Business Intelligence using Product Star Rating & Customer's Reviews

Implementation of Recommender Systems and NLP on Amazon Dataset

(Bhuvan Chopra, Fatemeh Rostamzadeh Renani, Junaid Qazi)

Abstract:

In this project, we implemented user and product recommendations system using Amazon data of Sports and Outdoors category with 3 million reviews. We compared the results of 7 machine learning algorithms used for collaborative filtering. We also classified reviews in categories of Good and Excellent by implementing NLP on text of reviews.

Data Modeling and Loading:

We got the reviews data from this public repository. For metadata, we had to request for access to the concerned person of the repository. The data is semi-structured in the form of JSON files. The 2 files were first loaded in HDFS using the 'scp' command. Then using Spark, the data was loaded in Cassandra. Fig. 1 shows what the reviews data contains.

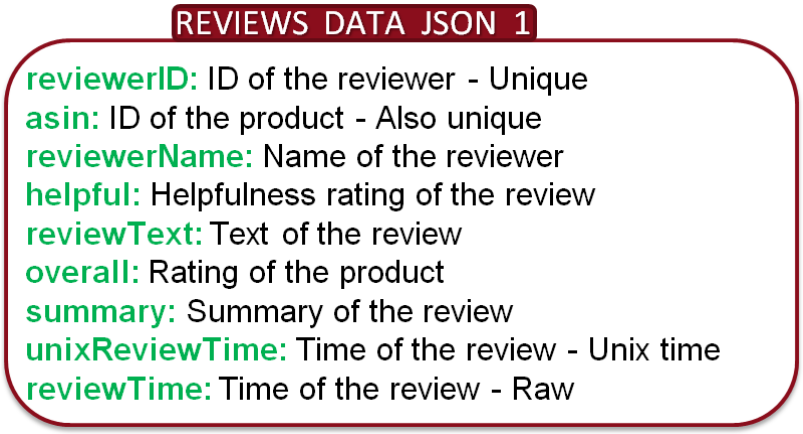


Fig. 1: Columns of reviews dataset of Sports and Outdoors category.

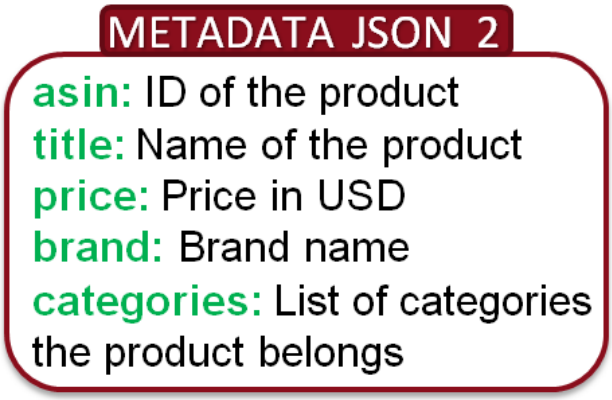


Fig. 2: Selected columns of metadata loaded in Cassandra.

The raw data was loaded into the Cassandra database with composite PRIMARY KEY as (asin, 'reviewerID'). The partitioning key is set as 'asin' so that efficient read happens while collecting reviews of a particular product. The metadata was also loaded with selected columns (shown in Fig. 2) in case we wanted to categorize our recommendations based on product category. This data was loaded with PRIMARY KEY as (asin). The null values in 'asin' column were dropped as the primary key cannot have NaN.

Tool(s)/Technique(s) used in this step and why: We had the option of using MongoDB, but as all of us were familiar and comfortable working with Cassandra, we chose it as our database in this project.

Data cleaning and analysis: As expected, the raw data needed cleaning, which involved handling the missing data, feature engineering, selection of useful features, renaming the columns for better understanding and so on. We created a smaller json file from Cassandra as a sample data to understand the data wrangling and to find the trend in the data. The code and all the steps for data cleaning is provided as jupyter notebook titled: "DataMining" in the git repository. Briefly, Fig. 3 shows the percentage of missing data in each selected column whereas on the right, Fig. 4 is the summary of the dataset after cleaning and feature engineering. After data wrangling, a file, "***data_clean.csv***" was created for further work. The file is provided in the git repository.

Columns	missing data %	unique values
asin	0.000	14232
brand	49.033	2026
categories	0.000	2824
helpful	0.000	894
overall	0.000	5
price	31.231	3061
reviewText	0.000	99883
reviewTime	0.000	3125
reviewerID	0.000	95891
reviewerName	0.363	81233
summary	0.000	74919
title	1.092	14141
unixReviewTime	0.000	3125

Fig. 3: Amount of missing data in each column.

After Cleaning, feature engineering etc

```
Data columns (total 14 columns):
asin                98545 non-null object
categories          98545 non-null object
reviewText          98545 non-null object
reviewTime          98545 non-null object
reviewerID          98545 non-null object
reviewerName        98545 non-null object
summary             98545 non-null object
title               98545 non-null object
unixReviewTime      98545 non-null object
rating              98545 non-null int64
help                98545 non-null int64
rev_len             98545 non-null int64
reviewerID_code     98545 non-null int32
asin_code           98545 non-null int16
```

Blue arrows points to the new columns

Fig. 4: New columns created for analysis.

Data Visualization: Range of plots were created to understand the trends in the data however, few useful plots are provided below:

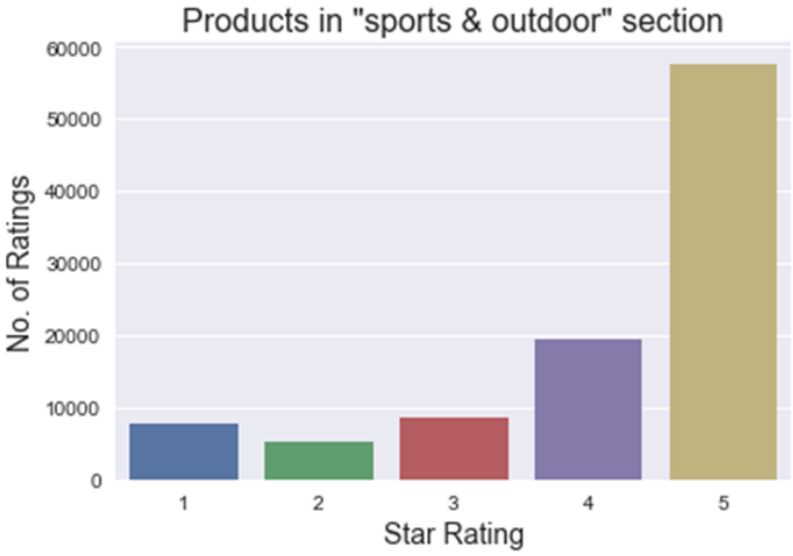


Fig. 5: Number of ratings on y-axis and star rating on x-axis.

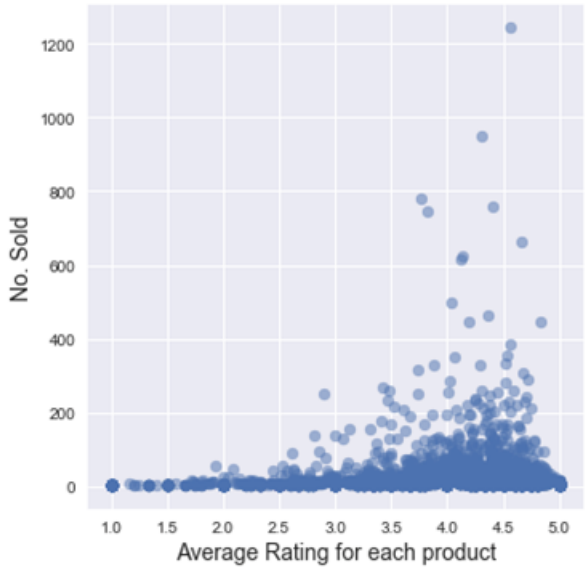


Fig. 6: Number of sold products on y-axis and average rating for each product on x-axis.

In Fig. 5. we clearly see that most of the customers are happy with the products in "Sports & Outdoors" section as the number of 4- and 5-star ratings are much higher than others. This suggest that the sellers are maintaining the quality of the products and over all the department is performing according to the customer's expectations. Fig. 6 shows that customers consider average rating of the product, higher the average rating is, higher the sold quantity is which is obvious. Another very interesting fact that we observed was the relationship between the length of the review and the star rating. We found that the happy buyers don't leave longer reviews, rather they are brief while posting reviews (Fig. 7).

Tool(s)/technique(s) used in this step and why: Indeed, it was a steep learning in a very short amount of time. We explored range of available technologies including **Spark** for ETL, **Cassandra** database, **ALS – Spark** for recommender systems, **Surprise** – a python library for recommender systems, **NLTK for NLP**, **scikit-learn** for machine learning, **pandas & numpy** for data wrangling, **matplotlib & seaborn** for data visualizations. We also **created pipelines** to compare the performance of three models in NLP code.

NLP – predicting star rating from reviews: NLP involves significant data preprocessing and preparation for machine learning algorithms such as tokenization, bag of word, vectorization and TF-IDF. This was an excellent experience to explore and learn all of these steps. Although, we started with predicting all-star ratings however, to make it simple a binary classification problem, we classified the ratings into two classes, 'Good' (for rating 1,2,3) and 'Excellent' (for rating 4,5). The code "NLP_Reviews.py" is provided with detailed comments in the git repository and the results are shown in Fig. 8 below.

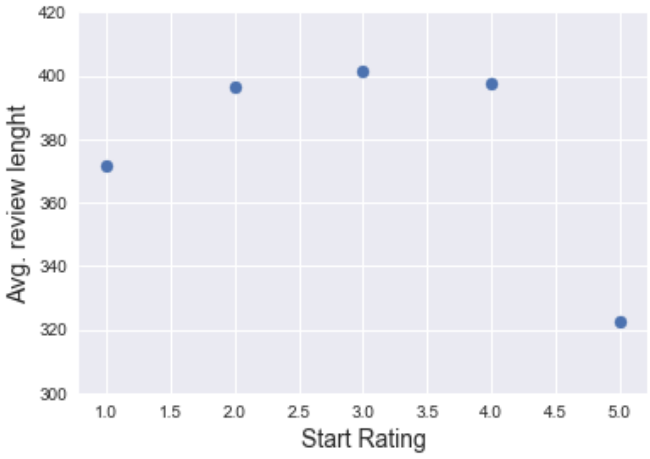


Fig. 7: Average review length on y-axis and star rating on x-axis.

MultinomialNB_model

	precision	recall	f1-score	support
Excellent	0.78	1.00	0.88	25415
Good	0.86	0.01	0.02	7091

LogisticRegression_model

	precision	recall	f1-score	support
Excellent	0.88	0.97	0.92	25415
Good	0.81	0.53	0.64	7091

RandomForest_model

	precision	recall	f1-score	support
Excellent	0.82	0.99	0.89	25415
Good	0.79	0.20	0.32	7091

Fig. 8: Performance comparison of the models used for NLP.

In Fig. 8, we can see the effects of unbalancing in the data from the results, specially the recall column. This is expected as we have fewer number of 1,2 & 3 stars in Good class whereas for 'Excellent" class we have much more data entries (Ref: Fig. 3).

Recommender Systems: Recommender systems are a family of methods that filters through large information (what the users have experienced/observed/bought/rated) and provides recommendations to users for which the user has not experienced. 35% of Amazon's revenue and about 70% of everything users watch on Netflix are generated by their recommendation engine. By narrowing down the selection options for the customer, it is more likely that they make a purchase. Many businesses greatly benefit from such recommender systems. In this project we focus on the Collaborative filtering algorithm.

In order to recommend appropriate products to customers based on the customer's rating, we have implemented the recommended system using two technologies. We have investigated and learned how recommender systems work in Spark using Collaborative Filtering. We also used the Surprise Library (a Python scikit for recommender systems) and found that each approach has its own advantages and requirements.

Alternating Least Squares (ALS) in Spark:

The main idea is to predict whether a customer would like a certain product (an item, a movie, or a song) to find the targeted customers for the product. The computational complexity increases with the size of a company's customer base and products. Therefore, the scalability is an important concern. This challenge can be addressed using Apache Spark MLlib which enables us to build recommendation models for billions of records. Spark MLlib implements a collaborative filtering algorithm called Alternating Least Squares (ALS). ALS models the user-item association matrix (R) as the product of two matrices U and V. Where U is a [u by f] matrix (users and hidden features) and V is a [f by i] matrix (hidden features and items). The idea is to learn these two matrices by minimizing the

reconstruction error of the observed ratings. ALS implementation in spark.ml has the following parameters:

NumBlocks is the number of blocks the users and items will be partitioned into in order to parallelize computation (defaults to 10). **Rank** is the number of latent factors in the model (defaults to 10). **MaxIter** is the maximum number of iterations to run (defaults to 10). **RegParam** specifies the regularization parameter in ALS (defaults to 1.0). **ImplicitPrefs** specifies whether to use the explicit feedback ALS variant or one adapted for implicit feedback data (defaults to false which means using explicit feedback). **Alpha** is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations (defaults to 1.0). **Nonnegative** specifies whether or not to use nonnegative constraints for least squares (defaults to false).

W ...Param
p ...at we
le

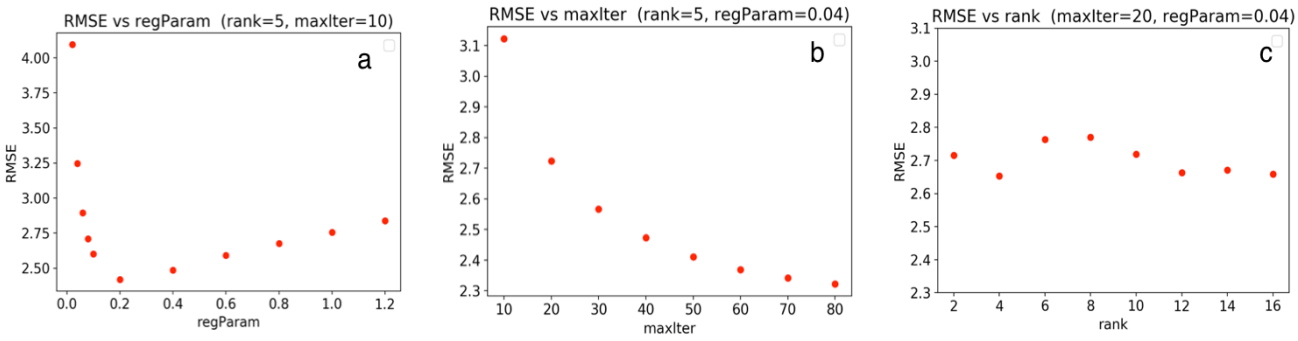


Fig. 9: Optimization of RMSE based on a) regParameter regularization parameter, b) maxIter maximum iteration, c) rank number of hidden features.

Key findings:

1. The "regParam" is the regularization parameter. Fig. 9a shows the grid-search actually found the parameters that minimize the error.

2. Fig. 9b shows that the accuracy gets better as the number of iteration increase (agrees with our intuition).However, Spark was not able to perform well for large "maxIter" values when we used our whole dataset. Thus we have used the maxIter default value which is 10.

3. As represented in Fig. 9c the "rank" does not change the accuracy of the predictions. Therefore, we select the default value.

4. We have found that setting nonnegative= True (which allows us to have negative constraints for least squares) improved the prediction, whereas the default value for this parameter is False.

Using Python's Surprise library: This library provides many predictive algorithms such as baseline algorithms, neighborhood methods and matrix factorization based algorithms such as SVD, SVD++.

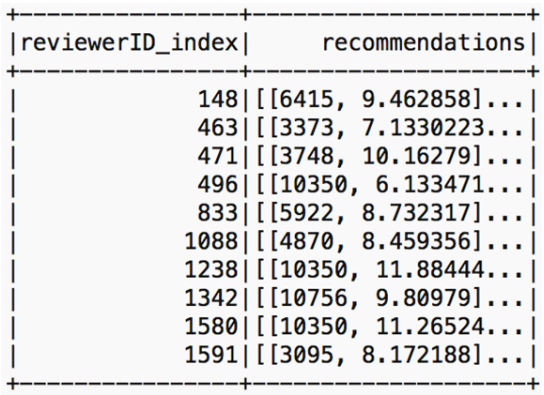
SVD and SVD++ are based on the matrix factorization-based method and produce better accuracy for our recommender system. Thus we investigated the recommender system using SVD++ in more detail to fine-tune its parameters. Here are the parameters of SVD++:

n_ephocs: The number of iteration of the SGD procedure. Default is 20.

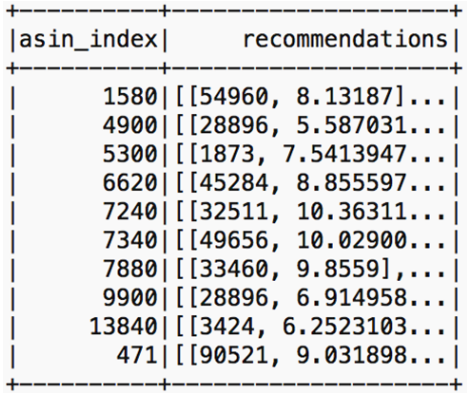
lr_all: The learning rate for all parameters. Default is 0.007.

reg_all: The regularization term for all parameters. Default is 0.02.

We got the **best RMSE (validation) = 2.512477** for ALS model trained with rank = 10, lambda = 0.04, and numIter = 30. Fig. 10a and 10b show the output of ALS model.



reviewerID_index	recommendations
148	[[6415, 9.462858]...
463	[[3373, 7.1330223...
471	[[3748, 10.16279]...
496	[[10350, 6.133471...
833	[[5922, 8.732317]...
1088	[[4870, 8.459356]...
1238	[[10350, 11.88444...
1342	[[10756, 9.80979]...
1580	[[10350, 11.26524...
1591	[[3095, 8.172188]...



asin_index	recommendations
1580	[[54960, 8.13187]...
4900	[[28896, 5.587031...
5300	[[1873, 7.5413947...
6620	[[45284, 8.855597...
7240	[[32511, 10.36311...
7340	[[49656, 10.02900...
7880	[[33460, 9.8559],...
9900	[[28896, 6.914958...
13840	[[3424, 6.2523103...
471	[[90521, 9.031898...

Algorithm	RMSE
SVD++	1.1931
SVD	1.1950
BaselineOnly	1.1996
KNNBaseline	1.1995
KNNBasic	1.2551
KNNWithMeans	1.2562
NMF	1.2733

Fig. 10a: Recommendations for each user. Fig. 10b: Same for each product. Table 1: Comparing RMSEs.

We found RMSE= 0.8793 implementing SVD++ where the parameters of the estimator are optimized by cross-validated grid-search over a parameter grid (n_epochs=15, lr_all= 0.009, reg_all= 0.06). We also found that our code in Python (using Surprise library) provides more accurate predictions however it is slower than spark. We were able to benchmark the RMSE for various algorithms in Surprise with their default parameters (see Table 1).

Learning and Problems faced in the project: Spark collaborative filtering (ALS) algorithm currently only supports integers for user and item ids. In our original dataset, these features are recorded as strings (combination of the numeric and alphabet). Thus, in order to perform the CF in spark we have to convert the review Id and item Id to unique integer IDs. We completed this task with two methods. First, we used “pandas” library for this conversion. Later we realized that this task can be done in spark using stringindexer module in spark ml. There were several other challenges while the implementation of project including learning NLTK, Surprise, and all the data processing and data visualization tools. However, we successfully learned all the required tools & libraries, write codes to implement different algorithms, performed grid-search for optimization, compared range of models for both NLP and recommender systems.

Citations:

1. Data - Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, R. He, J. McAuley

2. NLTK - https://www.nltk.org/

Project Summary:

In this project, we have successfully implemented a Business Intelligence solution using Amazon dataset for a single category/department. The overall performance of the sellers was observed to be above average and they work to maintain the quality of their product which makes Amazon one of the leading e-marketplace for both customers and sellers. To facilitate both buying and selling processes, our model suggests relevant products to the buyers based on their search. We have successfully explored the correlation between reviews and the star rating that can improve the e-marketplace experience for both the sellers and buyers.

Getting the data: (2 Marks)

The data was not publicly available. We requested a special link to access the data from Julia McAuley, who is the owner of the data portal. A special thanks to him. The portal range of datafiles for variety of products. However, after exploring for a while, we decided to work with the selected dataset.

ETL: (2 Marks)

ETL was performed using Spark into Cassandra database.

Problem: (1 Mark)

Business intelligence is one of the key areas of operational research. Because of the available technologies and interest of the customers to buy online, it is very important to understand the behavior for both, selling and buying processes. The star rating and reviews are two very common ways to lean about the performance of any product, department and platform's performance. People are seen to be expressive in giving reviews and telling their experience for e-purchases. Hence, this is very important to conduct the in-depth analysis using advance technologies to improve such experiences.

Algorithmic work: (5 Marks)

ALS & Surprise to implement the recommender systems

NLTK and scikit-learn to do Natural Language Processing and implementing Multinomial Naive Bayes, Logistic Regression and Random Forests machine learning algorithms.

Bigness/parallelization: (3)

Implemented parallelism and Biggness.

UI: (0 Mark)

Visualization: (2 Marks)

We created range of plots to understand the trends in the data, which is key step in any project. However, it is not trivial to visualize the larger/bigger datasets. We created smaller sample of our dataset to do most of the visualization part.

Technologies: (5 Marks)

Spark for ETL, **Cassandra** database, **ALS – Spark** for recommender systems, **Surprise** – a python library for recommender systems, **NLTK for NLP**, **scikit-learn** for machine learning, **pandas & numpy** for data wrangling, **matplotlib & seaborn** for data visualizations. We also **created pipelines** to compare the performance of three models in NLP code.