Amazon Recommendation System

Qiaofeng Wu (qw2235), Minhui Li (ml4026)
Department of Electrical Engineering
Columbia Univerisity
qw2235@columbia.edu, ml4026@columbia.edu

***Abstract*—When customers browsing online shopping websites, Amazon will recommend some related products which allows customers to make extra purchase. However, current recommendation system is still not well-directed thus customers are likely to just ignore the unrelated promotions. Our project combines the features of both customers (reviewers, users) and products (items), together with existing known ratings that users assign to items but not full-mesh, constructs an original algorithm to update the recommendation system and finally makes predictions about previously unknown ratings that users might give to items. Then we are able to recommend potential customers with certain items that are sorted according to our prediction scores. In this project, we use Spark, Python and Gephi as tools to do the prediction and visualization. The algorithm successfully predicts the unknown rating of a user towards an item with a root-mean-square error of approximately 0.5 by selecting adequate parameters.**

Keywords-Amazon; recommendation; similarity; prediction;

I. INTRODUCTION

Nowadays, online shopping has become inseparable to modern people's life. Amazon, an American electronic commerce and cloud computing company, is the largest Internet retailer in the world measured by revenue and market capitalization [1]. When you just finished a purchase and browse through the website, you will probably find a column down the page named "Sponsored products related to this item", which in fact can be regarded as a trick provided by the sales company to tempt customers to buy something else that may seem not necessary to them. However, some of these recommendation will remind customers of something that they really want, which both boost the revenue and bring customers convenience. However, these recommendations usually fail to arouse people's interests and becomes useless in some way.

Based on the above circumstances, our project aims to construct a more efficient and effective recommendation system for Amazon, helping the website to make more target-aimed products for potential customers.

II. RELATED WORKS

Julian McAuley, Christopher Targett, Qinfeng Shi and Anton van den Hengel seek to construct a model of human sense mainly based on the appearance two objects. For example, a pair of jeans from Levis may be seen to be alternative to another pair of jeans of Lee; a pair of jeans may also be seen to be complementary to a matching top shirt. This method is not based on fine-grained and sensitive features of user annotations, but capture as large as possible dataset and cast the whole as a network inference problem [2]. Ruining He and Julian McAuley build an original model for the One-Class Collaborative Filtering setting, aiming to estimate users' fashion-aware personalized ranking functions based on past feedback. This method first requires to understand the dimension of potential customers' preference as well as their dynamics, then combines high-level visual features, extracted from a deep convolutional neural network, past feedback of users, as well as evolving trends within the community [3].

III. SYSTEM OVERVIEW

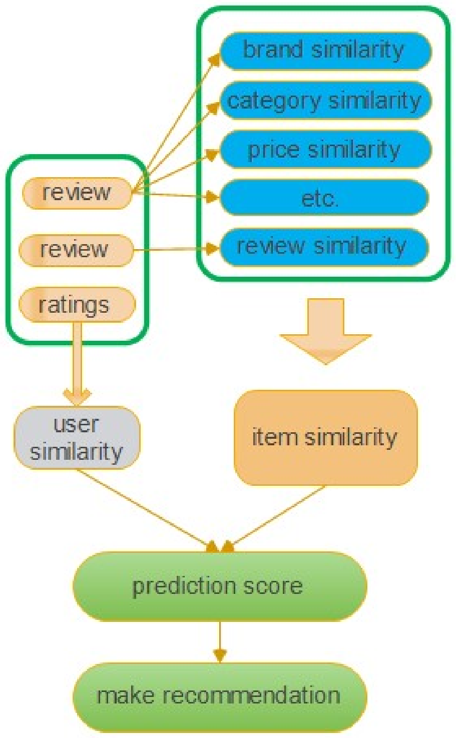


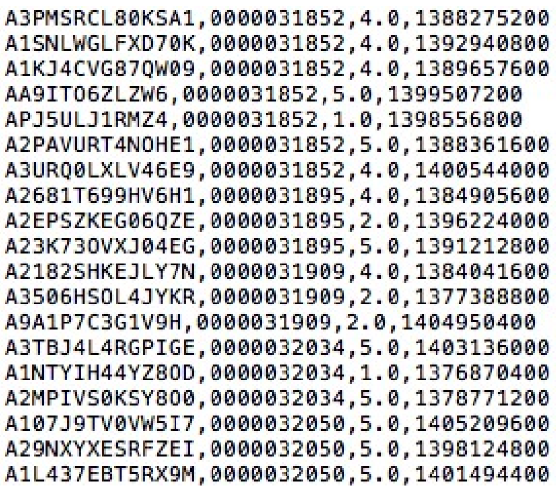
Fig3.1 major process of the project

The raw dataset "Amazon Product Data" we use is obtained from Julian MaCauly from UCSD. We pick a subset called “Sports and Outdoors”. We combine different features of products and the known ratings that customers have already assigned to the items to generate a new item similarity matrix and a new user similarity matrix. Then these two similarity matrices are applied to calculate the prediction score. Finally, for a selected user, we rank the items that the user has not rated according to prediction score and figure out which one is mostly likely to be purchased.

The whole dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014. Specifically, the subset of “Sports and Outdoors” contains 3,268,695 reviews and 532,197 products’ metadata information.

The review dataset includes ratings, text, helpfulness votes, and the product metadata includes descriptions, category information, price, brand, and links (also viewed/also bought graphs).

The original rating dataset is shown in Figure #!!!!, with four columns corresponding to Username, Itemname, Rating and Timestamp respectively. For this project, we definitely do not need to use timestamps so we’ll simply ignore it when preprocessing. The dataset contains 1,990,521 distinct users and 478,898 distinct items. The number of ratings in total is about two million.



```
A3PMSRCL80KSA1,0000031852,4.0,1388275200
A1SNLWGLFXD70K,0000031852,4.0,1392940800
A1KJ4CVG87QW09,0000031852,4.0,1389657600
AA9ITO6ZLZW6,0000031852,5.0,1399507200
APJ5ULJ1RMZ4,0000031852,1.0,1398556800
A2PAVURT4NOHE1,0000031852,5.0,1388361600
A3URQ0LXLV46E9,0000031852,4.0,1400544000
A2681T699HV6H1,0000031895,4.0,1384905600
A2EPSZKEG06QZE,0000031895,2.0,1396224000
A23K73OVXJ04EG,0000031895,5.0,1391212800
A2182SHKEJLY7N,0000031909,4.0,1384041600
A3506HSOL4JYKR,0000031909,2.0,1377388800
A9A1P7C3G1V9H,0000031909,2.0,1404950400
A3TBJ4L4RGPIGE,0000032034,5.0,1403136000
A1NTYIH44YZ80D,0000032034,1.0,1376870400
A2MPIVS0KSY800,0000032034,5.0,1378771200
A107J9TV0VW5I7,0000032050,5.0,1405209600
A29NXYXESRFZEI,0000032050,5.0,1398124800
A1L437EBT5RX9M,0000032050,5.0,1401494400
```

Fig3.2 Rating data description

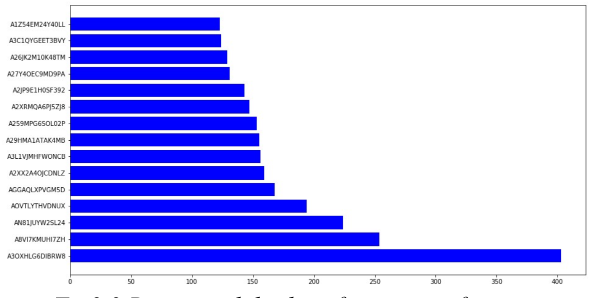


Fig3.3 Raters with highest frequency of rating

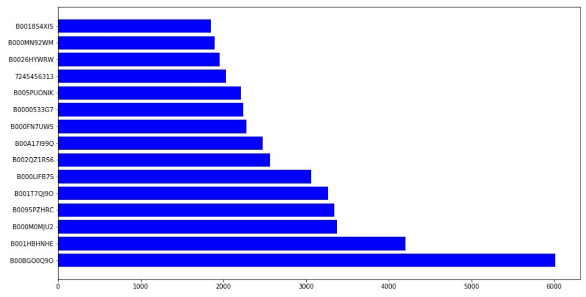


Fig3.4 Most frequently rated items

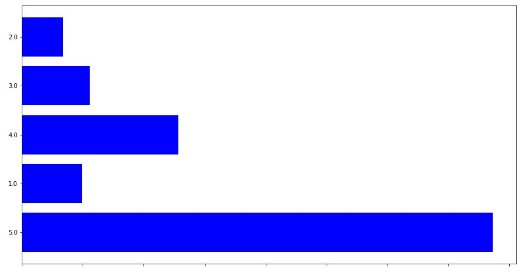


Fig3.5 Frequency of ratings

Figure 3.3, 3.4 And 4.5 Are three basic analysis for the dataset. We can see that user “A3OXHLG6DIBRW8” is really a big fan of Amazon shopping that he has rated about 400 items. “B00BGO0Q9Q” seems to be a popular merchandise that it is rated for about 6,000 times. Figure 3.5 suggests that most of the commodities are satisfying, which are given ratings over 4 stars.

IV. ALGORITHM

For visualization, we filtered some small nodes and edges according to the topology degree, i.e. if there exists a connection between two nodes, then we add 1 to each node’s degree, and there is no difference between in-degree and out-degree as the graph is undirected. Also, we did some graph distance metrics to better understand its features.

(a) Average Shortest Path: The graph distance, γ, between two nodes u and v is defined as the minimum number of edge-hops required to traverse the network, starting from node u and ending at node v (these values are identical in undirected graph, like this one, but not in directed graph. By applying average shortest path to all nodes in the graph, we can measure small world effect.

(b) Network Diameter: Imagine to create a circle in order to cover up all the graph, the diameter must satisfy that it is larger than the longest distance between any of two nodes within the graph. That is, the network diameter is the largest distance γ to be found in a network. Usually, average shortest path and network diameter are computed at the same time.

(c) Node Betweenness Centrality: This is the metric that indicates how often a node is found on a shortest path between two nodes in the network.

$$BC(v) = \sum_{u,w \in N, u \neq v \neq w} \frac{\sigma_{u,w}(v)}{\sigma_{u,w}}$$

where $\sigma_{u,w}(v)$ is the number of shortest paths that start form u and ends at w while passing through v, $\sigma_{u,w}$ is the summation of all shortest paths that start form u and ends at w with or without passing through v.

(d) Node Closeness Centrality: This is the metric that indicates how long it will take to transmit information start from a node u and reach other nodes in the graph/nework.

The average shortest path from a node v to all other nodes u is calculated by the below formula:

$$CC(v) = \sum_{u \in N, u \neq v} \frac{\gamma(v)}{N}$$

To be specific, we only take the non-infinite paths into consideration in case that there exist some nodes are not reachable by node v.

We use two layout that are ForceAtlas 2 and Yifan Hu to set the graph shape. ForceAtlas2 [4] is a force directed layout: it simulates a physical system in order to spatialize a network. Nodes repulse each other like charged particles, while edges attract their nodes, like springs. These forces create a movement that converges to a balanced state. This final configuration is expected to help the interpretation of the data. The force-directed drawing has the specificity of placing each node depending on the other nodes. This process depends only on the connections between nodes. Eventual attributes of nodes are never taken into account. This strategy has its drawbacks. The result varies depending on the initial state. The process can get stuck in a local minimum. It is not deterministic, and the coordinates of each point do not reflect any specific variable. The result cannot be read as a Cartesian projection. The position of a node cannot be interpreted on its own, it has to be compared to the others. Despite these issues, the technique has the advantage of allowing a visual interpretation of the structure. Its very essence is to turn structural proximities into visual proximities, facilitating the analysis and in particular the analysis of social networks. Yifan Hu [5] is also a force-directed algorithm that models the graph drawing problem by a physical system of bodies, with forces acting between them. The algorithm finds a good placement of the bodies by minimizing the energy of the system. To overcome some previous problems and advance in efficiency and high quality for large graphs, Yifan Hu combines multilevel approach which effectively obercomes local minimums, and also proposes an adaptivv cooling scheme for the basic force-directed algorithms and a scheme for selecting the optimal depth of octree/quadtree in Barnes and Hut algorithm.

We use the following formula to calculate the predication score of a selected user and a selected item. This formula was constructed by our own based on previous work [6, 7].

a: the selected user

i: the selected item

b: all the other users that have given known ratings to item i

j: all the other items that user a has given known ratings to

$$P_{a,i} = \overline{r_a} + \alpha \sum_{b=1}^{n} su(a,b)(r_{b,i} - \overline{r_b})$$
$$+ \overline{r_i} + \beta \sum_{j=1}^{m} si(i,j)(r_{a,j} - \overline{r_j})$$

$P_{a,i}$: the prediction rating that active user a assign to item j

$\overline{r_a}$: the average known ratings for user a

$\overline{r_b}$: the average known ratings for user b

su, si: similarity matrix for user and for item;

su(a,b): the similarity grade between user a and b

si(i,j): the similarity grade between item i and j

$r_{b,i}$: the rating that user b assign to item i

$\overline{r_i}$, $\overline{r_j}$: the average known ratings for item i and j

$r_{a,j}$: the rating that user a assign to item j

α,β: the given coefficient (can be modified later)

As seen from above, the formula has two symmetric part, we call the former one user-based part and the latter one item-based part, each part was construct accordingly by the format of:

*mod = similarity * difference_with_neighbor*

*pre_score = own_average + coefficient * sum(mod)*

To be specific and have more detailed understanding of the formula, we use an example to illustrate it. Imagine a user a has graded n items before he or she move on to purchase the next item, we want to predict the item that is most likely to arouse user a's interest, so we need to assign prediction score for every items that user a has not graded previously. After all the calculations are done, we are able to sort these items according to its prediction grade, i.e. we can rank these items from the most likeliness to the least likeliness.

(1) First, pick one specific item, say, item i, for user-based part, we need to calculate the average rating that user a has given, this is to set a base since some people prefer to give a fairly low grade no matter how good the product is, while other people are generous about the score they give. Also we need to calculate all the average ratings that user b ($b \neq a$, $b \in$ users) has given, which will be applied next.

(2) Second, we will need to figure out which users have rated item i, and calculate the difference between this rating and their averages to see whether item i is below or above these users' expectations.

(3) We then multiply this difference by the similarity likeliness between user a and user b, that is to say, we trust someone that is more similar to myself and believe he is more likely to have the same taste as me, so I will take his advice into more consideration. This arithmetic product can be regarded as a modification factor that we use to update the average rating we calculated when start.

(4) Next, we sum all these arithmetic products up and multiply the summation by a coefficient. The coefficient is used to adjust and control the inference of modification factors. Later we will talk about how to choose the value of coefficients.

(5) Finally add the average rating and get the user-based part of prediction score.

For item-based part, we do the steps correspondingly.

(1) First, for the picked item i, we also need to calculate the average rating that it received from all other users, this is to set a base give a general impression of how good or bad the item is. Also we need to calculate all the average ratings that other item j ($j\neq i$, $j\in$ items) has received, which will be applied next.

(2) Second, we will need to figure out which items have been rated by user a, and calculate the difference between this rating and their averages to see whether user a is mean or generous when giving ratings .

(3) We then multiply this difference by the similarity likeliness between item i and item j, that is to say, we believe user a has a fairly fixed tastes about items and is willing to assign similar ratings to similar items, so the more likely item i and item j are, the more influence that item j brings to item i will be considered. This arithmetic product can be regarded as a modification factor that we use to update the average rating we calculated when start.

(4) Next, we sum all these arithmetic products up and multiply the summation by a coefficient. The coefficient is used to adjust and control the inference of modification factors. As above, later we will talk about how to choose the value of coefficients.

(5) Finally add the average rating and get the item-based part of prediction score.

We can see from above that the calculation of similarity is a key point. For items, we combine two raw datasets (meta_Sports_and_Outdoors.json and review_Sports_and_Outdoors.json) and treat them as features of items, these features include the category, brand, price and plain text reviews that the products received; we use another raw dataset (ratings_Sports_and_Outdoors.csv) and treat it as features of users, in fact, it is more logical to use features like the living place, gender, salary level of customers, but as these information involves privacy we can not obtain any more detailed information other than the ratings that users give. Therefore, we will regard users that give similar ratings to a specified item as similar users.

For item-based part, there exist qualitative analysis and quantitative analysis as some features only involves “exactly the same” or “totally different” while others range between these two status. Take a quick example for qualititative analysis, if user i and user j are made by the same manufacturer, i.e. they have the same brand, then the i-th row and j-th column of brand similarity matrix is assigned 1, otherwise 0.

Take a quick example for quantitative analysis, the price of item i is price_i and the price of item j is price_j, then the i-th row and j-th column of brand similarity matrix is assigned

$$1 - \frac{|price\_i - price\_j|}{price\_i + price\_j}$$

i.e. the difference occupies less portion of their original price, the more similar they are.

Take another quick example for quantitative analysis, we will extract text features from customer reviews. Step by step, we tokenized the bag-of word, removed stop words like “the, and but” and did stemming to simplify all the words, transferring plural forms to singular forms. After we did so, we created a collection of review words for each item. If we want to calculate the review similarity score between item i and item j, we further need to create an intersection set and a union set from the two word collections. The review similarity score is then can be calculated as:

$$len(intersectioin) / len(union)$$

For user-based part, the only feature we can use that contains information about customers is the rating dataset. The similarity score between user a and user b can be calculated as below:

$$su(a,b) = \sum_i \frac{r_{a,i}}{\sqrt{\sum_{j\in I_a} r_{a,j}^2}} \frac{r_{b,i}}{\sqrt{\sum_{j\in I_b} r_{b,j}^2}}$$

where $r_{a,i}$, $r_{a,j}$ are the ratings that user a gives to item i and item j separately, I_a is the set of items on which user a has rated, and $r_{b,i}$, $r_{b,j}$ are the ratings that user b gives to item i and item j separately, I_b is the set of items on which user b has rated. In fact the above formula calculated the covariance of user a and user b.

Before trying to apply our collaborative filtering algorithm, we need to give each user and item an ID label as it is not available from the dataset. The operation is actually create a dictionary with usernames corresponding to positive integers. So we create an RDD with usernames zipped with their indexes, then create a dictionary with usernames as keys and indexes as value, and finally merge dictionaries in the RDD together as a big dictionary. In this way, each user or item has a unique ID label.

For collaborative filtering, Spark has a built-in ALS function. It works well for rating systems where items are frequently rated. But the rating matrix derived from Amazon is too sparse that the sparsity is regarded as 100%. This is reasonable because Amazon users averagely cannot even buy thousands of commodities out of millions, not to say rate them. ALS will predict missing ratings a value of approximately 0, which greatly increase the RMSE to more than 4.

That's the motive we give every prediction a baseline of mean values. But here, we do not want to make the RMSE as low as possible. Because according to Chebyshev's inequality [8],

$$P(|X-\mu| \geq k\sigma) \leq \frac{1}{k^2}$$

the value of a random variable is lowly probable to be too far away from the mean. If we set α and β both equal to 0, which means we simply predict according to the mean ratings of users and items without considering the similarities, we will definitely achieve a low RMSE. Then the problem of underfitting occurs and it loses the meaning to do collaborative filtering. Thus we set our target RMSE to be 0.5 because the 5-star rating is not anticipated to change much with an expected error of 0.5.

To guarantee the error to be close to 0.5, the hyper-parameters α and β need to be adequately selected. After a bunch of tests, we found out that when α and β both equal to 0.002, the test RMSE is 0.499148112264, which is close enough to 0.5.

For recommendation, take user "A35UHVTCH8150C" as an example, if we would like to recommend 5 items to him, we first go back to the user dictionary to find its ID. Next, we find out the IDs of 5 items with the highest predicted rating based on the prediction matrix. The following step is going through the item dictionary and find 5 corresponding item names, and finally print them out.

V. SOFTWARE PACKAGE DESCRIPTION

(1) Gephi:

This is the tool that we use to realize visualization. On its website, it parallelize itself as Photoshop but for graph data, where the user interacts with the representation, manipulate the structures, shapes and colors to reveal hidden patterns. The goal is to help data analysts to make hypothesis, intuitively discover patterns, isolate structure singularities or faults during data sourcing. Users are able to see real-time visualization that we can better understand how a change of a single parameter can influence the whole network.

It also provides different layout algorithms which gives the shape to the graph. The statistics and metrics framework offer the most common metrics for social network analysis (SNA) and scale-free networks. By dynamic filtering, users are able to select certain nodes or edges.

(2) Pandas:

This is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. We mainly use it to create a dataframe (DataFrame object for data manipulation with integrated indexing.) that combines all known features as a preparation for the next step.

(3) Numpy:

This is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

(4) Sklearn:

Scikit-learn is a free software machine learning library for the Python programming language. It features a various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

(5) Spark:

Apache Spark is an open-source cluster-computing framework. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Pyspark is Spark with Python Shell.

VI. EXPERIMENT RESULTS

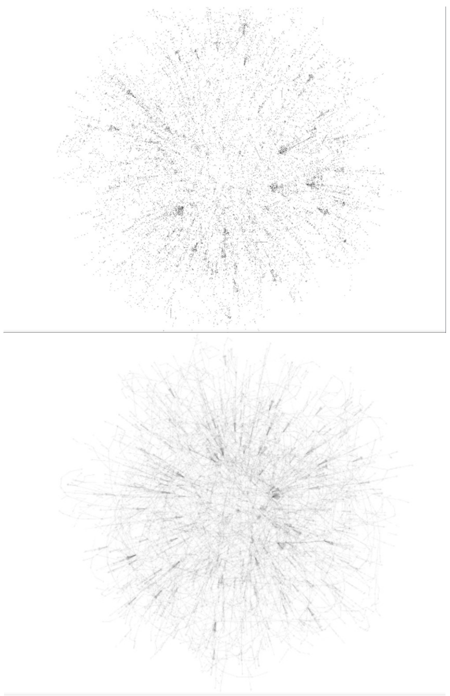


Fig5.1 Visualization of two subsets of t 1

Above are two subsets of the ratings that users give to items, we extracted them according to label features.

Zoom in one of the graph and take a closer look.

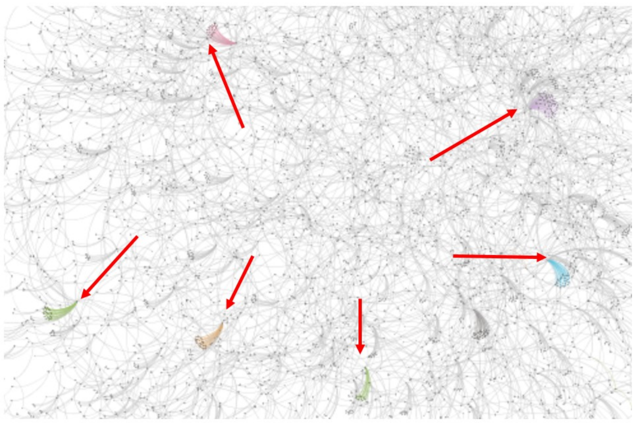
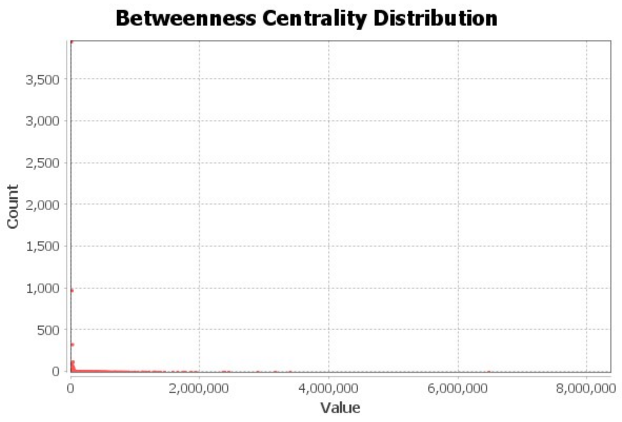
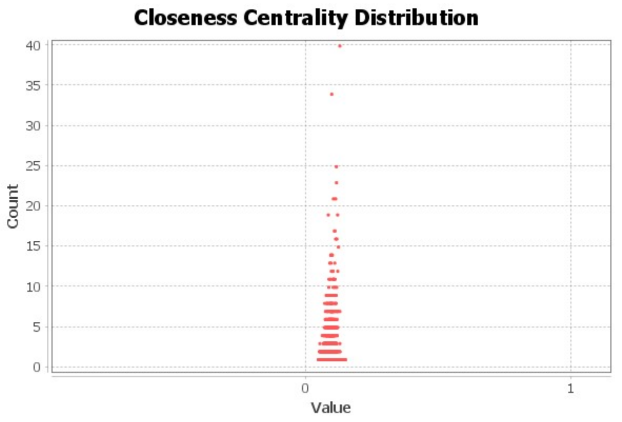


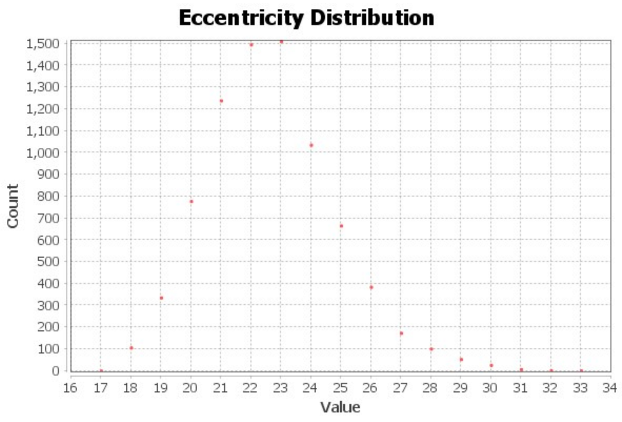
Fig5.2 Zoom in

The size is configured according to Betweeness Centrality and the color is configured according to Closeness Centrality.

We just display the distributions of one subset, other distributions are put shown in the appendix.







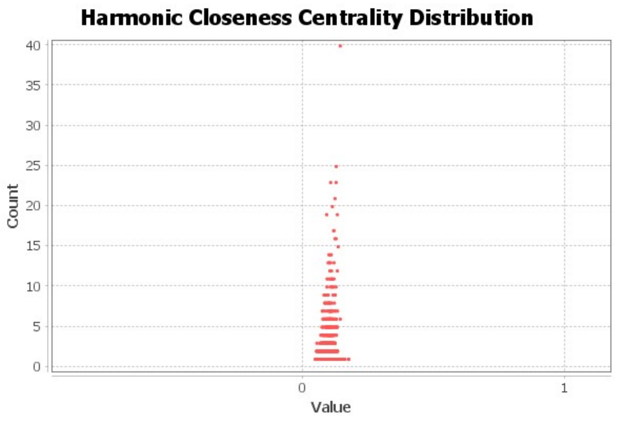


Fig5.3 Distributions of features

After we combine all the features of items (title, category, brand, price, review, etc.), we acquire an item similarity matrix. Here we show two layouts of its visualization.

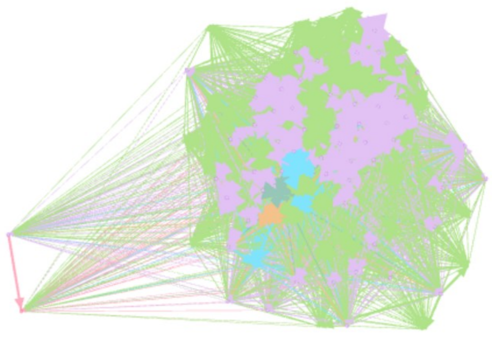


Fig 5.4 layout: ForceAtlas 2

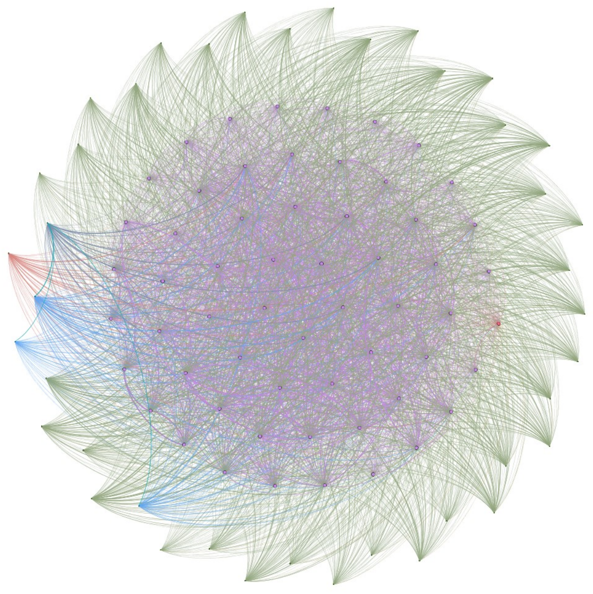


Fig5.5 layout: Yifan Hu

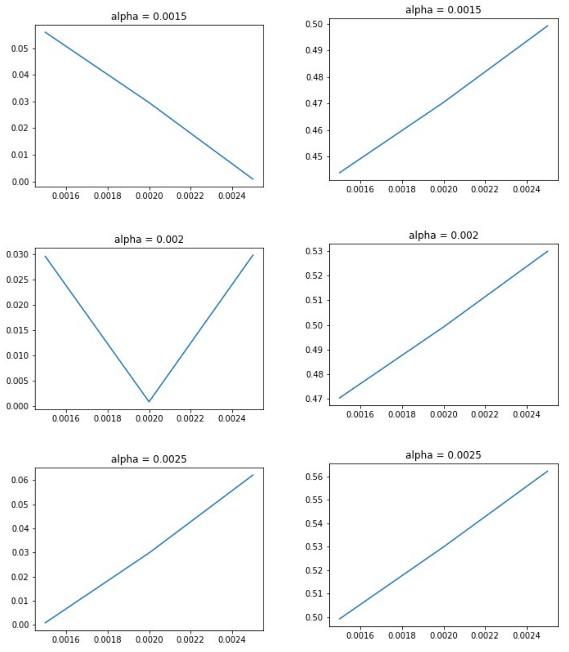
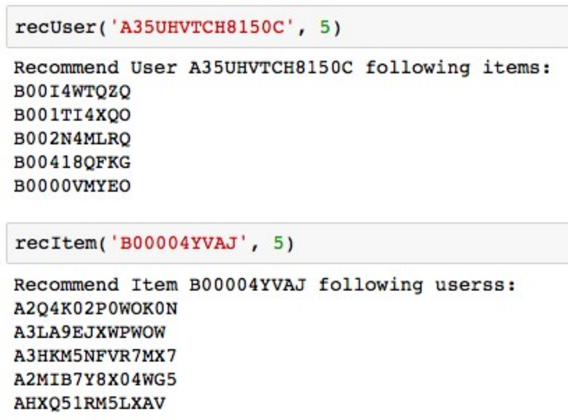


Fig5.3 Parameter selection

The upper subplot in Figure 4.3 is the difference between the RMSE and 0.5. The subplot below is the RMSE. Inferred from the figures, α = 0.0015, β = 0.0025 and α = 0.002, β = 0.002 are two locally optimal settings. We manually compared the two errors and found out that α and β both equal to 0.002 is a better setting.

Finally, after all parameters set, we simply do arithmetic calculation to predict all the ratings. I randomly pick 1 username and 1 itemname and ask the system to recommend 5 items and users.



```
recUser('A35UHVTCH8150C', 5)

Recommend User A35UHVTCH8150C following items:
B00I4WTQZQ
B001TI4XQO
B002N4MLRQ
B00418QFKG
B0000VMYEO

recItem('B00004YVAJ', 5)

Recommend Item B00004YVAJ following userss:
A2Q4K02P0WOK0N
A3LA9EJXWPWOW
A3HKM5NFVR7MX7
A2MIB7Y8X04WG5
AHXQ51RM5LXAV
```

Fig5.4 Sample output

VII. Conclusion

We successfully constructed an original recommendation system for Amazon, which can suggest customers with the most potentially could be purchased products given reviews, ratings and item information.

The self-developed collaborative filtering algorithm works well. The algorithm successfully predicts the unknown rating of a user towards an item with a root-mean-square error of approximately 0.5 by selecting adequate parameters. The algorithm recommends items to users and users to items based on both the similarities between users and items, as well. In future, we can further apply this collaborative filtering algorithm on other rating systems so that we can do movie, music or game recommendation an so on.

In the future, we may try:

- Search for datasets that may contain information related with users, like living area, age, gender, etc. which helps to make a more accurate similarity matrix for users.
- Update the similarity calculation by adding edges between items (i.e. use more graph related algorithm on "user also bought").
- Find a more efficient coefficient sets for similarity allocation.

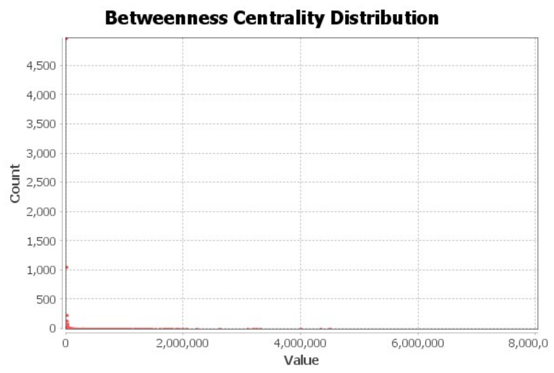
The contribution of each team member:

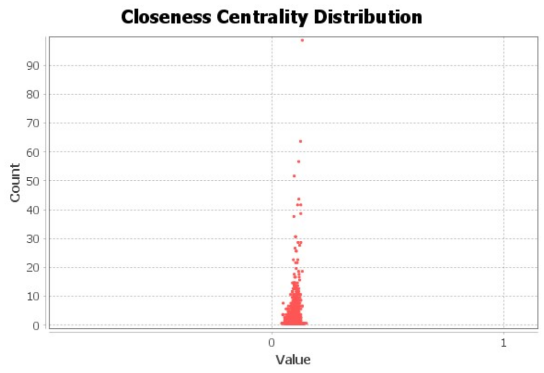
- Qiaofeng Wu: similarity matrix calculation, visualization, supporting slides & reports. Contribution Percentage: 50%
- Minhui Li: formula calculation, supporting slides & reports. Contribution Percentage: 50%

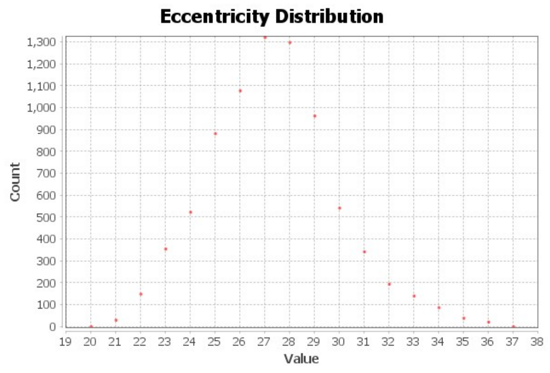
Acknowledgment

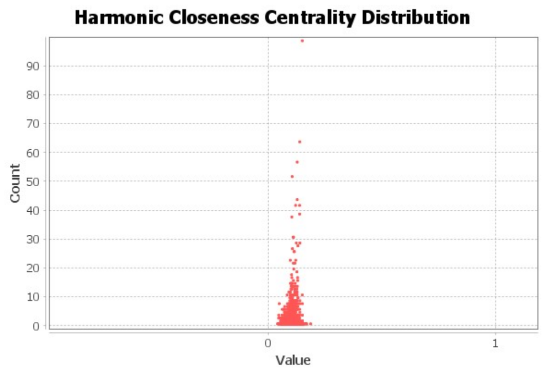
The authors would like to thank Ching-Yung Lin, Gongqian Li and Lingqing Xu.

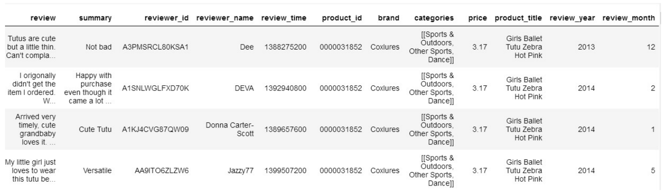
APPENDIX





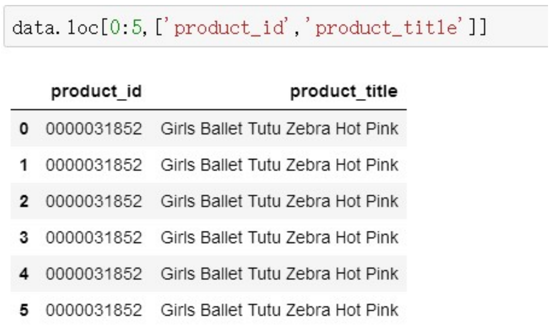






review	summary	reviewer_id	reviewer_name	review_time	product_id	brand	categories	price	product_title	review_year	review_month
Tutus are cute but a little thin. Can't compla...	Not bad	A3PMSRCL80KSA1	Dee	1388275200	0000031852	Coxlures	[[Sports & Outdoors, Other Sports, Dance]]	3.17	Girls Ballet Tutu Zebra Hot Pink	2013	12
I origonally didn't get the item I ordered. W...	Happy with purchase even though it came a lot ...	A1SNLWGLFXD70K	DEVA	1392940800	0000031852	Coxlures	[[Sports & Outdoors, Other Sports, Dance]]	3.17	Girls Ballet Tutu Zebra Hot Pink	2014	2
Arrived very timely, cute grandbaby loves it. ...	Cute Tutu	A1KJ4CVG87QW09	Donna Carter-Scott	1389657600	0000031852	Coxlures	[[Sports & Outdoors, Other Sports, Dance]]	3.17	Girls Ballet Tutu Zebra Hot Pink	2014	1
My little girl just loves to wear this tutu be...	Versatile	AA9ITO6ZLZW6	Jazzy77	1399507200	0000031852	Coxlures	[[Sports & Outdoors, Other Sports, Dance]]	3.17	Girls Ballet Tutu Zebra Hot Pink	2014	5

a quick look of dataframe



```
data.loc[0:5,['product_id','product_title']]
```

	product_id	product_title
0	0000031852	Girls Ballet Tutu Zebra Hot Pink
1	0000031852	Girls Ballet Tutu Zebra Hot Pink
2	0000031852	Girls Ballet Tutu Zebra Hot Pink
3	0000031852	Girls Ballet Tutu Zebra Hot Pink
4	0000031852	Girls Ballet Tutu Zebra Hot Pink
5	0000031852	Girls Ballet Tutu Zebra Hot Pink

extract out the product_title

REFERENCES

[1] B. Jopson, "Amazon urges California referendum on online tax," *The Financial Times,* vol. 4, 2011.

[2] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 43-52.

[3] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 507-517.

[4] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software," *PloS one,* vol. 9, p. e98679, 2014.

[5] Y. Hu, "Efficient, High-Quality Force-Directed Graph Drawing."

[6] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998, pp. 43-52.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 230-237.

[8] A. Papoulis, "Probability, random variables, and stochastic processes," 1965.