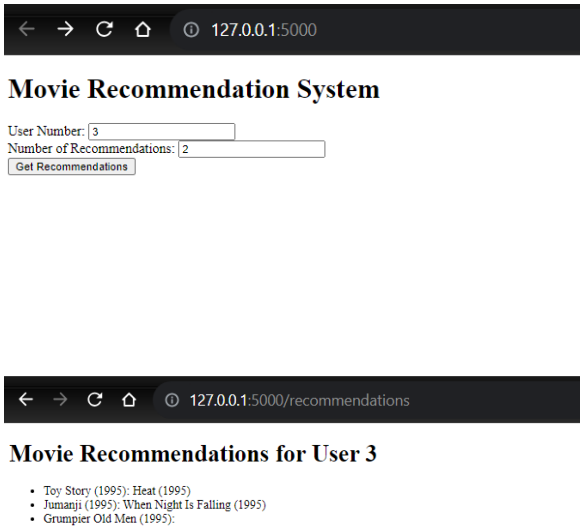

Fig.5. Our Flask Application

Engineering Tradeoffs:

The deployment of RBMs, while powerful in its ability to provide personalized recommendations, comes with certain engineering tradeoffs. One notable consideration is the computational complexity associated with training RBMs, especially with large datasets. The contrastive divergence algorithm, employed for training, demands significant computational resources. Therefore, the tradeoff involves a longer training time in exchange for the model's capability to discern complex user preferences and provide accurate recommendations.

7. PHASE-2

KMeans recommendation system

Data Preparation:

In the initial steps of the KMeans[7] recommendation system, the focus lies on curating a dataset that is conducive to effective clustering. Filtering the movies dataset based on essential criteria, such as a minimum vote count, non-null vote averages, and an English language preference, results in the creation of df_movie_english. Subsequently, relevant columns like 'id,' 'title,' 'vote_count,' 'vote_average,' and 'genres' are retained. The dataset is further enriched by extracting the release year from the 'release_date' column, handling data types, and refining the genres information through the application of an extraction function.

Simultaneously, the ratings dataset undergoes preprocessing to align it with

the movies dataset. Renaming the 'movieId' column to 'id' facilitates seamless integration with the movie dataset. The resulting df_final dataset represents a consolidated view, merging movie details with corresponding ratings, laying the foundation for subsequent clustering.

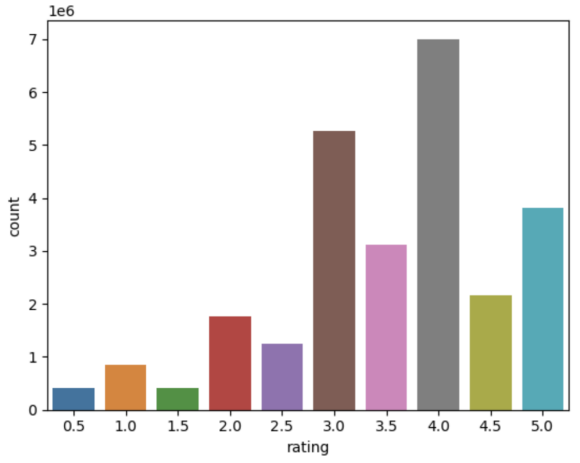

Fig 6: Number of ratings for each rating value by users

Data Preprocessing:

In the second step, the dataset is shuffled to eliminate any inherent biases. The train_test_split function is then employed to partition the shuffled dataset into training, testing, and validation sets. This ensures an unbiased distribution across these subsets. The features for the recommendation model are identified, encompassing variables like 'vote_count,' 'vote_average,' 'rating,' and 'year.' Additionally, the movie genres are encoded using a MultiLabelBinarizer to capture their categorical nature.

Feature Scaling and KMeans Clustering:

The subsequent step involves feature scaling to standardize the features and bring them to a comparable scale. This is achieved through the use of a StandardScaler. The KMeans algorithm is then applied to the scaled feature set with a specified number of clusters, in this case, 5. The training set is clustered, and these clusters are subsequently assigned to the test and validation sets. This clustering forms the basis for grouping movies based on their shared characteristics.

Recommendation Generation:

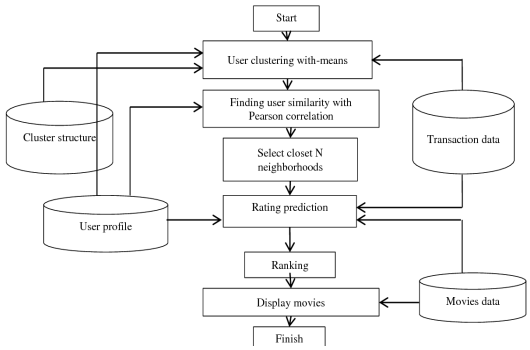

Fig 7: K Means Architecture for movies dataset

The KMeans recommendation system mathematically clusters movies based on features such as vote count (v), release year (y), and rating (r). The objective function seeks to minimize the sum of squared distances between data points and their assigned cluster centroids, as shown below.

$$\sum_{i=1}^{K} \sum_{j=1}^{n} ||x_j - \mu_i||^2$$

Eq 1: Function to minimize distance between data points and centroids

The next step in this process revolves around creating a function, recommend_movies, responsible for generating movie recommendations based on a user-input movie ID.

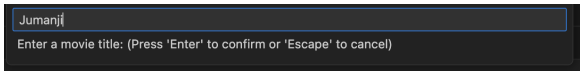

Fig 8(a): Input from user (movie name)

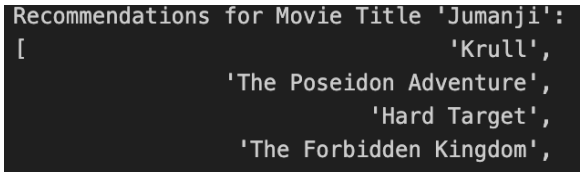

Fig 8(b): Recommendations with respect to movie entered by user.

This function extracts the cluster label of the specified movie from the dataset and identifies other movies belonging to the same cluster. These movies then constitute the recommendations, providing a set of relevant and similar films to the user's original choice. The output represents a user-friendly set of movie suggestions tailored to the individual's taste. This entire process is exemplified by recommending movies for a specific movie ID from the validation set, showcasing the practical application of the KMeans recommendation system.

Engineering Tradeoffs:

In contrast to RBMs, KMeans operates with significantly lower time and computational power requirements. The tradeoff here involves sacrificing some intricacy in capturing user preferences for the sake of faster processing times. KMeans excels in content-based clustering and offers valuable recommendations based on shared characteristics, making it a more computationally efficient option for recommendation systems.

8. CONCLUSION

We introduced a class of two-layer undirected graphical models (RBM's) and K means clustering model, suitable for modeling tabular or count data, and presented efficient learning and inference procedures for this class of models. We also demonstrated that RBM's can be successfully applied to a large dataset containing over 26 million user/movie ratings. The performance of a movie recommendation system depends on the characteristics of the dataset and the objectives of the recommendation. In the comparison between Restricted Boltzmann Machines (RBMs) and KMeans for movie recommendation, RBMs seems to outperform KMeans. RBMs excel in capturing intricate patterns and latent features in user-movie interactions, providing a more personalized and nuanced recommendation. They inherently handle

sparse data and consider user preferences more comprehensively.

KMeans, on the other hand, is effective in content-based clustering, grouping movies with similar attributes. While it offers valuable recommendations based on shared characteristics, it may lack the finesse of RBMs in capturing complex user preferences. Although, K Means using significantly lower time and computational power may be a factor of consideration before creating recommendation systems.

In conclusion, for a movie recommendation system that prioritizes personalization and user-specific patterns, RBMs perform better. However, a hybrid approach that leverages the strengths of both RBMs and KMeans could offer a well-rounded recommendation system, balancing individual preferences and content-based similarities.

Questions "Answered" After the Project:

Through this project, we've answered key questions related to the comparative performance of RBMs and KMeans in the context of movie recommendations. Specifically, we have demonstrated that RBMs excel in capturing intricate patterns and latent features in user-movie interactions, providing personalized recommendations. On the other hand, KMeans proves effective in content-based clustering, offering valuable recommendations based on shared characteristics. The project has provided a nuanced understanding of the strengths and limitations of these models, informing future decisions in the development of recommendation systems.

Still Curious About: While the project has delivered valuable insights, there remains a curiosity about the potential synergies between RBMs and other recommendation algorithms or models. Exploring hybrid approaches that combine RBMs with collaborative and content-based methods could potentially yield recommendation systems that offer a well-balanced mix of personalization and content relevance. This avenue of research holds the promise of further enhancing user satisfaction and engagement in the realm of recommendation systems.

In summary, this project serves as a stepping stone for future endeavors, prompting us to explore innovative approaches, refine existing models, and contribute to the continuous evolution of recommendation systems.

9. FUTURESCOPE

One major change with K Means could be the inclusion of user review on the movie along with their ratings. These reviews can be analyzed for emotion extraction which can then mathematically encoded to be used in K

Means algorithm. Understanding the users' perspective on movies would greatly help in providing accurate recommendations.

Recently, [9] derived a way to per- form fast, greedy learning of deep belief networks one layer at a time, with the top two layers forming an undirected bipartite graph which acts as an associative memory.

Looking forward, several avenues for extending this research present themselves. One notable extension involves the incorporation of additional contextual information, such as user reviews and sentiments, into the recommendation process. By leveraging natural language processing techniques, we can delve deeper into understanding user preferences and refine recommendations further. Additionally, exploring hybrid models that combine the strengths of RBMs and KMeans could potentially lead to even more robust and accurate recommendation systems.

References

1.Jayalakshmi, S.; Ganesh, N.; Čep, R.; Senthil Murugan, J. Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. Sensors 2022, 22, 4904. https://doi.org/10.3390/s22134904

2.Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of ACM World Wide Web Conference. 1. 10.1145/371920.372071.

3.R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in Proc. Int. Conf. Mach. Learning, Corvallis, OR, USA, June 2007, pp. 791–798.

4.Fischer, A., & Igel, C. (2012). An introduction to restricted Boltzmann machines. In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings 17 (pp. 14-36). Springer Berlin Heidelberg.

5.Shrivastava, R., & Singh, H. (2017). K-means clustering based solution of sparsity problem in rating based movie recommendation system. International Journal of Engineering and Management Research (IJEMR), 7(2), 309-314.

6.Hamerly, G., & Elkan, C. (2003). Learning the k in k-means. Advances in neural information processing systems, 16.

7.Sinaga, K. P., & Yang, M. S. (2020). Unsupervised K-means clustering algorithm. IEEE access, 8, 80716-80727.

8.Ralambondrainy, H. (1995). A conceptual version of the k-means algorithm. Pattern Recognition Letters, 16(11), 1147-1157.

9. Hinton, G.E., 2002. Training products of experts by minimizing contrastive divergence. Neural computation, 14(8), pp.1771–1800