Arda Ege İsker	Taner Furkan Göztok
Computer Engineering	*Computer Engineering*
TOBB University of Economic and Technology	*TOBB University of Economic and Technology*
Söğütözü, Çankaya Cd. No:43 06510, Çankaya/Ankara	*Söğütözü, Çankaya Cd. No:43 06510, Çankaya/Ankara*
aisker@etu.edu.tr	tfgoztok@etu.edu.tr

Abstract

The main goal to this project is to make inferences on raw "Amazon Electronic Product Reviews" dataset between 1996-2018.

I. Introduction

To get easy to read outputs, PySpark on Jupyter Notebooks used to distributed data processing to make fast and scalable computations. Outputs are optimized to be easy to read by using graphics and lists also explained with tables.

II. Relatable Projects

There are lots of open source projects on Amazon Product Reviews [1] [2] [3]. Most of these projects are based on classification, product recommendation or score prediction using machine learning alghoritms . The most common point of these projects are, all of them processes static and structured datasets.

III. Problem Explanation

Making sense of more than 20 million reviews of electronic products on Amazon between 1997 and 2018 and this process needs to be done in a short time with efficient methods. Thus, information about users and products can be obtained and this information can be marketed or it can be turned into a collective schema that product manufacturers should sample. The only purpose of the project is not only these. Also for students to have knowledge about how technologies such as distributed data structure and virtualization by Docker can be used.

IV. Dataset Resource

In this project the dataset of the University of San Diego in 2018 has been used. Overhaul from the data mining project from the publication. The data is made in .JSON file , there are a total of 20994353 reviews in it. The size of the file is 11.51 GB,It can be downloaded from Dataset link. With

disaggregated reviews for each category There is an option to download also raw data, removed duplicated reviews. There are also processed datasets, such as a filtered version of products with less than 5 reviews. In this project the raw form in the electronics category between 1996-2018 was used. We recommend to fill the form to contact Jianmo Ni, UCSD before download the dataset.

V. Technologies Used

In this project, Apache Spark (Pyspark), Jupyter Notebooks on Docker ve Python has been used. Apache Spark has used for distributed data processing events, Jupyter Notebooks has used for easily set the desired device resources in a virtual environment. Python is also used as the main programming language.

VI. Dataflow Diagrams

The project has been prepared on a single data set and the tables are fixed. Therefore, there is no Dataflow. But the schematic of the dataset can be seen below. Each product in the dataset has a column named “ASIN” which is an ID for that product. Review - product conversion is provided by the ASIN value included in the review.

```
[('asin', 'string'),
 ('image', 'array<string>'),
 ('overall', 'double'),
 ('reviewText', 'string'),
 ('reviewTime', 'string'),
 ('reviewerID', 'string'),
 ('reviewerName', 'string'),
 ('style',
  'struct<Capacity::string,Color Name::string,Color::string,Colorj::string,Colour::string,Configuration::string,Des
ign::string,Digital Storage Capacity::string,Edition::string,Flavor Name::string,Format::string,Grip Type::string,I
tem Display Length::string,Item Package Quantity::string,Length Range::string,Length::string,Material Type::string,
Metal Type::string,Model Name::string,Model Number::string,Model::string,Number of Items::string,Offer Type::string
,Package Quantity::string,Package Type::string,Pattern::string,Platform for Display::string,Platform::string,Preamp
lifier Output Channel Quantity::string,Processor Description::string,Product Packaging::string,Service plan term::s
tring,Shape::string,Size Name::string,Size::string,Style Name::string,Style::string,Team Name::string,Tension Suppo
rted::string,Wattage::string,Width::string,processor_description::string,style::string,style_name::string>'),
 ('summary', 'string'),
 ('unixReviewTime', 'bigint'),
 ('verified', 'boolean'),
 ('vote', 'string')]
```

VII. Details of Computation Algorithms

We performed our calculations for three different time periods. These are;

1) All Times
2) Year Based Computations
3) Season Based Computations

While performing the calculations involving all times, we converted the entire dataset into a "Spark Dataframe" and performed the calculations on this dataframe. A new column "year" has been added to the Dataframe while making year-based computations. This column is made using the reviewTime column that already in the dataset. First, the reviewTime column, which is kept as a string, is converted to date,After converting to date, the year of each row is written in the "year" column and groupBy operation has used. While performing seasonal calculations, the years of the seasons are given as a range with SQL's WHERE query by using the Spark SQL library.

```
#Reformatting reviewTime column, String to Date in order to make things work efficiently and well.
from pyspark.sql.types import IntegerType

df = df.withColumn(
            'reviewTime',
                F.to_date(
                    F.unix_timestamp('reviewTime', 'MM d, yyyy').cast('timestamp')))\
        .withColumn(
            'vote',
                df["vote"].cast(IntegerType())
        )
```

With the code seen above, conversion from string to date is performed. Then, a "year" column was added to a newly created Dataframe and groupBy operations were performed using this column.

```
adjustedDf = df.withColumn("reviewTime", to_date(col("reviewTime"), "yyyy-MM-dd"))\
    .withColumn('year', year("reviewTime"))
```

When performing seasonal filtering:

```
spark.sql("SELECT * FROM TABLE where reviewTime between '"+str(year) +"-01-01' and '"+str(year) +"-03-01' ")
```

SQL queries used.

URL conversions of output products has calculated by "https://www.amazon.com/dp/" +ASIN number of the product function.

```
def getProductURL(asin):
    return "https://www.amazon.com/dp/" + asin
```

We used Python's URLLib library to print the image to the screen using an image's web link.

```
urllib.request.urlretrieve(imageURLs[i],"image.png")
img = Image.open("image.png")
img.show()
```

We obtained the calculations of the "most"s by groupBy with the product's ASIN number and then sorting in descending order. Then we took as many rows as we wanted and assigned them to a list (number of rows effects computation time). Then, we looped through each list element and presented the information about the product to the user.

```
voteDF = df.filter(df.vote.isNotNull()) #Task number 15
voteDF = voteDF.withColumn("vote", voteDF["vote"].cast(IntegerType()))
listOftopReviews = voteDF.orderBy('vote', ascending=False).take(10)
topReviewTexts=[]
topReviewTextsAsin=[]

for i in range(10):
    text = listOftopReviews[i].reviewText
    asin = listOftopReviews[i].asin
    topReviewTexts.append(text)
    topReviewTextsAsin.append(asin)

for j in range(10):
    print(str(j+1) + ")")
    print("Product ASIN No:" + topReviewTextsAsin[j])
    print("Product URL: " + getProductURL(topReviewTextsAsin[j]))
    print(topReviewTexts[j])
    print("////////////////////////////////////////////////////")
    print("////////////////////////////////////////////////////")
    print("////////////////////////////////////////////////////")
```

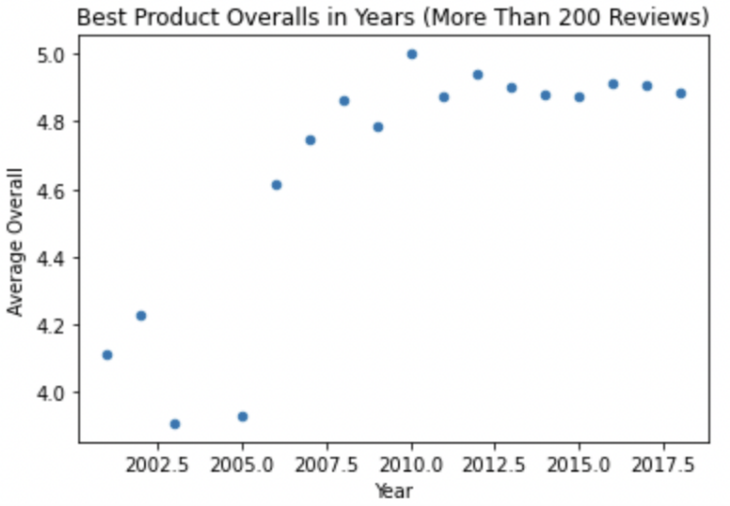
Since every column we use is a string data type, we performed a conversion from string to integer when we needed to do mathematical calculations.

VIII. Results

Desired results were obtained from the "Amazon Electronic Product Review" dataset, graphs and lists were drawn and the dataset was given meaning. Below you can see the results we achieved within the scope of the project.

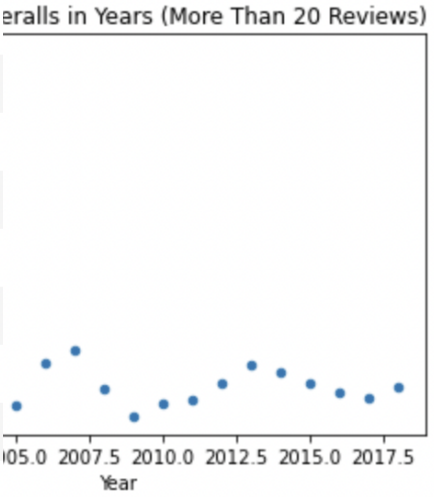
```
#HER YILA GORE EN IYI PUANA SAHIP URUNLERIN LISTESI VE GRAFIGI
```

Year	ASIN	Average Overall
2001	B00004SB92	4.112195
2002	B00004SB92	4.230047
2003	B00005ARK3	3.909091
2005	B00007KDVI	3.933071
2006	B000A3WS84	4.616858
2007	B0002IWC9C	4.744755
2008	B000A5TAT2	4.862595
2009	B00007E7JU	4.784000
2010	B000065BP9	5.000000
2011	B00004ZCJJ	4.876106
2012	B00001WRSJ	4.942222
2013	B00005N6KG	4.901288
2014	B00004T8R2	4.878378
2015	B000051ZOA	4.876251
2016	B00002EQCS	4.910769
2017	B000067S60	4.910000
2018	B00004T8R2	4.887613



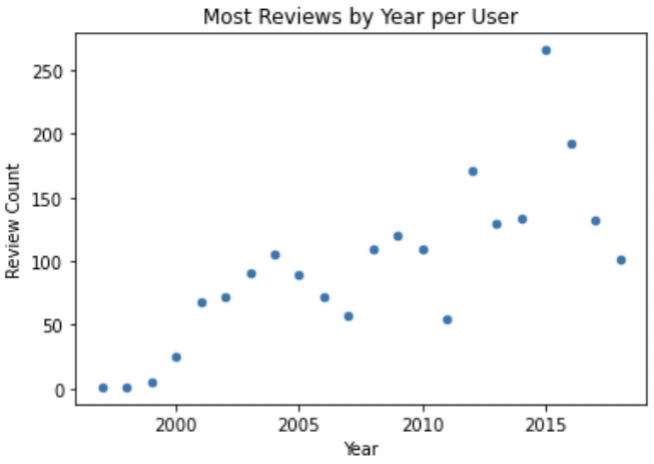
List and graph of the worst rated products by year

Year	ASIN	Average Overall
1999	B00000JFMK	3.704545
2000	B00000JMUG	1.888889
2001	B00004YRVY	2.090909
2002	B000063V7N	1.857143
2003	B00001P4XA	1.833333
2004	B00000J061	1.333333
2005	B00005A9AU	1.115385
2006	B00004VUGJ	1.428571
2007	B000065BPB	1.521739
2008	B00000J1V3	1.238095
2009	B00004T1XE	1.034483
2010	B00005LEN4	1.125000
2011	B00004WINT	1.150000
2012	1400501466	1.275862
2013	B00004THCZ	1.413793
2014	B00004VX39	1.357143
2015	B000066R6M	1.280000
2016	998504780X	1.214286
2017	B00004SY4H	1.166667
2018	B00004SABB	1.250000



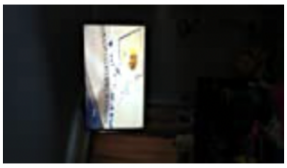
Top reviewer per year and its graphic

Year	User ID	Review Count
1997	A3B12RM18TB8OQ	1
1998	A2CT25FY12PYNF	1
1999	APKNTDAV6EPI4	5
2000	AJIC64C630FTT	25
2001	A3NJ1W51XSRC29	68
2002	A30NZYWSJ5WT7I	72
2003	A3GQBTDQYMJYA0	91
2004	A3PWF2AQ5XYALW	106
2005	A7OR8OWWGK55W	89
2006	A221XWNYJZX863	72
2007	A1CQ0FUOKPQZOV	57
2008	A1CKDRO4VYQQQC	110
2009	AVFR88UIGCQS6	120
2010	AESUMN2OKV2H8	110
2011	ALKZQOZM9Y72H	54
2012	A2OFPBKQ6DTVW7	171
2013	A1HBJHGRLTM5XH	129
2014	A3OXZRN6VAXPBG	133
2015	A3HWHZNT6R7B3P	266
2016	A0008052W2M3QH4G1NJZ	192
2017	A1YY8PAYNR1CNG	132
2018	AVRDNNVGWOTZY	102



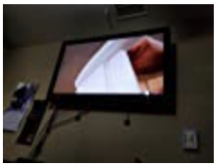
Product images of most reviewed products that has imaged reviews.

Image reviews of most reviewed products that has imaged reviews.
1) https://www.amazon.com/dp/B00B5HE3UU

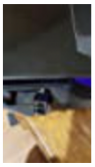


2) https://www.amazon.com/dp/B00B5HE3UU

3) https://www.amazon.com/dp/B00B5HE3UU



4) https://www.amazon.com/dp/B00B5HE3UU



5) https://www.amazon.com/dp/B00B5HE3UU

Ratio of approved products to all products

```
Rate of verified products to all products: %88.58140091290262.
```

Most liked product of 2016's Winter

ASIN	Average Overall
B00004T8R2	4.914634

Most disliked product of 2016's Winter

ASIN	Average Overall
B00005ATMI	1.222222

Apache Spark was run from two identical computers (Macbook Air M1). Different resource combinations are given to Docker Containers so that computational performance can be measured. For example, a 6-core processor and 8GB memory configuration on a machine and the for other machine, 4-core processor and 4GB memory configuration has been used. It has been seen that the biggest factor that provides awareness at the time of execution is the memory size. It has been observed that the number of blocks Apache Spark processes decreases in settings where the memory size is small, so there is a performance difference of nearly 30% between 8GB and 4GB memory.

Also, in some processes where the memory size is insufficient “RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch.” error has been received. When this error was investigated, it was seen that the memory allocated to Spark was full and some data in the memory was written to the disk because memory usage was required. This has been one of the most important factors affecting performance. While the speed of the memory in the devices used is officially 204GB/s, the disk speed is stated as 2400MB/s on average.

IX. References

1. https://github.com/minimaxir/amazon-spark
2. https://github.com/Currie32/Text-Summarization-with-Amazon-Reviews
3. https://github.com/aws-samples/amazon-codeguru-reviewer-sample-app
4. https://spark.apache.org/docs/latest/api/python/
5. https://sparkbyexamples.com/pyspark-tutorial/
6. https://hub.docker.com/r/jupyter/pyspark-notebook
7. https://docs.docker.com/desktop/mac/install/
8. https://jupyter-notebook.readthedocs.io/en/stable/