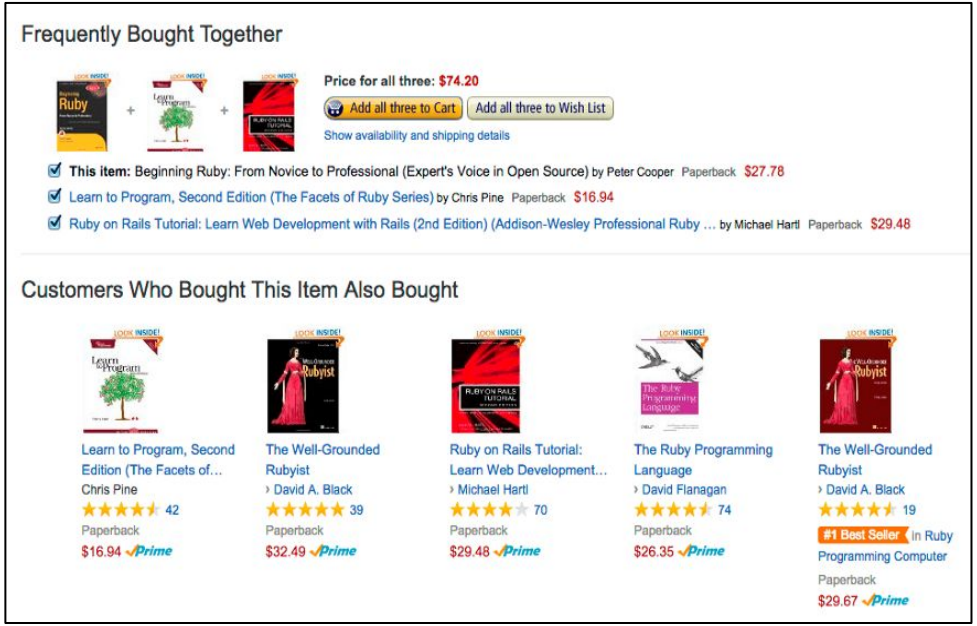
Recommendation System for Products & Services

Mansi Kathuria (01901012017)
Rashmi Kumari (05801012017)
Yashasvi Kapoor (06801012017)

Group ID - 12
Mentor - Prof. D.K Tayal

Problem Statement

Building a **generic** recommender system as a **filter** that seeks to **predict** and show the items that a user would like to purchase, to boost **customer experience** and enhance customer retention for online businesses and facilities.

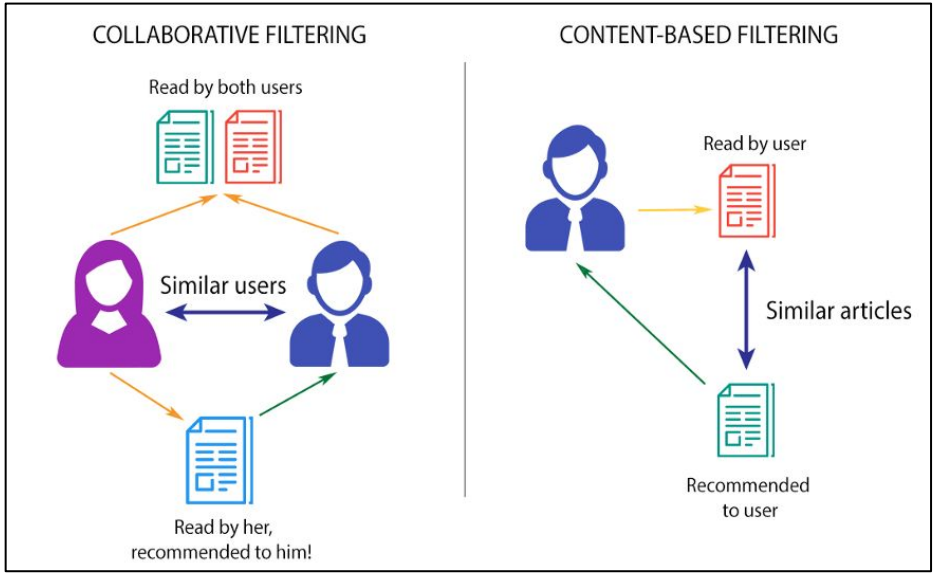


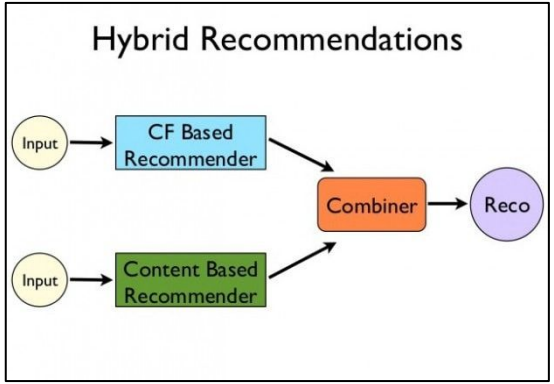
Recommendation System Types

➔ **Collaborative Filtering**
filtering method is usually based on collecting and analyzing information on **user's behaviors**, their activities or preferences and predicting the results.

➔ **Content Based Filtering**
filtering methods are based on the **description of an item** and a profile of the user's preferred choices.

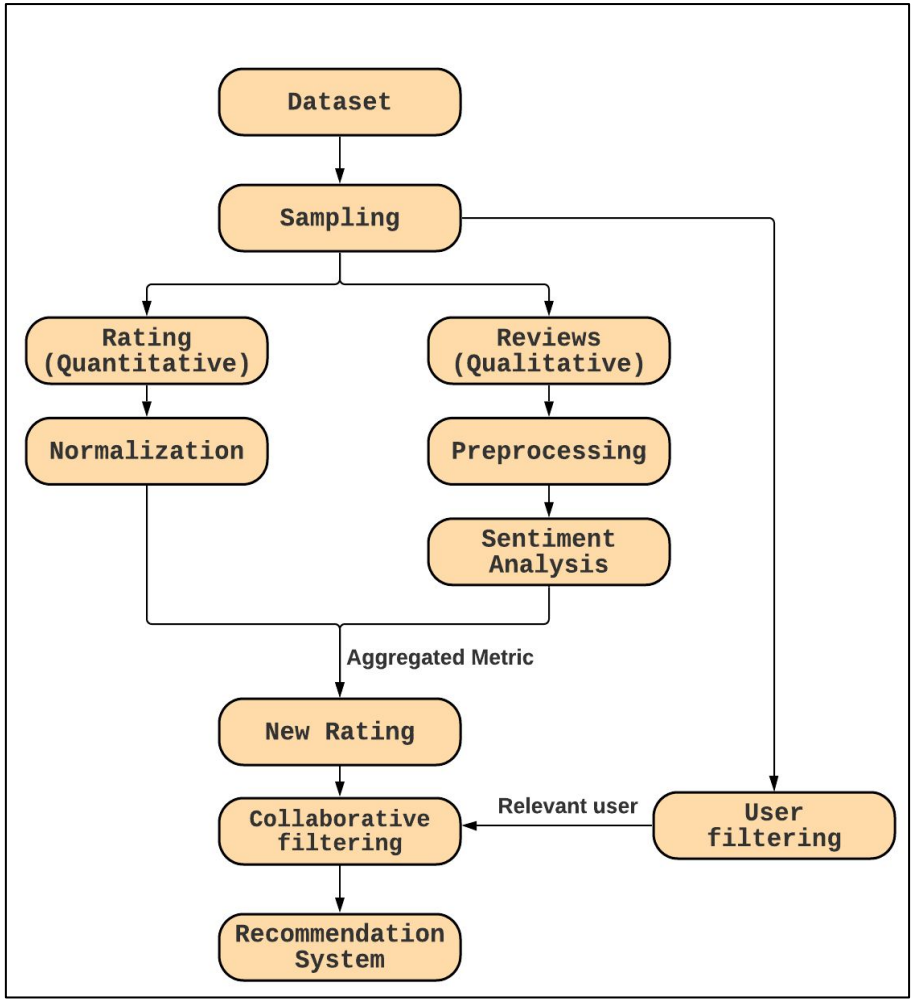
➔ **Hybrid Recommendation System**
combining collaborative and content-based recommendation can be more effective. Can be implemented by making content-based and collaborative-based predictions separately and then combining them.





Proposed Solution

1. Collaborative filtering uses similarity metrics of different users without considering the relevance of a user having availed that product in the past.
2. Similarly, content based filtering looks up similar products and generates recommendations.
3. Our solution proposes that using collaborative filtering will constitute one part of the recommendation algorithm.
4. Relevant users can be found based on their activity. This will act as a filter to get rid of irrelevant recommendations and produce genuine results.
5. The second part of the algorithm will identify a set of similar users out of the relevant users, the comment reviews and ratings provided by whom can be of interest to a user looking up for recommendations.



Implementation Procedure

1. Obtaining user ratings and review comments from external sources.
2. Applying sentiment analysis to find sentiment score for each product.
3. Combine sentiment score and ratings of products to find new rating of every product.
4. Apply collaborative filtering algorithms to generate recommendations.

1. Exploration of libraries for link analysis (networkx)
2. Representing users and product entities in graph data structure format
3. Applying centrality measures to understand user relevance
4. Identifying anomalies (users) irrelevant to product recommendations by page rank algorithm

Filtered list of product recommendations

Step 1- Sentiment Analysis

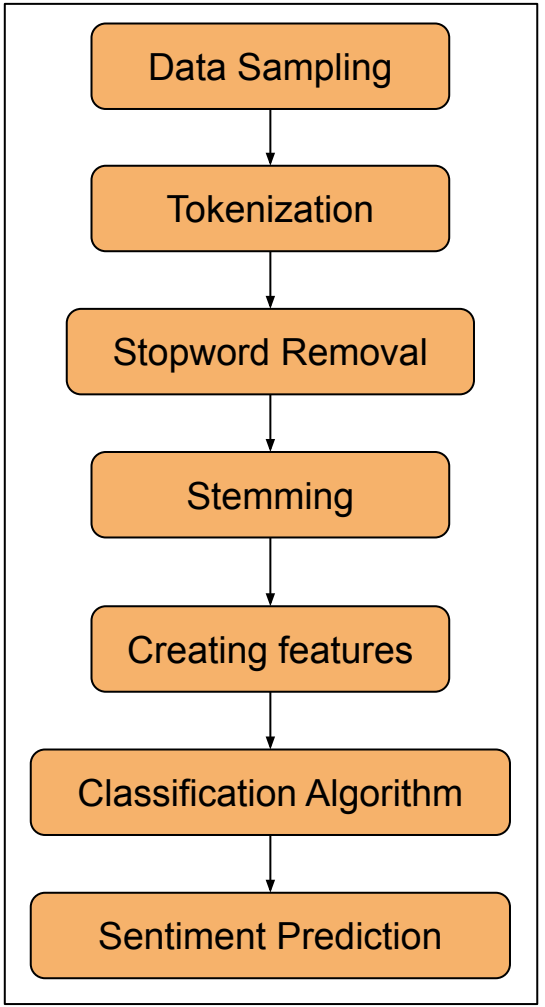
Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative our neutral.

➔ **Natural Language Processing**

NLTK library is used for Natural language processing. Steps involved in NLP are: Data Sampling, Stemming, Tokenization, Creating features (Unigram or Bigram),

➔ **Classification Algorithm used for Sentiment Analysis**

Naive Bayes - Multinomial and Bernoulli, Support Vector machines.



Sentiment Analysis Results...

We used twitter sentiment analysis dataset and Amazon Book Reviews dataset.

After comparing various classification algorithm we noticed that SVM with a combination of Unigrams and Bigrams gives best accuracy.

Therefore we will be using SVM in our recommendation system for sentiment analysis.

Twitter Data

S.No.	Classification Algorithm	Feature	Accuracy
1.	Multinomial Naive Bayes	Unigram	71.90%
2.	Multinomial Naive Bayes	Unigram+Bigram	68.59%
3.	Bernoulli Naive Bayes	Unigram	74.54%
4.	Bernoulli Naive Bayes	Unigram+Bigram	70.36%
5.	SVM	Unigram	86.86%
6.	SVM	Unigram+Bigram	94.75%

Amazon Books Dataset

S.No.	Classification Algorithm	Feature	Accuracy
1.	Multinomial Naive Bayes	Unigram	82.99%
2.	Multinomial Naive Bayes	Unigram+Bigram	78.96%
3.	Bernoulli Naive Bayes	Unigram	79.09%
4.	Bernoulli Naive Bayes	Unigram+Bigram	78.83%
5.	SVM	Unigram	80.98%
6.	SVM	Unigram+Bigram	84.13%

Collaborative Filtering

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user.

```
In [16]: bookmat = data.pivot_table(index="reviewerID", columns="asin", values="overall").fillna(0)
         bookmat.tail(n=10)
```

Out[16]:

reviewerID	006055584X	006055665X	006055973X	006056038X	006056167X	006056198X	006056251X	006056346X	006056458X
AZRYYS3PLALH0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZSFP4R60R2L3	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0
AZSV99SDJC242	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZT57V3XQBRAC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZVXGCT3CHIS5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZW2TLAQZEYDF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZWW1U604W0N	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZYIHKBD3FZBI	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZZ4GD20C58ND	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZZHHNN8RQ7YQ	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0

10 rows × 170 columns

Types of Collaborative filtering

- **Memory Based** - Find similar users based on cosine similarity or pearson correlation and weighted average of the ratings.

  Eg - item based, user based, KNN

- **Model Based** - Use ML to find user ratings of unrelated items.

  Eg - SVD, Neural Network, Matrix factorization

"We are leaving the age of information and entering the age of recommendation."

Memory Based Collaborative Filtering

Item-Item Collaborative Filtering: It takes an item, find users who liked that item, and find other items that those users or similar users also liked.

"Users who liked this item also liked …"

The steps we took in this technique are:

1. Specify the target user
2. Find similar items which have similar ratings with items target user rated.
3. Predict the ratings for similar items
4. If the predicted ratings are above the threshold, then recommend them to target user

Similarity Metrics:

- Cosine similarity
- Pearson correlation

```
In [17]: predictions = predict_books("60572965")
         predictions.head()
```

Out[17]:

	Correlation	num of ratings
asin		
60572965	1.000000	307
60572973	0.452506	164
006057299X	0.328458	135
60562536	-0.004056	120
60562498	-0.005295	128

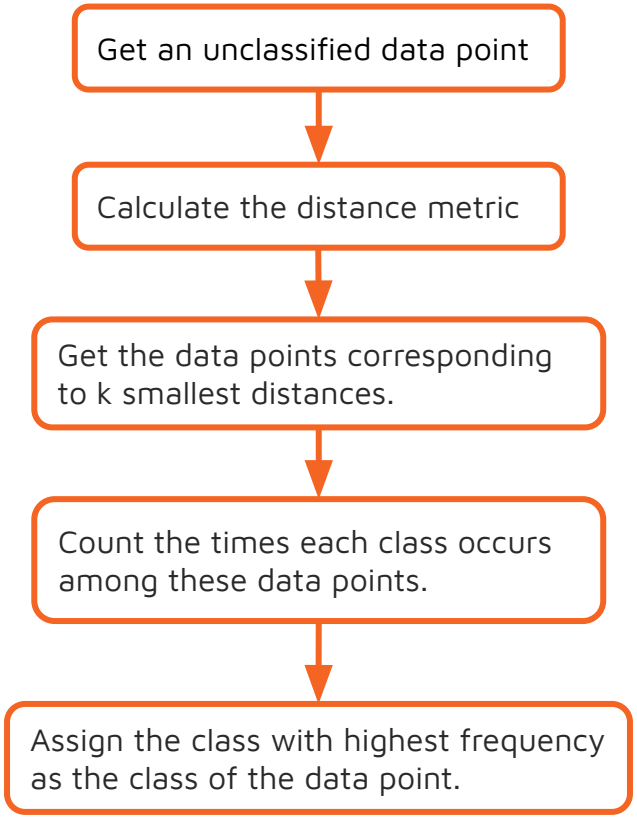
Memory Based Collaborative Filtering

K-Nearest Neighbors:

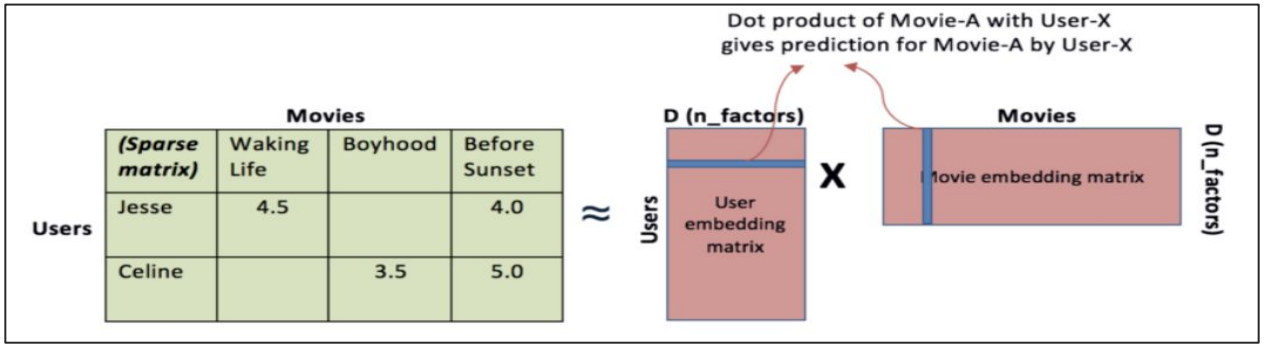
When a KNN makes a prediction, it will calculate the distance between the target item and every other item in its database. It then ranks its distances and returns the top k nearest neighbor item as the most similar item recommendations.

Distance metrics: Euclidean, Manhattan, Minkowski or Weighted

```
recommender('60572965', bookmatcsr, model_knn,6)

You have input book: 60572965
......

Searching for recommendations.....
......

Index       Book Id
----------------------------
3356    006056542X
4282      60568984
1973      60561785
2221      60562498
4660           NaN
2599      60562560
Name: asin, dtype: object
----------------------------
```



Matrix Factorization - Latent factor model



Embeddings:

- Latent factor model is an approach that tries to explain the ratings by characterizing both items and users on 20 to 100 factors call embeddings inferred from the ratings patterns. Such factors represent human created categories of items
- For movies, the discovered factors might measure obvious dimensions such as comedy versus drama, amount of action, or orientation to children; less well-defined dimensions such as depth of character development or quirkiness; or completely uninterpretable dimensions.
- For users, each factor measures how much the user likes movies that score high on the corresponding factor

Steps followed:

1. Specify target user
2. Decompose user-item rating matrix to user and item embeddings
3. New ratings for unrated movies are predicated through multiplication of above 2 matrices.
4. Unrated items are sorted on the basis of new ratings.
5. Top items are recommended

Matrix Factorization Models - Implementation

SVD

- The singular value decomposition is a method of decomposing a matrix into three other matrices as : $A = U\Sigma V^T$
- In case of collaborative filtering, A is the user-item rating matrix given as input to find relationship between users and items on the basis of k features.
- U represents the user matrix containing values for each user which tells about the extent to which user is inclined towards a feature
- Similarly, V^T represents the item matrix containing values for each item
- Σ is a diagonal matrix where values at diagonal represents the scale to which the features are important in deciding user ratings

```
print(all_user_predicted_ratings)
type(all_user_predicted_ratings)
```

```
[[ 6.44838854e+00  2.95045020e+00  1.62598149e+00 ... -4.40875253e-03
   7.97805823e-03  5.36655280e-02]
 [ 2.33427841e+00  1.15585625e-01 -1.01177576e-01 ...  6.23701750e-03
   3.03678476e-02 -2.80431016e-02]
 [ 3.27603425e-01 -2.73722127e-01 -1.43939143e-01 ...  3.08582703e-02
   6.64717634e-03  2.02154921e-03]
 ...
 [ 3.09693646e+00 -4.04433484e-02  5.42197828e-01 ... -2.03503842e-03
  -4.12365264e-03 -1.31629574e-02]
 [ 9.54203280e-01  5.99508248e-01  4.84349206e-01 ...  1.18627050e-02
   1.97773478e-02  2.09692603e-03]
 [ 1.41502259e+00  2.85803399e+00  1.78005861e+00 ... -1.73804392e-02
   1.16284009e-02  4.63633730e-02]]
```

```
def recommend_movies(predictions, user_index, movies, org_matrix, num_recommendations=5):
    sorted_movies = (-predictions[user_index]).argsort()
    boolArr = (org_matrix[user_index] == 0)
    unknown_sorted_movies = [x for x in sorted_movies if boolArr[x]]
    recommendations = [movies[x] for x in unknown_sorted_movies][:num_recommendations]
    return recommendations
```

```
top_7_movies = recommend_movies(all_user_predicted_ratings, 7, movies.values, R, 7)
top_7_movies
```

```
[array([161, 'Top Gun (1986)'], dtype=object),
 array([226, 'Die Hard 2 (1990)'], dtype=object),
 array([265, 'Hunt for Red October, The (1990)'], dtype=object),
 array([230, 'Star Trek IV: The Voyage Home (1986)'], dtype=object),
 array([62, 'Stargate (1994)'], dtype=object),
 array([4, 'Get Shorty (1995)'], dtype=object),
 array([68, 'Crow, The (1994)'], dtype=object)]
```

– ALS

Following are the steps used in ALS:

1. Initialize the user latent vectors, U , and item latent vectors V randomly
2. Calculate new user-item ratings and minimize the square of the difference between actual ratings and predicted ratings.
3. To avoid overfitting weighted regularization terms are also added to above expression and then function is minimized

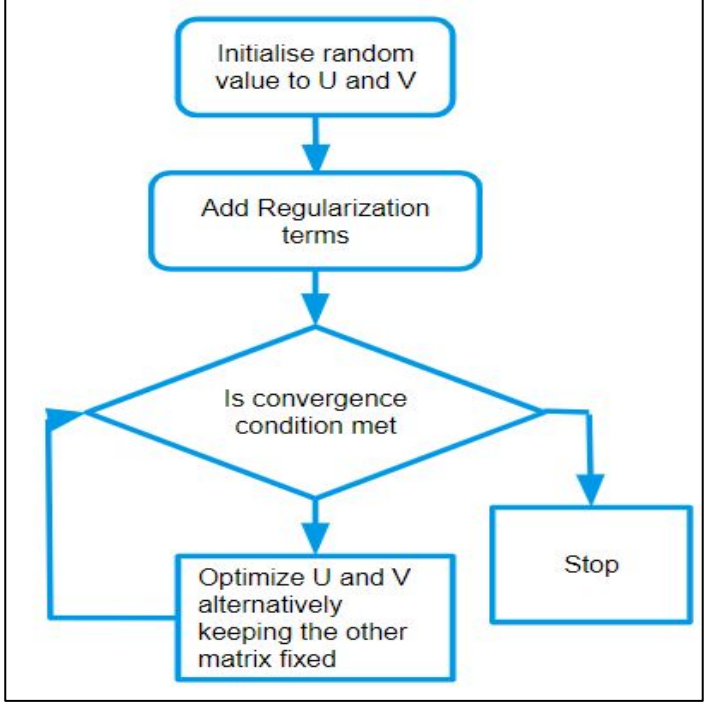
$$\min \sum_{u,i \in K} (r_{ui} - p_u q_i^T)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2),$$

4. Since optimizing U and V simultaneously is hard to solve, U or V is kept constant and the other matrix is optimized by taking gradient and vice versa. Below expressions are used to find new U and V after every iteration:

$$\mathbf{x}_u = \mathbf{r}_u Y (Y^T Y + \lambda I)^{-1}$$

$$\mathbf{y}_i = \mathbf{r}_i X (X^T X + \lambda I)^{-1}$$

5. Step 4 is carried out back and forth until convergence



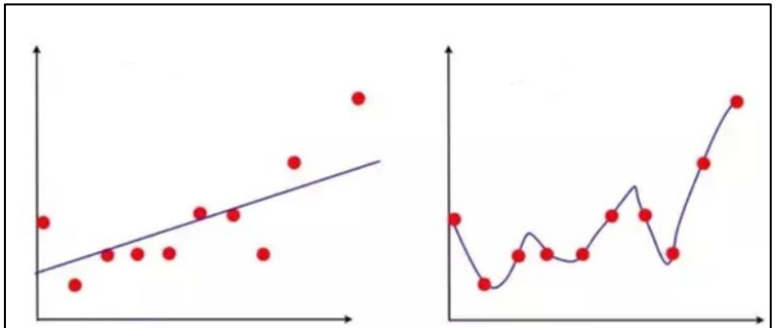


Figure 4: Underfit vs Overfit

ALS - Implementation

In [31]:
```
als=ALS(maxIter=5,
        regParam=0.09,
        rank=25,
        userCol="newUserId",
        itemCol="newBookId",
        ratingCol="rating",
        coldStartStrategy="drop",
        nonnegative=True)
model=als.fit(training)
```

In [33]:
```
evaluator=RegressionEvaluator(metricName="rmse",
                              labelCol="rating",
                              predictionCol="prediction")
predictions=model.transform(test)
rmse=evaluator.evaluate(predictions)
print(rmse)
predictions.show()
```

```
1.6502403716193452
+------+---------+---------+----------+
|rating|newUserId|newBookId|prediction|
+------+---------+---------+----------+
|     5|     3757|      148|  3.884962|
|     5|     5036|      148|  5.206557|
|     5|     3016|      148|  3.884962|
|     4|     3853|      148| 3.0492332|
|     5|      984|       31| 3.8800292|
|     5|     4144|       31| 1.8428248|
```

SVD vs ALS

Algo	RMSE
SVD	1.039
ALS	1.650

Model based vs Memory based collaborative filtering

Model based

- Latent factors / parameters are learnt by reduction of dimensionality using optimization algorithms like gradient descent
- Dimension reduction results in formation of dense matrices, which can generate prediction
- With matrix factorization, less-known movies can have rich latent representations as much as popular movies have, which improves recommender's ability to recommend less-known movies
- Personalized recommendation is given according to the latent factors calculated for each user

Memory based

- No parameters are learnt and closest user or items are calculated only by using Cosine similarity or Pearson correlation coefficients
- Performance reduces when data is sparse. Hence, it's non scalable
- Memory based system recommends the movies with the most interactions
- This system recommends the movies without any personalization

Link Analysis

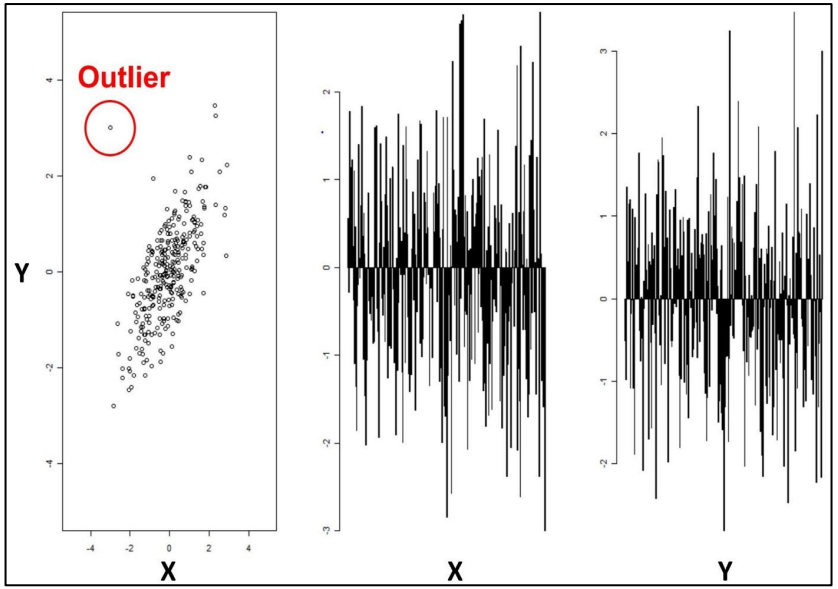
What is link analysis ? the process of investigating social structures by the use of networks and graphs.

Application domains -

marketing influence maximization, recommendation systems, fraud user detection

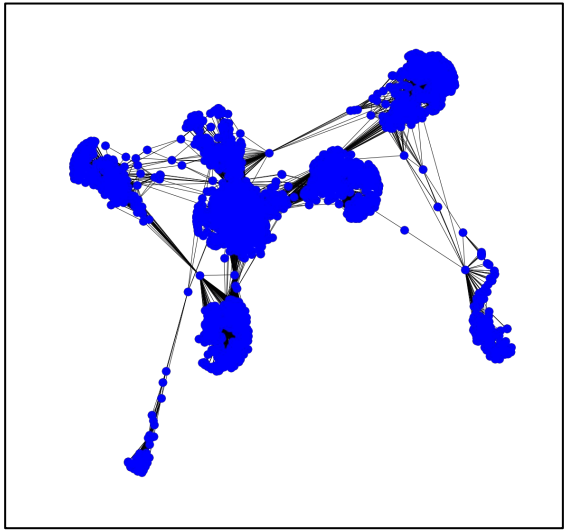
How does Link Analysis help with our use case ?

- identifying anomaly nodes of a network (irrelevant users) who can be turned off to refine produced set of recommendations
- predicting new links on already built social network to interrelate a new product with older ones having similar features



How Abouts of Link Analysis

- We make use of a python library, called networkx, for link analysis processing mechanisms and running algorithms against user-product data.
- The page rank algorithm is run against the graph generated from user-product data to identify sets of users who are core users of the product/similar products and viable enough to produce valuable recommendations.
- Other users, who are outliers to this cluster are irrelevant users whose product recommendations are estimated to not hold much value and such recommendations are identified.
- The use case can be further extended by incorporating link-prediction algorithms for data entering the system in real time
- We are currently exploring other domains of link prediction in graph based social networks, product irrelevance using product specifics datasets.

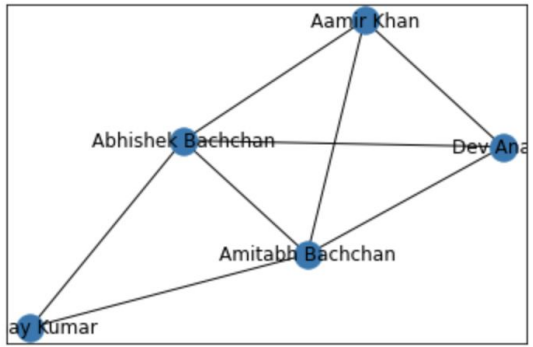


Concepts and Basic Methods for a small network

```
In [1]: import networkx as nx
```

```
In [3]: G = nx.Graph()
        G.add_edge('Amitabh Bachchan','Abhishek Bachchan')
        G.add_edge('Amitabh Bachchan','Aamir Khan')
        G.add_edge('Amitabh Bachchan','Akshay Kumar')
        G.add_edge('Amitabh Bachchan','Dev Anand')
        G.add_edge('Abhishek Bachchan','Aamir Khan')
        G.add_edge('Abhishek Bachchan','Akshay Kumar')
        G.add_edge('Abhishek Bachchan','Dev Anand')
        G.add_edge('Dev Anand','Aamir Khan')
```

```
In [4]: nx.draw_networkx(G)
```




```
In [7]: nx.degree(G, 'Dev Anand')
Out[7]: 3

In [10]: nx.average_clustering(G)
Out[10]: 0.8666666666666666

In [11]: nx.shortest_path_length(G, 'Aamir Khan', 'Akshay Kumar')
Out[11]: 2

In [15]: nx.eigenvector_centrality(G)
Out[15]: {'Amitabh Bachchan': 0.5100364187624349,
          'Abhishek Bachchan': 0.5100364187624349,
          'Aamir Khan': 0.43904190094642953,
          'Akshay Kumar': 0.3069366734339046,
          'Dev Anand': 0.43904190094642953}
```

Centrality Measures

Degree Centrality : This is based on assumption that Important nodes have many connections.

Cdeg of node ‘a’ = Degree of node ‘a’ no. of nodes in the network — 1

Closeness Centrality : This is based on the assumption that important nodes are close to other nodes.

Closeness centrality of node ‘a’ = No of nodes in the network — 1 d(a,u)

Clustering Coefficient

Triadic closure : It is the tendency of the nodes (people) who share common connections in a social network to get connected.

Local Clustering Coefficient : Fraction of pair of Node’s friend that are friends with each other.

Global Clustering Coefficient : It measures the degree to which the nodes in the whole network tend to cluster or form triangles.

Extending Link Analysis for our Dataset (constructing large network)

Steps of Implementation :

1. construct undirected and multigraphs from dataset using the logic that an edge of relation will exist between two users if they avail the same product
2. analyze the network through their degree and betweenness centrality measures
3. apply the page rank algorithm to find the page ranks of every node of the graph, the results of the algorithm will depend on the graph degrees and edges correlation between nodes of the graph

In [10]:
```
#unweighted graph (visualization)

G = nx.Graph()
for i in range(len(df)):
    for j in range(len(df)):
        if i!=j and df['Title'][i] == df['Title'][j]:
            G.add_edge(i,j)
```

In [11]:
```
nx.draw_networkx(G)
plt.savefig("path.jpg")
```

Figure : graph construction

In [12]:
```
#weighted graph


# creating a multigraph able to contain multiple edges between nodes,
# weight between edges being the mean of user ratings given by both

GW = nx.MultiGraph()
for i in range(len(df)):
    for j in range(len(df)):
        if i!=j and df['Title'][i] == df['Title'][j]:
            GW.add_edge(df['reviewerID'][i],df['reviewerID'][j], weight = (int)(df['overall'][i]+df['overall'][j])/2)
```

In [20]:
```
# visualizing network with degree
nx.draw(GW)
plt.savefig("gw_path.jpg")
```

Figure : graph visualization

In [21]:
```
print(nx.info(GW)) # same node for repeated user
print(nx.info(G)) # different node for repeated user
```

```
Name:
Type: MultiGraph
Number of nodes: 5630
Number of edges: 1181258
Average degree: 419.6298
Name:
Type: Graph
Number of nodes: 7338
Number of edges: 590629
Average degree: 160.9782
```

In [22]:
```
max_node_list = sorted(GW.degree, key=lambda x: x[1], reverse=True)
max_node_list[:10]
```

Out[22]:
```
[('AFVQZQ8PW0L', 6828),
 ('A1X8VZWTOG8IS6', 4052),
 ('A1M4NJYP0WNL8Q', 3570),
 ('A3H2CKTFZ3B3GD', 3446),
 ('A2NHD7LUXVGTD3', 3224),
 ('ABMX8XUNPR3LP', 2840),
 ('A2MPXQB68PGSM8', 2772),
 ('A3RTNA9THAU2OP', 2772),
 ('A2F6N60Z96CAJI', 2762),
 ('A2ZATPER188K3J', 2732)]
```

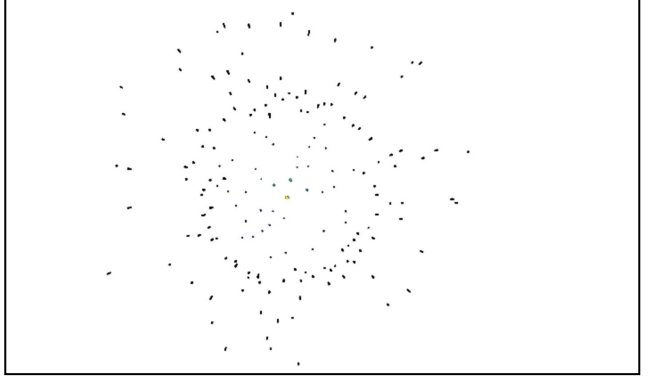
In [42]:
```
print(max_node_list[-1])
max_node_list_unweighted = sorted(G.degree, key = lambda x: x[1], reverse=True)
print(max_node_list_unweighted[-1])
```

```
('A1W0Z6DWSQAFHN', 18)
(6445, 9)
```

Figure : analyzing network by degree and betweenness centrality

In [24]:
```
# visualizing network by betweenness centrality

pos = nx.spring_layout(G)
betCent = nx.betweenness_centrality(G, normalized=True, endpoints=True)
node_color = [200.0 * G.degree(v) for v in G]
node_size =  [v * 100 for v in betCent.values()]
plt.figure(figsize=(8,8))
nx.draw_networkx(G, pos=pos, with_labels=False,
                 node_color=node_color,
                 node_size=node_size )
plt.axis('off')
```

Out[24]:
```
(-1.0789565235376357,
 1.0989979296922683,
 -1.0322396516799928,
 1.0445556402206422)
```



In [33]:
```
pr = nx.pagerank_numpy(G) # will return dict of user node page ranks
print(pr)
```

```
{0: 0.0001362769146882824 8, 1: 0.0001362769146878562, 2: 0.0001362769146882911 3, 3: 0.0001362769146878362 3, 4: 0.0001362769146878362 2, 5: 0.0001362769146878362 2, 6: 0.0001362769146878362 2, 7: 0.0001362769146878362 2, 8: 0.0001362769146878362 2, 9: 0.0001362769146878362 2, 10: 0.0001362769146878362 2, 11: 0.0001362769146904875 3, 13: 0.0001362769146904875 3, 14: 0.0001362769146904875 3, 15: 0.0001362769146904875 3, 16: 0.0001362769146904875 3, 17: 0.0001362769146904875 3, 18: 0.0001362769146904875 3, 19: 0.0001362769146904875 3, 20: 0.0001362769146904875 3, 21: 0.0001362769146904875 3, 22: 0.0001362769146904875 3, 23: 0.0001362769146904875 3, 24: 0.0001362769146904875 3, 25: 0.0001362769146904875 3, 26: 0.0001362769146904875 3, 27: 0.0001362769146904875 3, 28: 0.0001362769146904875 3, 29: 0.0001362769146904875 3, 30: 0.0001362769146904875 3, 31: 0.0001362769146904875 3, 32: 0.0001362769146904875 3, 33: 0.0001362769146904875 3, 34: 0.0001362769146904875 3, 35: 0.0001362769146904875 3, 36: 0.0001362769146904886 7, 37: 0.0001362769146904886 7, 38: 0.0001362769146904886 7, 39: 0.0001362769146904886 7, 40: 0.0001362769146904886 7, 41: 0.0001362769146904886 7, 42: 0.0001362769146904886 7, 43: 0.0001362769146904886 7, 44: 0.0001362769146904886 7, 45: 0.0001362769146904886 7, 46: 0.0001362769146904886 7, 47: 0.0001362769146904886 7, 48: 0.0001362769146904886 7, 49: 0.0001362769146904886 7, 50: 0.0001362769146904886 7, 51: 0.0001362769146904886 7, 52: 0.0001362769146904886 7, 53: 0.0001362769146904886 7, 54: 0.0001362769146904886 7, 55: 0.0001362769146904886 7, 56: 0.0001362769146904886 7, 57: 0.0001362769146904886 7, 58: 0.0001362769146904886 7, 59: 0.0001362769146904886 7, 60: 0.0001362769146904886 7, 61: 0.0001362769146904890 8, 62: 0.0001362769146904890 8, 63: 0.0001362769146904890 8, 64: 0.0001362769146904890 8, 65: 0.0001362769146904890 8, 66: 0.0001362769146904890 8, 67: 0.0001362769146904890 8, 68: 0.0001362769146904890 8, 69: 0.0001362769146904890 8, 70: 0.0001362769146904890 8, 71: 0.0001362769146904890 8, 72: 0.0001362769146904904 6, 73: 0.0001362769146904904 6, 74: 0.0001362769146904904 6, 75: 0.0001362769146904904 6, 76: 0.0001362769146904904 6, 77: 0.0001362769146904904 6, 78: 0.0001362769146904904 6,
```

In [32]:
```
gm = nx.google_matrix(GW, alpha = 0.89)
print(gm)
```

```
[[1.95381883e-05 6.00992280e-03 1.54233843e-02 ... 1.95381883e-05
  1.95381883e-05 1.95381883e-05]
 [7.98913331e-02 1.95381883e-05 1.02711846e-01 ... 1.95381883e-05
  1.95381883e-05 1.95381883e-05]
 [5.34195382e-02 2.67195382e-02 1.95381883e-05 ... 1.95381883e-05
  1.95381883e-05 1.95381883e-05]
 ...
 [1.95381883e-05 1.95381883e-05 1.95381883e-05 ... 1.95381883e-05
  1.62341131e-02 1.80357325e-02]
 [1.95381883e-05 1.95381883e-05 1.95381883e-05 ... 1.81008023e-02
  1.95381883e-05 1.81008023e-02]
 [1.95381883e-05 1.95381883e-05 1.95381883e-05 ... 1.80357325e-02
  1.62341131e-02 1.95381883e-05]]
```

Figure : page rank algorithm and results
(list of irrelevant users)

In [36]:
```
# finding min page rank nodes from graph
minval = min(pr.values())
res = [k for k, v in pr.items() if v==minval]
print(res)
```

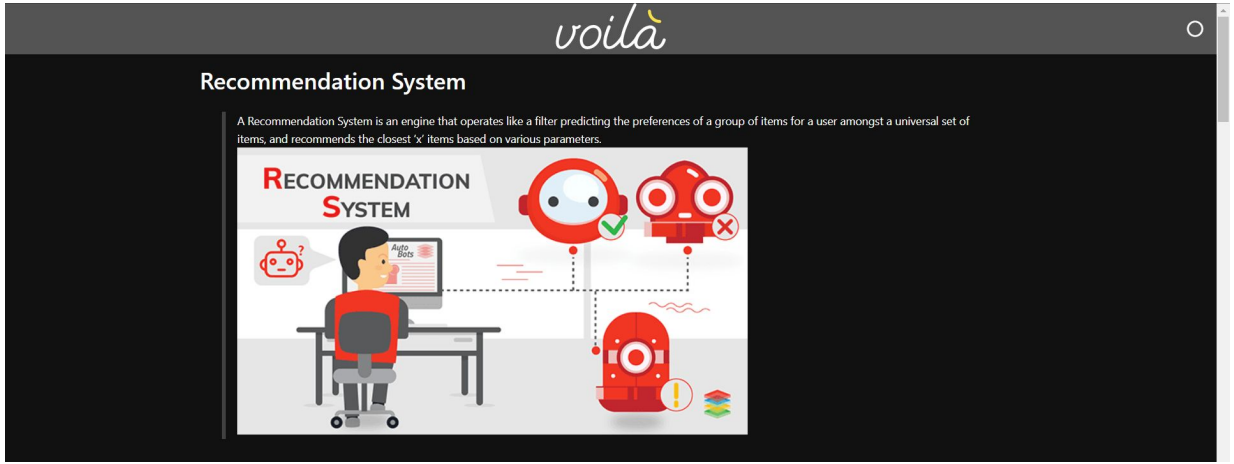
```
[3, 4, 5, 6, 7, 8, 9, 10]
```

In [52]:
```
# irrelevant users (based on page rank scores) ->
irrelevant_users = df[3:11]
print(irrelevant_users)
```

	reviewerID	asin	Title	reviewerName
3	A1CT8ENDZSYTX3	60554800	Red Tide	Lisa B.
4	A2SI6BNK5SWSMD	60554800	Red Tide	L. J. Roberts
5	A1SSYYL2WTAK4Y	60554800	Red Tide	Mtnhi 5
6	AVZ1LIW6EUQIK	60554800	Red Tide	Robert J. Unger "Dr. Gates"
7	A3QJQQZTKFV7BJ	60554800	Red Tide	Sandy "WR Gma"
8	A3OMPCA27U6WL2	60554800	Red Tide	Teri Tipton
9	A31WHFXF6T06DR	60554800	Red Tide	Watson McFestus "Watson McFestus"
10	A13HLAGY514ICM	60554800	Red Tide	William W. Mouroux

Deployment

We have deployed our complete model (jupyter notebook) using voila, to get an interactive interface of our work



voilà

Train Model

Amazon Book Reviews Dataset

	reviewerID	asin	reviewerName	reviewText	overall	title
0	A15Q7ABIU9O9YZ	60554800	Larry Scantlebury	This is my first GM Ford book and I will read ...	3	Red Tide: A Novel (Ford, G. M.)
1	AUIIDXNYVTEA8	60554800	Les Stockton	I liked the story. I thought the book added a...	4	Red Tide: A Novel (Ford, G. M.)
2	A20N5GOON55TE9	60554800	lila	As always, G.M. Ford does not disappoint. I st...	5	Red Tide: A Novel (Ford, G. M.)
3	A1CT8ENDZSYTX3	60554800	Lisa B.	I love Ford's Leo Waterman series and the firs...	3	Red Tide: A Novel (Ford, G. M.)
4	A2SI6BNK5SWSMD	60554800	L. J. Roberts	It was nice to see Corso working with the poli...	3	Red Tide: A Novel (Ford, G. M.)

Data Analysis

Filtering Irrelevant Users using Link Analysis

What is Link Analysis

Link analysis is the process of investigating social structures by the use of networks and graphs. To help with the use case we are trying to solve, link analysis helps in the following ways:

Identifying anomaly nodes of a network (irrelevant users) who can be turned off to refine produced set of recommendations
Predicting new links on already built social network to interrelate a new product with older ones having similar features

voilà

Collaborative Filtering using SVD

What is Collaborative Filtering

Collaborative filtering technique is based on the idea that users that preferred certain items in the past are likely to agree again in the future. It filters out products that a user might prefer on the basis of activity by similar users.

What is SVD

SVD is Singular Vector Decomposition. What it does is that it decomposes a matrix into constituent arrays of feature vectors corresponding to each row and each column.

```
Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.8889  0.9411  0.9309  0.9203  0.0226
MAE (testset)     0.7106  0.7489  0.7432  0.7342  0.0169
Fit time          0.35    0.27    0.36    0.33    0.04
Test time         0.29    0.08    0.05    0.14    0.11
```

voilà

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9341  0.9120  0.9115  0.9192  0.0105
MAE (testset)     0.7448  0.7211  0.7357  0.7338  0.0098
Fit time          0.16    0.19    0.15    0.16    0.02
Test time         0.03    0.03    0.02    0.03    0.01

Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
```

Top Books

	book_id	book_name
0	144	Septimus Heap, Book Two: Flyte: 02 (Septimus H...
1	46	The Nanny
2	95	Eels: An Exploration, from New Zealand to the ...
3	57	The Barefoot Princess: The Lost Princesses #2 ...
4	58	The Prince Kidnaps a Bride: The Lost Princesse...
5	90	A Dark Champion: 3 (Brotherhood of the Sword S...
6	167	By the Shores of Silver Lake: Full Color Editi...

Recommendation System

Books similar to Valdez Is Coming Low Price are:

	book_id	book_name
0	137	Warlord: A Life of Winston Churchill at War, 1...
1	31	Cadillac Beach: A Novel: 6 (Serge Storms)
2	96	A Year with C. S. Lewis: Daily Readings from H...
3	41	Gentlemen and Players: A Novel
4	0	Mission Compromised
5	99	Scots on the Rocks: A Bed-and-Breakfast Myster...
6	128	Where God Was Born: A Journey by Land to the R...
7	158	Useful Idiots: How Liberals Got It Wrong in th...
8	131	Busting Vegas: The MIT Whiz Kid Who Brought th...
9	48	Ruby Holler

Project Timeline

August 2020
Project and Use Case Discussion

September 2020
Finalizing approach, solution and timeline

October 2020
Sentiment analysis

November 2020
Minor Presentation

January - February 2021
Collaborative filtering and Link Analysis Exploration

March 2021
User Segregation addition and API building (deployment)

April 2021
Project Completion and presentation

Minor

Major

Work Repository

https://github.com/MarleyKuinn18/RecommendationSystem-v1

https://github.com/MarleyKuinn18/link-analysis

Thank you

Equation for Generating an Aggregated Sentiment score from Comments

1. Sentiment scores are predicted from sentiment analysis as positive, negative and neutral; holding values +1, -1, 0 respectively.

2. Sentiment Coefficient ->
   X = number of neutral / total, iff s = 0
   = number of positive / total, iff s = +1
   = number of negative / total, iff s = -1

3. We then take a **scalar product** of sentiment coefficient and rating coefficient for every **product**.

Equation for Generating “new rating” and Training the data

Rating coefficient (aij) ->

$$aij = log(Lij) \times 1/ (sij - \bar{s}j)/\sigma j$$

Where,

Lij is a comment length of the reviewer i for a product j

$\bar{s}j$, σj is an average and a standard deviation of scores for a product j, respectively.

reviewer i evaluates a product j as a score sij (= 1,2, . . . ,5)