# AMAZON BOOK REVIEWS ANALYSIS

*Big Data Engineering – Shashwat Mehra*

08/18/2017

## REPORT

## TABLE OF CONTENTS

## 1. OBJECTIVES


Amazon is one of the largest online shopping websites in the world. The site is widely known for its wide selection of books, electronics, apparel, furniture, music etc.

Customers provide reviews regarding the product and the whole experience about buying products.

By developing a predictive model that accurately classifies sentiments of customers regarding a product it can help other customers in buying a product. Also, it helps sellers and other associated stake holders in understanding and improving the products and the process. Along with that we have taken a data set provided by AWS (Amazon Web Services) which has all the information regarding the book products, its respective rating and the users who gave the rating. We have done an extensive exploratory analysis on each of the datasets which has been used.

## 2. TECH. STACK USED:

| | |
|---|---|
| ▪ Apache Hadoop HDFS | ▪ MongoDB |
| ▪ MapReduce | ▪ HIVE |
| ▪ Mahout | ▪ Sentiment Analysis |
| ▪ AWS (Amazon Web Services) | ▪ Redis |
| ▪ Python | ▪ R |
| ▪ Spring MVC | ▪ Flask |
| ▪ Tableau | ▪ EMR (Elastic Map Reduce) |

## 3. WORKFLOW STEPS

- Data wrangling and cleansing.
- Upload the data in HDFS
- Create MR jobs to perform initial analysis
- Store the MR result in MongoDB and Mahout for faster access
- Created recommendation model using Mahout
- Scraped amazon website using R
- Stored scrape data in HDFS.
- Perform sentiment analysis using MapReduce.
- Visualization on the MR results.

## 4. DATASET USED:

- BX-Books.csv:

  Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (`Book-Title`, `Book-Author`, `Year-Of-Publication`, `Publisher`)

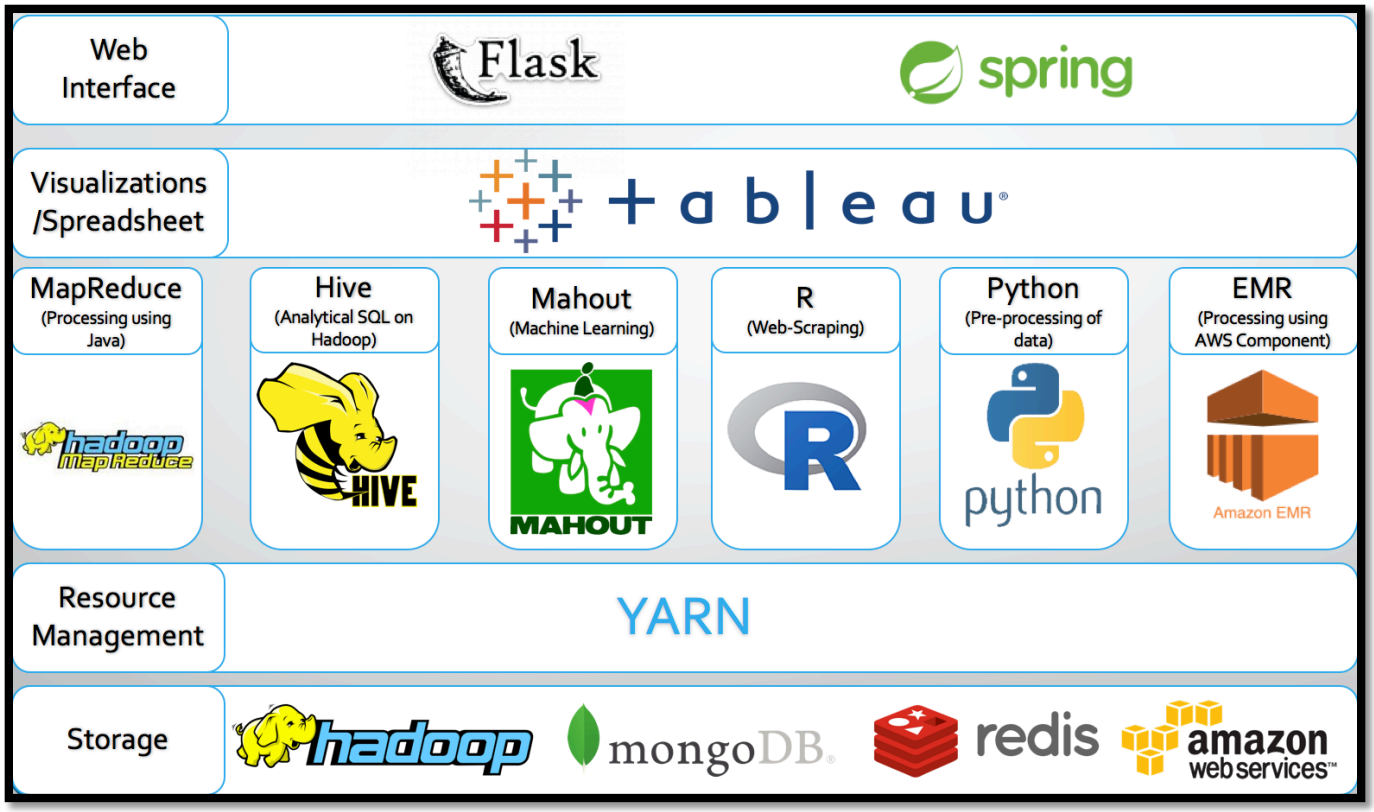
- BX-BookRatings.csv:

  Contains the book rating information. Ratings (`Book-Rating`) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0

- BX-Users.csv:

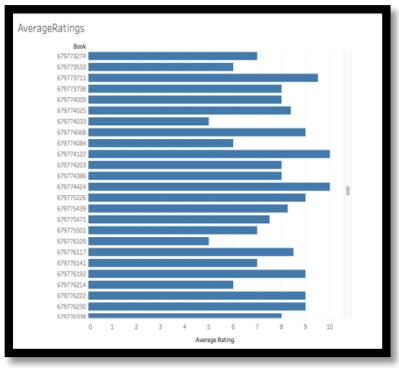
  Contains the users. Note that user IDs (`User-ID`) have been anonymized and map to integers. Demographic data is provided (`Location`, `Age`) if available.

## 5. ECOSYTEM:



# 6. MAP REDUCE ANALYSIS:

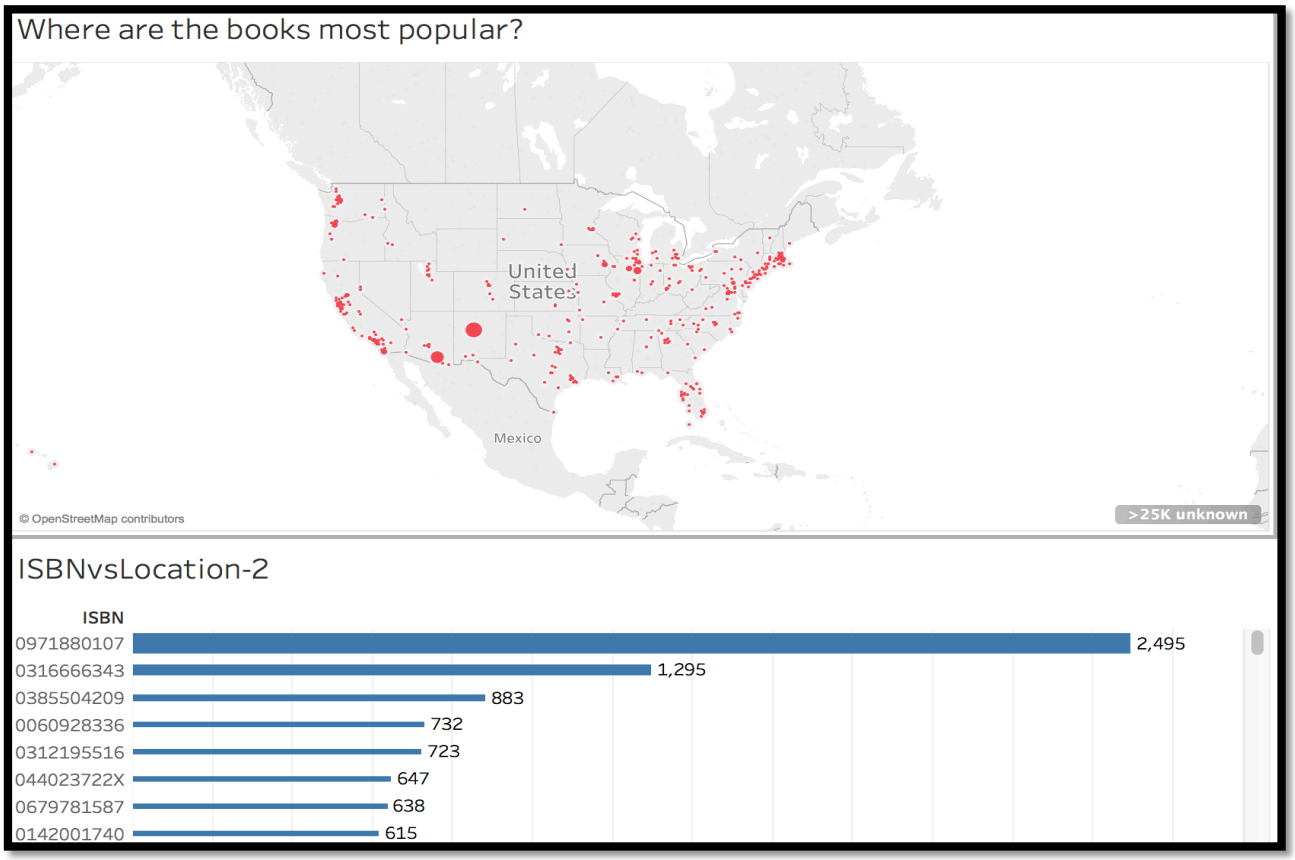
## 6.1.Map Reduce 1:




We have created first map reduce to get the average rating of all the books w.r.t it's ISBN. We have visualized the graph using Tableau. Through this graph its easy to check the average rating of all the ISBN's which can be selected by the filter option provided in the tableau workbook.

## 6.2.Map Reduce 2:

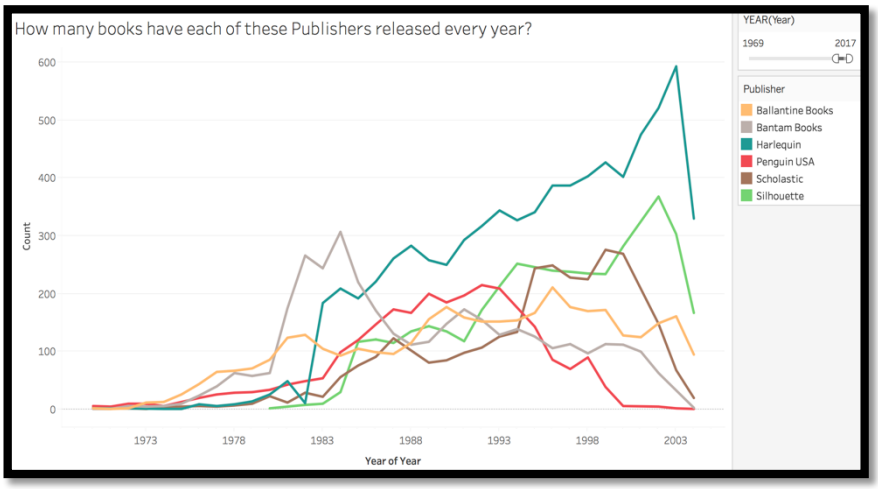
In the second map reduce we have done chaining of 2 mappers and reducers. The




objective is to get all those locations where the sales are maximum as per the ISBN.

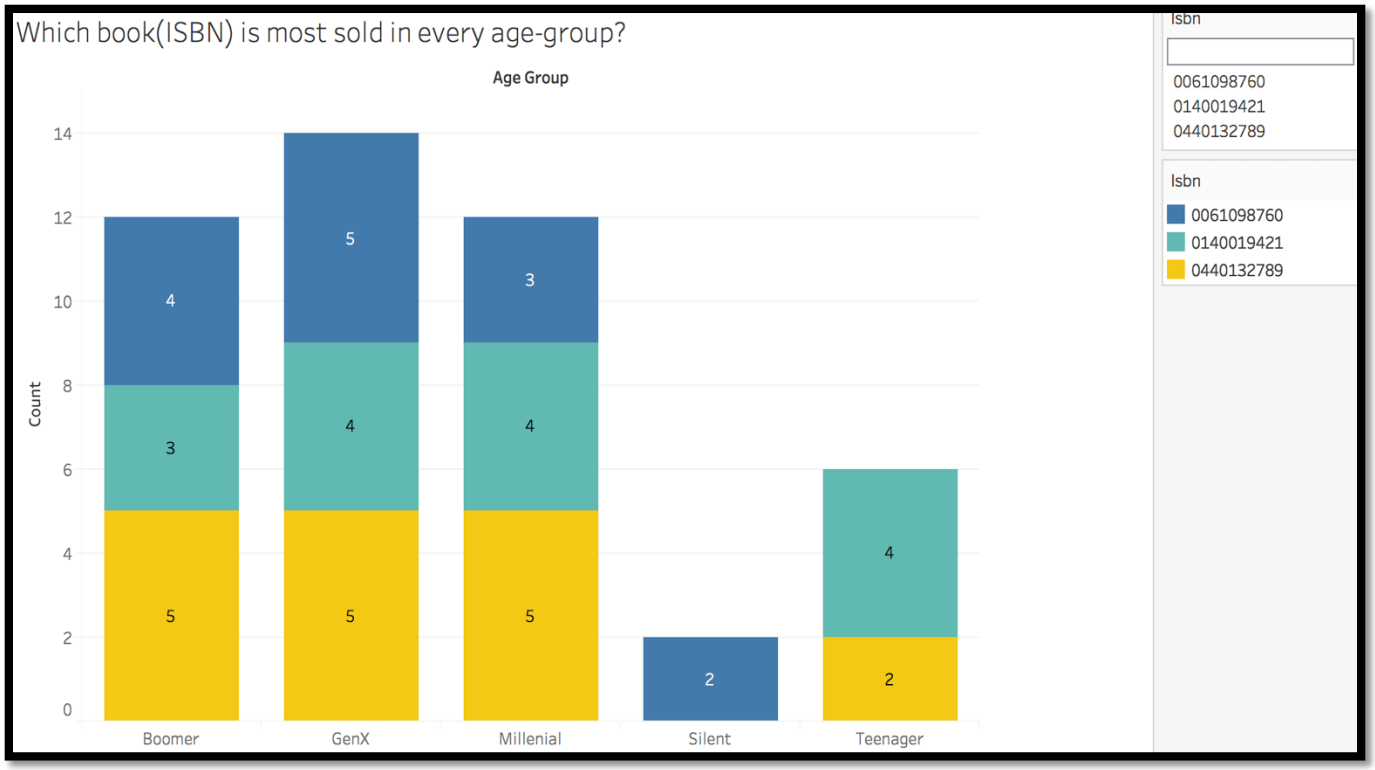
## 6.3.Map Reduce 3:

The objective of this map reduce was to find out how many publishers have published their book over the years. This will give us a time-series analysis of the publisher's performance over the year.




## 6.4.Map Reduce 4:

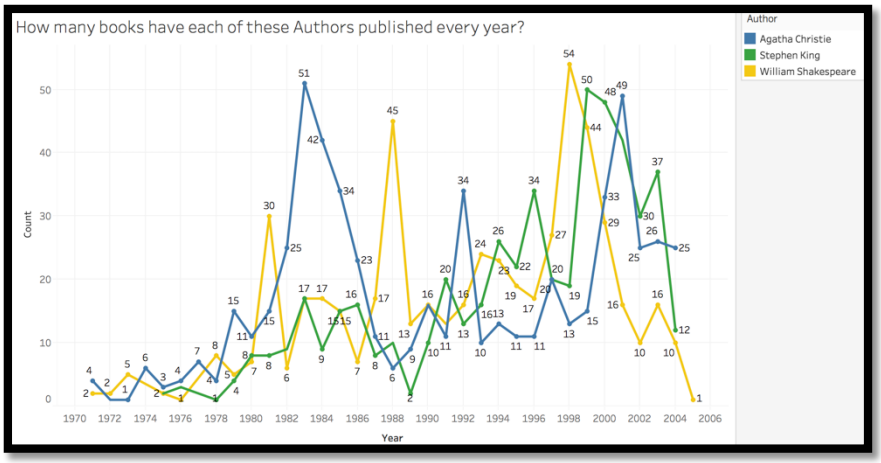
We have done a chaining for this map reduce analysis as well, because we must join




2 files book-ratings and user's info. We have divided the age-groups of each users as Teenage, Boomer, Gen-X, Millennial and Silent. This will help us decide which all books are famous as per the age group.

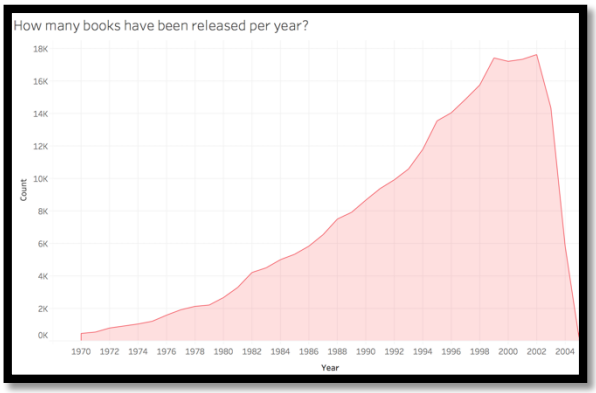
### 6.5.Map Reduce 5:

The objective of this map reduce was to find out how many authors have published their book over the years. This will give us a time-series analysis of the author's performance over the year.




### 6.6.Map Reduce 6:

We have calculated number of books published each year thus, giving us a count of all the books published every year.




### 6.7.Map Reduce 7:

We have calculated the year which has the highest number of books published.

## 7. HIVE ANALYSIS:

We have executed some queries in hive as well to see the output and how much time it takes to execute the same queries compared to map reduce algorithm.

### 7.1.Query 1:

To find the count of book released per year.

```
hive> SELECT YearOfPublication, COUNT(BookTitle) FROM BookData GROUP BY YearOfPublication;
```

We have stored the data in Book-Data table and described each field as string, except the year of publication which is integer.

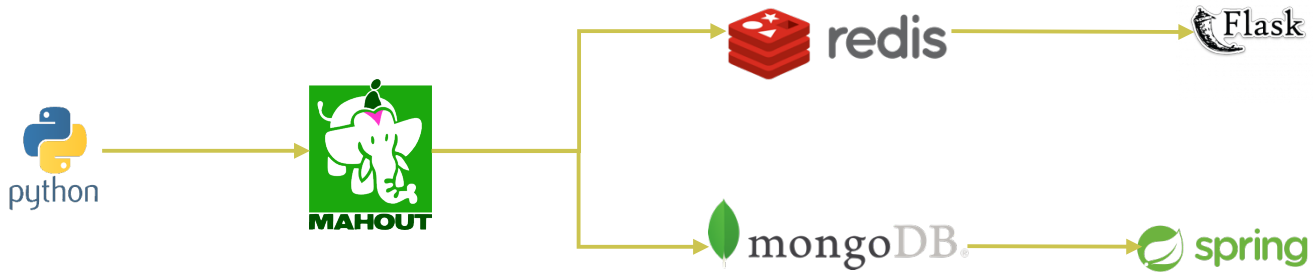
```
hive> describe BookData;
OK
isbn                    string
booktitle               string
bookauthor              string
yearofpublication       int
publisher               string
```

### 7.2.Query 2:

In the second query, we are getting books published each year by Author.

```
hive> SELECT Publisher, BookAuthor, YearOfPublication, COUNT(BookTitle) FROM BookData GROUP BY Publisher, BookAuthor, YearOfPublication;
```

## 8. MAHOUT RECOMMENDATION:



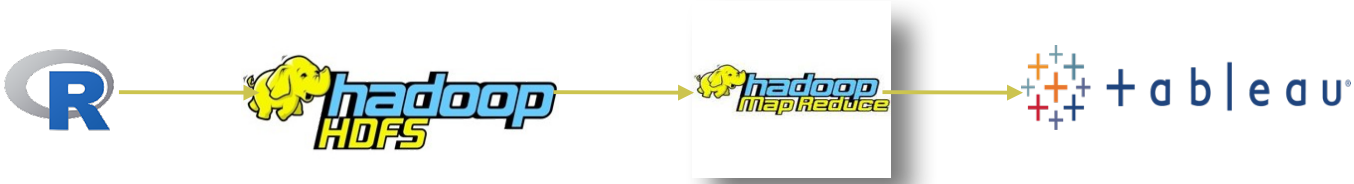

To provide the best mahout recommendation for the data, we have to first change the data as mahout accepts only integer values and since ISBN and TITLE are strings we have created a dictionary/csv file which has indexed value from 1 to n for Book ISBN - Book Title.

***Sequence for executing python script:***

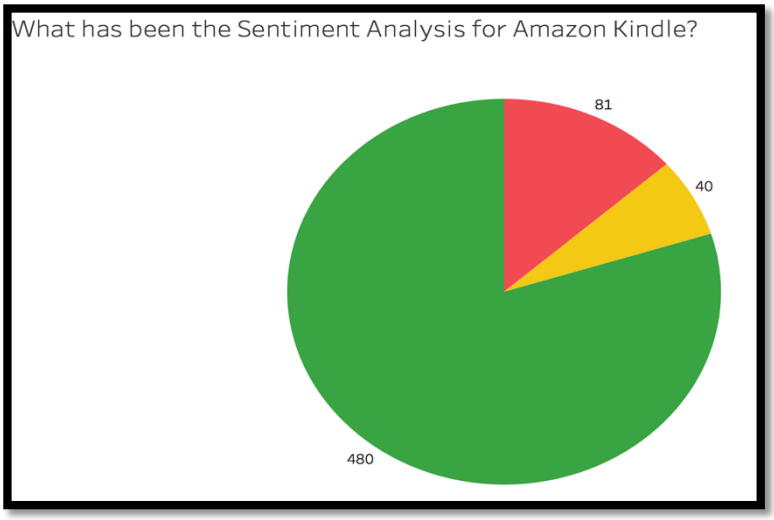
- Run BookTitle.py, this will create a dictionary for ISBN and title, with ISBN as keys and title as values
- Run BookMapping.py, this will create 2 csv files, first csv file has isbn, bookTitle and indexes, second csv file has user, index, rating. The indexes refer to the ISBN and Book Title from the first csv
- Start HDFS and run the following mahout command:
- mahout recommenditembased --input /New-Book-Ratings.csv --output recommendations --numRecommendations 10 --similarityClassname SIMILARITY_COSINE
- After the command is executed a new success file under /user/user_name/recommendations/part**** is created which has all the recommendations.
- Run Redis server - redis-4.0.1/src/redis-server
- Open Redis CLI - Check if it's working
- Run Mongo Daemon <mongod> to instantiate the server.
- Run redisMapping.py which inserts the recommendations from HDFS into REDIS database.
- Run mongoMapping.py which inserts the recommendations from mongoDB into Redis database.

- Run webServer.py to initiate flask, and you can test by passing the User ID as input parameters to check the output. (http://localhost:5000/user/242)
- Run the MVC application on local host and enter the User ID to view its details and recommendations.

## 9. SENTIMENT ANALYSIS




We have performed sentiment analysis using R scraping from amazon website and storing it into Hadoop HDFS. Once the data is loaded in Hadoop a map reduce job is then execute to perform sentiment analysis. The data that we receive is stored in csv format which is then pushed to Hadoop-dfs environment, in R script when we enter a product id it scrapes all the reviews from the amazon webpage for that product id. Sentiment Analysis is done by removing stop words from the reviews, and then comparing each word with the dictionary which has key value pairs of all good/bad words and their respective score. Once the score is calculated, a second map reduce job is executed which scores the reviews into positive, negative and neutral, which helps us to get the overall sentiment analysis of the reviews related to the product id initially entered.




## 10. LINK TO GITHUB:

https://github.com/Shashwat21/Big-Data-FInals