Content Recommendation System- Interim Report

Shubham Kumar
2015098
IIIT- Delhi
shubham15098@iiitd.ac.in

Ishmeet Kaur
2015042
IIIT- Delhi
ishmeet15042@iiitd.ac.in

1. Introduction

With increasing no. of blogs and articles all over the web these days, one might get lost in irrelevant articles. So, we aim to build algorithms to recommend content(articles) that are similar to those that a user liked in the past.

2. Related Work

2.1. List of related works we referred to

1. **Quick Guide to Build a Recommendation Engine in Python:**
   Access the resource- Here
   It covers the fundamentals of creating a recommendation system using GraphLab in Python. It gives a basic idea how recommendation work using basic popularity model and a collaborative filtering model.

2. **Process Mining with sequence clustering:**
   Access the resource- Here
   This paper talks about sequence clustering- a technique of bioinformatics that is used to discover the properties of sequences by grouping them into clusters and assigning each sequence to one of those clusters.

3. **Towards Effective Research-Paper Recommender Systems and User Modeling:**
   Access the resource- Here
   This is a bible for recommender systems, having dozens of research papers and related articles.

2.2. Best results obtained so far

Our dataset is a pretty rare and confidential one, so there aren't many research papers written on it(we couldn't find any).. However, research papers which worked on similar content recommendation on different datasets achieved .9166 accuracy.

3. Dataset and Evaluation

3.1. A small description

The dataset is a log of users and articles they have accessed along with any action they took like like/comment etc over a period of one year. The link to the dataset is mentioned in references. The data is diverse, having information of users and complete article text along with timestamp, location, browser information, language etc.

3.2. Preprocessing and no. of samples in training, validation and test set

In our dataset, a user can interact with the articles in four ways:- VIEW, LIKE, COMMENT and BOOKMARK.Interactions must have different weight. Thus we made a new column called event strength which saves the weight of each interaction.
event type strength = VIEW: 1.0, LIKE: 2.0, BOOKMARK: 2.5, FOLLOW: 3.0, COMMENT CREATED: 4.0
Before any preprocessing, these were the stats:
No. of user interactions: 72312
No. of distinct articles: 2987
No. of distinct users: 1895
Then for preprocessing, we are only considering those users which have atleast 5 interactions. Also, a user can both share and remove articles, but, we are only considering those users which have shared the article. After this we also divide our dataset into training, validation and test set(60-20-20 respectively).
No. of interactions on Train set: 23462
No. of interactions on Validation set: 7822
No. of interactions on Test set: 7822

3.3. Feature extraction

We are using tfidf vectorisation which involves token extraction, filtering stop words, etc.

3.4. Evaluation metrics

1. **Cross Validation Technique** Dataset divided into training set, validation set and test set. Validation set

used to improve our model and adjust the hyperparameters. Finally applied on test set to get the results.

2. **Top-N accuracy metrics** It evaluates the accuracy of the top recommendations provided to a user, comparing to the items the user has actually interacted in test set.
   For each user:
   For each item the user has interacted in test set

   - Sample 100 other items the user has never interacted.
   - Ask the recommender model to produce a ranked list of recommended items, from a set composed one interacted item and the 100 non-interacted (non-relevant!) items
   - Compute the Top-N accuracy metrics for this user and interacted item from the recommendations ranked list

   Recall@5 = x%, means x% of interacted items in test set were ranked by Popularity model among the top-5 items (from lists with 100 random items)
   Recall@10 = x%, means x% of interacted items in test set were ranked by Popularity model among the top-10 items (from lists with 100 random items)

4. Methadology

4.1. The design choices

1. **Popularity Model**
   We computed the most popular articles based on number of views on an article.The graphs shows the accuracy of this model.

2. **Content-Based Filtering Model**
   For articles, we are using TF-IDF vectorisation. For every user we calculate the most interacted article by him/her. Then using this result we found the relevant tokens. Computes the cosine similarity between the user profile and all item profiles.Predict the most similar item.
   To improve upon this method, what we did was instead of calculating the similarities using tfidf cosine similarity, we used vector embedding, glove. Glove first vectorises the words and then compares them. We took an average of all the useful words in the article and then compared teh similarities.
   The results improved significantly. Hyperparameter setting: Referring to the sklearn page on tfidf vectorisation, we set our parameters. the best ones were chosen using grid search. Here's the list of parameters we used:

   - ngram_range
   - tuple (min_n, max_n): The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min_n ¡= n ¡= max_n will be used.
   - Max_df
   - Min_df
   - Norm : l1, l2 or None, optional: Norm used to normalize term vectors. None for no normalization.
   - Smooth_idf: we had also used this hyperparameter. But it was not making any difference in accuracy. Thus, we eliminated it.

   The difference in accuracies on changing these can be seen in the graphs. Also, the change in results and inference has been discussed in detail in the next section. Glove defintely improved the recommendations drastically, reaching an accuracy of 0.86!

3. **Collaborative Filtering model**
   Matrix Factorization is done via Singular Value Decomposition. First we make a matrix R (with users in rows and items in columns). Initially it is a sparse matrix, we try to remove its sparsity. Matrix factorization is the breaking down of one matrix into a product of multiple matrices.
   R = U Sigma Vt SVD- implemented using svds in scipy library.After the factorization, we try to to reconstruct the original matrix by multiplying its factors. The resulting matrix is not sparse any more.
   Now SVd results were not that good, so we tried to use another method for sparsity removal. The method is listed below.
   However, accuracy did not improve, rather it fell down. So, we tried the hybrid of both methods: SVD and the other one, accuracy improved slightly, but still not as good as content-based. We have inferred the results in the next section.

   Hyperparameters:

   k : int, optional Number of singular values and vectors to compute. Must be 1 ¡= k ¡ min(A.shape). We take k = 1,5,10,15,20,25 in our grid search and found k = 15 was best.
   Content-Based approach is being benefited by the rich item attributes (text) for a better modeling of user's preferences. Hence, performing much better than collaborative filtering.

4. **Hybrid Recommender**

   In this technique, we combine both the models-content-based and collaborative filtering. We simply, multiply the CF score with the Content-Based score, and ranking by resulting score.

5. **Matrix sparsity removal based on document similarity:** For CF model, we used SVD for removing the sparsity of matrix. Now we will use a better technique. We had a matrix R (with users in rows and items in columns). To remove sparsity in R,

   New_Matrix = R X C

   Where C is item x item matrix. When we multiply R with C, we will get a New_Matrix with removed sparsity, which we will further use for CF.
   To obtain this matrix, we used cosine similarity obtained by tfidf vectorisation. Using this content x content matrix, we remove sparsity of the user x content matrix.
   However, the results which we were expecting were not upto the mark. So, later we used both methods - SVD and this one, accuracy improved, but overall it was still not upto the mark.
   This method failed.

6. **Sequence clustering using Markovs Chain and Expectation maximisation:**
   Sequence clustering is used to discover the properties of sequences by grouping them into clusters and assigning each sequence to one of those cluster.

   Each cluster is associated with a first-order Markov chain, where the current state depends only on the previous state. The probability that an observed sequence belongs to a given cluster is in effect the probability that the observed sequence was produced by the Markov chain associated with that cluster. For a sequence x = x0, x1, x2,,xL-1 of length L this can be expressed simply as:

   p(x — ck) = p(x0, ck). mult(xi — xi -1, ck)

   To implement this method, we decided to use Microsoft visual studio to implement the algorithm, as was suggested in the research paper we referred to and to manage our database and server we used microsoft sql server management tool and for configuring the two together, microsoft server configurator was deployed.
   We gave 2 weeks of time to this, saw tons of tutorials and when Shubham was stuck, I also dug deep into it, but alas, we were not able to implement this.

   The screenshots of the problem are attached. Also, we were so close to it, couldn't figure out the problem.
   Also, there are hardly any tutorials on this, we figured everything out mostly from the main documentation itself. We first saw all the tutorials and were now ready to implement it, but we couldn't move beyond a point. The database and the algorithm were setup, but there was some problem with the server.

   We were not able to accomplish it, even after a galore of efforts.

4.2. Supporting Evidence

Graphs have been attached towards the end of this report. Our project is not based on classification, it's a recommendation system. Hence, the graphs show the plots of accuracy achieved in prediction using different methods. Also, it shows the results under different conditions of hyperparameters. Thus, helping us chose the best one.
Screenshots for the problem in visual studio have also been attached.

5. Results and Analysis

5.1. Results obtained and Interpretation

1. We achieved best accuracy for content based model(0.86)

2. For popularity model, we plotted a graph varying the weights of interaction and number of minimum interactions. We found best result came with equal weights and (interactions ¿=5).

3. However accuracy was only 375 in the best case, under recall@10.

4. So, we start looking for better methods, which take the user into consideration while making recommendations.

5. In Content Filtering model, tuning the hyperparameters increased the accuracy steadily. Best accuracy achieved was 64%.

6. Till now this was the best accuracy, but then came into play- Glove- the gamechanger.

7. Now the accuracy for content based recommendations boomed upto **0.886174**!!!!

8. Compared to others, this is the best accuracy among all the models.

9. Now we implemented Colaborative filtering.

10. The accuracy of this model is greater than popularity model but less than content-based filtering model. Its about 45%.

11. This is because Content-Based approach is being benefited by the rich item attributes (text) for a better modeling of user's preferences.

12. Collaborative filtering v2.0: used a different method for matrix sparsity removal.

13. We were disappointed with the poor accuracy of just 30%. This is even worse than before.

14. Then when we combined bothe SVD and cosine similarity for sparsity removal, we achieved best results, about 48% accuracy.

15. Overall, collaborative was a poor method, but better than popularity method. Then we switched to hybrid method.

16. Hybrid technique was initially better than CF and content-based approach. But in some cases if we find optimal parameters for content-based approach, it results better than hybrid technique.

17. But now, after incorporating glove, content based was always better. Hybrid ave accuracy of about 70%. Good but not great!

18. We were pretty happy with the content based accuracy, as it is really close to the state-of the-art accuracy.

5.2. Insights

1. Learnt the basic functioning of recommendation systems and its importance in e-commerce.

2. Method to implement recommendation system and how to evaluate its result using recall.

3. Natural Language processing: tools like tfidf and GloVe. Choosing the best out of both.

4. Selecting the best hyper parameter using ablative analysis.

5. Theory of Markov's chain and sequence clustering. How a sequence can be linked to a cluster and further used for recommendation.

6. Sparsity removal of a matrix using various methods to obtain the best result.

5.3. Gap between our model's performance and the state-of-the-art

Our results are not bad as we achieved about 88%accuracy. The best we found was about 91%.

6. Contributions

6.1. Deliverables

The deliverables in the report were as follows:
Ishmeet: Matrix sparsity removal based on document similarity, Data Selection and training the model
Shubham: Clustering of activity sequence using expectation maximization and Markov chains.
We've successfully implemented the matrix sparsity removal method together, although, the results were not as we had expected. We tried to implement sequence clustering as well till the last minute, but failed ultimately. But, we have implemented 4 other models in place of that from scratch, which have given us a great accuracy as well.Overall:

6.2. References & Citations

Here's the link to the resources we followed:
Dataset
Matrix factorisation recommender
Sklearn tfidf vectorisation
tutorial
tutorial
tutorial
wiki
tutorial
tutorial
tutorial
quora
tutorial
tutorial
tutorial
tutorial
tutorial
tutorial
tutorial
VS
VS
VS
VS
VS
VS

6.3. Individual Contributions:

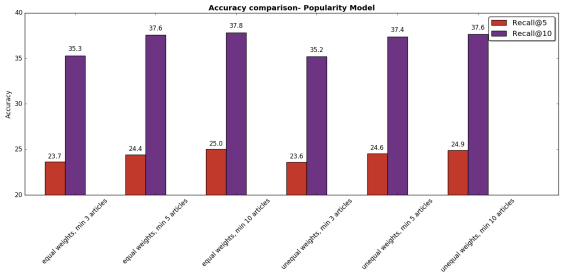
Each team member worked equally hard and also shared the load of the other one when he/she was stuck.

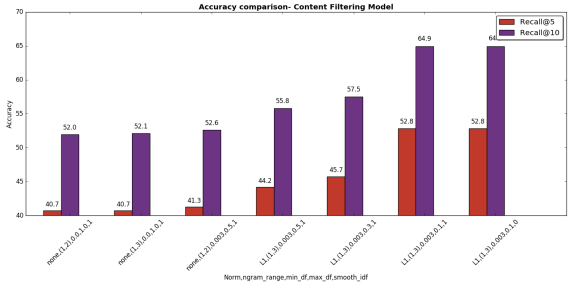
Overall contribution:
Ishmeet: Finding both the parts of the very rare dataset, complete implementation of Popularity Model, complete implementation of content based model, creating the content x content matrix for sparcity removal, attempts at glove for creating the content x content matrix, visual studio,

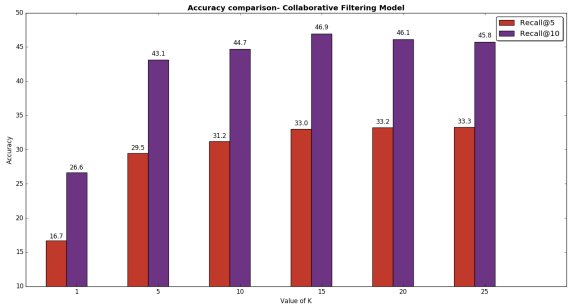
sql server management studio and server configuration software for sequence clustering.
Shubham: Preprocessing of dataset(removing users with inconsiderable amount of interactions and taking the action of user into consideration like VIEW/LIKE/SHARE etc.), complete implementation of collaborative filtering model, removing sparsity from the matrix given by Ishmeet and analyzing the accuracy, visual studio, sql server management studio and server configuration software for sequence clustering along with implementation tutorials

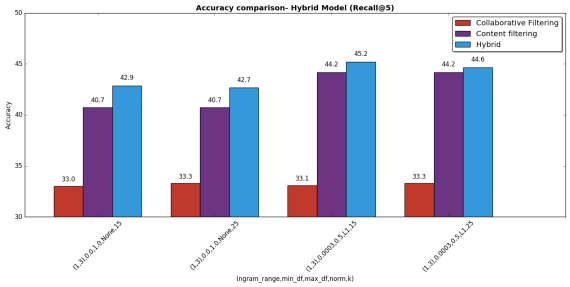
List of files:
Ishmeet: contentbased.py, popularity.py
Shubham: hybrid.py, cf.py, pickle files

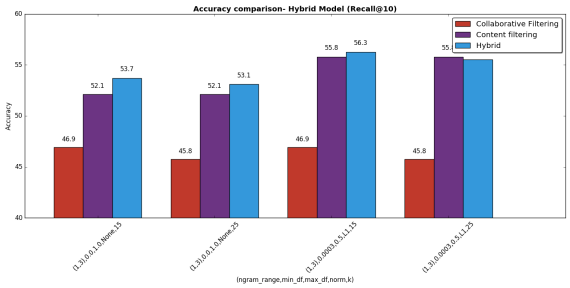
7. Graphs and Screenshots

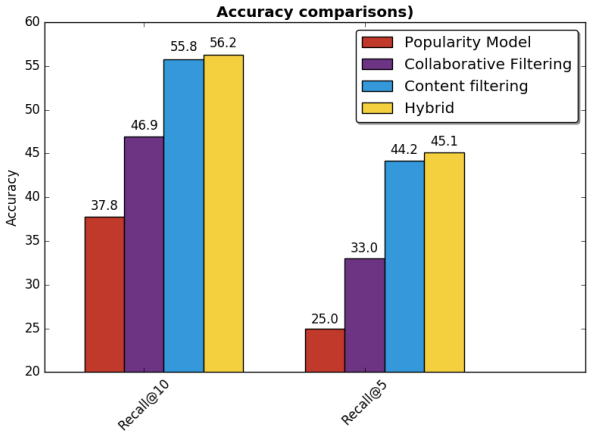




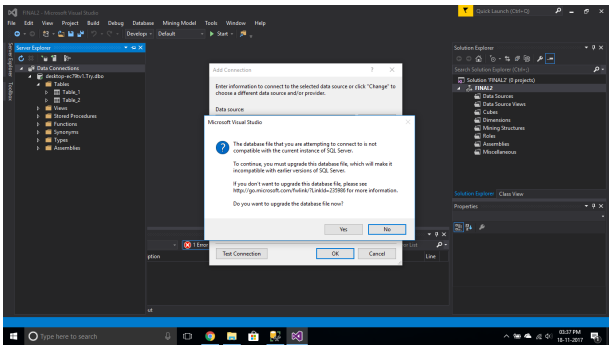


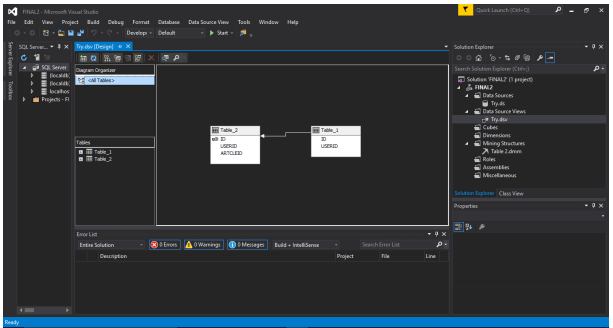


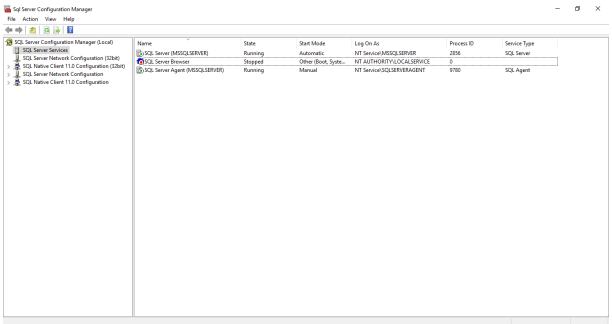


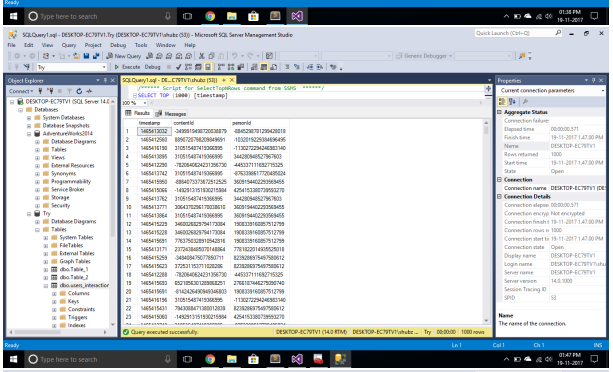


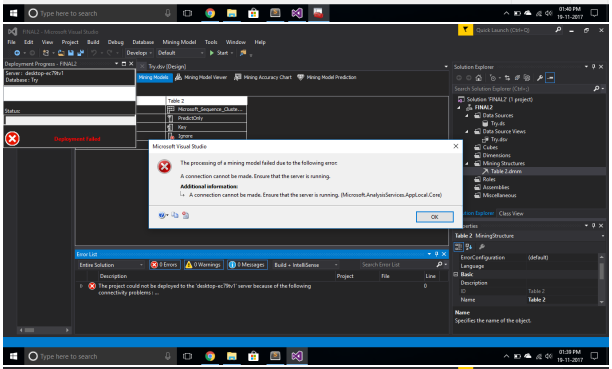
The screenshots for visual studio:

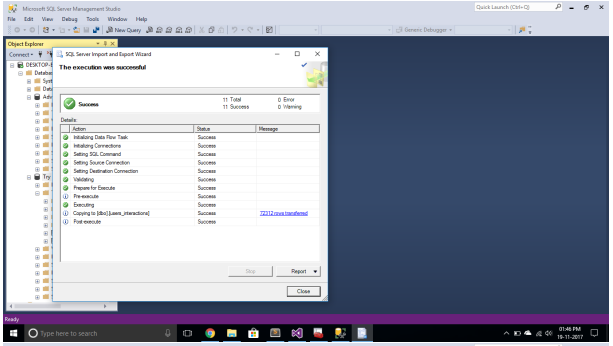


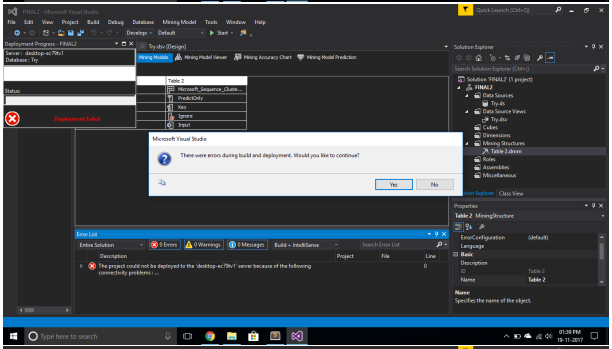


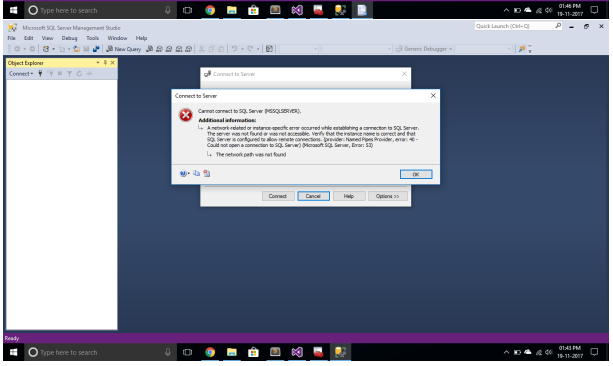


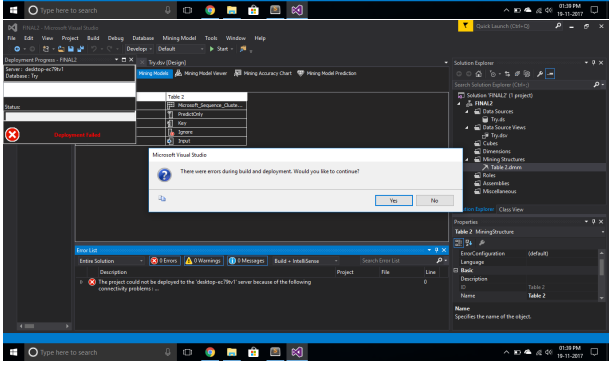


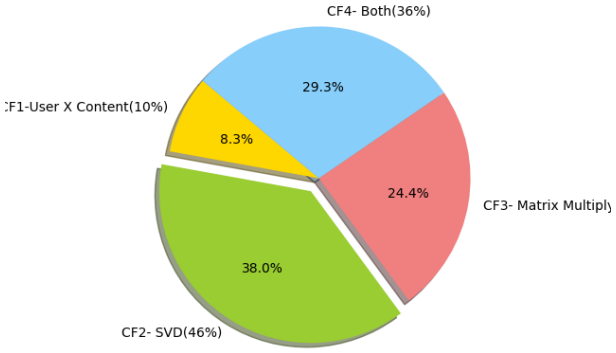


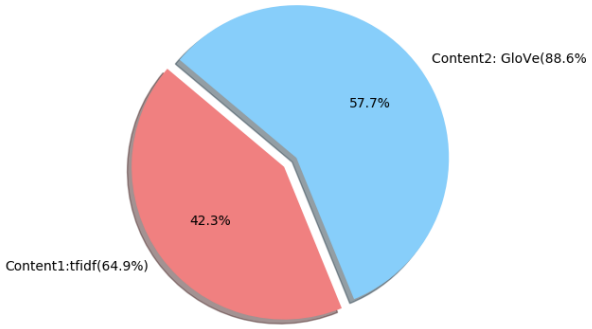


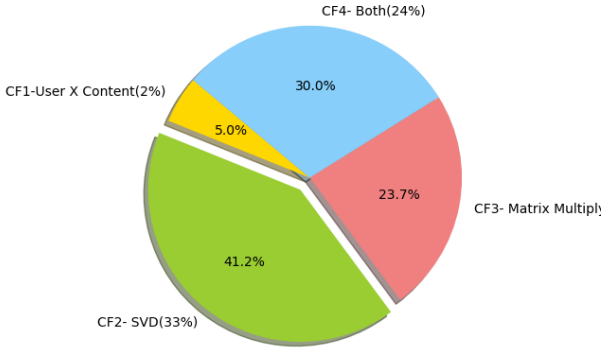


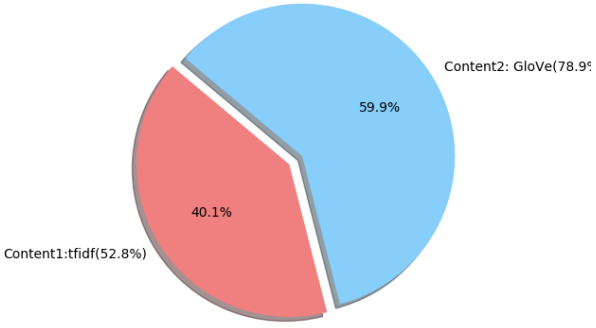





CF4- Both(36%)
29.3%
CF1-User X Content(10%)
8.3%
24.4%
CF3- Matrix Multiply
38.0%
CF2- SVD(46%)


Content2: GloVe(88.6%
57.7%
42.3%
Content1:tfidf(64.9%)


CF4- Both(24%)
CF1-User X Content(2%)
30.0%
5.0%
23.7%
CF3- Matrix Multiply
41.2%
CF2- SVD(33%)


Content2: GloVe(78.9
59.9%
40.1%
Content1:tfidf(52.8%)