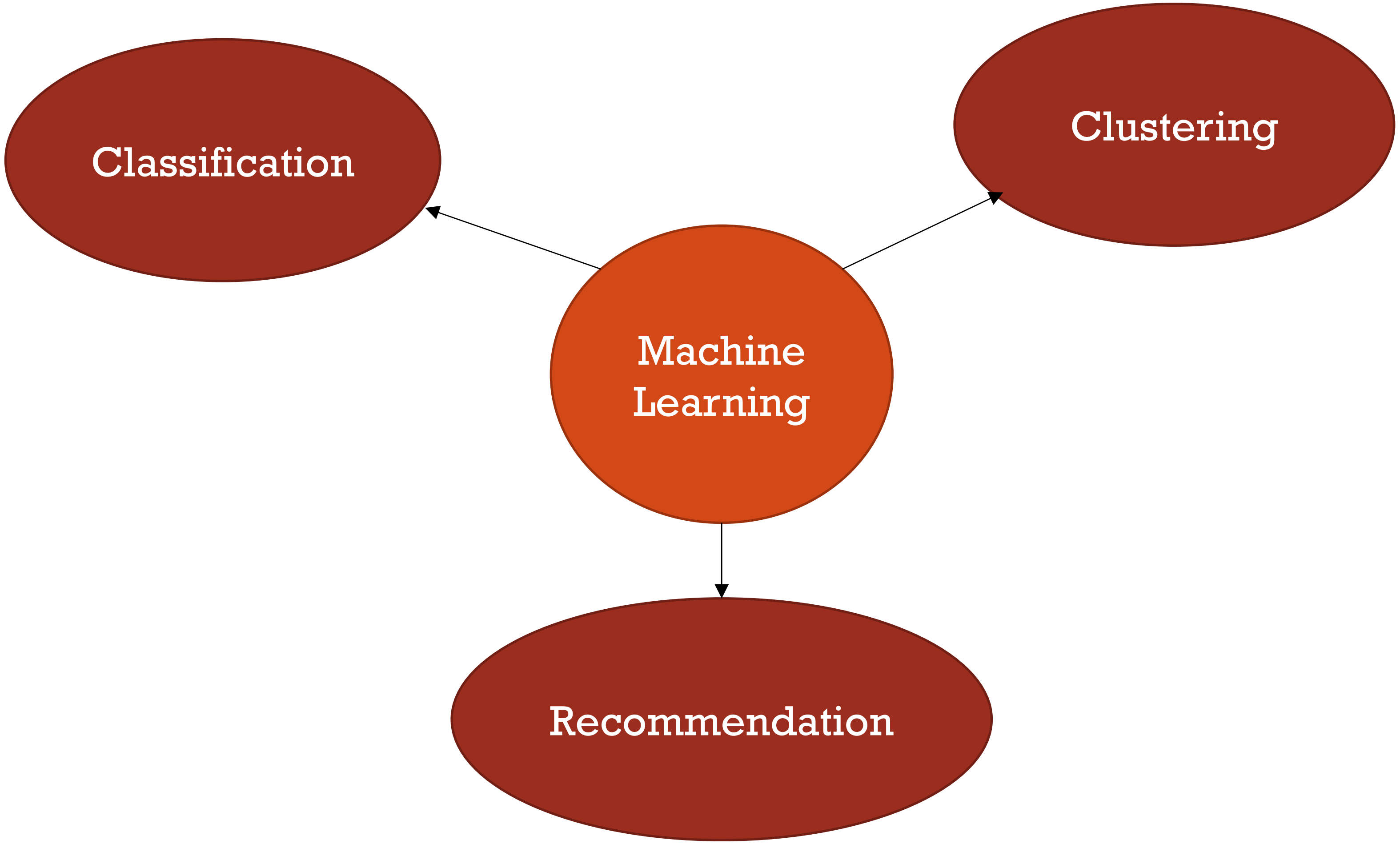
RECOMMENDER SYSTEMS

AN INTRODUCTION

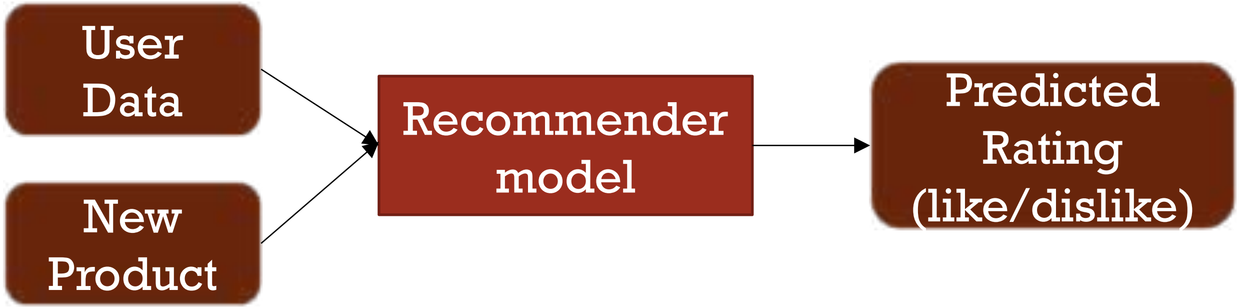
-By Tehreem

THREE COMMON CATEGORIES OF TECHNIQUES



WHAT IS A RECOMMENDER?

- Information filtering system - personalize information coming based on interests, relevance, etc.



- Input: User's history, item, other useful features (such as other similar users)
  - Implicit feedback: Captured automatically. E.g.: clicks a user makes, amount of time a user stays on a page, buying a product, etc.
  - Explicit feedback: Given consciously by the user, thus explicitly given by the user. E.g. ratings, reviews, feedback forms, etc.
- Output: Predicted rating, preference

WHERE IS IT USED?

- Almost everywhere…
  - E-commerce (e.g.: cross selling)
  - Social media (suggested friend requests, news feeds, pages you might like)
  - Web portals
  - Music recommendations (e.g.: Last.fm, Spotify)
  - Personalization
  - News/Media

OUR USE-CASE

- Recommend books to the user which they should read next or books they might like reading.

Get the ratings from user

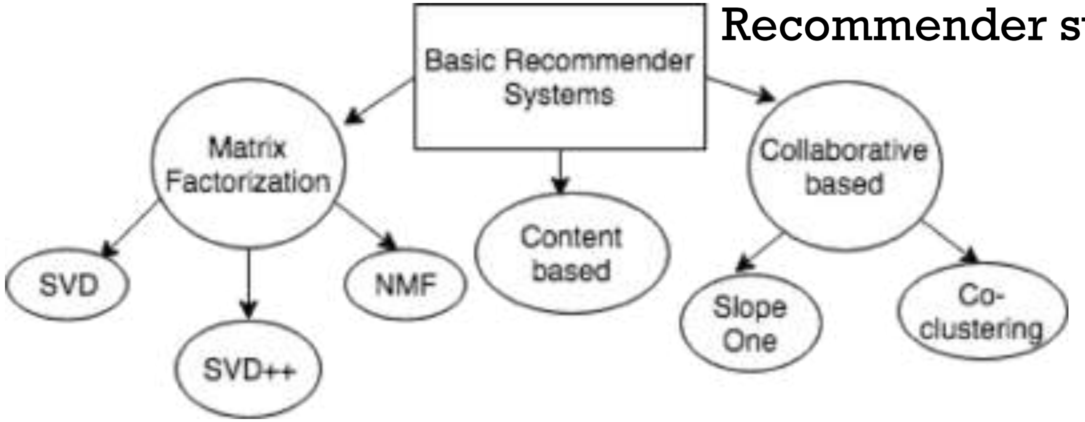
Give recommendations

STEPS WE FOLLOWED

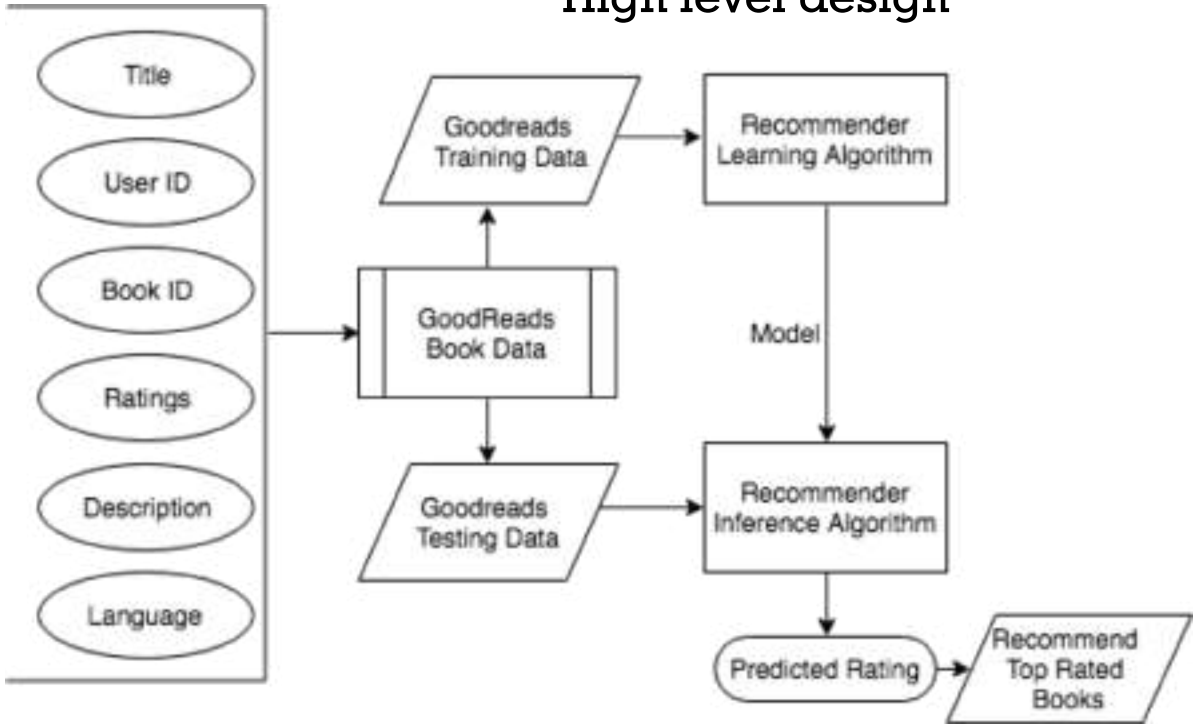
- Data Collection
- Scraping additional information for the data (ie books)
- Feature selection
- Divide the data set into 2, preferably 3 sets: Training Data, Cross Validation Data and Testing Data. In our case, we used 70% for Training data and 30% for Testing.
- Using training data, train the models. We used Graphlab and Surprise library.
- Cross validation to check if the model is generalized enough. In our case, we have not separated cross validation and test data.
- Test the model
- Check how the model behaved on the test data using evaluation metrics. We have used RMSE (Root Mean Square Error), MAE (Mean Absolute Error) and FCP (Fraction of Concordant Pairs) as our metrics.
- Once the evaluation metrics is available, we compared them and concluded why they behaved as they did.

APPROACH

Recommender system algorithms



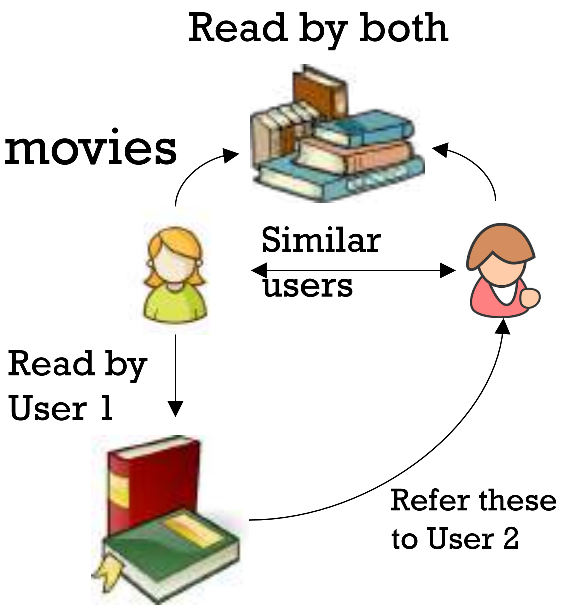
High level design



- Collaborative based
  - Slope Once
  - Co-clustering
- Content based
- Hybrid
- Matrix Factorization
  - Singular Value Decomposition (SVD)
  - SVD++
  - Non –ve matrix factorization

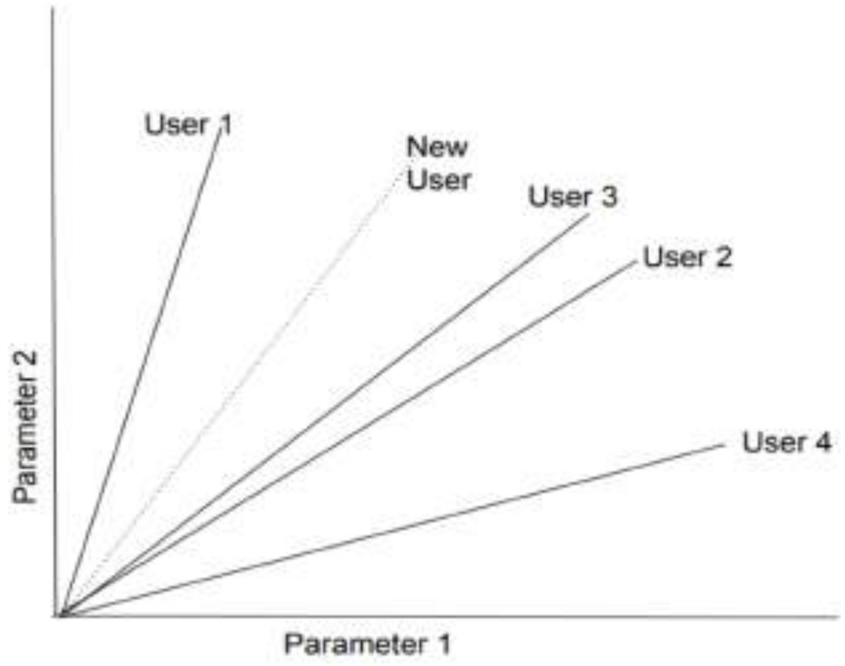
COLLABORATIVE BASED

- Collecting, analyzing information on users' behaviors, activities or preferences and predicting what users will like based on past history and their similarity to other users.
- Does not rely on machine analyzable content.
- Capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself.
- Clustering algorithms as k-NN for identifying similar users.
- Based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.



SHORT COMINGS OF COLLABORATIVE BASED

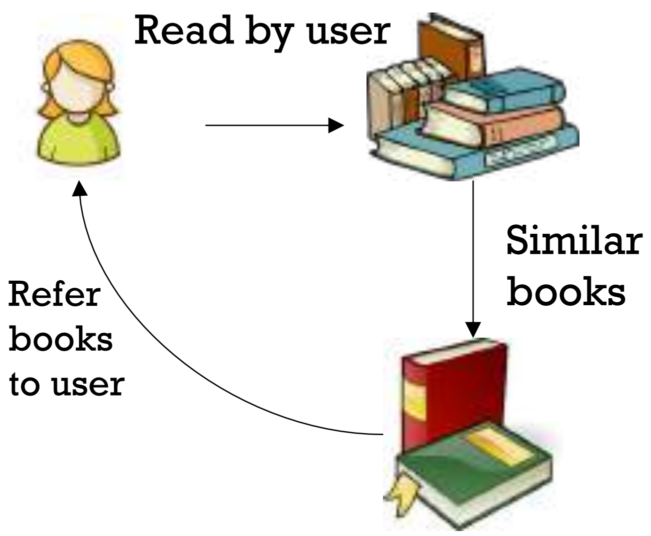
- Cold start: These systems often require a large amount of existing data on a user in order to make accurate recommendations.
- Scalability: Large amount of computation power is often necessary to calculate recommendations.
- Sparsity: The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings.
- Popularity Bias: Tends to recommend popular items.



- Early rater problem: Cannot provide recommendations for new items since there are no user ratings on which to base a prediction.

CONTENT BASED

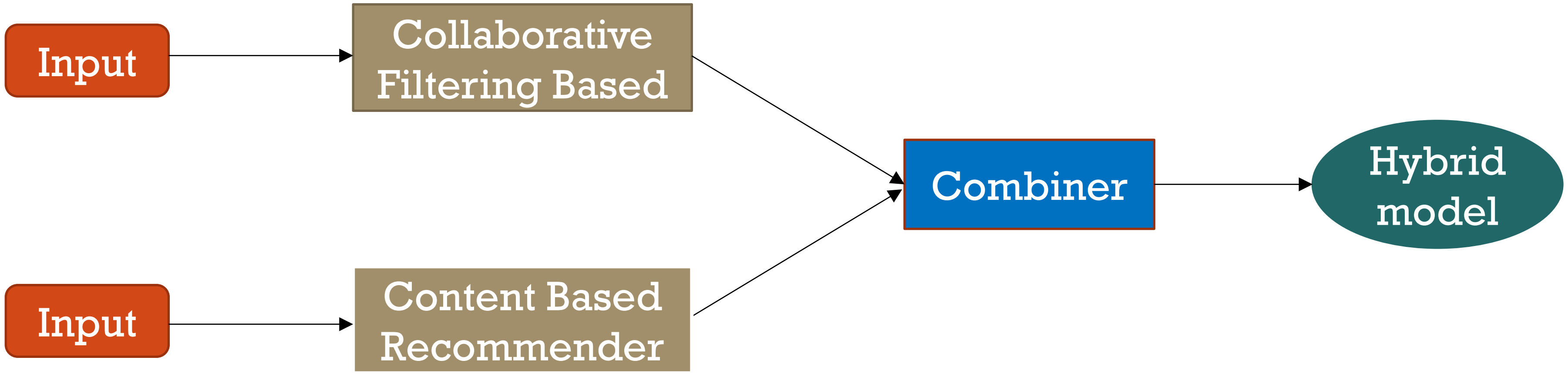
- Focuses on the products themselves and recommends other products that have similar attributes.
- Doesn't rely on other users to interact with the products before making a recommendation.
- Profile of the user's preferences (items liked/dislike in the past).
- To check if an item is similar to another item, item presentation algorithm is used. Eg: TF-IDF 'Term Frequency' on one side and 'Inverse document Frequency' on the other side.



SHORT COMING OF CONTENT BASED

- System is limited to recommending content of the same type as the user is already using, i.e. Over-specialization.
- For example, recommending news articles based on browsing of news is useful, but would be much more difficult when music, videos, products, discussions etc. from different services need to be recommended based on news browsing.
- Subjective domain problem: I.e. difficulty in distinguishing between subjective information such as points of views and humor.
- Content description is not always easy, e.g. music or videos.

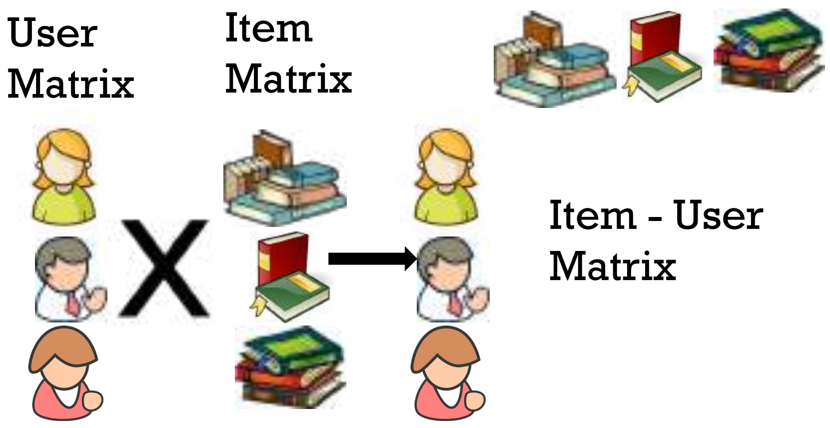
HYBRID APPROACH



- Combining collaborative filtering and content-based filtering to give a more effective model.
- Approaches:
  - Content-based and collaborative-based predictions separately and then combining them.
  - Adding content-based capabilities to a collaborative-based approach (and vice versa).
  - Unifying the approaches into one model.

MATRIX BASED

- Based on "factorizing a matrix".
- Find out two (or more) matrices such that when multiplied them you will get back the original matrix.
- Useful in discovering latent features underlying the interactions between two different kinds of entities.
- Group of users and a set of items. Given that each users have rated some items in the system, we would like to predict how the users would rate the items that they have not yet rated, such that we can make recommendations to the users.



EXAMPLE

- Calculating a user's rating: Assuming user ratings are reflection of how much a book appeals to the **user's unique set of interests.**
- M: Model how much a book appeals to every possible interest.
- U: Model's the user's interest.
- User's ratings - Measure how well the user's interests match the book's attributes. (U X M).
- U and M are called Latent vectors.
- Note: Latent means not directly observable. The common use of the term in PCA and Factor Analysis is to reduce dimension of a large number of directly observable features into a smaller set of indirectly observable features.

U X M = User Rating

$$\underbrace{\begin{array}{ccccc} \begin{bmatrix} 5 & -2 & 1 & -5 & 5 \end{bmatrix} \\ \text{ACTION} \quad \text{DRAMA} \quad \text{THRILLER} \quad \text{COMEDY} \quad \text{HORROR} \end{array}}_{\text{user1}} \times \begin{array}{cc} \text{book1} & \text{book2} \\ \multicolumn{2}{c}{\begin{bmatrix} 5 & -5 \\ -2 & 5 \\ 0 & 1 \\ -5 & 4 \\ 4 & -5 \end{bmatrix}} \end{array} = \begin{bmatrix} 74 & -79 \end{bmatrix}$$

SHORT COMING OF MATRIX FACTORIZATION

- Huge computation and memory required as it is based on matrix multiplication.
- Highly dependent on the size of latent factors (i.e. User vector and item vector).
- Dense metrics give more information than sparse, however the output generally is a sparse matrix.

Books Users	1	2	3	4	5	6
1	4		1	3		5
2		4			3	
3	3			3		
4			2			5
5	4	3		2		

Sparse Matrix

Books Users	1	2	3	4	5	6
1	4	3	1	3	2	5
2	3	4	3	2	3	4
3	3	3	3	3	2	3
4	4	4	2	4	1	5
5	4	3	3	2	1	4

Dense Matrix

SVD (SINGULAR VALUE DECOMPOSITION)

- Decompose original and very sparse matrix into two low-rank matrices that represent user factors and item factors.
- Done by using an iterative approach to minimize the loss function.
- Is a method of decomposing a matrix into three other matrices: $A=USV^T$.
  - A is an $m \times n$ matrix
  - U is an $m \times n$ *orthogonal* matrix
  - S is an $n \times n$ *diagonal matrix*
  - V is an $n \times n$ orthogonal matrix
- Used to reduce the number of features of a data set by reducing space dimensions from N to K where K < N.
- Is a type of matrix factorization technique.

SVD VS PLAIN MF

- For SVD we have: $A=USV^T$.
- It comes with stronger guarantees than Matrix Factorization's:
  - S is a diagonal matrix having the singular values of A on its diagonal. A common convention is to list the singular values in S in a descending order.
  - U and V are orthonormal matrices (their columns are orthogonal and their norm equals 1)
  - SVD solution is unique

SVD++

- Takes care of "implicit" ratings, i.e. user rating an item is in itself an indication of preference.

  SVD→ $min_{p,q,b} \sum_{u,i} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$

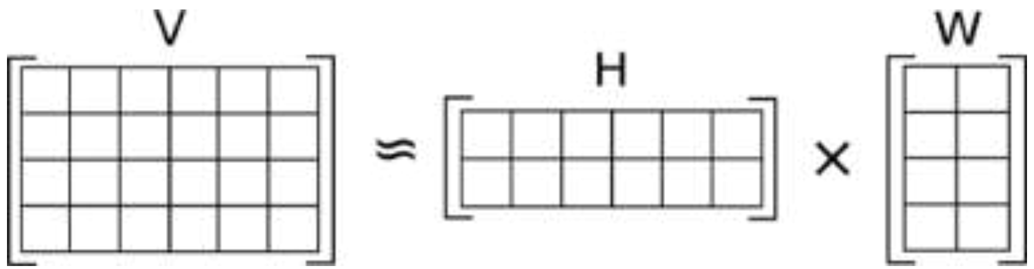
  SVD++→ $min_{p,q,b} \sum_{u,i} (r_{ui} - \mu - b_u - b_i - q_i^T (p_u + |N(u)|^{-1/2} \sum_{j \epsilon N(u)} y_i))^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2 + \sum_{j \epsilon N(u)} ||y_i||^2)$

- Difference is the addition of the factor. $|N(u)|^{-1/2} \sum_{j \epsilon N(u)} y_i$ and $\sum_{j \epsilon N(u)} ||y_i||^2$

- SVD++ is including the effect of the "implicit" information as opposed to SVD's p(u) that only includes the effect of the explicit one.

NMF (NON NEGATIVE MATRIX FACTORIZATION)

- Matrix V is factorized into (usually) two matrices W and H, with the property that all three matrices have no negative elements.
- This non-negativity makes the resulting matrices easier to inspect.
- Non-negativity is inherent to the data.
- A 'document-term matrix' is constructed with the weights of various terms from a set of documents. This matrix is factored into a term-feature and a feature-document matrix. The features are derived from the contents of the documents, and the feature-document matrix describes data-cluster of related documents.

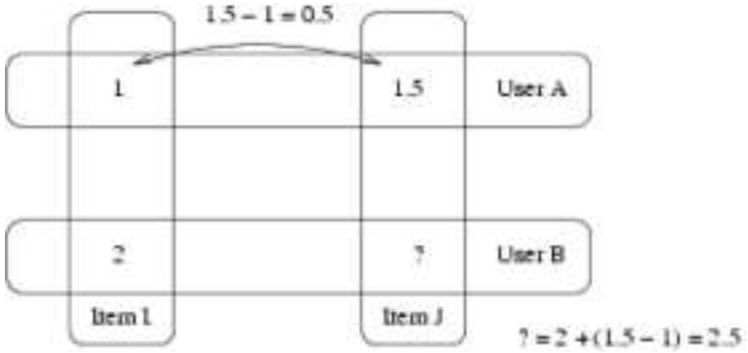


- Numeric attributes are normalized.
- Missing numerical values are replaced with the mean.
- Missing categorical values are replaced with the mode.

SLOPE ONE

- Drawbacks of Item-based collaborative filtering (internally typically using linear regression):
  - Overfitting
  - Slow performance
  - Complicated compared to Slope One.
- Hence, **Slope One**.
- Instead of using linear regression from one item's ratings to another item's ratings ($f(x) = ax + b$), it uses a single free parameter ($f(x) = x + b$). The free parameter is then simply the **average difference** between the two items' ratings.

EXAMPLE



- User A gave a 1 to Item I and a 1.5 to Item J.
- User B gave a 2 to Item I.
- The Slope One answer is to say 2.5 (1.5-1+2=2.5).

Customer	Item A	Item B	Item C
John	5	3	2
Mark	3	4	Didn't rate it
Lucy	Didn't rate it	2	5

- The average difference in ratings between item B and A is (2+(-1))/2=0.5. Hence, on average, item A is rated above item B by 0.5.
- The average difference between item C and A is 3. Hence, if we attempt to predict the rating of Lucy for item A using her rating for item B, we get 2+0.5 = 2.5. Similarly, if we try to predict her rating for item A using her rating of item C, we get 5+3=8.

CO-CLUSTERING

- Co-clustering/Biclustering: simultaneous clustering of the rows and columns of a matrix.
- Method of co-grouping two types of entities simultaneously, based on similarity of their **pairwise interactions**.
- I.e., both **similar users and similar documents** into, categories or interests, synchronously.
- Co-clustering is extremely useful when pairwise interactions signal is sparse.
- E.g: 2 users with **similar affinity to the same news categories**. Even if the two match perfectly in their preferences, it is **extremely unlikely** that the two will read **exactly the same articles**, due to the huge variety of news articles offered daily on the net.

  Moreover, the amount of articles **read by both users** is **very low.** In this case, clustering similar users based on the articles they read (or on the opposite side, clustering articles based on overlapping readers) seems pretty useless. And it is the case where co-clustering approach is useful, comparing to "regular" unimodal clustering.

RECOMMENDATION SYSTEM EXAMPLES

Business/Applications	Recommendation Interface	Recommendation Technology
Amazon.com		
Customers who Bought	Similar Item	Item to Item Correlation *Purchase data*
Eyes	Email	Attribute Based
Amazon.com Delivers	Email	Attribute Based
Book Matcher	Top N List	People to People Correlation *Likert*
Customer Comments	Average Rating Text Comments	Aggregated Rating *Likert* *Text*
CDNOW		
Album Advisor	Similar Item Top N List	Item to Item Correlation *Purchase data*
My CDNOW	Top N List	People to People Correlation *Likert*

CONT.

eBay		
Feedback Profile	Average Rating Text Comments	Aggregated Rating *Likert* *Text*
Levis		
Style Finder	Top N List	People to People Correlation *Likert*
Moviefinder.com		
Match Maker	Similar Item	Item to Item Correlation *Editor's choice*
We Predict	Top N List Ordered Search Results Average Rating	People to People Correlation *Aggregated Rating* *Likert*
Reel.com		
Movie Matches	Similar Item	Item to Item Correlation *Editor's choice*
Movie Map	Browsing	Attribute Based *Editor's choice*

From here

DATASET AND FEATURE ENGINEERING

```
book_id,user_id,rating
1,314,5
1,439,3
1,588,5
1,1169,4
1,1185,4
```

Ratings Table

Unnamed: 0	author	description0	Language	NumberofRatings	GoodReadAverageRating
0	J.K. Rowling	Harry Potter's life is miserable. His parents ...	English	4879247	4.44
1	Harper Lee	The unforgettable novel of a childhood in a ...	English	3368077	4.26
2	F. Scott Fitzgerald	, F. Scott Fitzgerald's third book, stands as ...	English	2807833	3.89

book_id_original	description1	title	book_id	goodreads_title
3	Harry Potter's life is miserable. His parents ...	Harry Potter and the Philosopher's Stone ...	2	Harry Potter and the Sorcerer's Stone (Harry ...
2657	Compassionate, dramatic, and deepl ...	To Kill a Mockingbird	4	To Kill a Mockingbird (To Kill a Mockingbird #1) ...
4671	, F. Scott Fitzgerald's third book, stands as ...	The Great Gatsby	5	The Great Gatsby

Scraped Data (using scrapy)

```
book_id,goodreads_title,goodreads_book_id,goodreads_work_id
1,"The Hunger Games (The Hunger Games, #1)",2767052,2792775
2,"Harry Potter and the Sorcerer's Stone (Harry Potter, #1)",3,4640799
3,"Twilight (Twilight, #1)",41865,3212258
4,To Kill a Mockingbird (To Kill a Mockingbird #1),2657,3275794
5,The Great Gatsby,4671,245494
```

Books Data

Feature engineering: manually designing what the input x's should be.

DEMO


GoodReads Books Recommender System
Please rate as many books you like
DARTH VADER
THE CASE FOR FAITH
LEE STROBEL
SARA PARETSKY
INDEMNITY ONLY
UGLY LOVE
COLLEEN HOOVER
Darth Vader and Son
1 2 3 4 5
1 2 3 4 5
Indemnity Only
1 2 3 4 5
Ugly Love
1 2 3 4 5
MATTHEW

EVALUATION METRICS

- RMSE: Root Mean Squared Error :

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}.$$

- MAE: Mean Absolute Error:

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

- FCP: Fraction of Concordant pairs: A pair is concordant if the subject ranked higher on X also ranks higher on Y.

IMPLEMENTATION OF MATRIX FACTORIZATION

```
Schema
------
User ID                          : user_id
Item ID                          : book_id
Target                           : rating
Additional observation features  : 0
User side features               : []
Item side features               : []

Statistics
----------
Number of observations           : 3346828
Number of users                  : 53424
Number of items                  : 10000

Training summary
----------------
Training time                    : 88.4635
```

```
Model Parameters
----------------
Model class                      : RankingFactorizationRecommender
num_factors                      : 32
binary_target                    : 0
side_data_factorization          : 1
solver                           : auto
nmf                              : 0
max_iterations                   : 25

Regularization Settings
-----------------------
regularization                   : 0.0
regularization_type              : normal
linear_regularization            : 0.0
ranking_regularization           : 0.25
unobserved_rating_value          : -1.79769313486e+308
num_sampled_negative_examples    : 4
ials_confidence_scaling_type     : auto
ials_confidence_scaling_factor   : 1
```

```
Final objective value: 0.700294

Final training RMSE: 0.613973
```

1.RMSE for each book

```
{'rmse_by_item': Columns:
        book_id int
        count   int
        rmse    float

Rows: 10000

Data:
+---------+-------+----------------+
| book_id | count |      rmse      |
+---------+-------+----------------+
|   7899  |   31  | 1.02627624187  |
|   5288  |   57  | 1.09901040599  |
|   3143  |   63  | 1.08130707845  |
|   6769  |   32  | 1.32712029334  |
|   5604  |   37  | 0.541691928137 |
|   2779  |   58  | 0.79439914799  |
|   8455  |   26  | 1.49920750411  |
|   118   |  873  | 0.937006143214 |
|   3988  |   68  | 1.00087314943  |
|   5783  |   21  | 1.28743827249  |
+---------+-------+----------------+
[10000 rows x 3 columns]
Note: Only the head of the SFrame is printed.
```

2. RMSE for each User

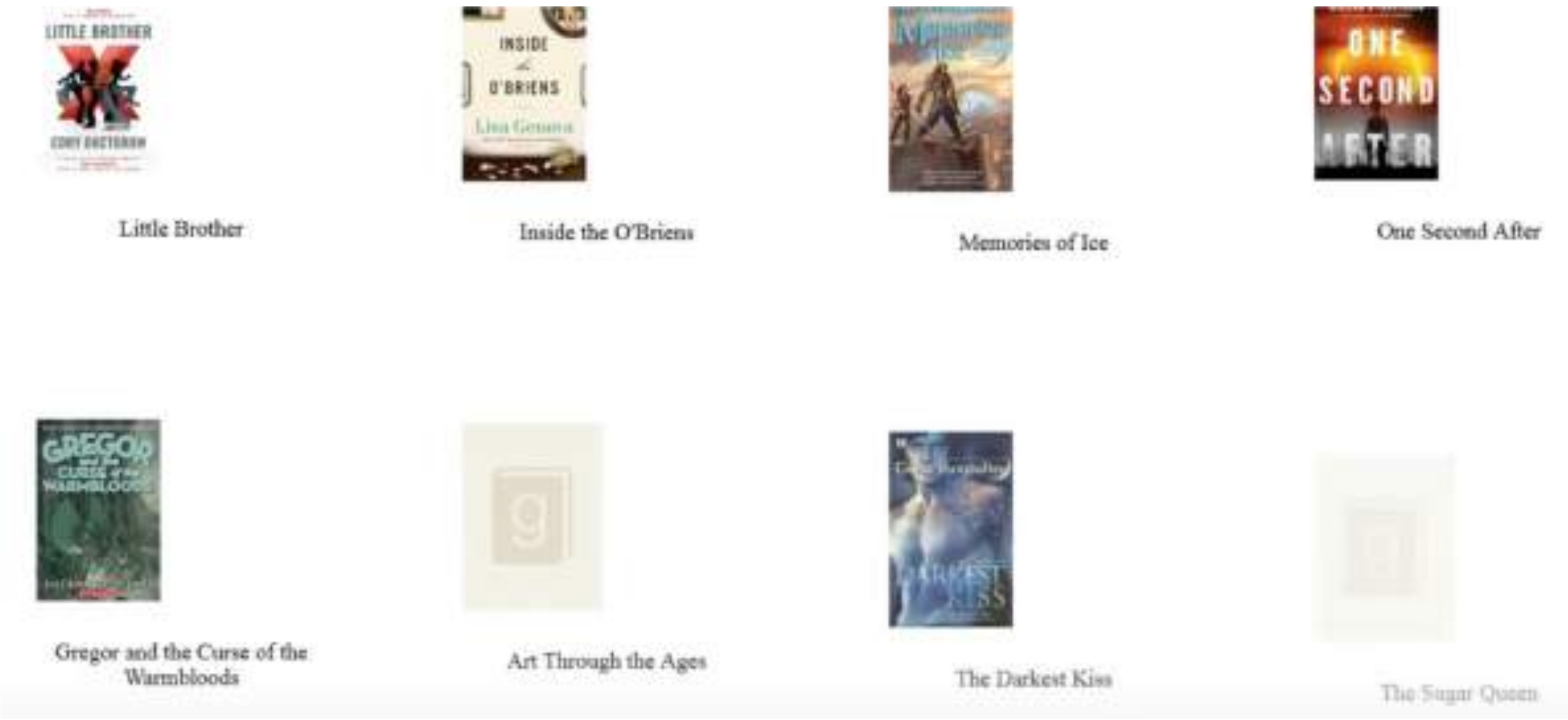
```
'rmse_by_user': Columns:
        user_id int
        count   int
        rmse    float

Rows: 53418

Data:
+---------+-------+----------------+
| user_id | count |      rmse      |
+---------+-------+----------------+
|  21855  |   19  | 0.979131781233 |
|   7899  |   10  | 0.961549571754 |
|  25263  |   12  | 1.38863605735  |
|  30021  |   7   |  1.1473691638  |
|  43116  |   18  | 1.39985775277  |
|  27112  |   20  | 0.96780091402  |
|  26319  |   12  | 0.816475172189 |
|  26439  |   10  | 0.624185622184 |
|   5288  |   15  | 0.787275412956 |
|  19584  |   13  | 1.07429253237  |
+---------+-------+----------------+
[53418 rows x 3 columns]
Note: Only the head of the SFrame is printed.
```

```
'rmse_overall': 1.06426161111 2098}
```

Recommended books for the user



Focus User

	user_id	_num_items
	35637	49

Top Recommended Items

Score	book_id	rank	_num_users
5.08	4220	1	183
5.01	1624	2	378
4.80	638	3	1085
4.80	424	4	1623
4.77	168	5	3375
4.71	2274	6	399
4.69	239	7	2077
4.68	1055	8	711
4.67	2259	9	405

Recommendation for the Picked Focus User

IMPLEMENTATION OF CONTENT BASED

```
Column                   Type   Interpretation  Transforms                              Output Type
-----------------------  -----  --------------  --------------------------------------  -----------
author                   str    categorical     None                                    str
description0             str    long_text       2-Word NGram Counts -> TFIDF            dict
Language                 str    categorical     None                                    str
NumberofRatings          int    numerical       None                                    int
GoodReadAverageRating    float  numerical       None                                    float
book_id_original         int    numerical       None                                    int
description1             str    long_text       2-Word NGram Counts -> TFIDF            dict
title                    str    short_text      3-Character NGram Counts -> TFIDF       dict
goodreads_title          str    short_text      3-Character NGram Counts -> TFIDF       dict
goodreads_book_id        int    numerical       None                                    int
goodreads_work_id        int    numerical       None                                    int

Schema
------
User ID                         : user_id
Item ID                         : book_id
Target                          : rating
Additional observation features : 0
User side features              : []
Item side features              : ['author', 'description0', 'Language', 'NumberofRatings', 'GoodReadAverageRatin
g', 'book_id_original', 'description1', 'title', 'book_id', 'goodreads_title', 'goodreads_book_id', 'goodreads_work
_id']

Statistics
----------
Number of observations          : 3346028
Number of users                 : 53424
Number of items                 : 10000

Training summary
----------------
Training time                   : 3.0000

Model Parameters
----------------
Model class                     : ItemContentRecommender
threshold                       : 0.0001
similarity_type                 : cosine
training_method                 : auto

Other Settings
--------------
degree_approximation_threshold   : 4096
sparse_density_estimation_sample_size : 4096
max_data_passes                  : 4096
target_memory_usage              : 8589934592
seed_item_set_size               : 50
nearest_neighbors_interaction_proportion_threshold : 0.05
max_item_neighborhood_size       : 64
```

1.RMSE for each book

book_id	count	rmse
7899	31	4.17782570653
5288	57	3.7231274309
3143	63	4.24108002892
6769	32	3.80554907543
5684	37	3.85015066581
2779	58	3.86992049737
8455	26	4.38529009654
118	873	3.94103561533
3988	68	4.28144006087
5783	21	3.71789391895

[10000 rows x 3 columns]
Note: Only the head of the SFrame is printed.

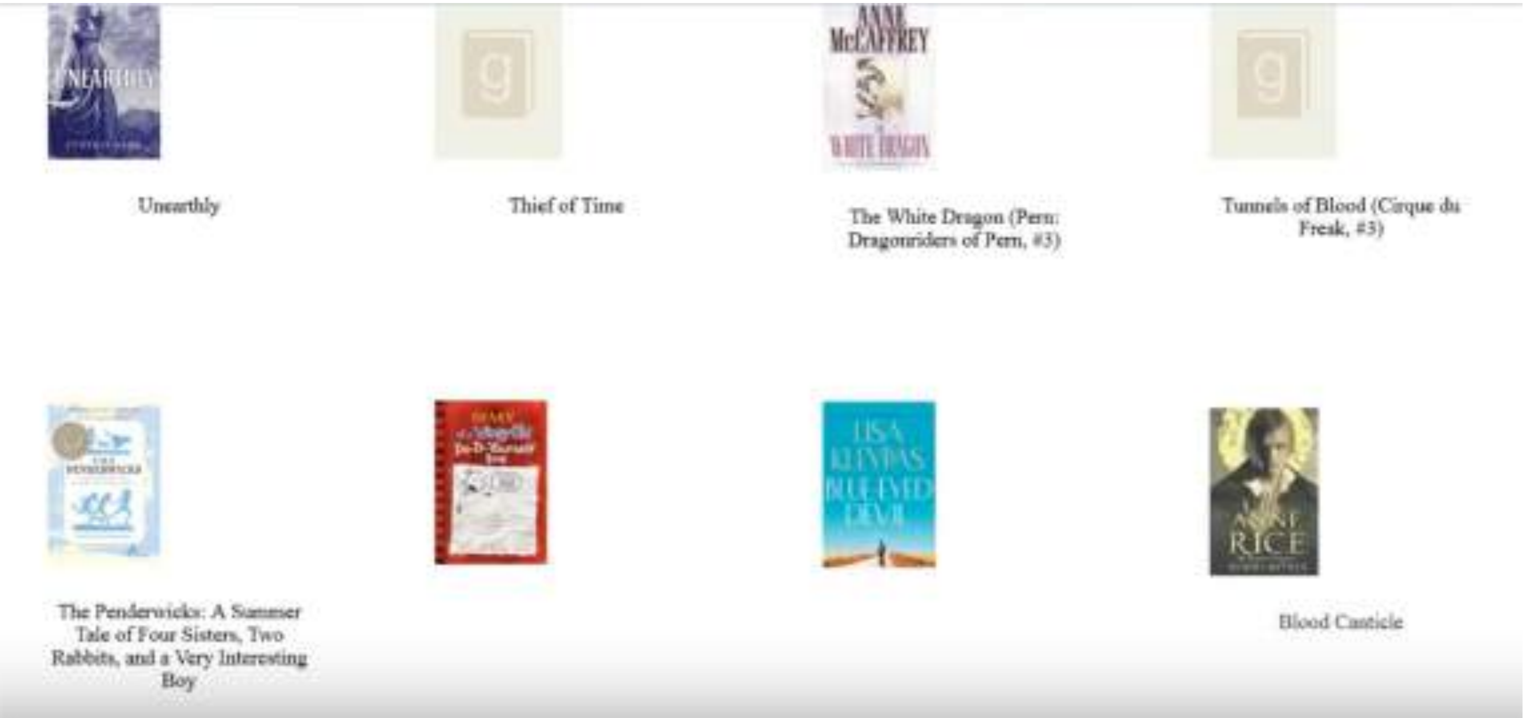
2. RMSE for each User

user_id	count	rmse
21855	19	4.36597786815
7899	10	4.65757999996
25263	12	4.44391585597
30621	7	4.01287800657
43116	18	3.88410814434
27112	20	4.14687098502
26319	12	4.53782266941
26439	16	4.63632111327
5288	15	2.7748605771
19584	13	4.12905667863

[53418 rows x 3 columns]
Note: Only the head of the SFrame is printed.

'rmse_overall': 3.967757942326331}

Recommended books for the user



Focus User

	user_id	_num_items	
	49944	61	

Top Recommended Items

Score	book_id	rank	_num_users
0.74	1089	1	786
0.65	1768	2	551
0.64	2452	3	462
0.64	4565	4	235
0.64	2677	5	408
0.63	2658	6	412
0.62	1343	7	643
0.61	1368	8	696
0.61	5099	9	223
0.61	2881	10	392

Recommendation for the Picked Focus User

IMPLEMENTATION OF COLLABORATIVE BASED

```
Class                           : ItemSimilarityRecommender

Schema
------
User ID                         : user_id
Item ID                         : book_id
Target                          : rating
Additional observation features : 0
User side features              : []
Item side features              : []

Statistics
----------
Number of observations          : 4183535
Number of users                 : 53424
Number of items                 : 10000

Training summary
----------------
Training time                   : 29.8580
```

```
Model Parameters
----------------
Model class                     : ItemSimilarityRecommender
threshold                       : 0.001
similarity_type                 : pearson
training_method                 : auto

Other Settings
--------------
degree_approximation_threshold   : 4096
sparse_density_estimation_sample_size : 4096
max_data_passes                 : 4096
target_memory_usage             : 8589934592
seed_item_set_size              : 50
nearest_neighbors_interaction_proportion_threshold : 0.05
max_item_neighborhood_size      : 64
```

1.RMSE for each book

book_id	count	rmse
7899	39	0.73831464459
5288	36	0.934596878462
3143	82	1.06583845906
6769	38	0.896043841403
5684	46	1.04445265004
2779	47	1.0449453836
8455	13	1.15672902445
118	849	0.981942447782
3988	77	0.882446200879
5783	28	1.17739724005

[10000 rows x 3 columns]
Note: Only the head of the SFrame is printed.

2. RMSE for each User

user_id	count	rmse
21055	15	0.974991106731
7899	11	1.09575745825
25263	12	1.02981920094
30621	15	0.774463730657
43116	20	0.852094430341
27112	17	0.911763489292
26319	15	0.843977213497
26439	14	1.10285776157
5288	14	1.15514905204
19584	17	0.723789839405

[53419 rows x 3 columns]
Note: Only the head of the SFrame is printed.

'rmse_overall': 0.94769676585914l2}

Recommended books for the user

Focus User

	user_id	_num_items	
	52152	66	

Top Recommended Items

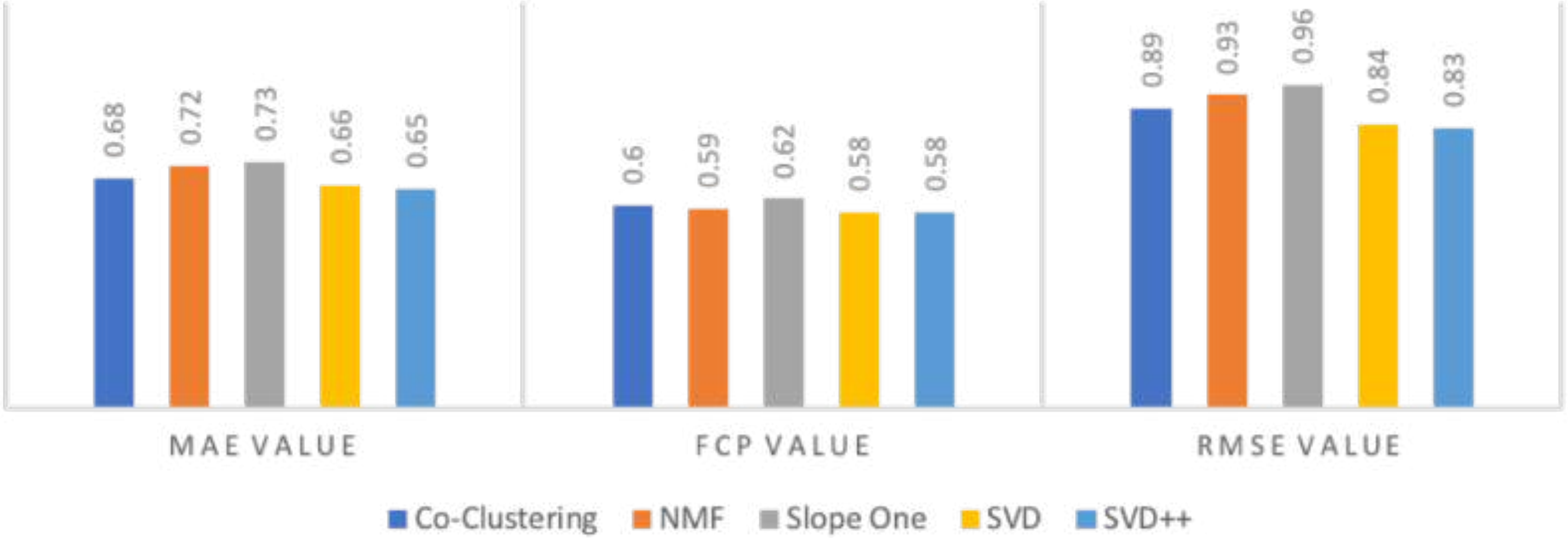
Score	book_id	rank	_num_users
4.84	3628	1	319
4.82	8978	2	110
4.82	7947	3	60
4.79	9566	4	107
4.79	6361	5	175
4.77	4483	6	247
4.76	6590	7	169
4.76	8569	8	84
4.76	6920	9	159
4.73	1308	10	665

Recommendation for the Picked Focus User

RMSE RESULTS OF OTHER MODELS

Model	RMSE
Matrix Factorization	1.064
Item Similarity Collaborative Based	0.947
Content Based	3.967
SVD	0.8594
SVD++	0.8355
NMF	0.9327
Slope One	0.9633
Co-clustering	0.8959

EVALUATION GRAPH



OBSERVATIONS

- Matrix Factorization and Collaborative (along with their variations) filtering based on user provided better results as lower the RMSE, better the result.
- SVD++ has least RMSE and MAE, thus is the best model for Goodreads Book recommendation (case specific).
- Content based recommender performed bad due to following reasons:
  - Not enough content to discriminate items precisely
  - Over-specialization
  - Not enough information to build solid profile for new user

OTHER MODELS AND FUTURE WORK

- Using Deep Learning along with collaborative filtering.
- Hybrid recommendations, using collaborative and content based together using a combiner.
- Using clustering techniques, recommended products go well together.
- Using Feature Weighted Linear Stacking for producing ensemble from a collection of heterogeneous models.
- **MAPLE** (**M**arketing **A**utomation **P**rogram and **L**earning **E**ngine) is a machine learning recommendation engine built to support Amazon's financial products. MAPLE's mission is to recommend the best payment product at the right time to Amazon's customers.

IDEAS

- Family customers based recommendations, either for products (showing stationaries at the start of schools), or promotions.
- Predefined loadable options (load recommendation) in Add Money Page.
- Ranking Recommendations using Customer Intent.
- Recommending Tickets in Amazon's Trouble Ticket Website using Unsupervised Learning.
- Recommending promotional messages according to their age and gender, i.e. demographics.
- Extract implicit negative ratings from purchase data. Purchasing something does not mean liking it, such as analysis of returned/cancelled products.
- Produce an indication of the price sensitivity of the customer for a given product, to offer each product at the price that maximizes the lifetime value of the customer to the site.

LASTLY

- Jeff Bezos "If I have 2 million customers on the Web, I should have 2 million stores on the Web."

- Wiki Link: Goodreads Recommender
- Other papers and posters: Recommendations at Amazon-AMLC2018