Tropefy
(An interactive movie recommendation system using TV Tropes)

Girish U Rao: guru3@gatech.edu
Ramesh: rravi37@gatech.edu
Kunal: ksingh98@gatech.edu
Ritu:rparathody3@gatech.edu
Satya: vkamisetti3@gatech.edu
Narendra: dgadidasu3@gatech.edu

Abstract

The existing recommendation systems fall short of providing precise recommendation based on story lines. So, we propose a novel recommendation system that integrates three different entities: TV tropes, recommendation algorithms and interactive UI. Data is collected by scraping the website, which is fed into recommendation algorithm for apt recommendations and then these recommendations are presented to user through interactive interface.

1. Introduction

A trope can be defined as a storytelling device or convention, a storyteller can assume the audience will recognize. By looking at the tropes associated with a movie, one can get an understanding of its plot. The same can be used for movie recommendation.

Existing recommendation systems for movies currently operate by utilizing the user's watch history and ratings which doesn't necessarily be interesting to user. We aim to utilize the trope data to help solve this along with providing an interactive interface for recommendation. Our target users would be those who wish to explore their movie recommendations based on tropes in an interactive fashion.

2. Literature Survey

2.1. Recommendation Algorithms

Collaborative filtering based on user preference has the problem of cold-start and sparsity, which is partially over come in paper [1], by using genre correlation matrix based on the user's selection. There is also cold-start problem associated with an item(node).However, paper[2] calculates semantic similarity between items on the same level in a tree and provides more accurate recommendation with this data. Collaborative filtering via matrix factorization uses un-interpreted latent factors to make recommendations [3]. This paper [3], deals with generating a new recommendation model based on these interpretable latent factors such as tropes. In addition to matrix factorization methods, TagiCoFi[8] aims to combine user submitted tags so as to capture content based features into the model.

Sentiment analysis [4] using the positive or negative sentiment of the reviews posted online to suggest recommendations based on current movie being reviewed and doesn't

reflect user interest. Parsing the movie plot summaries to get the character information in [5] can be applied to get the story tropes proposed here. Surveying the user ratings to find user's interest for each of the movie and use this to cluster users to get the nearest neighbor to recommend a movie was discussed in [6]. For live and catch up TV applications, using the implicit and explicit feedback available to learn the ranking of the shows to improve the ranking system was proposed in [7]. An approach used in [12] is a modified K-means based clustering algorithm on users to find the most similar user to the target user to provide recommendations.

2.2. TV Tropes

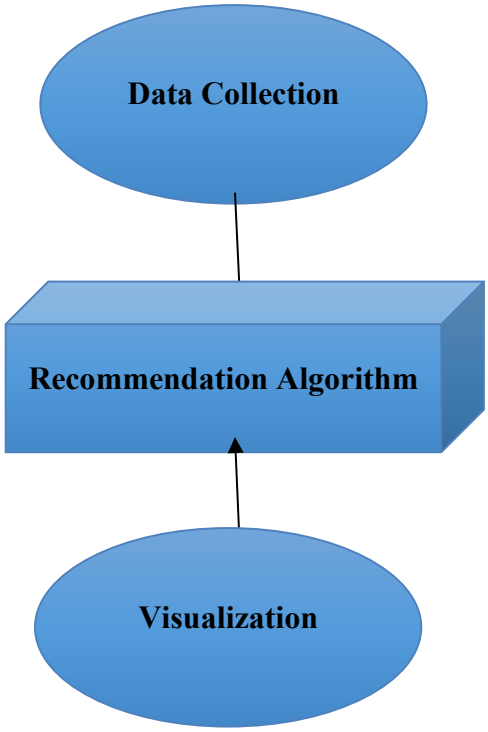
TV tropes are wiki that helps the viewers to identify and document the similar tropes in the media and literature. The paper [9], explains the bipartite graphs of the tropes that were implemented using network analysis techniques to study the propagation of the tropes. The paper [10] provides the descriptive analysis of the dataset that makes the future recommendation system smarter with the help of tropes by films and films by tropes approach. This paper gave help in suggesting the movies by popularity by giving good ontology. The paper [11], uses skip forward that makes use of the TV tropes in proving the content based recommendation using annotation recommenders. The paper fell short in including the overlapping annotations.

2.3. Visualization

Most of the existing recommendation engines present results as sorted lists. As a result, user can't understand the structure of the recommendations. To tackle this, a graph like representation is suggested in [13]. [14] Talks about an application called SmallWorlds that also suggests graph based interface for recommendations based on facebook. Paper [15] also presents graph based interface which is much more interactive and user driven and enhances sense-making by providing users the options to populate similar members, grouping them, fixing them and filtering further results based on groups. Based on [13], [14] and [15] the graph like representation was chosen for the project, Also, [16] suggests that, in addition to distance based structure, various levels of hierarchy of the clusters could provide some insight and proposes a method for the same. Also [17], suggests content based user profiling and then representing them visually using a cartoon giving the user, an overview of his own profile [18] combines the usual recommendation technique using social data across various platforms and marries that with a novel interactive interface which lets user to define weights of various inputs to recommendation engine, giving him better understanding of the output [19] proposes usage of interactive visualizations to involve user in the process of recommendation by visualizing the results through various suitable methods specific to various attributes of the recommendations.

PeerChooser [20], proposes an interface where connected users are positioned at a distance from our target user based on their similarity with them. Additionally the interface has genre nodes. They allow the user to move the genre nodes around, causing the connected users to rearrange thus affecting the recommendation results based on it.

3. Implementation



3.1. Data Collection

3.1.1. TV Tropes Website

- First approach was to retrieve TVTropes data via DBTropes in RDF format. Following drawbacks with RDF format were identified:
    - The linked data contained information about different sources and was tough to filter only movie data
    - The original RDF would have forced us to conform to the ontology defined for a semantic web domain which made it complex.

  Web scraping was suitable approach. Initial plan was to search for movie in the tvtropes website and then scrape data from the results. However, the result didn't have a set pattern to isolate tropes. Hence, a second approach was taken, that would scrape a list of movies and its hyperlink in the website. Then, navigate to the website for each movie in the master list and traverse through the content listing tropes. Python along with BeautifulSoup python library to parse the html was used. Each movie's tropes are then captured in csv file which will serve as data for the algorithms.

3.1.2. TMDB data:

The Movie DB is a huge movie data base which had user ratings and popularity numbers associated with each movie. This data is readily available for analysis via REST api, with movie name as the search key. The initial master list of movies scraped from the TvTropes database was used as the input master list to

fetch the TmDb ratings and movie description from the TmDb website using REST Http Get calls.(

3.1.3. Survey Data from users:

To evaluate the effectiveness of the recommendation system, a survey (https://www.surveymonkey.com/r/F69XXHB) was created to get user satisfaction results with respect to Tropefy and another popular movie recommendation site called tastedive.

3.2. Recommendation Algorithm

- Analysis and evaluation of the clustering algorithms
  - The raw data fed directly into the Clustering based collaborative filtering algorithm for finding the movies with similar tropes. The algorithm was implemented in python using scikit-surprise package. K-Nearest neighbors' (KNN) model was used to gauge the similarity output. As shown in Table [1], the results were not optimal enough to consider for further experiments. The raw data obtained was not fully sufficient to find the similarity. The data needs pre-processing. The pre-processing involves, including user preferred movies list and weighted tropes.

Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

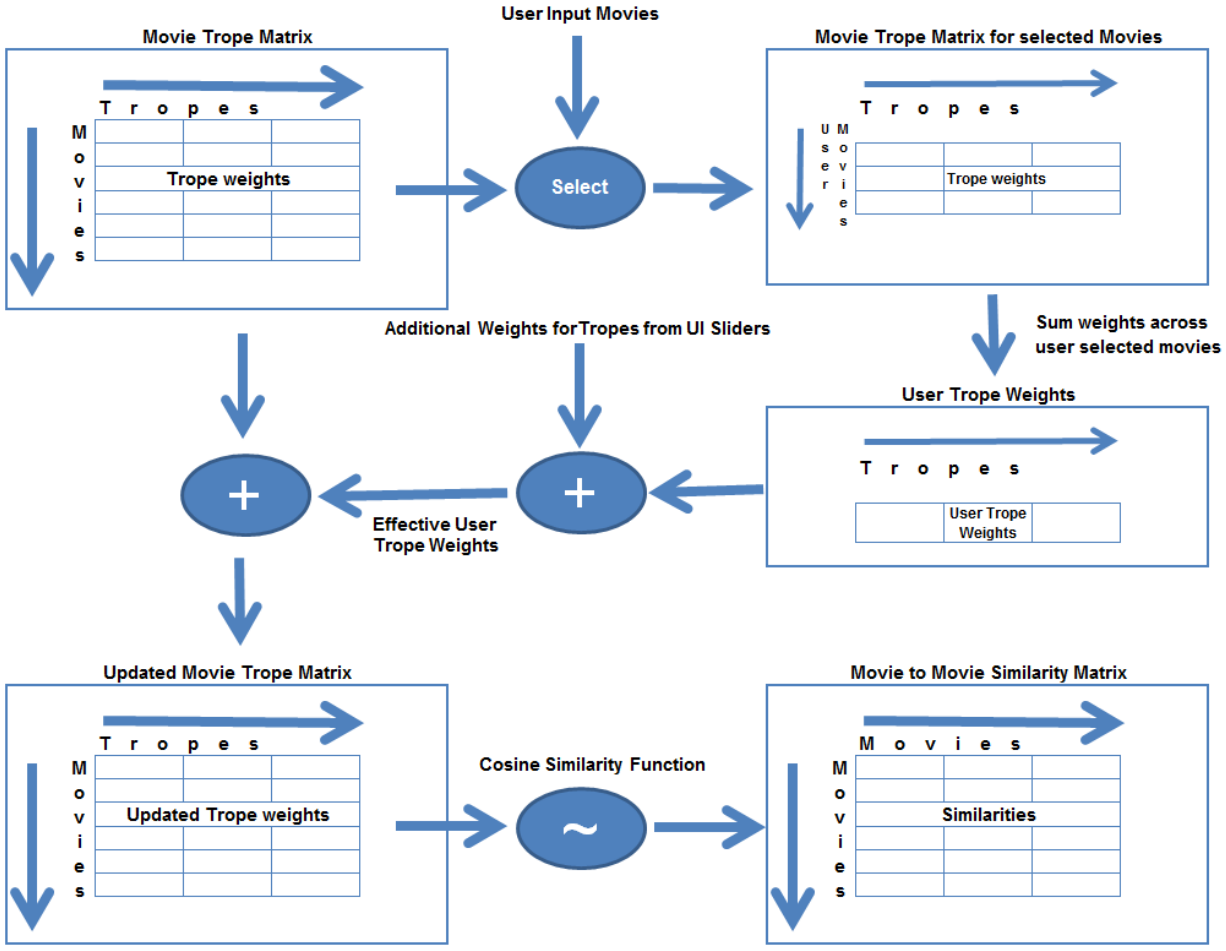
	Fold1	Fold2	Fold3	Fold4	Fold5	Mean	Std
MAE(testset)	8453.176	8565.126	8258.653	8146.744	8423.642	8369.468	148.3982
RMSE(testset)	13861.58	13884.02	13527.21	13374	13699.06	13669.18	195.628

Table 1: Clustering results

- Analysis and evaluation of collaborative filtering algorithms.
  - A matrix with all the movies as rows and the tropes as columns, which indicate the presence of that particular trope in the movie was created. Similarity measurement based on this input was good enough to find the common tropes and the nearest neighbor.
- Decision was made to continue with the second approach with following innovations.
  - Using the combination of frequency of a trope and weight of a trope to instead of just plain presence indicator.
  - Use of separate frequency and weightage measurement for the tropes present in the current user interest movies.

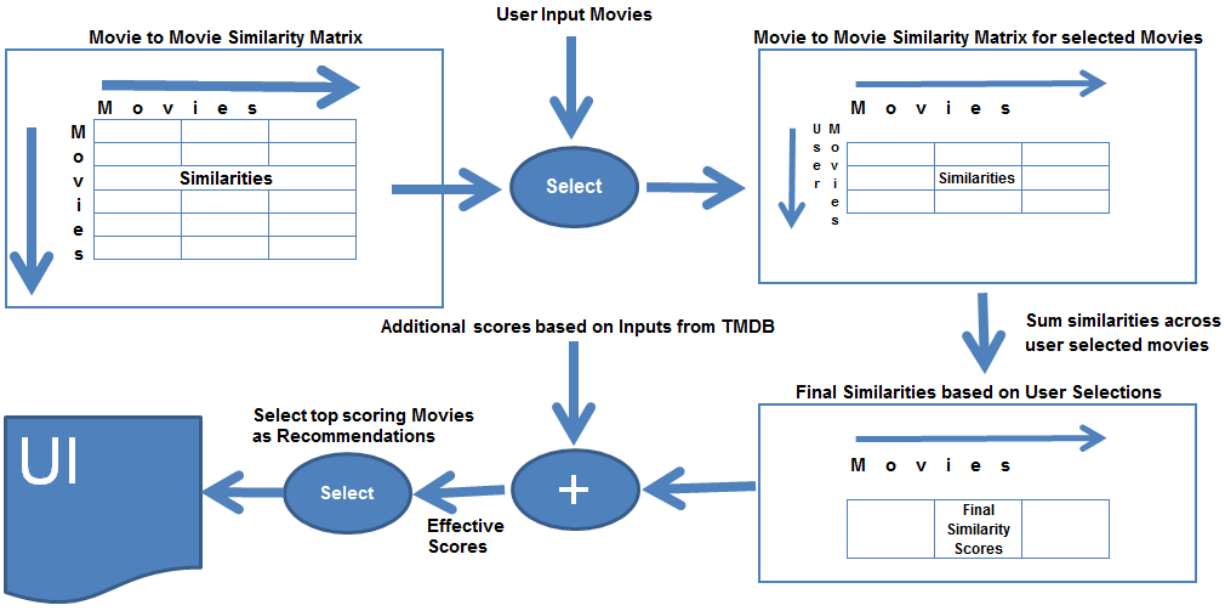
The frequency is the number of times a trope occurred in the whole movie database.

- **Proposed Method:** To start with, a matrix where each row represents a movie and each column represents a trope would be created. We call this Movie Trope Matrix (MTM). Each of the elements of the matrix, will be either 0, if a trope is not present in movie or it would be the Trope Weight (TW), if it is present. The Trope Weight (TW) of each trope will be based on its frequency in the entire set of movies being considered and it is defined as the value we obtain when the frequency corresponding to a trope is scaled using a scale whose domain is from 2 to maximum frequency of any trope and whose range is 1 to 0.5. This inverse mapping is to give low weightage to commonly occurring tropes.



Then, an input of one or more movies is taken from the user and for each trope, User Trope Weight (UTW) is calculated as the Trope Weight (TW) times the number of

user selected movies in which the trope is present. The top 10 tropes having the highest User Trope Weight (UTW) would be sent to the UI and presented to the user.



Now, each of the elements of the Movie Trope Matrix (MTM) would be multiplied by the User Trope Weight (UTW) of the corresponding trope of the element to get the updated Movie Trope Matrix (MTM). Using MTM, a cosine similarity score is generated for each and every pair of movies. Similarity between movies i and j is defined as $\sum_{t=1}^{T} w_{it} w_{jt}$, where t = 1, 2...T are various tropes and w_{xy} represent the weight of trope 'y' in movie 'x'.

Then, Final Similarity Score (FSS) is derived as the sum of the similarity scores of each of the user selected movies (This would be a single row with a score for all the movies in the dataset). Finally, 30 movies having the highest final similarity score would be shortlisted and the final similarity scores would be normalized to range from 0 to 1. This top 30 represent similar movies to user selection but we would like to show popular movies in these to user. So, for each of the 30 shortlisted movies, the properties, popularity and vote average would be extracted from The MovieDB (TMDB) website using the TMDB API (for speed we stored it locally). Then, popularity and vote average of each movie would be normalized to range from 0 to 1.

Then, the Effective Score (ES) for each of the 30 shortlisted movies would be the sum of normalized values of Final Similarity Score (FSS), vote average and popularity.

Finally, the 10 movies having the highest Effective Score (ES) would be given as the recommended movies to the user.

In addition, the user can give extra weights to the tropes present to him by using the slider in UI and these extra weights would be added to the User Trope Weight (UTW) and the entire algorithm would be re-triggered and new recommendations would be generated.

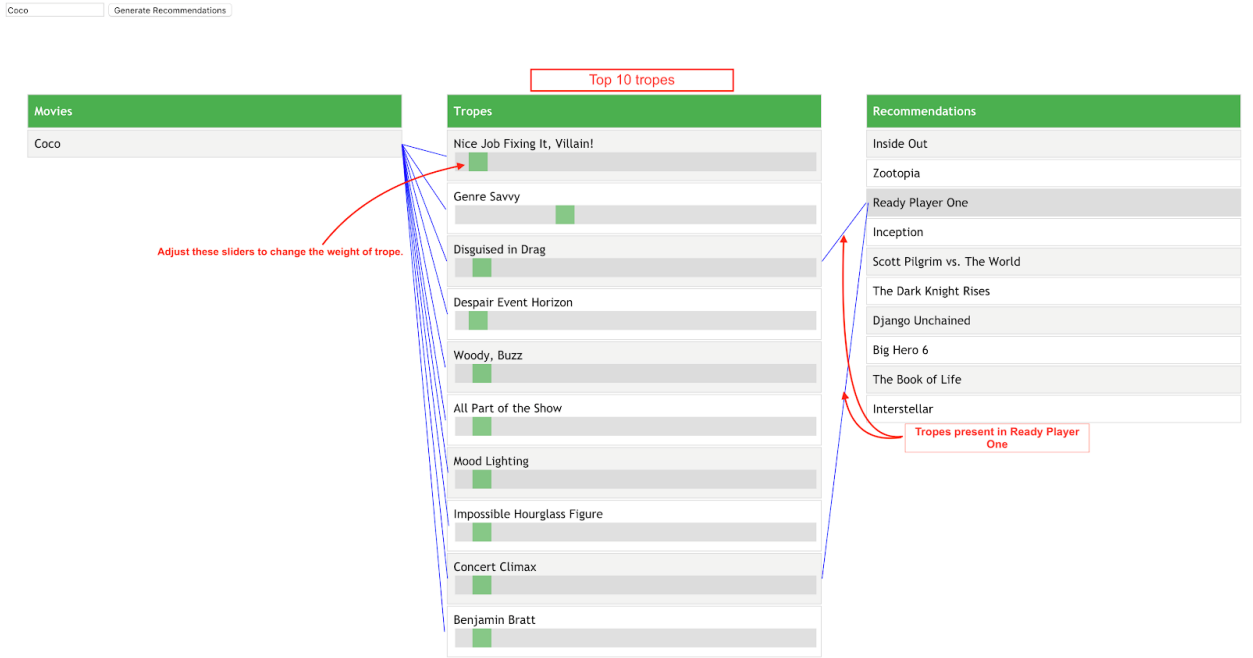
3.3. Visualization

Proposed approach was more focused on representing the relationships between the recommended movies and allows user to give feedback by controlling the recommendations itself by deleting items from the list. This approach doesn't capture as to why the user disliked the particular recommendation and gives little feedback to the algorithm to refine new recommendations.

The new visualization approach is as follows:

- Collect some movies as the baseline to understand user's current mood.
- Generate initial set of recommendations and present them along with the tropes used in generating them
- The tropes are shown as sliders which the user can navigate based on his liking of the recommendations provided.
- As the tropes are calibrated, new recommendations are presented to the user.

It provides explanation ability via the tropes presented and gives user control via the slider for it.



- We originally tried implementing the interface with D3.js and react but the Code was quickly getting quite complicated to add features. We shifted back to HTML/JavaScript and implemented our proposals given our time constraints.
- We created an API server using Python/Flask for our recommendation algorithm to serve the UI.
- We have implemented the visualization by creating three types of elements on the UI (Movies, tropes and recommendations).
- On obtaining the recommendations, we determine the positions for the links, which we receive from the API and draw them as zero width HTML elements with borders to mimic lines.
- Additionally upon hovering over any trope or movie, it shows only its connected links so as to present its influence in the entire recommendation flow.
- The sliders present along with each Trope indicates the weight of the tropes with respect to the current recommendations. These can be adjusted by the user which in turn produces new recommendations based on his changes.

4. Evaluation

Due to the nature of the idea, there is no ground truth data to systematically evaluate the performance of the results. So, we continued with user survey which takes in the feedback about our system and relative comparison with existing product tastedive.

Figure 1: Transparency of recommendations

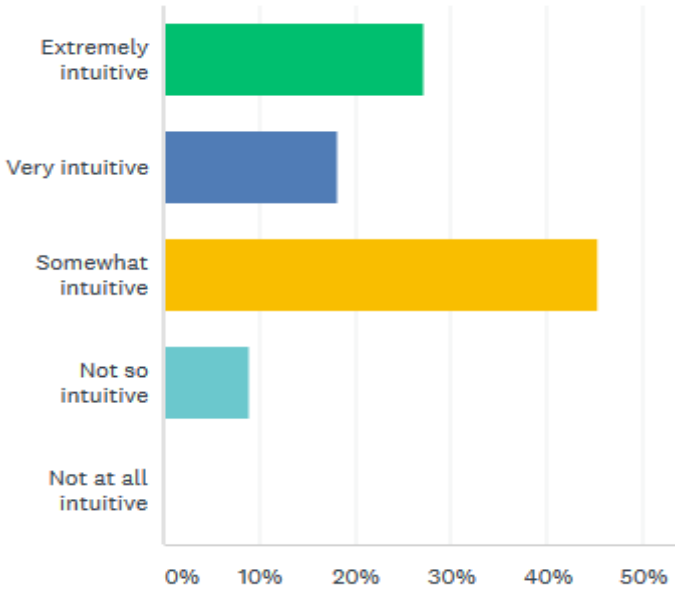


Figure 2: Transparency of recommendations

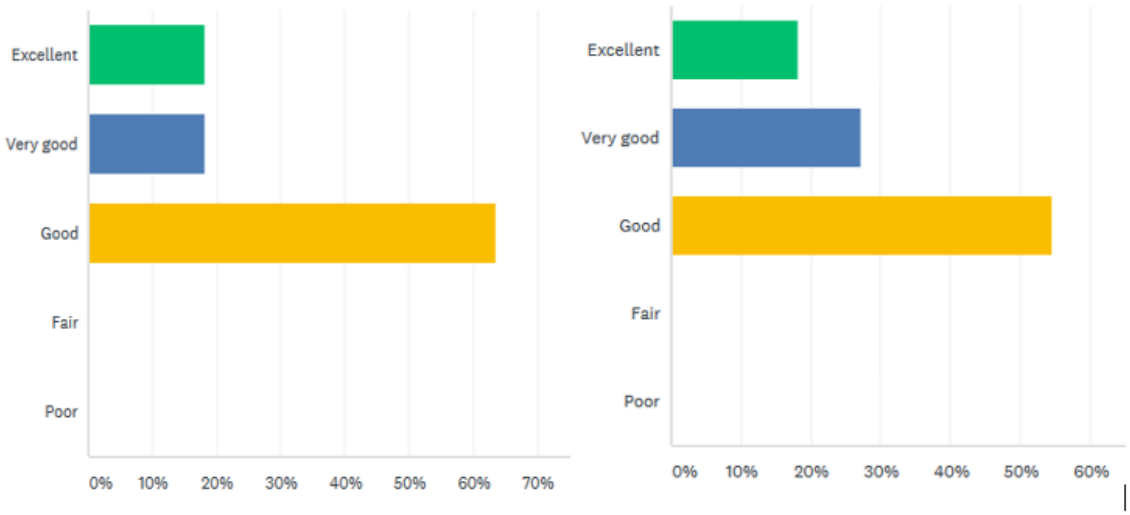


Figure 3 Tropefy vs Tastedive

Over all, end uses in our limited testing using surveys gave good rating for our system. And, they mainly liked the transparency in knowing how these recommendations were created.

5. Cost and Development Time

We used our personal computers to collect the data and develop the product. For final product demo, we hosted it at http://ramesharvind.pythonanywhere.com/ui. This costs us $5 for each month of hosting. As the system was developed with low memory foot print and optimized processing speed in mind, no additional high end equipment is needed.

It took 1.5 months after the approach is finalized to complete the work.

6. Distribution among team members:

Item	Main team member(s)
Data Collection	Ritu & Kunal
Explore and Finalize Algorithm	Narendra, Girish & Satya
Implement Algorithm	Narendra, Girish & Satya
Develop User Interface	Ramesh
User Surveys	Ritu & Kunal
Evaluation based on User Surveys	All
Enhancements based on Survey	All
Final Report and Presentation	All

7. Risks & Future work

Focusing only on the tvtropes data of a given movie. It might turn out to be insufficient data to make good recommendations for some particular set of users whose interest go in sync with other basic attributes such as genre and rating.

8. Future work

- Implement the enable/disable feature for tropes
- Tune the weights and mapping function for better results
- Removing outliers from the data

9. Conclusion

The recommendations presented with this method are really good in our limited experimentation and would like to enhance the work to make it a full product.

10. References

[1] Choi, S., Ko, S., & Han, Y. (2012). A movie recommendation algorithm based on genre correlations. Expert Syst. Appl., 39, 8079-8085.

[2] B. Juan, Collaborative filtering recommendation algorithm based on semantic similarity of item, 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), Nanjing, 2012, pp. 452-454

[3] Datta, A., Kovaleva, S., Mardziel, P., & Sen, S. (2017). Latent Factor Interpretations for Collaborative Filtering. CoRR, abs/1711.10816.

[4] Guimarães, R., Rodríguez, D. Z., Rosa, R. L., & Bressan, G. (2016, September). Recommendation system using sentiment analysis considering the polarity of the adverb. In Consumer Electronics (ISCE), 2016 IEEE International Symposium on (pp. 71-72). IEEE.

[5] Bamman, D., O'Connor, B., & Smith, N. A. (2013). Learning latent personas of film characters. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 352-361).

[6] Yu, Y., & Zhou, Y. (2017, March). Research on recommendation system based on interest clustering. In AIP Conference Proceedings (Vol. 1820, No. 1, p. 080021). AIP Publishing.

[7] Gonçalves, D., Costa, M., & Couto, F. M. (2016). A flexible recommendation system for cable TV. arXiv preprint arXiv:1609.02451.

[8] Zhen, Y., Li, W. J., & Yeung, D. Y. (2009, October). TagiCoFi: tag informed collaborative filtering. In Proceedings of the third ACM conference on Recommender systems (pp. 69-76). ACM.

[9] Mellina, C., & Svetlichnaya, S. (2011). Trope Propagation in the Cultural Space.

[10] García-Ortega, R. H., Merelo-Guervós, J. J., Sánchez, P. G., & Pitaru, G. (2018). Overview of PicTropes, a film trope dataset. arXiv preprint arXiv:1809.10959.

[11] Kiesel, M., & Mittag, F. (2011, October). Personalization in Skipforward, an Ontology-Based Distributed Annotation System. In SPIM (pp. 90-97).

[12] Wang, Z., Yu, X., Feng, N., & Wang, Z. (2014). An improved collaborative movie recommendation system using computational intelligence. Journal of Visual Languages & Computing, 25(6), 667-675.

[13] Z.Karni, L.Shapira IS&T/SPIE Electronic Imaging, 2013, Visualization and exploration for recommender systems in enterprise organization.

[14] Brynjar Gretarssony, John O'Donovanz, Svetlin Bostandjievx, Christopher Hall, Tobias Höllererk Department of Computer Science, University of California, Santa Barbara Eurographics/ IEEE-VGTC Symposium on Visualization (2010) SmallWorlds: Visualizing Social Recommendations

[15] Duen Horng (Polo) Chau, Aniket Kittur, Jason I. Hong, Christos Faloutsos Apolo: Making Sense of Large Network Data by combining Rich User Interaction and Machine Learning

[16] Wei Wang, Guangquan Zhang, and Jie Lu, Senior Member, IEEE Hierarchy Visualization for Group Recommender Systems

[17] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, Perfecto Herrera Semantic audio content-based music recommendation and visualization based on user preference examples

[18]Svetlin Bostandjiev, John O'Donovan, Tobias Höllerer TasteWeights: A Visual Interactive Hybrid Recommender System

[19]Christian Richthammer, Johannes Sanger, Gunther Pernul Interactive Visualization of Recommender Systems Data

[20] O'Donovan, J., Smyth, B., Gretarsson, B., Bostandjiev, S., & Höllerer, T. (2008, April). PeerChooser: visual interactive recommendation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1085-1088). ACM.