|  |
| --- |
| Product Recommendation and Reviews Analysis  An analysis of Amazon product reviews and product recommendation  IDS 561 Course Project  Abhijay Patne  Aditi Vishwasrao  Mehul Parmar  B:\new_sync\Box Sync\academics\sem2\553\our_presentation\COL.CBA.LGSB.LOCKB.SM.BLK.PNG |

Contents

[MOTIVATION 3](#_Toc449987738)

[DATASET 4](#_Toc449987739)

[CHALLENGES 5](#_Toc449987740)

[APPROACH 6](#_Toc449987741)

[TOOLS AND TECHNOLOGIES 8](#_Toc449987742)

[ANALYSIS AND EXPLORATION USING HIVE AND TABLEAU: 9](#_Toc449987743)

[FUTURE WORK 13](#_Toc449987744)

[CITATIONS 14](#_Toc449987745)

[REFERENCES: 14](#_Toc449987746)

[ATTACHMENTS 14](#_Toc449987747)

[APPENDIX 15](#_Toc449987748)

# MOTIVATION

With the rapid development of the internet, increasing number of web applications and more and more devices connecting to the internet, data is increasing at an exponential rate. Handling this data efficiently and analyzing it so that we can make most out it is very important. Proper tools and techniques must be used, we wanted to learn these techniques and get some hands-on experience with analyzing the data. We were looking for some data which is very close to our day-to-day lives so that we can relate it to the real world and make more sense of it.

Shopping is an essential part of our daily routine. Due to widespread internet access, convenience, free home delivery and many such features e-commerce portals are being very popular. Because of the huge user base, a lot of product listings and a number of reviewers, we decided to study Amazon’s product review data. If we are planning to buy something, apart from the product description, product’s reviews play an important role in making the decision. When a user logs in to the e-commerce portal, there are a number of categories, so many products listed, he gets confused what to buy and what not. In this situation, recommender systems play an important role by helping his decide which item should he buy. These recommendations might be based on his previous purchase history, his friends’ purchase history, user’s interests recorded in the system or items which are on sale or trending products on the website. At the same time if a user purchases an item A and we recommend an item B, which might be frequently bought together or might be a necessary part of item A then the user is happy and he purchases it. This helps the customer and the retailer, thus, it’s a win-win situation.

Apart from recommending items, review data can be utilized in more efficient way. Reviews analysis can tell which product is in more demand, which category is performing well, how reviews are helping customers make their purchases, customer’s purchase trend depending on the location, season and several other patterns can be found. Considering all the factors discussed till now, we decided to study Amazon product review data and develop a product recommendation engine.

# DATASET

Our dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.

Dataset is divided into two parts:  
1) reviews (ratings, text, helpfulness votes)

2) product metadata (descriptions, category information, price, brand, and image features) and links (also viewed/also bought graphs).

62 GB of review data - no duplicates whatsoever (82.83 million reviews)

Format is one-review-per-line in loose json.

The dataset contains the following fields:

reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B

asin - ID of the product, e.g. 0000013714

reviewerName - name of the reviewer

helpful - helpfulness rating of the review, e.g. 2/3

reviewText - text of the review

overall - rating of the product

summary - summary of the review

unixReviewTime - time of the review (unix time)

reviewTime - time of the review (raw)

Metadata

10 GB Metadata includes descriptions, price, sales-rank, brand info, and co-purchasing links:

The metadata contains information for 9.4 million products contains the following fields:

asin - ID of the product, e.g. 0000031852

title - name of the product

price - price in US dollars (at time of crawl)

imUrl - url of the product image

related - related products (also bought, also viewed, bought together, buy after viewing)

salesRank - sales rank information

brand - brand name

categories - list of categories the product belongs to

All the reviews and product metadata is divided into several categories. As the total data across the categories was 72 GB, it was difficult to dprocess it on our local machines. So we decided to study one of the major categories, “Sports and outdoors” which had 6.2 GB of review data and 3.2 GB of product metadata. So total size of the dataset we analyzed was 9.4 GB

# CHALLENGES

1. **Improper Data Format**

Though the provided data claims to be in JSON format, it was not it strict JSON format. Review text contained some special characters which made it difficult to parse the data for JSON reader libraries. To overcome this, we wrote a python script which will read the data and convert it to strict JSON format.

1. **Compute Limitations**

As our data was huge, it was consuming a lot of system resources and when we tried to load the entire data to our MapReduce jobs. We tried creating pseudo distributed clusters and running the same jobs again. Same result again and we realised that even if we create a pseudo cluster, our underlying resources were the same. To overcome this problem and use more resources, we tested our MapReduce jobs locally and then ran the program on Amazon EC2 instance and stored our data on Amazon S3 buckets. We used 4 EC2 medium instances (2 vCPUs, 4 GB RAM each), so we could use more compute power.

# APPROACH

We implemented a recommender system to identify similar item sets using Hadoop Distributed programming, Python MRJob and AWS EMR. The items-sets were filtered based on the concept of cosine / correlation similarity to further identify most related items. The details about the Item Based Collaborative Filtering Python MRJob Code are as follows:

**Item Based Collaborative Filtering using Python MRJob:**

Parameters for running Python MRJob on EMR:

* Input File. Eg: “input/review\_Sports\_and\_Outdoors.json”
* Output File. Eg: “output/review\_Sports\_and\_Outdoors.txt”
* -r emr: Runs an [MRJob](https://pythonhosted.org/mrjob/job.html#mrjob.job.MRJob) on Amazon Elastic MapReduce
* num\_ec2\_instances: Total number of instances to start up; basically the number of core instance you want, plus 1 (there is always one master instance). Default is 1. We made use of 4 EC2 instances.

Note: For running EMR Job from your Windows command prompt, you need to have an AWS account and you will need to setup user environment variables on your Window machine – AWS\_ACCESS \_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY.

Input Files for the Program:

* JSON file (Amazon Product Reviews)

Sample record:

{"reviewerID": "A2SUAM1J3GNN3B", "asin": "0000013714", "reviewerName": "J. McDonald", "helpful": [2, 3], "reviewText": "I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. The music is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!", "overall": 5.0, "summary": "Heavenly Highway Hymns", "unixReviewTime": 1252800000, "reviewTime": "09 13, 2009"}

The Python MRJob Program consists of the following components:

To define multiple steps, we override [steps()](https://pythonhosted.org/mrjob/job.html#mrjob.job.MRJob.steps) to return a list of [MRStep](https://pythonhosted.org/mrjob/step.html#mrjob.step.MRStep)s and then we write our mapper and reducer methods to get the desired result. The input, output and the task performed by each Mapper and Reducer methods are as follows:

Mapper1 - mapper\_parse\_input

This Mapper takes JSON file as the input, parse it and generates key value pairs with key = User ID and value = Item details (Item ID, Rating)

Reducer1 - reducer\_ratings\_by\_user

The Reducer1 class takes input from Mapper1 and further processes, i.e., generates a single output line for multiple values of the same key (User ID).

Mapper2 - mapper\_create\_item\_pairs

This Mapper takes the output from Reducer1 as the input and processes it to generate key value pairs. In this Mapper, the key is the co-rated item pairs and the value is the corresponding ratings for the items.

Reducer2 - reducer\_compute\_similarity

The Reducer2 takes input from Mapper2 and further processes, i.e., generates a single output line (similarity score, number of pairs) for multiple values of the same key (Co-rated item pairs).

Mapper3 - mapper\_sort\_similarities

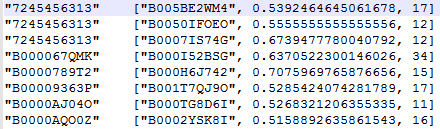
The Mapper3 takes input from Reducer2 and Shuffle things around so the key is (item1, score) so we have meaningfully sorted results.

Reducer3 - reducer\_output\_similarities

The Reducer3 takes input from Mapper3 and Output the results.

# Item => Similar Item, score, number of co-ratings

Sample output:



Finally, we thought it would be interesting to see if the item recommendations made by our algorithm are in sync with the item recommendations made by Amazon on the basis of Market Basket Analysis. We thus built a program using JDBC with Hive to retrieve the products suggested by Amazon for each user and we will be using this to compare the results returned by our item recommender. This is, however, a future goal as Amazon performs their recommendations across categories and we are currently restricted to within the categories. A screenshot of the output produced by the java program is shown [here in the appendix](#amazonRecommendation).

# TOOLS AND TECHNOLOGIES

**Hive**

Our dataset had many records and running a MapReduce job every time to query the data was very inefficient. So we decided to load our entire dataset into Hive. As we had already converted our data into strict JSON format, we used Json SerDe library to load the entire data into a Hive table. We partitioned the table according to year. When we execute a query in Hive, it runs MapReduce jobs in the backend and returns the result. We can also specify the number of mappers and reducers explicitly. This helped us query the records faster and create new tables according to several use cases discussed further in analysis section. Considering all these benefits, we decided to use Hive as our primary data store and querying tool.

**Hadoop**

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. We ran our MapReduce program on Hadoop to achieve faster processing of our dataset and obtain the desired result.

**Python MRJob**

MRJob lets you write MapReduce jobs in Python 2.6+/3.3+ and run them on several platforms.You can:

* Write multi-step MapReduce jobs in pure Python
* Test on your local machine
* Run on a Hadoop cluster
* Run in the cloud using [Amazon Elastic MapReduce (EMR)](http://aws.amazon.com/documentation/elasticmapreduce/)

**Amazon EMR, EC2, S3**

Amazon EMR simplifies big data processing, providing a managedHadoop framework that makes it easy, fast, and cost-effective for you to distribute and process vast amounts of your data across dynamically scalable Amazon EC2 instances. In order to run our job on multi-node cluster and achieve distributed computing we ran our MapReduce job on Amazon EMR services.

**Enthought Canopy**

Enthought Canopy is a comprehensive Python analysis environment that provides easy installation of the core scientific analytic and scientific Python packages, creating a robust platform you can explore, develop, and visualize on. We made use of this editor to write and test our Python MRJob code.

**Tableau**

We made use of Tableau desktop software to analyse the result obtained from our MapReduce code and Hive queries in order to evaluate the effectiveness of our Item Based recommender and perform a detail analysis of the review and meta data file. This helped us visualizing some of the interesting patterns in our data.

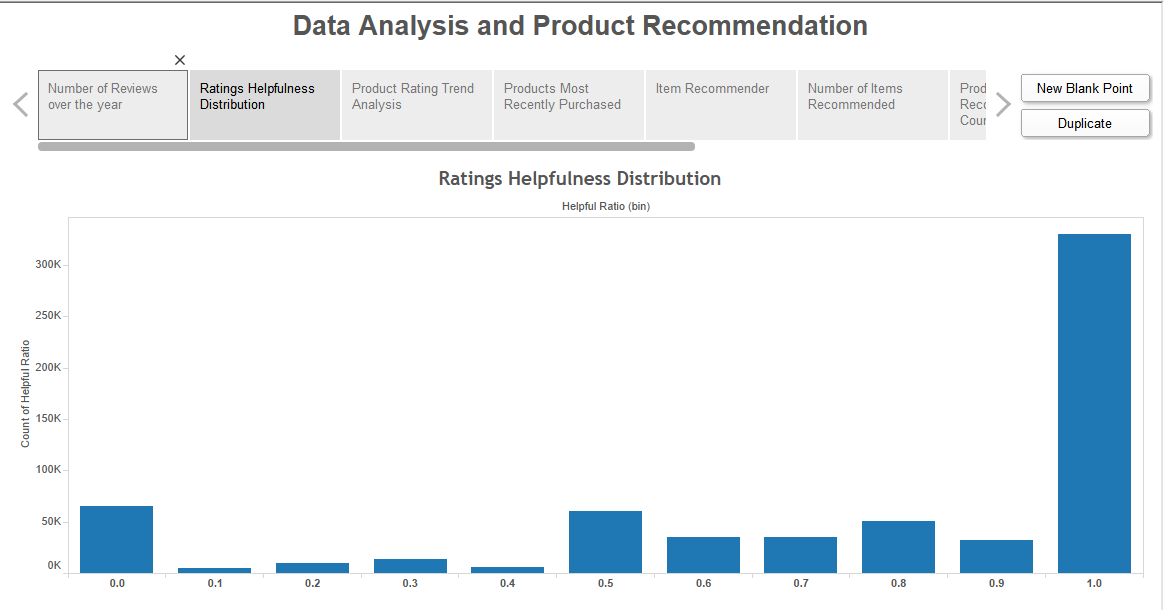
# ANALYSIS AND EXPLORATION USING HIVE AND TABLEAU:

The queries for all the explorations and analysis we performed on the dataset have been summarized in the text file attached below. We retrieved the results of each of these queries in the form of csv files which we than used to visualize in tableau. Screenshots of a few of these files have been shown in the [appendix](#appendix).

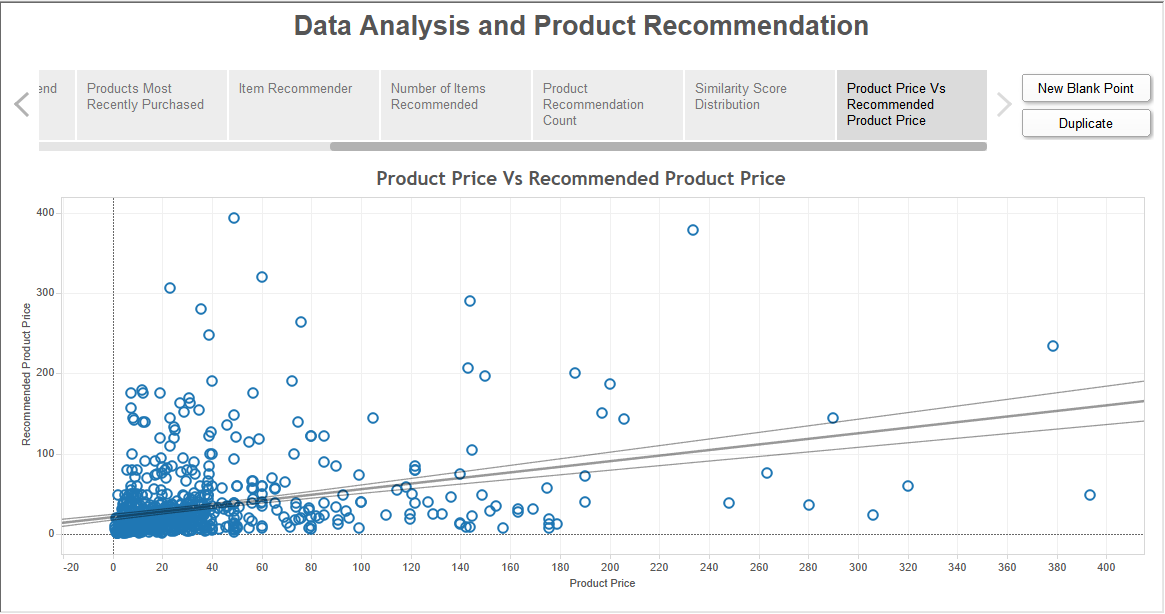


Our findings can be summarized as below:

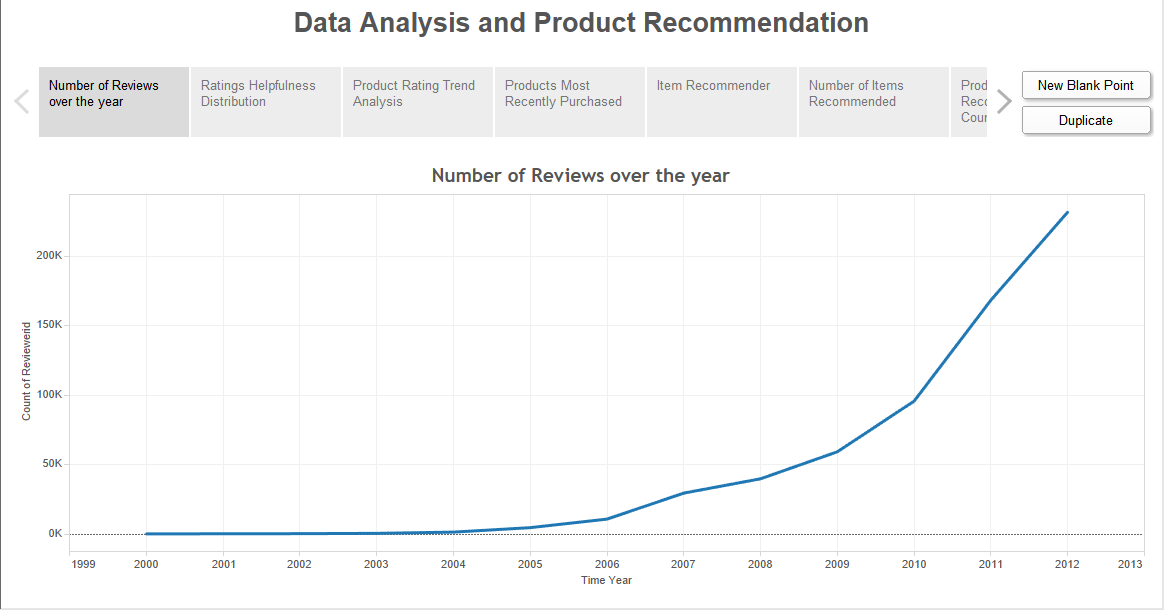
1. Looking at the plot of number of reviewers against the helpfulness ratio, it looks like most product reviews on Amazon are helpful to users.



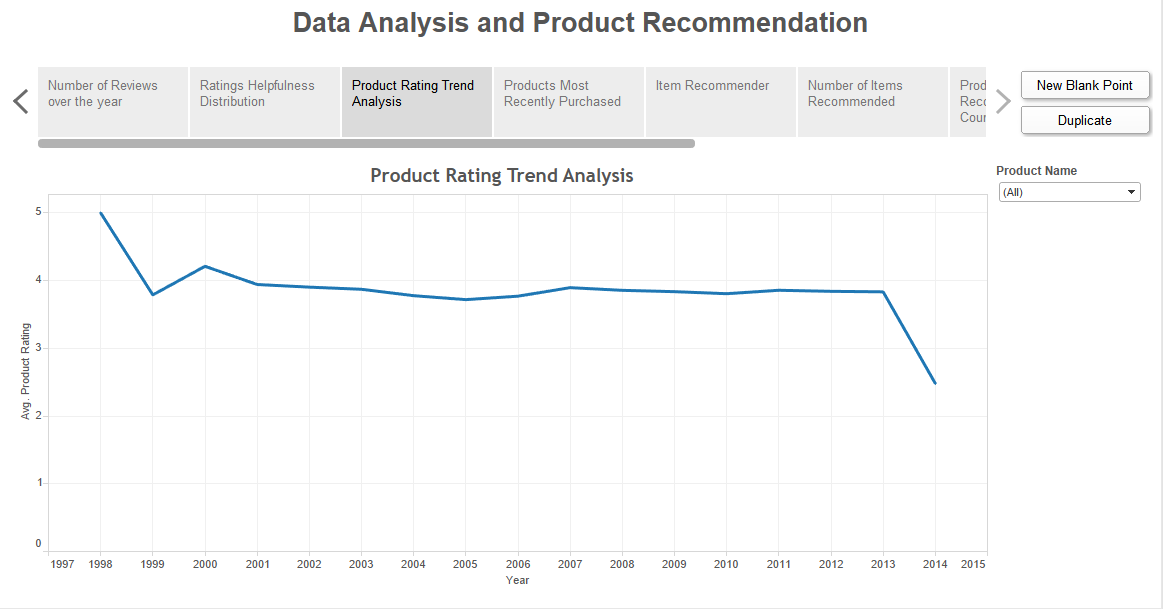
1. Most recommended products fall in the same price range as the product for which it is recommended with the exception of a few outliers. These can be ignored as our dataset is very large.



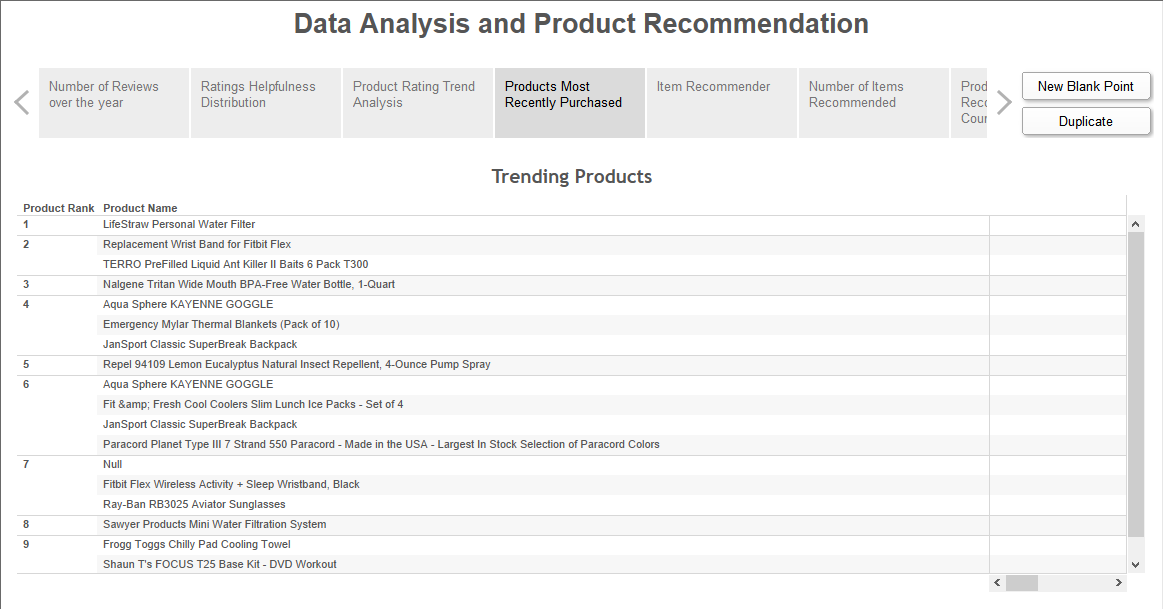
1. The number or reviewers for Amazon exponentially have increased over the years from 1996 to 2014. This shows the growing popularity of Amazon reviews and the increased awareness of users with regards to product reviews



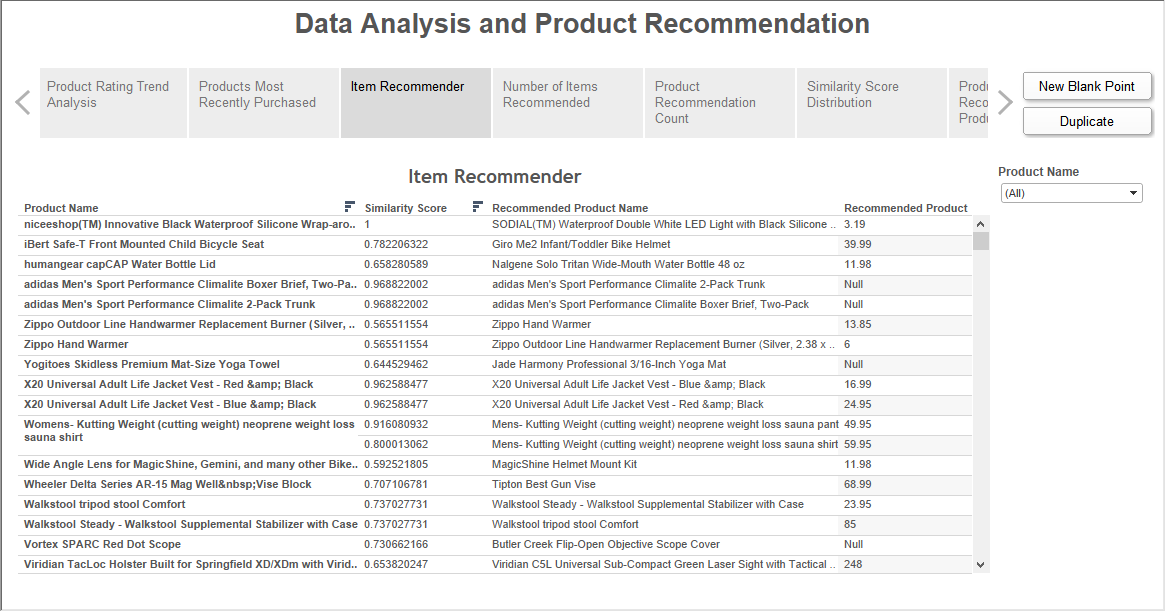
1. The following screenshot shows how the average rating of a product has changed over the years. This can give us an idea of whether the quality of a product is increasing or decreasing over the years. A similar analysis can be performed for a particular brand to determine its trend.



1. The salesrank of a product gives us an idea of its current trend in the online store and how many users are buying that product. When products are arranged by their sales rank, we know the recently trending products and that they should be highlighted on the Amazon store page.



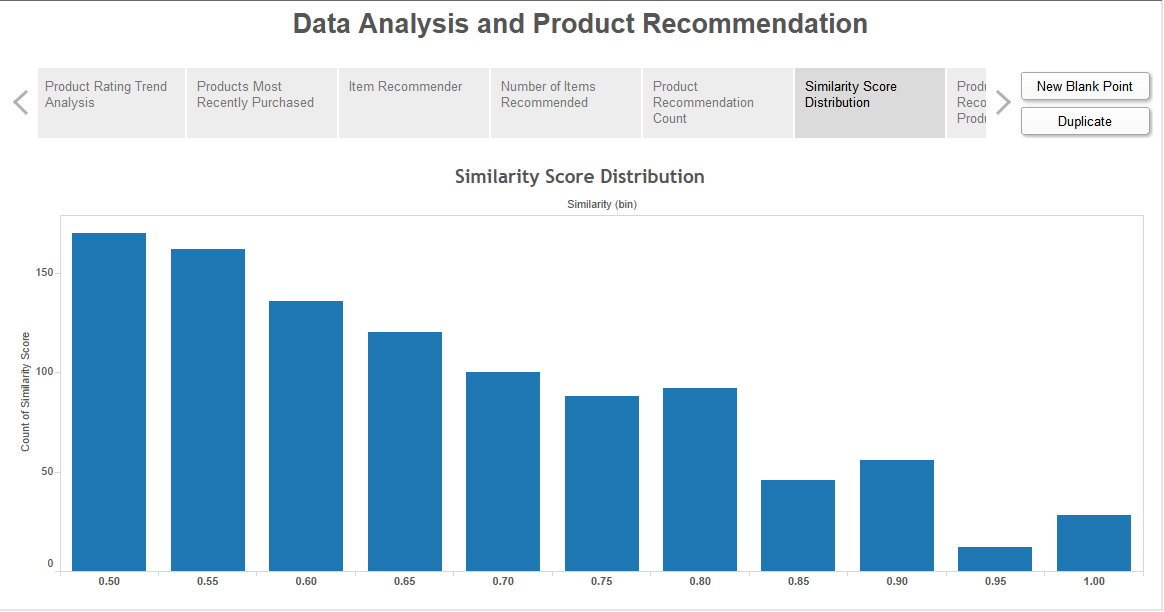
1. The following screenshot shows the item recommender similarity measures for the recommended products and the price of the recommended product. This can help us determine if the recommended product is in the price range of the product for which the recommendation is conducted. This is because, if the price for the recommended product is too large than the original product, customers are less likely to buy those.



1. The number of items recommended for a product determines how much the revenue generating potential of the product is. If a large number of items are recommended to be bought together, customers are more likely to buy more number of products from among those rather than with the product for which no recommendation is made.



1. The similarity score distribution shows the amount of similaritywith which recommendations have been made for a particular product. We see that our item recommender is recommending products with a great deal of products.



# FUTURE WORK

**Analysis across categories**

Currently we have analysed and recommended products only within a specific category. However, customers do not restrict their purchases to only one category. For example, a person buying a tennis racquet can also buy a book. Thus we need to perform analysis across the categories. This poses a challenge as we need to explore better ways of handling a large data efficiently without noticeable performance impact on our clusters and avoid very large processing time.

**Use sales rank and price for item recommendations**

Currently recommendations are made on the basis of who bought them and with which products. However, it is observed in everyday purchase behaviour of customers that most of them tend to buy cheaper products or products within a certain price range. Also most customers tend to buy currently trending products. Thus, including the sales rank and price of a product will produce greatly accurate results in our item recommendations.

**Streaming Data**

As of now we are using static data archived by Amazon and de duplicated by the organisation which provided us this dataset. In the future if possible it would be interesting to see if our application can accommodate real time stream of data. It would be interesting to explore tools like Apache Flink for this purpose.

**Comparison of Amazon item recommendations and our item recommender algorithm**

We would use the “viewed together”, “bought together”, “also bought” and “bought after viewing” data that Amazon generates by market basket analysis to compare the results returned by our algorithm against the real data. The challenges we face here are that our recommender algorithm recommends products in the same category whereas Amazon performs market basket analysis across all categories. This future goal corresponds with our future goal to perform cross category analysis.

# CITATIONS

**Image-based recommendations on styles and substitutes**  
J. McAuley, C. Targett, J. Shi, A. van den Hengel  
SIGIR, 2015

**Inferring networks of substitutable and complementary products**  
J. McAuley, R. Pandey, J. Leskovec  
Knowledge Discovery and Data Mining, 2015

# REFERENCES:

Dataset:

<http://jmcauley.ucsd.edu/data/amazon/links.html>

Python MRJob:

<https://pythonhosted.org/mrjob/>

Amazon Elastic MapReduce Documentation:

<http://aws.amazon.com/documentation/elastic-mapreduce/>

Apache Hadoop:

<https://en.wikipedia.org/wiki/Apache_Hadoop>

# ATTACHMENTS

1. **Reader\_meta.py**This script reads the review / metadata file, converts it into strict JSON format and write it is a separate file.
2. **AmazonItemRecommenderCosineSim.py**This script creates map and reduce task to calculate product recommendation based on cosine similarity of ratings of the products.

****

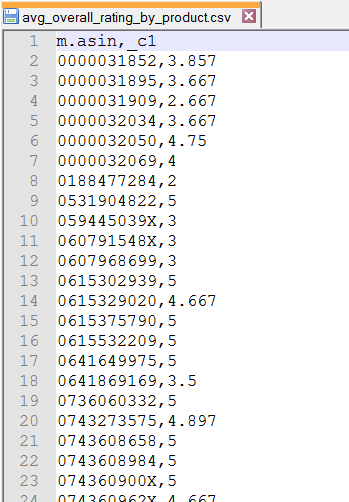
1. **AmazonItemRecommenderPearsonSim.py**This script creates map and reduce task to calculate product recommendation based on pearson similarity of ratings of the products.

****

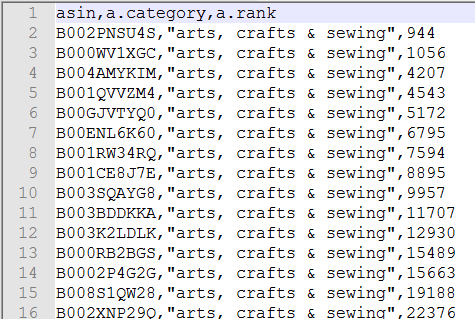
1. **Queries.txt**This file contains all the Hive queries executed to create table, retrieve certain results.

# APPENDIX

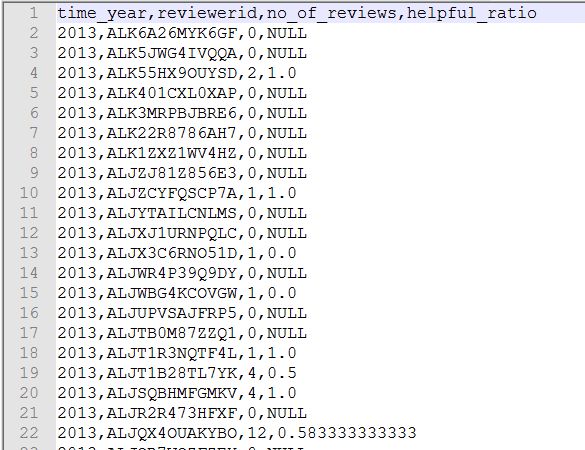
**Products sorted by their average overall rating:**

****

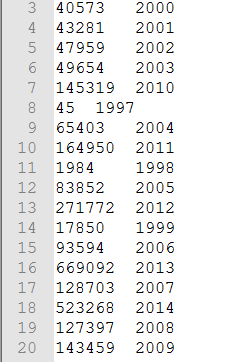
**Products, categories and Rank in the respective category:**

****

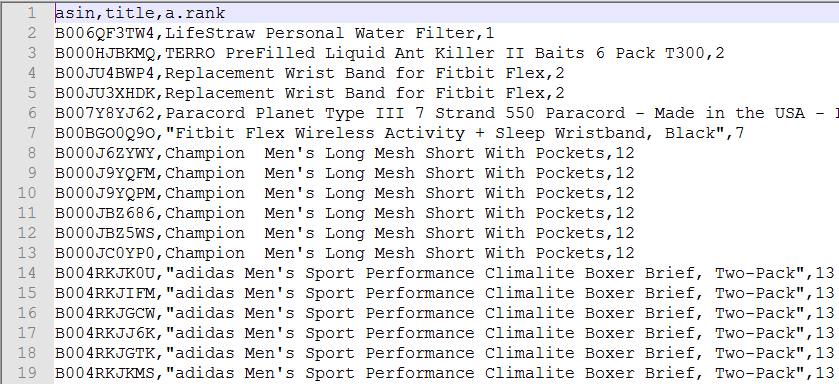
**Helpfulness ratios of reviews by reviewers and users:**

****

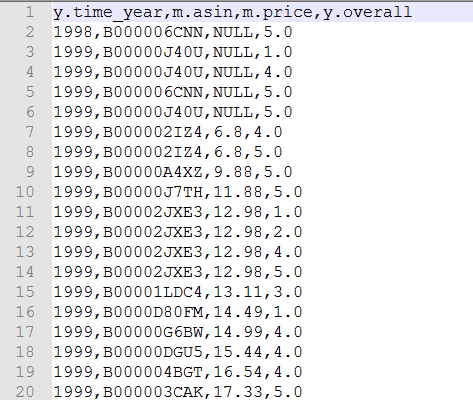
**Number of reviewers grouped by year:**

****

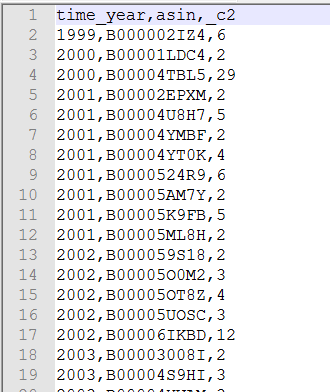
**Product, title and its sales rank:**

****

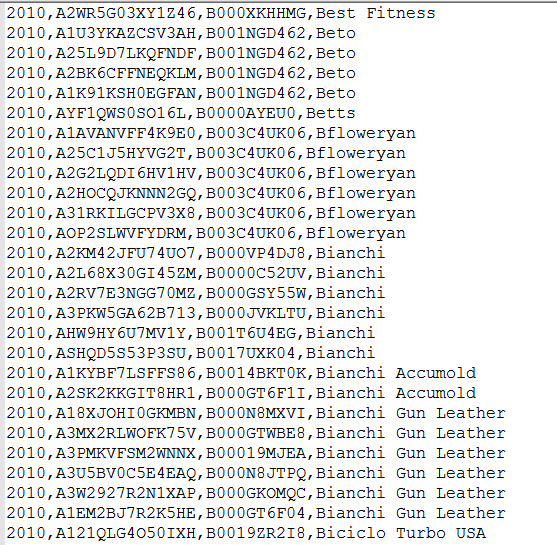
**Price of a Product and its overall rating grouped by year**

****

**Number of reviewers for each product grouped by years:**

****

**Products grouped by brand:**

****

**Retrieving the products recommended by Amazon**



