UNIVERSITÀ
DI PAVIA

From Raw Data to Informed Decisions: Analyzing Amazon Book Reviews

Alberti A.,Ligari D., Andreoli A.[1]

[1] *Data Science and Big data Analytics course, University of Pavia, Department of Computer Engineering (Data Science), Pavia, Italy*
Github page: https://github.com/DavideLigari01/data-science-project

Date: September 18, 2023

Abstract —prova

Keywords —Big Data • Hadoop • Spark • ML • MongoDB • Data Analysis • Data Visualization • Python

Contents

1 Introduction 1
2 Discovery 1
a Team 1
b Framing 1
c Tools 1
3 Data Preparation 2
4 Local Hypotheses Testing 3
5 Spark Hypotheses Testing 4
6 Helpfulness Prediction 4
7 Conclusions 4
8 Python Script 4

1. Introduction

In the age of digital commerce, customer reviews play a pivotal role in shaping product perception and influencing purchasing decisions. With the proliferation of online bookstores, Amazon has amassed an immense repository of book reviews spanning nearly two decades. These reviews contain valuable insights, sentiments, and trends that can unlock a treasure trove of information for authors, publishers, and book enthusiasts. This project embarks on a journey to harness the power of data, employing a comprehensive workflow to dissect and understand the vast collection of Amazon Books Reviews. Our mission is to develop a scalable solution that allow us to discover patterns, sentiment trends, and hidden correlations within the world of book reviews. We leverage cutting-edge tools and technologies, including Hadoop, Spark, MongoDB, and Python libraries such as Pandas and Scikit-learn. In this report, we embark on a detailed exploration of our project, delving into each stage of our workflow, from initial data discovery and preparation to feature extraction, model building, and rigorous evaluation.

2. Discovery

To begin our data science campaign was crucial to compose the team, clearly fix our goal and exploring about the available resources.

a. Team

The team is composed by three members:
Andrea Alberti: github.com/AndreaAlberti07
Davide Ligari: github.com/DavideLigari01
Cristian Andreoli: github.com/CristianAndreoli94

b. Framing

The **goal** of the project is to develop a **scalable** solution to analyze a dataset of reviews of books from Amazon and eventually build a model able to predict the helpfulness of a review based on its content.

c. Tools

The tools were chosen with the aim of developing a scalable solution, able to work in a **Big Data** environment.

- **Virtual Machine**: to work in a controlled environment.
- **Hadoop**: to store data in a distributed file system and to perform MapReduce operation on it.
- **Spark**: used as an improved alternative to MapReduce, to perform operations on the distributed dataset.
- **Python**: as the reference programming language thanks to its libraries.
- **MongoDB**: used as NoSQL database sandbox to safely work with local data.
- **GitHub**: to share and collaborate on the project.
- **LaTeX**: to write the report.

3. Data Preparation

To start with the project, we needed to retrieve and prepare the data.

Data Retrieval and Prior Analysis

The chosen dataset is composed by 2 tables and about 3 million reviews, accessible at the following link: Amazon Books Reviews.
Once the dataset has been acquired we performed the following steps:
1. **HDFS loading** using the following commands:

```
# Create HDFS directories
hdfs dfs -mkdir -p "$HDFS_PATH/ratings"
hdfs dfs -mkdir -p "$HDFS_PATH/
    books_info"

# Copy local files to HDFS
hdfs dfs -copyFromLocal "$LOCAL_PATH/
    ratings.csv" "$HDFS_PATH/ratings/"
hdfs dfs -copyFromLocal "$LOCAL_PATH/
    books_info.csv" "$HDFS_PATH/
    books_info/"
```

2. **Prior analysis** using *pyspark* to get a better understanding of the data. In this phase we defined a schema for our data and we computed some statistics together with the percentage of missing values and unique values for each field or our dataset.

Hypothesis Generation

After the prior analysis, we started to think about some hypotheses that we could test on our data. We came up with the following ones:

- **H1**: Reviews with longer text have higher helpfulness ratings.
- **H2**: Reviews with more positive sentiment words receive higher helpfulness ratings.
- **H3**: Reviews with higher book ratings have higher helpfulness ratings.
- **H4**: The rating score is influenced by individual users, which may either overestimate or underestimate a book's quality. Anonymous tends to overrate the books.
- **H5**: The review score is influenced by the category of the book.
- **H6**: The larger the number of books published for a category or publisher, the higher the review score.

Data Cleaning

In this step we cleaned the data, removing duplicates, useless columns for our analysis and any symbol that could have interfered with the reading of the csv files. All the cleaning operations were performed using *pyspark*.

Data aggregation

The MapReduce job was created to perform the inner join operation on the "Data table" and the "Rating table" based on the title. The output of the MapReduce job is a single file containing the joined records from both tables.

Mapper

The Mapper script processes the input data line by line, where each line represents a distinct record. It transforms these lines into a key-value structure, where the key corresponds to the book title, and the value contains the remaining content of the line.
Given that the Mapper deals with data from two distinct sources, it becomes crucial to distinguish between records belonging to the 'Data table' and those in the 'Rating table'. This distinction is essential because it mandates a specific order of processing records from the 'Data' table need to be joined with corresponding records from the 'Rating' table in the Reducer phase. Consequently, the Reducer should process 'Data' table records before 'Rating' table records. To ensure this orderly processing, the Mapper augments the key with a special character for each table type. Specifically, it appends a hyphen ('-') as the second key element for records from the 'Data table' and 'www' for records from the 'Rating table.' By doing so, and thanks to Hadoop's sorting task made after, the Mapper guarantees that 'Data table' records are encountered and processed prior to 'Rating table' records during the subsequent phases of MapReduce.

Reducer

The Reducer script is responsible for processing the intermediate output records generated by the Mapper. Its primary role is to perform the join operation between the 'Data' table and the 'Rating' table, taking advantage of the pre-sorting of records by title. During its execution, the Reducer reads the records in a sequential order. As it encounters a record from the 'Data' table, it stores the information in one variable. Conversely, when it comes across a record from the 'Rating' table, it stores that information in another variable. Once both 'Data' and 'Rating' records for the same title are available, the Reducer performs the join operation by combining the data from these records.

MongoDB loading

Once all the previous operations were completed, we loaded the data into MongoDB. To do so the steps executed are:

- Connect to MongoDB using *'pymongo'*
- Connect to HDFS and read the data using *'spark.read.csv'*
- Select a subset of the Spark DataFrame to import using *'sample'* method
- Transform the data into a dictionary using *'to_dict'* method
- Insert the data into MongoDB using *'insert_many'* method

We loaded in MongoDB the two tables *ratings* and *books_info*, together with the joined table produced by MapReduce. These data have been used to perform the local hypotheses testing described below.

4. Local Hypotheses Testing

Hypothesis 1

Reviews with longer text have higher helpfulness ratings.

Description and Results The data cleaning and 'review/helpfulness' transformation process ($helpfulness = \frac{x}{y}\sqrt(y))$ was performed using the *'pymongo'* library to exploit the MongoDB efficiency. Specifically, we defined a pipeline to perform the needed operations. As regards the 'review/text' transformation, we used the *'nltk'* library to tokenize the text, remove punctuation, stopwords and eventually count the number of words.
The correlation coefficient between the two variables is 0.3313 with a p-value < 0.05, indicating a statistically significant correlation.
A graphical confirmation is provided by Figure 1. Indeed there is a positive correlation until around 400 words, after which the boxplot stabilizes. Thereby, we decided to analyze the correlation within the review length groups. As a results (Table 4) we got that the correlation is positive and statistically significant for the reviews with length between 0 and 400 words, while it becomes negative and statistically significant for the reviews with length greater than 750 words. As regards the reviews in the middle (i.e. between 400 and 750 words), the correlation is negligible.
Conclusion: The hypothesis is confirmed, but the correlation is not very strong and changes depending on the review length.

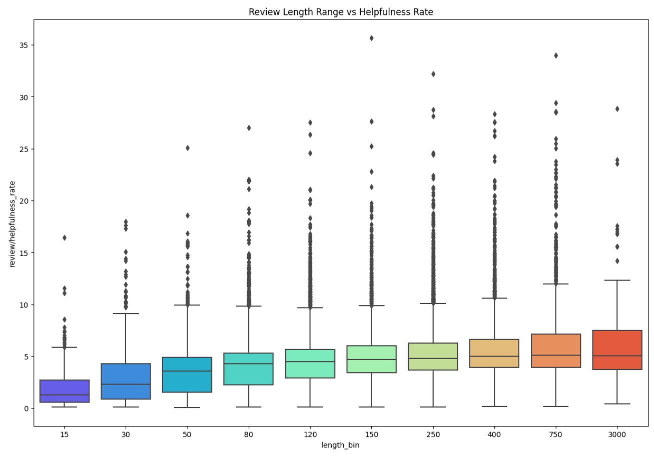

Fig. 1: Correlation between review length and helpfulness score for different review length groups

Table 1: Correlation Coefficients and P-values for Different Groups

Group Number	Correlation Coefficient	P-value
400	0.2216	0.0000
750	-0.0188	0.2585
3000	-0.1418	0.0065

Hypothesis 2

Hypothesis 3

Reviews with higher book rating have higher helpfulness ratings.

Description and Results As in the previous hypothesis, we decided to deal with missing values and data transformation directly with a MongoDB query. With the data ready to be processed, we decided to perform a prior investigation on the distribution of votes on the 4 rating categories. As shown by Figure 2, there is a **positive bias** as people are more prone to vote a positive reviews rather than a negative one. Specifically, the large majority of votes for rating 5, were composed by a number of total votes just equal to 1. This introduces a bias in our results, since according to the formula used to compute the helpfulness score, such a small number of total votes would lead to a small helpfulness score. To avoid this we decided to retain only the reviews with a number of total votes greater than 20.

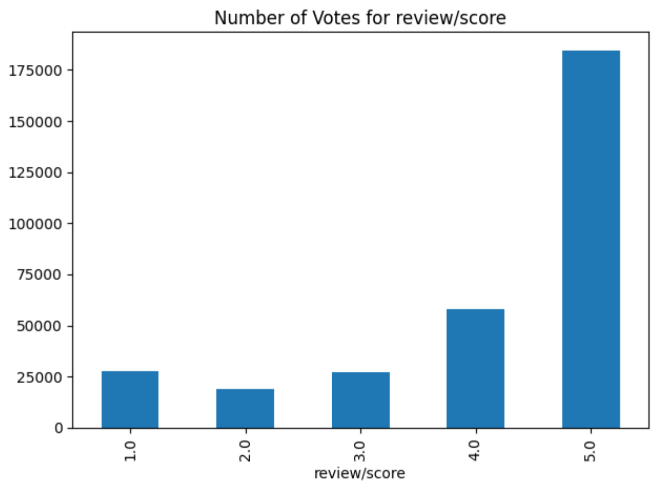

Fig. 2: Distribution of votes on the 4 rating categories

The Spearman correlation coefficient between the two variables is equal to 0.5247, with a p-value of 0.0.
Conclusion: The hypothesis is confirmed as there is a positive and statistically significant correlation between the two variables. This result is confirmed by the boxplot in Figure 3.

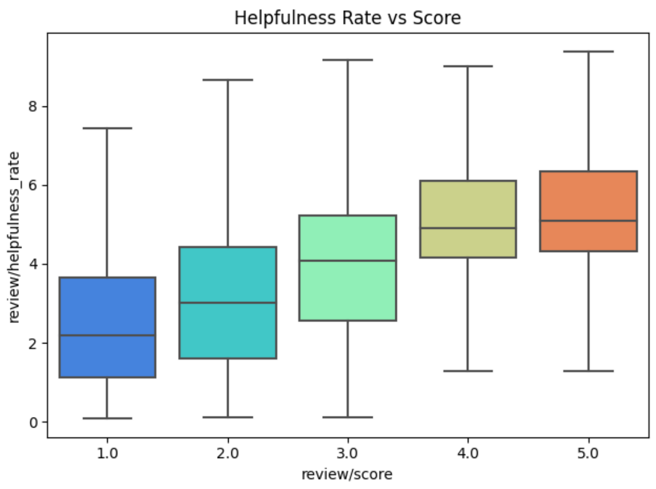

Fig. 3: Boxplot of the correlation between review rating and helpfulness score

Hypothesis 4

Hypothesis 5

Hypothesis 6

- The larger the number of books published for a category, the higher the review score.
- The larger the number of books published by publishers, the higher the review score.

Description and Results All the data cleaning and transformation steps were performed using MongoDB with the aggregation pipeline, to have a more efficient and faster computation.
Specifically the data cleaning steps were performed using the *$match* operator, while the data transformation steps were performed using the *$group* operator with *$avg* operation. Finally a *$project* operator was used to select the fields of interest. To reduce bias, we removed the categories having less than 50 books and the publishers having less than 20 books. The results are shown in Table 2.
Conclusion: The hypotheses are falsified since the metrics shows no correlation between the two variables in both cases.

Table 2: Correlation Values and P-values for Categories and Publishers

Variable	Correlation Value	P-value
Category	-0.0806	0.558
Publisher	-0.0673	0.151

Curiosity We performed two complex MongoDB queries (reported as example in section 'Code') to answer to two questions:

- Which are the best publishers? (i.e. capable of getting avg scores above 4.5 in lots of categories)
- In which categories are the best publishers focused?

The results are reported in Figure 4 and Figure 5.

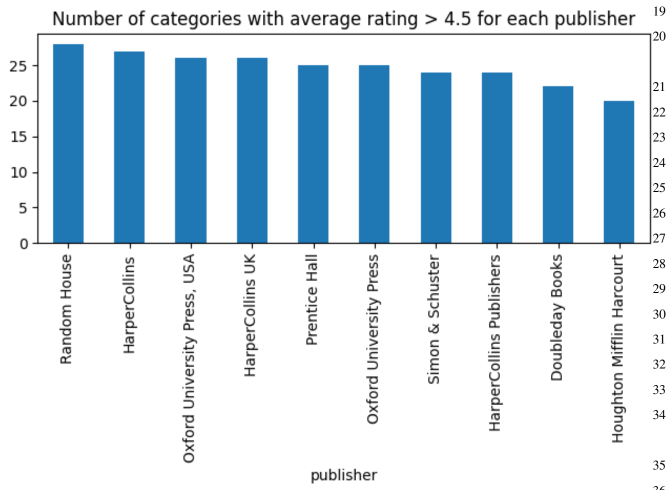

Fig. 4: Which are the best publishers?

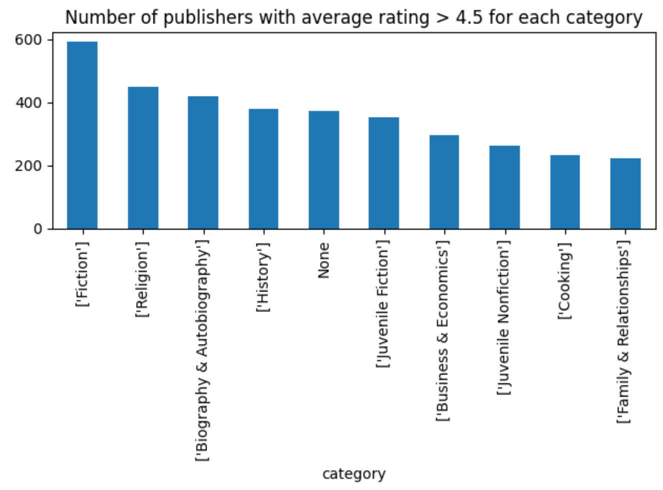

Fig. 5: In which categories are the best publishers focused?

5. Spark Hypotheses Testing

6. Helpfulness Prediction

7. Conclusions

8. Python Script

This is the script to answer the question: In which categories are the best publishers focused?

```
# Deal with missing values
pipeline_missing = {'$match': {
    'review/score': {'$exists': True, '$ne': 0},
    'publisher': {'$exists': True, '$ne': None},
    'categories': {'$exists': True},
}
}

# Compute average rating for each tuple category,
    publisher
pipeline_average_rating = {'$group': {
    '_id': {
        'category': '$categories',
        'publisher': '$publisher',
    },
    'avg_score': {'$avg': '$review/score'},
    'count': {'$sum': 1}
}
}

# Show average rating for category for each
    publisher
pipeline_publisher = {'$group': {
    '_id': '$_id.category',
    'avg_score/publisher': {
        '$push': {
            'publisher': '$_id.publisher',
            'avg_score': '$avg_score',
            'count': '$count'
        }
    }
}
}

# Unwind the list of categories
pipeline_unwind = {'$unwind': '$avg_score/
    publisher'}

# Remove categories or publisher with less than '
    threshold' reviews
```

```
threshold = 0
pipeline_remove = {'$match': {
    'avg_score/publisher.count': {'$gte': threshold}
}
}

# Count the number of categories with average rating > 4.5
pipeline_counts = {'$project': {
    'category': '$_id',
    '_id': 0,
    'publisher': '$avg_score/publisher.publisher',
    'count': {
        '$sum': {
            '$cond': {

                'if': {'$gt': ['$avg_score/publisher.avg_score', 4.5]},
                'then': 1,
                'else': 0
            }
        }
    }
}
}

# Sum the results for each publisher. If Total > 10, then the hypothesis is False
pipeline_sum = {'$group': {
    '_id': '$category',
    'total': {'$sum': '$count'}
}
}

pipeline_sort = {'$sort': {
    'total': -1
}
}

results = books.aggregate([pipeline_missing, pipeline_average_rating, pipeline_publisher,
                          pipeline_unwind, pipeline_remove, pipeline_counts,
                          pipeline_sum, pipeline_sort])
```