Group Project

19ZO02 - Social and Economic Network Analysis

Comparative Analysis of Social Recommender Systems

19Z306 – Aravind M

19Z309 – Bhooshaan A

19Z313 – Dharun Bharathi

19Z321 – Hrithik B

19Z329 – Pawan Kumar Babu

Problem Statement:

Technology world today revolves around making robust use of data available in abundance to gain insights from it and incorporate them in business activities to improve their services to consumers. One such domain of interest is the domain of product purchases in which there is a Long Tail Problem, wherein the number of popular products is very small when compared to the total volume of products available. Thus, recommendation systems prove helpful in recommending the unpopular products based on the users' interests. There still prevails a "Cold Start" issue where the new users to the system have very few preferences and "Data Sparsity" issue wherein the number of items each user rates are very small compared to the total number of items. Hence, based on the principle of Homophily, incorporating social information provides a solution and helps in making recommendations based on neighbor preferences. This task is popularly known as the social recommendation task. This project aims at performing a comparative analysis on the social recommendation models and inferring how they perform based on the network structure i.e., the communities that are present in the user-user interaction data (Social Information).

Dataset Description:

The social recommender systems require user-user interaction information as well as user-item interaction information. Three commonly uses datasets are chosen namely, Ciao, Epinions, and Filmtrust dataset.

Epinions:

Epinions is a trust network-based dataset of an online product rating site. It consists of the user's ratings on the items and the explicit trust information between the users. The nodes of the network are the users and items, thus forming a bipartite graph. The edges are directed and weighted which signifies the rating that a particular user has given to an item. The columns present in the user-item dataset are User ID, Item ID, Category ID, Rating, Helpfulness, Timestamp. The user-user dataset consists of two columns both of which are the user ids. The graph statistics of the dataset are shown below:

Users	18069
Items	261246
Ratings	762938
Rating Density	0.0162%
Social Connections	355530
Social Connection Density	0.1089%

Ciao:

Ciao dataset contains the rating information of the users given to the items and also contains the categorical information about the items. The columns present in the user-items dataset are: User ID, Item ID, Category ID, Rating, Helpfulness. The user-user dataset consists of two columns both of which are the user ids. The graph statistics of the dataset are shown below:

Users	7317
Items	104975
Ratings	283320
Rating Density	0.0369%
Social Connections	111781
Social Connection Density	0.2088%

FilmTrust:

FilmTrust consists of user ratings for movies and social trust relationships between the users. The columns present in the user-items dataset are User ID, Movie ID, Movie Rating.

Users	874
Items	1957
Ratings	18662
Rating Density	1.0911%
Social Connections	1853
Social Connection Density	0.2426%

<u>**Visualization of FilmTrust Dataset Using Gephi:**</u>

<u>User-Item Interaction Visualization:</u>

Red Nodes – Users
Blue Nodes – Items
Directed Weighted Edges indicate the ratings (Thickness)

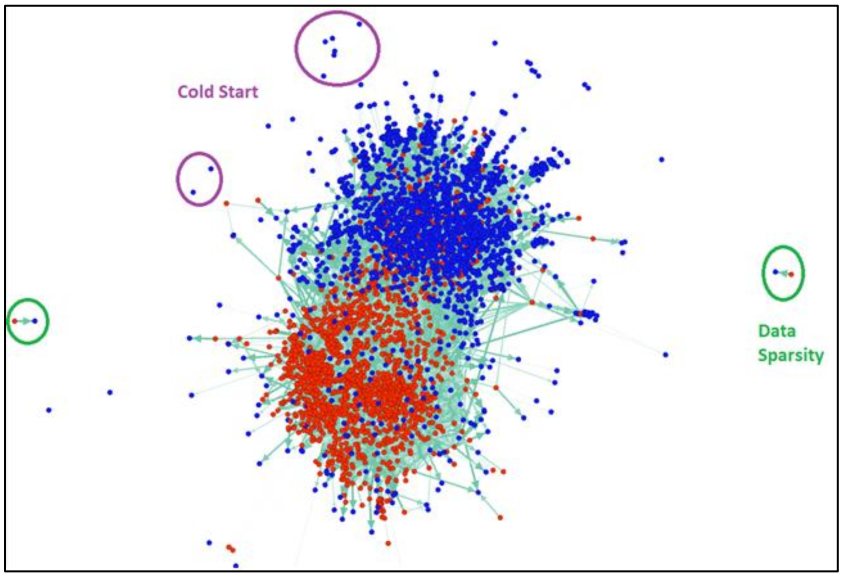


Figure 1. User-Item Interaction Graph

<u>Social Interaction Visualization:</u>

Nodes are colored based on their degrees.
The trust network is visualized as follows.
Some Graph based metric values were observed as follows

Avg. Path Length: 4.691	Avg. Clustering Coefficient: 0.13
Avg. Degree: 2.12	Network Diameter: 15

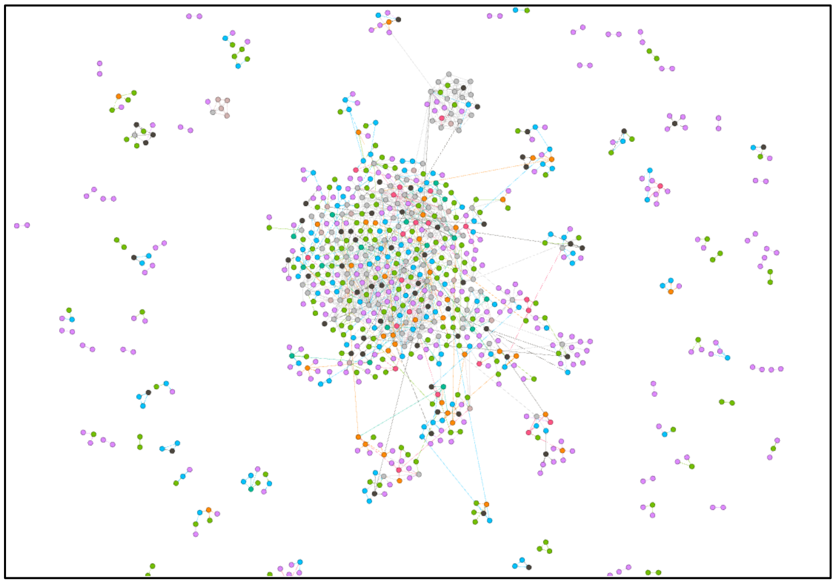


Figure 2. Trust Network Visualization

Tools Used:

Gephi:

Gephi is a visualization application used for network analysis. Gephi is also used to extract and analyze different graph metrics such as graph density, clustering-coefficient, avg-path length and etc.

NetworkX:

NetworkX is python graph library which is used to construct, manipulate and analyze the structure, features and metrics of the input graph.

PyTorch:

Pytorch is a python machine learning framework which provides tensor computation (makes use of GPU performance) and deep learning platform for high flexibility and speed.

Challenges Faced:

- The models due to its complexity took long execution times on large datasets such as Ciao and Epinions, even when enabling GPU on Google Colab. Hence, GPU server was made use of.
- Dependency issues due to change in versions of packages used were encountered and some versions of python packages used were difficult to obtain.
- When mutli-fold cross validation was performed, certain models had the issue of weight explosion and overflow in the values caused termination of code.
- ConsisRec model uses the dataset in pickled format. But the pickled format of epinions dataset was not available and the format in which the dataset must be pickled was also not provided by the authors of the model. Hence, Epinions dataset was not tested with the ConsisRec model.

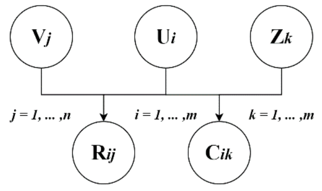
Contribution of Team Members:

Member	Contribution
Aravind M	ConsisRec Model Community Detection Algorithm
Bhooshaan A	GraphRec Model Inferential Plots, Slides and Report Work
Dharun Bharathi	SoRec Model
Hrithik B	RSTE Model TrustWalker Model Report Work
Pawan Kumar Babu	SoReg Model

Annexure I: Code:

SoRec[1]:

SoRec takes both user-item ratings and social network structure into consideration for recommendation predictions. The social network is considered in such a way that each cell will be denoted as w_{ij} representing how much the user u_i trusts the other user u_j ranging in a value of [0,1]. The matrix factorization is done using U^TZ and U^TV where U represents the user latent feature space, V denotes the low-dimensional item latent feature space and Z represents the factor matrix in social network graph. Here, $U \in R^{l \times m}$, $Z \in R^{l \times m}$ and $V \in R^{l \times n}$ with column vectors U_i representing user-specific, V_j representing item-specific and Z_k representing factor-specific latent feature vectors. The main idea of social network matrix factorization is deriving a *l*-dimensional feature representation U of users with high quality based on analyzing the social network graph. These matrices factorize to provide the matrices R and C where R represents the user-item interaction and C represents the social network matrix.



Some of the advantages of using this approach are, (1) method can deal with missing values problem (2) this method is interpreted using a probabilistic factor analysis model (3) as it scales linearly with the number of observations made, this approach can be applied to large datasets too. Considering information diffusion and propagation of ideas or interests will help in increasing the accuracy much more. This data fusing method using probabilistic matrix factorization is not only applicable in social recommendation, but also be applied to other popular research topics related to social search, information retrieval and data mining.

Code Adopted From:

https://github.com/hongleizhang/RSAlgorithms/blob/master/model/social_rec.py

SoReg[2]:

In this model, social information is incorporated as regularization terms along with the user-item matrix factorization. They propose two kinds of regularization namely,

Average-based regularization:

It is based on the principle that the average of the interests of a user's connections will approximate the user's interests. But in many cases, this may not be true as a user having more connections will not share a similar taste with most of his/her connections. Hence a similarity function is incorporated into the regularization term allowing the model to treat each connection of a user differently.

Individual-based regularization:

The average-based regularization constrains the user's interests to the average of connections' interests. When a user has diverse connections, it leads to information loss and inaccurate user modeling. Thus, another regularization term is incorporated to impose constraints on a user and their connections individually. The advantage of using this model is that it indirectly models the propagation of user interests.

SoReg takes into account all the user's connections in order to make recommendations. But, the existence of certain social connections retards the recommendation performance. Hence, it is necessary to choose a suitable group of connections for different recommendation tasks.

Code Adopted From:

https://github.com/hongleizhang/RSAlgorithms/blob/master/model/social_reg.py

RSTE[3]:

RSTE stands for **“Recommend with Social Trust Ensemble”.** This model extended the SoRec model and improved the efficiency of probability matrix factorization. The RSTE aims to improve the probability of finding the rating matrix (R) given the user-latent (U) feature matrix, item-latent (V) feature matrix and social graph (S) i.e.,

$$U^T V \cong R \ (1)$$

Equation 1 considers only user-item relationship for matrix factorization.

$$S.R \cong R' \ (2)$$

Equation 2 considers social network and predicts the rating matrix R', Each element in R' represents the rating influenced by their trusted neighbors only. Combining 1 and 2 we get,

$$S.U^T.V \cong R'$$

So here the main goal is to minimize the difference between $S.U^T.V \cong R'$, in turn to maximize the probability of finding R given U,V and S.In terms of conditional probability,

$$\text{Maximize}(P(\mathbf{R|U,V,S}))$$

Code Adopted From:

https://github.com/hongleizhang/RSAlgorithms/blob/master/model/social_rste.py

GraphRec[4]:

GraphRec iteratively aggregates the feature information from the local graph neighborhood using graph neural networks. It addresses the challenges faced by GNNs in building social recommender systems as both social and user-item graphs provide information about users from different perspectives. It aggregates information from social and user-item graphs, captures interactions and opinions between users and items jointly, and distinguishes the social relationships with heterogenous strengths. The model architecture consists of three components namely, user modeling, item modeling, and rating prediction. In the user modeling phase, it makes use of item aggregation and social aggregation in order to identify the user latent factors. Item aggregation deals with inferring information about a user by considering the interaction between the users and items in the user-item graph. Social aggregation deals with inferring information about a user by considering the social network interactions of a user. In the item modeling phase, it makes use of user aggregation to identify the latent factors of the item. The rating prediction phase combines the inferred user and item latent factors and makes use of an MLP which predicts the rating for a particular user-item pair. This is compared with the original rating and the loss is calculated which in turn is used to tune the model parameters. The results showed that this model outperformed most of the other models in terms of accuracy as it combines both user-item and user-user interactions in an effective manner. The main advantage of this model is that it considers heterogeneous strengths of social relationships and is able to distinguish the strengths of the ties. It has a limitation in that it considers the rating and social information as static whereas they are actually dynamic in real-world networks.

Code Adopted From:

https://github.com/Wang-Shuo/GraphRec_PyTorch

ConsisRec[5]:

Most of the GNNs handle social recommendation tasks by combining the user-item and user-user interactions simultaneously to learn node embeddings from them. Yet, there arise social inconsistencies where the social links are not necessarily consistent with predicted ratings. Having such inconsistent neighbors may result in the degraded performance of recommender systems. Social inconsistencies are categorized into two levels namely,

Context-level: Two users may be socially connected but may have interests in totally different contexts/domains.

Relation-level: Two users who are socially connected may rate the same item contrastingly.

In order to address these social inconsistencies, the paper proposes a framework named ConsisRec which learns embeddings only by considering consistent neighbors.

This model consists of four phases, namely

Embedding layer: Each user and item is represented as node embeddings and an additional relational embedding vector for each user-user relation is added to address relation level social inconsistency.

Query layer: In order to handle the social inconsistency problem, only the consistent neighbors are to be aggregated to learn the node embeddings. In order to handle both context level and relation level inconsistencies, this layer queries to identify consistent neighbors for each user-item pair. It then generates a query embedding by concatenating user and item embeddings.

Neighbor Sampling: A consistency score between the query embedding and the neighbor embeddings is calculated and based on this score aggregation of neighbors for a node is done by giving proper emphasis to consistent neighbors and giving lower importance to inconsistent ones.

Relation Attention: This module takes into account the relations to learn the importance of the sampled neighbors. The embeddings of the consistent neighbors that are sampled are aggregated and embeddings for each user and item are learned.

Rating Prediction: Using the embeddings generated for the user and the item, the rating is found for the user-item pair by taking the inner product of the embeddings.

Hence, this proves efficient in addressing the social inconsistency issues in the recommender systems.

Code Adopted From: https://github.com/YangLiangwei/ConsisRec

TrustWalker[6]:

This model proposes a random walk approach to find a target rating R_{CI} i.e., rating of a particular item I by user C. The result is an aggregation of multiple random walks. A single random walk starts from the user C and item I to predict R_{CI}. Next step of the random walker from the current user C will depend on the user-user network. The random walker moves from user C to user B, if and only if C trusts B. If the node B has rated the target item I, then the rating of item I by user B is returned. But if the path between the target user and the user (whose rating has been returned) is too large, the precision of the rating may get affected. To avoid this, sometimes rating of item similar to I is returned which in turn reduces the path length. Evaluation on epinions dataset was performed, demonstrating that Trust Walker out performs both collaborative filtering methods and purely trust-based methods specially in terms of coverage.

Code Adopted From:
https://github.com/hongleizhang/RSAlgorithms/blob/master/model/trust_walker.py

Community Detection Algorithms on Trust Network of Datasets:

Two algorithms Louvain and were used to detect the number of communities in the social network information.

Louvain Algorithm:

Louvain algorithm is an unsupervised approach that is divided into two phases. The first phase deals with finding the modularity and optimizing it. The second phase uses the modularity values in the previous phase to aggregate nodes into communities. This method does not take as input the number of communities or their sizes prior to execution. Both the steps will be carried out up until the network has undergone all possible alterations and has reached its maximum modularity.

Leiden Algorithm:

The Louvain method is improved by the Leiden algorithm, which ensures that communities are highly interconnected. It also benefits from the notions of speeding up local node movement and shifting nodes to randomized neighbors. The algorithm involves three stages: local node shifting, partition refining, and network aggregation based on the partitions that are refined, with the aggregate network's initial partition being made from the non-refined partition. Compared to the Louvain algorithm, this algorithm is significantly more complicated.

Code:

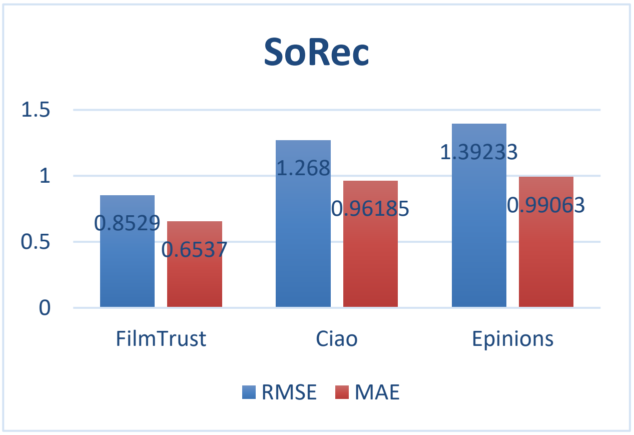
https://colab.research.google.com/drive/1jKi5uui3VFjEOA042jggEQ81Gw4FHXcC

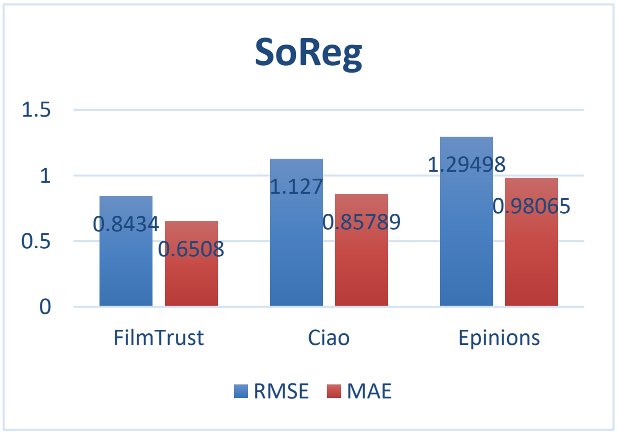
Inferring Graph Metrics of Trust Network of Datasets:

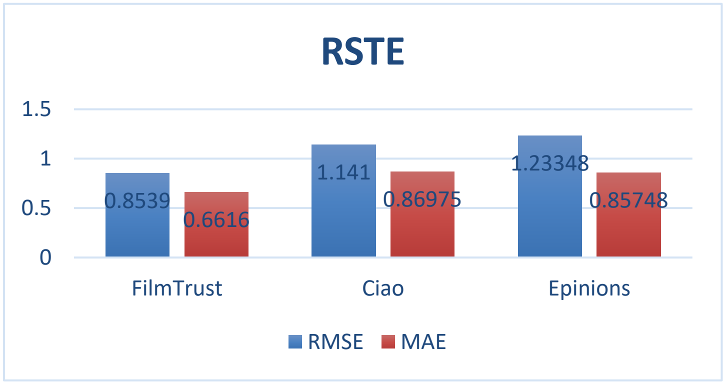
Since, the Epinions and Ciao datasets are comparatively large in size, the graph statistics would not be possible to be inferred from Gephi tool. So, networkx module of python was used to infer statistics using the code below.

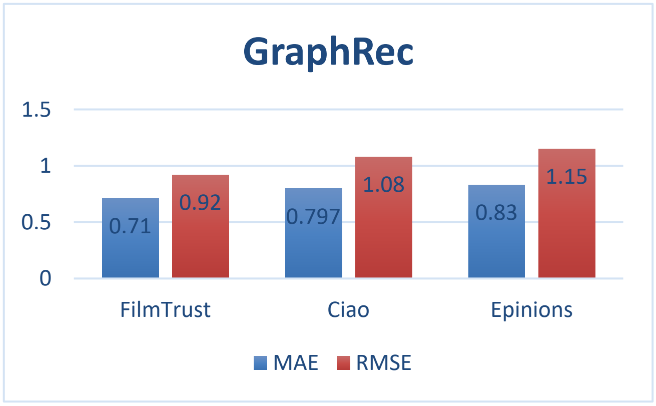
Code:
https://colab.research.google.com/drive/1Cpm8Mo1rwi_MfaxXlzZQExJQGEvjI2EW?usp=sharing

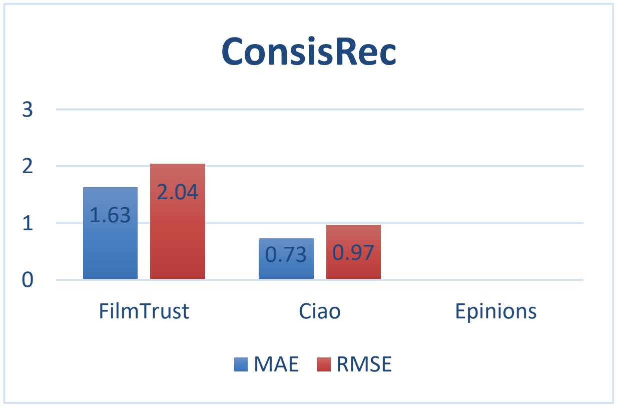
Annexure II: Snapshots of Output – Inferential Plots:

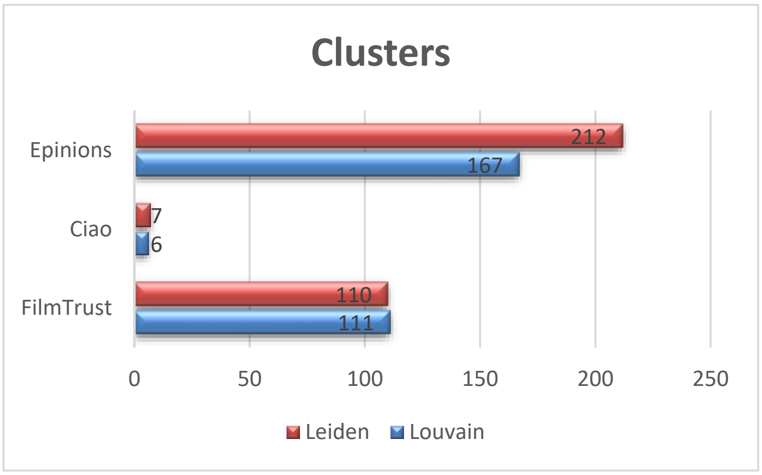


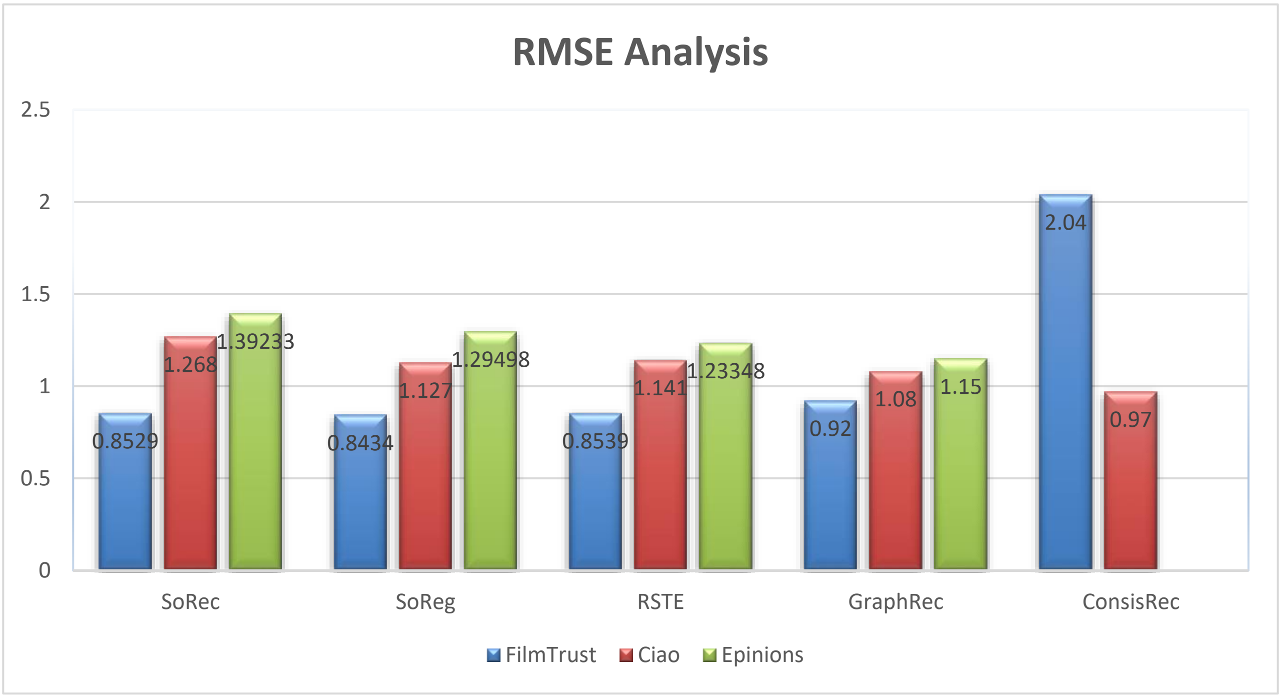












RMSE Analysis for Epinions, Ciao and Filmtrust

Inferences from Plots:

The number of communities were inferred using both Louvain and Leiden algorithms and it is found that both the algorithms resulted in similar number of communities except for the Epinions dataset. The number of communities present in the dataset was found not to influence much on the recommendation performance of the models due to the fact that though there is a huge variation in the number of communities, the average clustering coefficient of all the three datasets was found to be in the similar range (0.13 to 0.16). Since, clustering coefficients decide on how dense the friends circle are, it can be observed that all the models perform in a similar manner for each of the datasets. GraphRec model proved to be proficient in both Ciao and Epinions datasets in terms of their RMSE values as they employ Graph Neural Networks. The SoRec, SoReg and RSTE models perform equivalently and the latter models perform slightly better than the other since each are improvements over the previous models.

Hence, it is found that the clustering or the density of the datasets have an impact on recommendation performance and that graph neural network-based methods perform effectively. Further work can be presented on how graph neural network-based methods can be improvised based on other aspects such as privacy preservation in recommendations.

References:

[1] SoRec: Social Recommendation Using Probabilistic Matrix Factorization By Hao Ma, Haixuan Yang, Michael R. Lyu, Irwin King

[2] Recommender Systems with Social Regularization By Hao Ma, Dengyong Zhou, Chao Liu

[3] Learning to Recommend with Social Trust Ensemble By Hao Ma, Irwin King and Michael R. Lyu

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation.

[5] ConsisRec: Enhancing GNN for Social Recommendation via Consistent Neighbor Aggregation By Liangwei Yang, Zhiwei Liu, Yingtong Dou, Jing Ma, Philip S. Yu

[6] TrustWalker: A Random Walk Model for Combining -Trust-based and Item-based Recommendation By Mohsen Jamali and Martin Ester

[6] Gephi – Network Visualization: https://gephi.org/users/

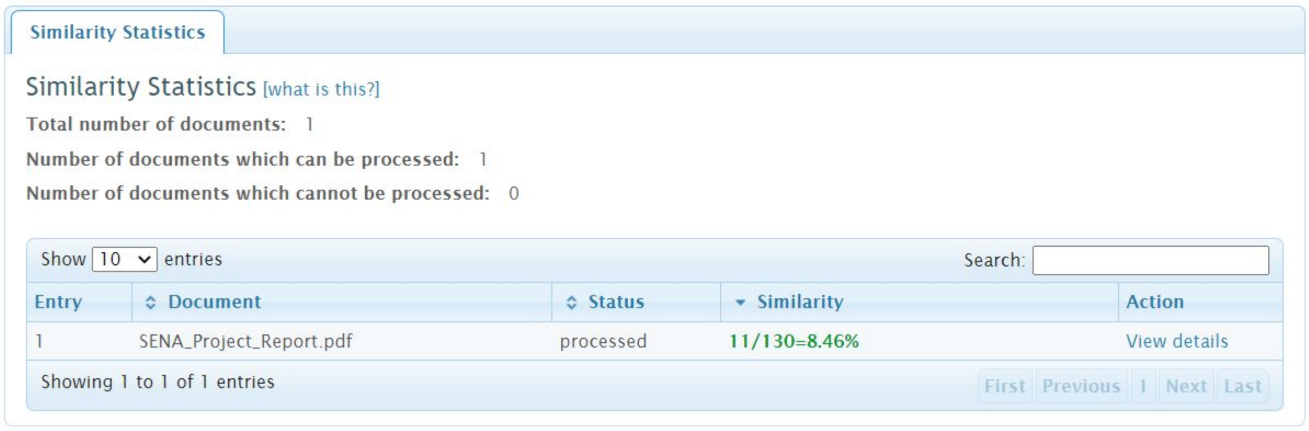
[7] Community Detection Algorithms: https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae

[8] Epinions Dataset: https://snap.stanford.edu/data/soc-Epinions1.html

[9] Ciao Dataset: https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm

[10] FilmTrust Dataset: http://konect.cc/networks/librec-filmtrust-trust/

Plagiarism Report:


Similarity Statistics

Similarity Statistics [what is this?]

Total number of documents: 1

Number of documents which can be processed: 1

Number of documents which cannot be processed: 0

Show 10 entries Search:

Entry	Document	Status	Similarity	Action
1	SENA_Project_Report.pdf	processed	11/130=8.46%	View details

Showing 1 to 1 of 1 entries