Sorting Amazon Kindle Reviews + Ratings
Sentiment Analysis of Kindle Reviews

Anh Nguyen

Table of Contents

1. Introduction

- Aims at studying a pre-existing dataset of Amazon Kindle Reviews
- Restructure this flattened dataset so it can be loaded into a SQL database
- Conduct some basic exploratory data analyses and sentiment analysis → Draw conclusions

2. Methodology

01 **Data Collection**

02 **Data Pre-processing**

03 **Exploratory Data Analysis**

04 **Sentiment Analysis**

05 **Conclusion**

Block Diagram

2.1 Data Collection

- Kaggle: https://www.kaggle.com/datasets/bharadwaj6/kindle-reviews/data
- File format: JSON
- Size: 827.8MB
- A small subset of dataset of product reviews from Amazon Kindle Store
- Timeframe: May 1996 - July 2014
- Contains 982619 entries
- Each reviewer has at least 5 reviews and each product has at least 5 reviews

2.2 Data Pre-processing

- Use SQLite library to extract the dataset to a Jupyter Notebook

```
# Import necessary libraries

import json
import sqlite3
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Load JSON data from the Kaggle dataset

data = []
with open('kindle_reviews.json') as json_file:
    for idx, line in enumerate(json_file):
        data.append(json.loads(line))

print(len(data))
```

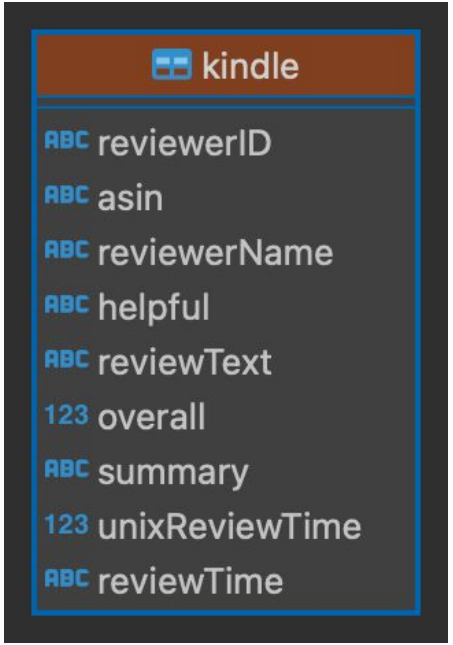
```
982619
```

2.2 Data Pre-processing

- Set up and connect to the SQLite database
- Create a table and store data from JSON file in it

```
# Set up and connect to the SQLite database

conn = sqlite3.connect('kindlereviews.db')
cursor = conn.cursor()

# Create a table to store JSON data

cursor.execute("""CREATE TABLE IF NOT EXISTS kindle (
    reviewerID  TEXT,
    asin TEXT,
    reviewerName  TEXT,
    helpful   TEXT,
    reviewText  TEXT,
    overall    REAL,
    summary    TEXT,
    unixReviewTime INTEGER,
    reviewTime TEXT)"""
);
```

```
# Insert the JSON data into the table

for item in data:
    if "reviewerName" not in item:
        item["reviewerName"] = ""
    values = [str(value) for value in item.values()]
    values_tuple = tuple(values)

    cursor.execute("INSERT INTO kindle (reviewerID,asin,reviewerName,helpful,reviewText,overall,
                   summary,unixReviewTime,reviewTime) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)", values_tuple)
```



2.2 Data Pre-processing

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A1F6404F1VG29J	B000F83SZQ	Avidreader	[0, 0]	I enjoy vintage books and movies so I enjoyed ...	5.0	Nice vintage story	1399248000	05 5, 2014
1	AN0N05A9LIJEQ	B000F83SZQ	critters	[2, 2]	This book is a reissue of an old one; the auth...	4.0	Different...	1388966400	01 6, 2014
2	A795DMNCJILA6	B000F83SZQ	dot	[2, 2]	This was a fairly interesting read. It had ol...	4.0	Oldie	1396569600	04 4, 2014
3	A1FV0SX13TWVXQ	B000F83SZQ	Elaine H. Turley "Montana Songbird"	[1, 1]	I'd never read any of the Amy Brewster mysteri...	5.0	I really liked it.	1392768000	02 19, 2014
4	A3SPTOKDG7WBLN	B000F83SZQ	Father Dowling Fan	[0, 1]	If you like period pieces - clothing, lingo, y...	4.0	Period Mystery	1395187200	03 19, 2014

2.2 Data Pre-processing

- Remove non-numeric data
- Covert data types

```
# Remove rating data that is not numerical to count the number of (numerical) reviews for each rating

new_ratings = ratings[pd.to_numeric(ratings['ratings'], errors='coerce').notna()]
new_ratings = new_ratings.reset_index(drop=True)
new_ratings
```

3. Sorting Reviews and Ratings

Statistical analysis for all ratings

```
count    978821.000000
mean          4.348605
std           0.954134
min           1.000000
25%           4.000000
50%           5.000000
75%           5.000000
max           5.000000
Name: overall, dtype: float64
```

Average reviews per product

```
no_asin['count'].mean()
```

```
15.86558271708593
```

Average ratings of all reviewer IDs

```
no_id['count'].mean()
```

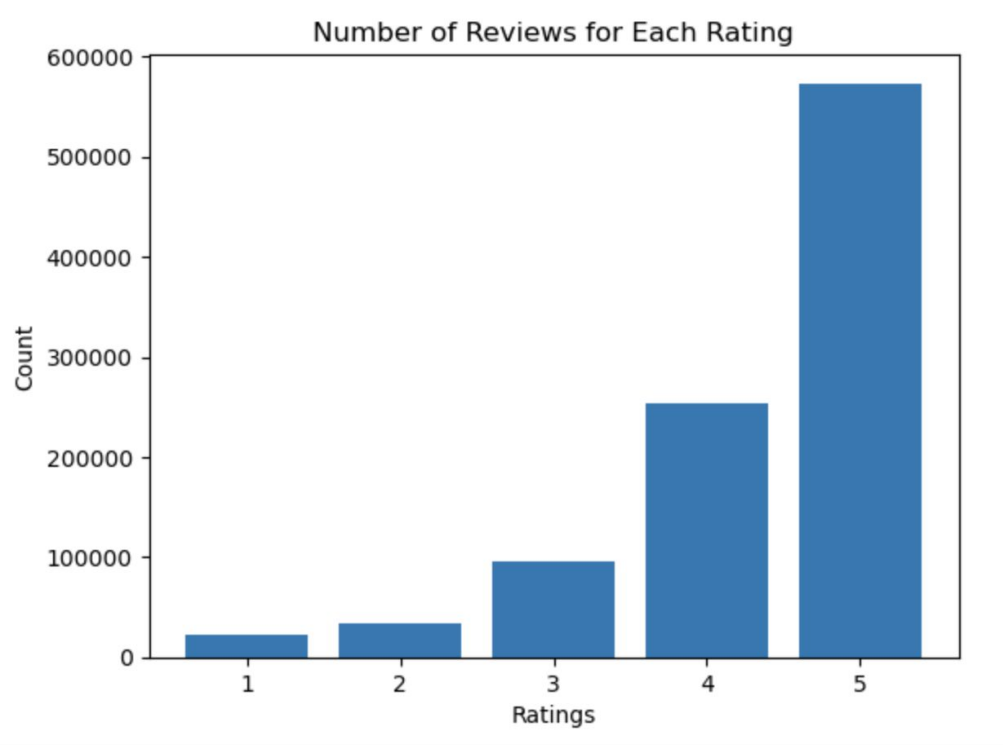
```
14.40304589361359
```

3. Sorting Reviews and Ratings

```
# Get the number of reviews each rating has

query = """
select overall as ratings, count(overall) as count
from kindle
group by overall"""

ratings = pd.read_sql_query(query, conn)
ratings
```

	ratings	count
0	1.0	22814
1	2.0	33932
2	3.0	95730
3	4.0	253087
4	5.0	573258

3. Sorting Reviews and Ratings

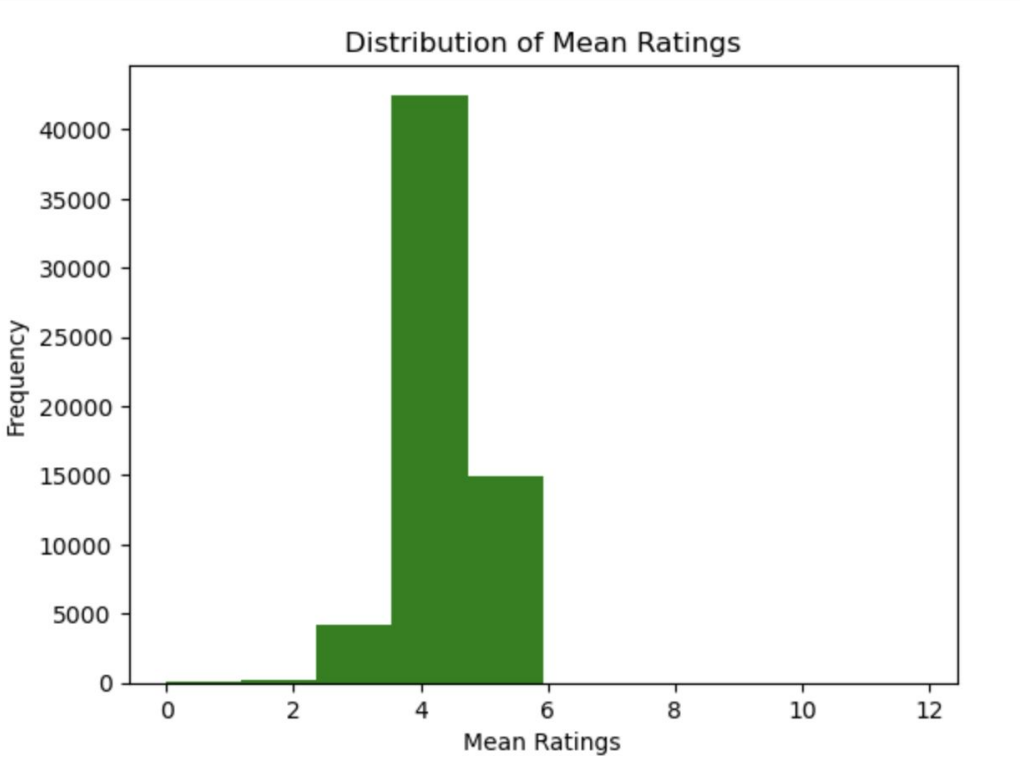


3. Sorting Reviews and Ratings

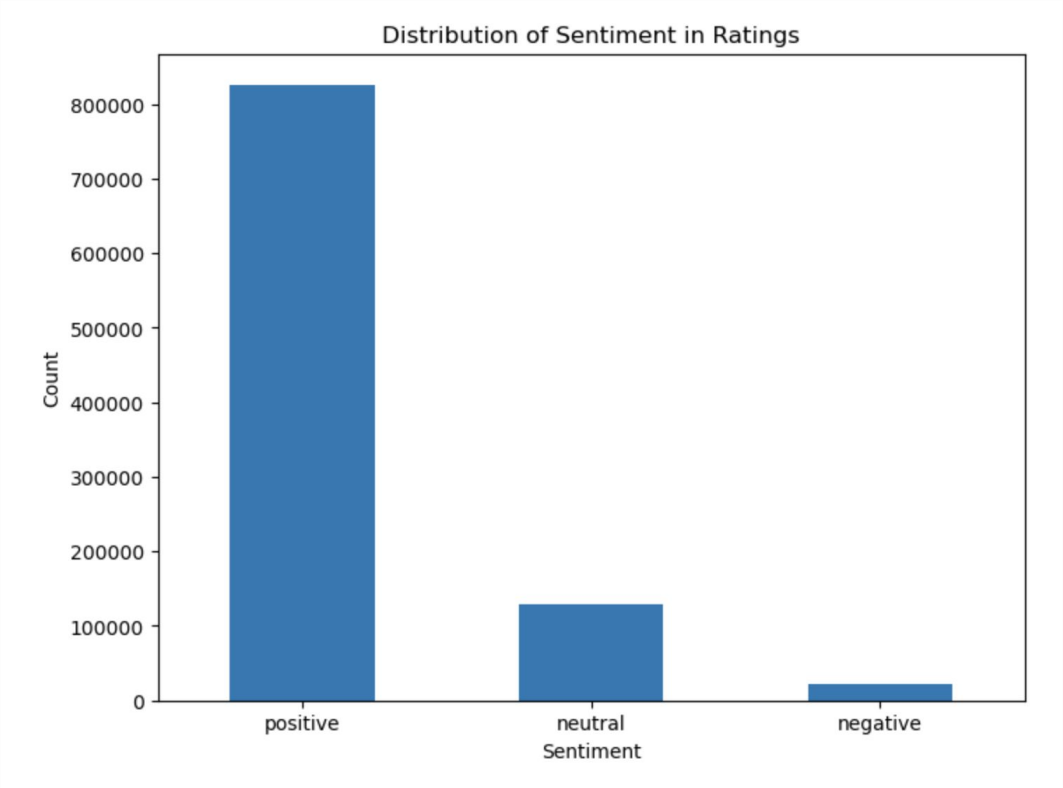
```
query = """
select asin, count(overall) as count, avg(overall) as mean_ratings
from kindle
group by asin
order by mean_ratings desc"""

df_asin = pd.read_sql_query(query, conn)
df_asin
```

	asin	count	mean_ratings
0	B00AFH6O4O	31	11.838710
1	B0093MU7QS	307	7.677524
2	B006244Q44	5	7.600000
3	B007HAWNYW	18	5.166667
4	B00LZ9OBWI	7	5.000000
...	...	...	...
61929	B0086824IE	34	0.382353
61930	B0082UPBWC	22	0.181818
61931	B0061YAUG8	287	0.090592
61932	B00JDZIK3W	6	0.000000
61933	B001O2SCKI	12	0.000000

3. Sorting Reviews and Ratings



3. Sorting Reviews and Ratings



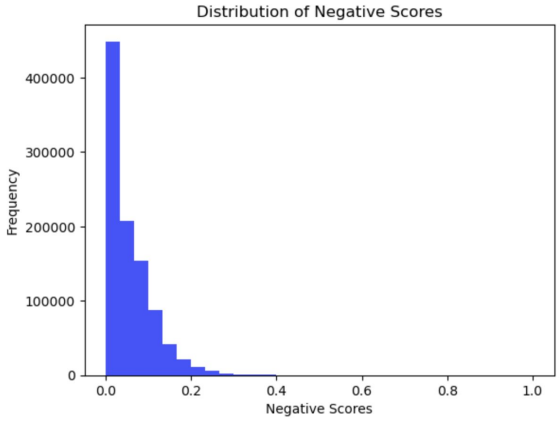
2.3 Sentiment Analysis of Review Text

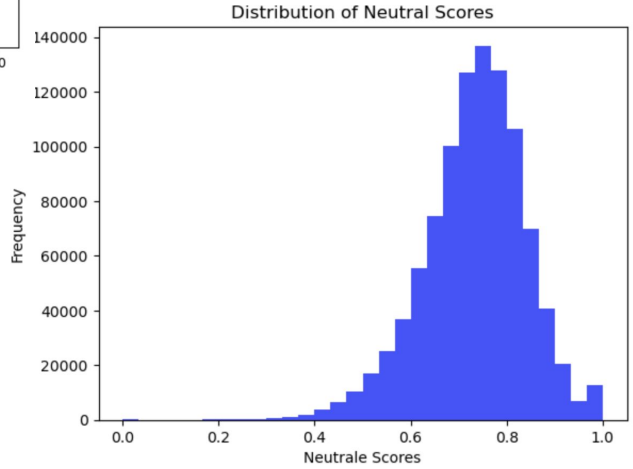
- Use a built-in Natural Language Toolkit (nltk) library for Sentiment Analysis
- Get the review data from table using query to run sentiment analysis on

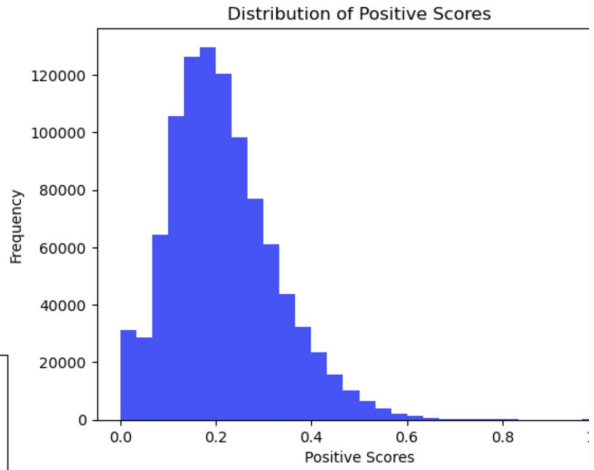
```
from nltk.sentiment import SentimentIntensityAnalyzer

# Create a sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Get the review text column from the table
query = """
select reviewText
from kindle"""

cursor.execute(query)

# Get all rows from the query result
rows = cursor.fetchall()
```

	Negative	Neutral	Positive
0	0.00	0.809	0.191
1	0.05	0.775	0.175
2	0.00	0.741	0.259
3	0.00	1.000	0.000
4	0.00	0.787	0.213

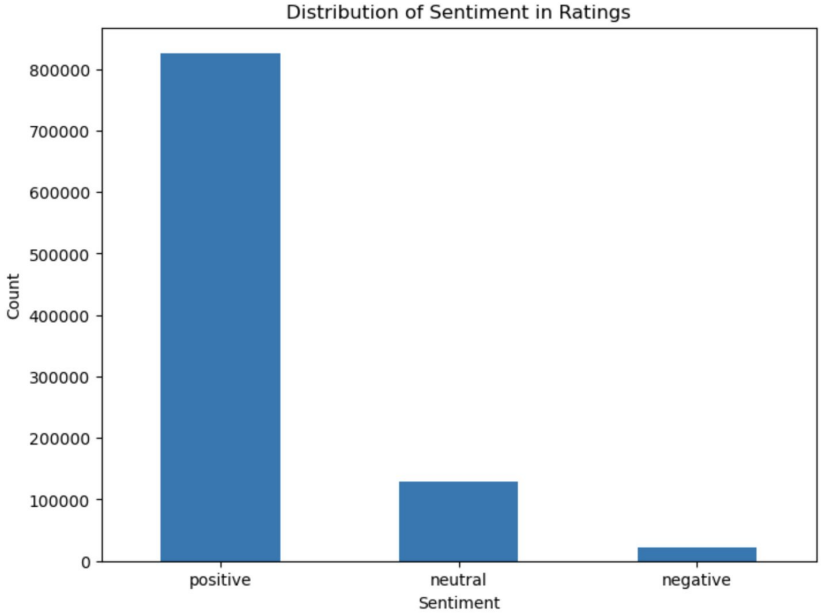
4. Sentiment Analysis of Review Text



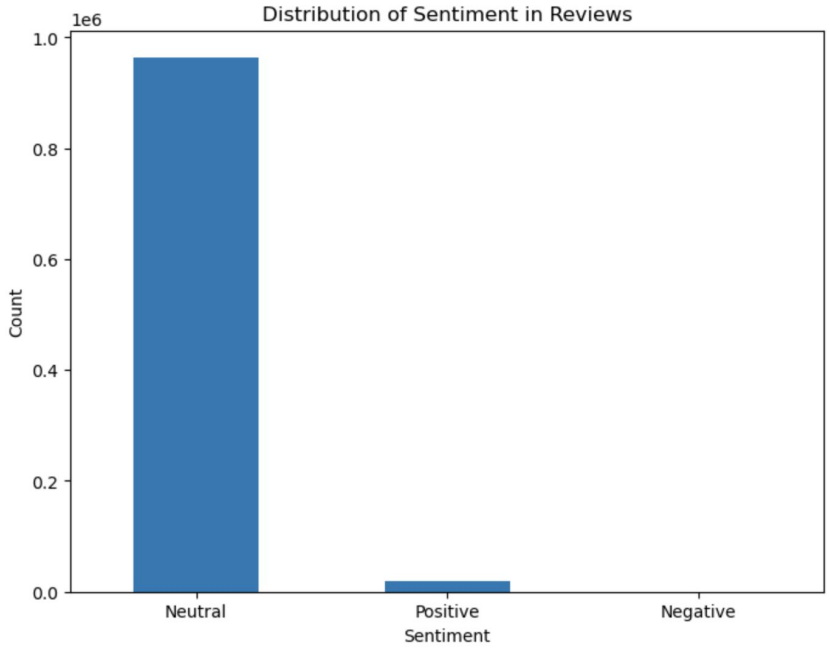




5. Conclusions



Exploratory Data Analysis



Sentiment Analysis

5. Conclusions

Restructuring JSON data and storing it in a SQLite database

Ensure the consistency and integrity of the data

Prevent data inconsistencies and errors that might happen when working with raw JSON

Support indexing and provides efficient querying capabilities

Involve an upfront transformation step
→ May require additional effort compared to working directly with JSON

Questions