

Analysis on Amazon Review Dataset

CIS 602-01- Scalable Data Analysis

Abhishek Manoj Kumar

ID: 01675536

Contents:

Introduction

Background and Dataset Information

Scalability Challenges

Solutions to Scalability Challenges and their Implementation

Analysis and Results

Conclusion

References

Introduction

The dataset I have chosen is the “Amazon product dataset” which consists of data about the product reviews, ratings etc which I will describe in detail in the next section. The dataset which was made available was about 16.6 GB which was divided into 24 json files based on categories. The largest json file was about 8.8 Gb which I could not load on my PC due to limited memory. My goal was to do analysis on this data which required me to combine all the files together and load them. This was the first issue I had to deal with, since the data is too large. To resolve this issue, I decided to create a database using MongoDB and query from the database to obtain the required data for analysis. Also for tasks like finding the word count and performing topic modelling, required me to tokenize the words and apply the LDA model provided by the genism library which again took a long time when done serial. To speed-up this process I was able to parallelize the code and distribute the tasks on multiple cores which gave significant speedup.

Dataset Information

Dataset URL: <http://jmcauley.ucsd.edu/data/amazon/>

Dataset Description:

The complete dataset consists about 142.8 million reviews and the dataset I could access was a subset consisting of multiple files of 5-core review data with a combine size of 16.6 GB and had limited attributes as well. The Dataset is divided into multiple json files, where each json file consists reviews of each product category. Since the subset alone was 16.6 GB, it was large enough for the project. The dataset gives us reviews made on amazon products between the year 1996-2014. Also, the size of the data varies for each category and varies from 7MB to 8.8 GB.

























Name	Date modified	Type	Size
 Amazon_Instant_Video_5.json	3/2/2016 1:56 PM	JSON File	27,450 KB
 Apps_for_Android_5.json	3/2/2016 2:00 PM	JSON File	329,483 KB
 Automotive_5.json	3/2/2016 2:08 PM	JSON File	13,943 KB
 Baby_5.json	3/2/2016 2:10 PM	JSON File	118,529 KB
 Beauty_5.json	3/2/2016 2:16 PM	JSON File	137,493 KB
 Books_5.json	3/2/2016 5:10 PM	JSON File	9,236,338 KB
 CDs_and_Vinyl_5.json	3/2/2016 6:09 PM	JSON File	1,332,515 KB
 Cell_Phones_and_Accessories_5.json	3/2/2016 6:29 PM	JSON File	138,370 KB
 Clothing_Shoes_and_Jewelry_5.json	3/2/2016 6:59 PM	JSON File	149,564 KB
 Digital_Music_5.json	3/2/2016 7:07 PM	JSON File	86,880 KB
 Electronics_5.json	3/2/2016 7:35 PM	JSON File	1,444,303 KB
 Grocery_and_Gourmet_Food_5.json	3/2/2016 7:50 PM	JSON File	111,112 KB
 Health_and_Personal_Care_5.json	3/2/2016 8:01 PM	JSON File	255,587 KB
 Home_and_Kitchen_5.json	3/2/2016 8:18 PM	JSON File	411,435 KB
 Kindle_Store_5.json	3/2/2016 8:33 PM	JSON File	808,404 KB
 Movies_and_TV_5.json	3/2/2016 8:47 PM	JSON File	1,938,890 KB
 Musical_Instruments_5.json	3/2/2016 8:59 PM	JSON File	7,272 KB
 Office_Products_5.json	3/2/2016 9:03 PM	JSON File	54,603 KB
 Patio_Lawn_and_Garden_5.json	3/2/2016 9:07 PM	JSON File	14,204 KB
 Pet_Supplies_5.json	3/2/2016 9:10 PM	JSON File	108,017 KB
 Sports_and_Outdoors_5.json	3/2/2016 9:26 PM	JSON File	203,199 KB
 Tools_and_Home_Improvement_5.json	3/2/2016 9:34 PM	JSON File	109,742 KB
 Toys_and_Games_5.json	3/2/2016 9:42 PM	JSON File	126,670 KB
 Video_Games_5.json	3/2/2016 9:46 PM	JSON File	311,986 KB

Image showing all the json files based on category of the product.

Attribute Information:

Snapshot of each json object with the object information.

```
_id: ObjectId("5a332d0e5132162d811c8f3c")
reviewerID: "A1RJP1GRSNX4PW"
asin: "B000H00VBQ"
reviewerName: "J. Kaplan "JJ""
✓ helpful: Array
  0: 0
  1: 0
reviewText: "Mysteries are interesting. The tension between Robson and the tall bl..."
overall: 4
summary: "Robson Green is mesmerizing"
unixReviewTime: 1383091200
reviewTime: "10 30, 2013"
```

reviewerID - ID of the reviewer

asin - ID of the product. This ID can be used to find the particular product on Amazon.

reviewerName - name of the reviewer

helpful - helpfulness rating of the review. Shows if people found the review to be helpful

reviewText – The review provided by the consumers

overall - rating of the product provided by the consumer

summary - summary of the review

unixReviewTime - time of the review (unix time)

reviewTime - time of the review (raw)

Scalability Challenges

The challenge I faced was loading the large json files and I got memory errors while trying to do so. I also needed to combine all the files for analysing them and this was not possible since loading the files wasn't possible in the first place. Also, text tokenization and LDA used for topic modelling took long time to run serially.

Solutions to Scalability Challenges and their Implementation

Since I was not able to load the files with the memory I had, I decided to create a database using MongoDB and access the data from there. I was able to merge all the files into a single collection of the database and access the required attributes for analysis from the database.

Steps involved in setting up mongoDB,

Step 1: Download mongo DB and select where to install

https://www.mongodb.com/download-center?jmp=tutorials&_ga=2.25467773.1918750684.1513403815-39964168.1512945759#community

Step 2: Create 2 folders data and log and in data create db where data is saved

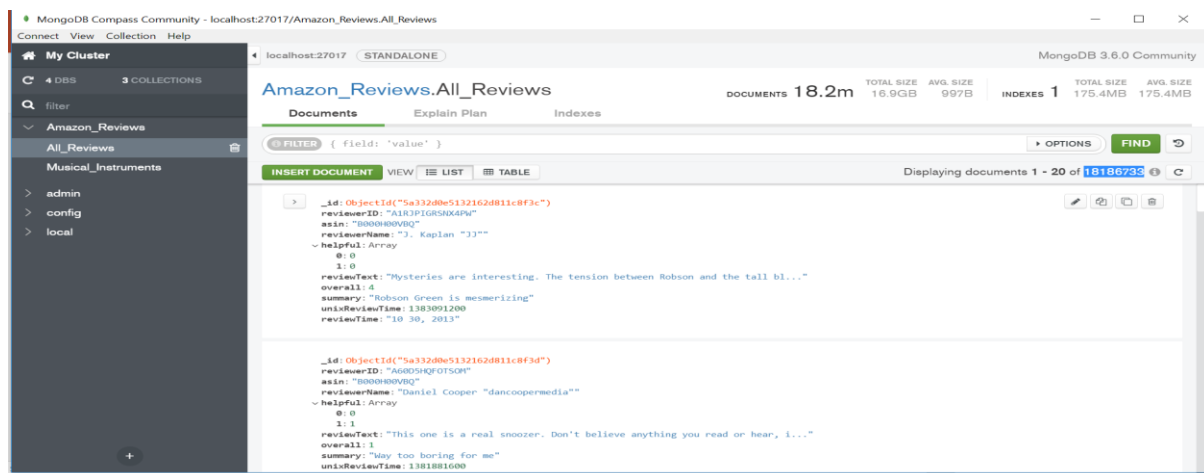
step 3: enter bin and run to add the flags

```
mongod --directoryperdb --dbpath C:\mongodb\data\db --logpath  
C:\mongodb\log\mongo.log --logappend--rest --install
```

Step 4: Enter another cmd as an administrator and type “net start MongoDB” to start mongoDB

Step 5: Enter the mongo shell by typing “mongo” and create a db by typing “use Amamzon_Reviews” which will automatically create and shift to this db, then create a collection using “db.createCollection(‘All_Reiews’)”.

Step 6: To insert the files into one collection, exit the mongo shell and enter the bin in the mongo folder to file mongoimport and use the following command,
mongoimport --db <db-name> --collection <coll-name> --type json --file seed.json
--jsonArray

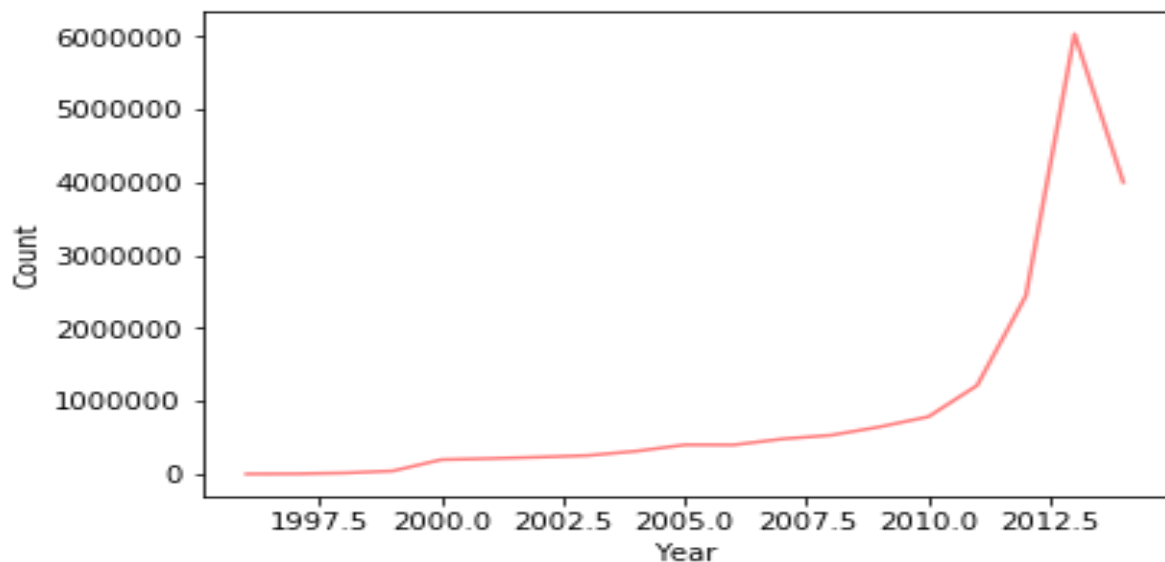


Snapshot of the collection with all the json files viewed from the mongoDB compass community.

After I was able to load the data, I had to tokenize the text and use LDA for topic modelling which required a lot of time and I was able to speed up both the processes by using multiprocessing library which divided the task into multiple cores and speed-up the process significantly.

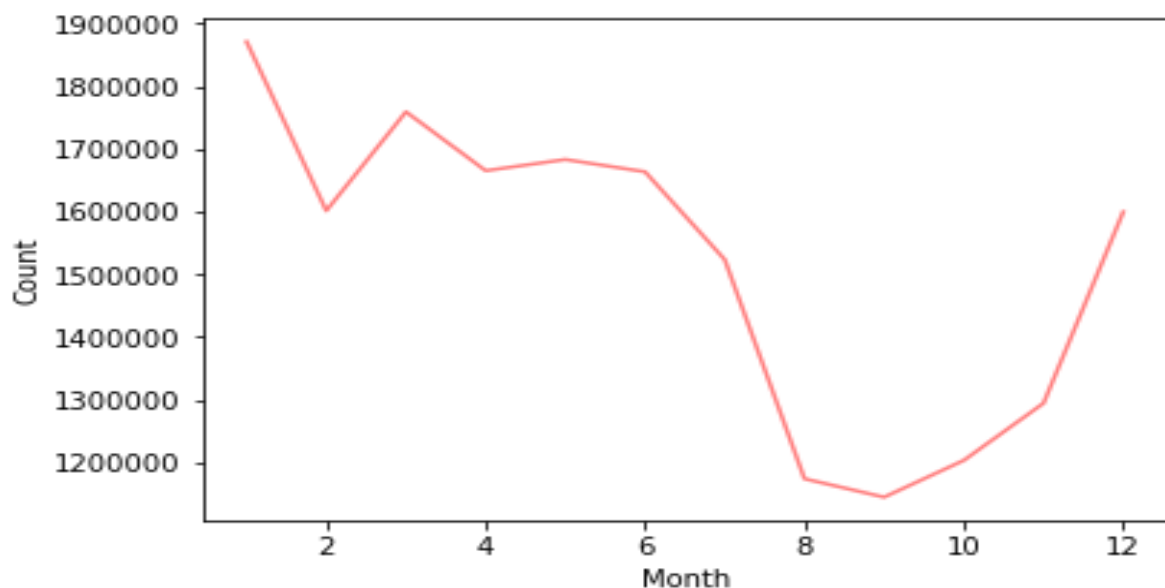
Analysis and Results

Analysis 1: Number of reviews over the years 1996 to 2014.



From this chart it is seen that the number of reviews increase gradually over the years from 1996 to 2010 and then spike at 2013 and then drop at 2014. From this chart we can see that amazon started getting popular after 2010 and their sales as well as customer base went up. The drop in 2014 might be because many online vendors came into the market and competition for amazon increased.

Analysis 2: Number of reviews based on months.



This visualization shows that there is highest number of reviews during Christmas and new years which shows that the sales must be high during that period since people write reviews after buying from the site. The number of reviews drop from July to November.

Analysis 3: What words occur the most in a review based on Rating?

Ratings: 1

2

3

```
[('work', 27236),
 ('bad', 25092),
 ('disappoint', 24995),
 ('wast', 24268),
 ('good', 21789),
 ('one', 20058),
 ('time', 19585),
 ('money', 18973),
 ('bore', 18785),
 ('read', 17601),
 ('like', 16340),
 ('poor', 16155),
 ('buy', 15849),
 ('star', 14760),
 ('get', 14655),
 ('terribl', 14578),
 ('worst', 13816),
 ('horribl', 13720),
 ('movi', 13100)]
```

```
[('disappoint', 41921),
 ('book', 34776),
 ('great', 24501),
 ('like', 22771),
 ('work', 21700),
 ('read', 20650),
 ('bore', 17648),
 ('much', 17509),
 ('stori', 16988),
 ('one', 16363),
 ('bad', 16070),
 ('ok', 15814),
 ('poor', 15476),
 ('get', 14957),
 ('better', 14430),
 ('realli', 12620),
 ('star', 11104),
 ('time', 10820),
 ('expect', 10475)]
```

```
[('read', 94058),
 ('ok', 84582),
 ('great', 83495),
 ('book', 80771),
 ('stori', 61170),
 ('okay', 53207),
 ('interest', 44116),
 ('work', 41516),
 ('better', 41147),
 ('like', 40174),
 ('bad', 39444),
 ('star', 35971),
 ('fun', 32248),
 ('one', 30923),
 ('nice', 30430),
 ('love', 29672),
 ('disappoint', 28945),
 ('littl', 28934),
 ('much', 28376)]
```

4

5

```
[('great', 404629),
 ('read', 310119),
 ('book', 229497),
 ('stori', 161685),
 ('love', 141941),
 ('fun', 139965),
 ('nice', 109740),
 ('work', 100648),
 ('like', 94351),
 ('enjoy', 81070),
 ('interest', 78905),
 ('well', 78879),
 ('one', 76812),
 ('seri', 73151),
 ('star', 62602),
 ('better', 56102),
 ('excel', 54157),
 ('littl', 53369),
 ('best', 52278)]
```

```
[('love', 793215),
 ('book', 764861),
 ('read', 584972),
 ('good', 508208),
 ('best', 370175),
 ('excel', 297064),
 ('stori', 281664),
 ('one', 250778),
 ('work', 247394),
 ('awesom', 216525),
 ('fun', 197242),
 ('seri', 195162),
 ('perfect', 173108),
 ('wonder', 168843),
 ('amaz', 162717),
 ('anoth', 142461),
 ('must', 140931),
 ('well', 133340),
 ('nice', 132641)]
```

Words taken from the summary attribute which gives the summary of the review text. It is clearly seen that for rating 1 and 2 there are more negative words like bad, disappoint, poor, terrible. For a rating of 3 it gets more neutral and positive with words like okay, fun, nice. For rating 4 and 5 the words are positive like great, love, perfect, excellent. Some words seem to be incomplete in a few cases because I was stemming the words to not differentiate words like eat and eating and I was stemming the words I didn't want as well.

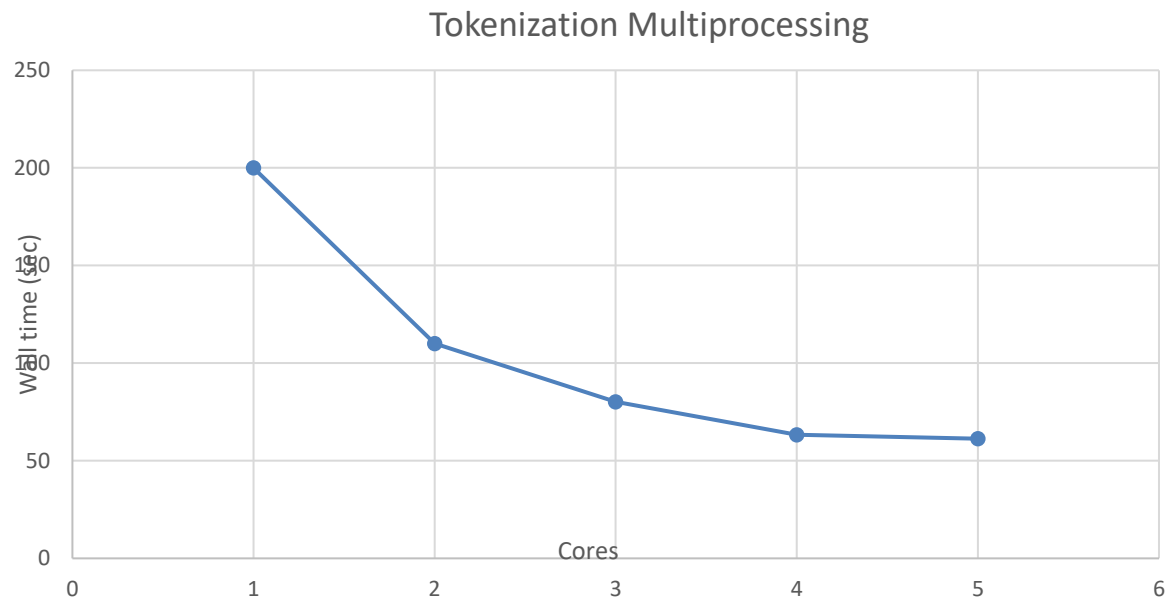


Chart showing varying tokenization time based on the number of cores.

Tokenization is performed on the text data before finding the count and this process took about 204 seconds for a sample of 1,00,000 review texts when I run it serially. Python provides a package for multiprocessing and when I use that there is significant speed up. For 4 and 5 cores the time in seconds is about 60 seconds with speed up of more than 3 times.

Analysis 4: Topic modelling based on reviews.

Topic modelling is a statistical model for discovering the abstract “topics” that occur in the collection of documents. I made use of the `gensim` library and the LDA (Latent Dirichlet Allocation) model. Here we need to provide the number of words and the number of topics. In the example below, I give words and topics = 5 and the model returns the 5 most significant words for 5 topics. Below, I ran the model for the musical instruments data and found words like guitar, tune, amp, sound, cable etc. which were most significant which tells us what the topic of discussion was.


```
In [31]: lda_model.show_topics(num_words=5, num_topics=5)
```

```
Out[31]: [(0,  
u'0.063*"use" + 0.055*"record" + 0.051*"qualiti" + 0.048*"good" + 0.040*"cabl"'),  
(1,  
u'0.150*"pick" + 0.110*"34" + 0.095*"strap" + 0.056*"guitar" + 0.051*"like"'),  
(2,  
u'0.156*"string" + 0.069*"guitar" + 0.056*"tune" + 0.054*"tuner" + 0.044*"sound"'),  
(3,  
u'0.105*"pedal" + 0.089*"sound" + 0.082*"amp" + 0.043*"like" + 0.043*"tone"'),  
(4,  
u'0.087*"guitar" + 0.063*"one" + 0.060*"work" + 0.060*"use" + 0.044*"stand"')]
```

Even the LDA model used here could be parallelized using the LDA multicore function. For 10,000 rows, When I ran the LDA the model on a single core, it took about 367.67 seconds. When I moved to multiple cores, there was a significant speed-up and with 5 cores it takes about 140 seconds.

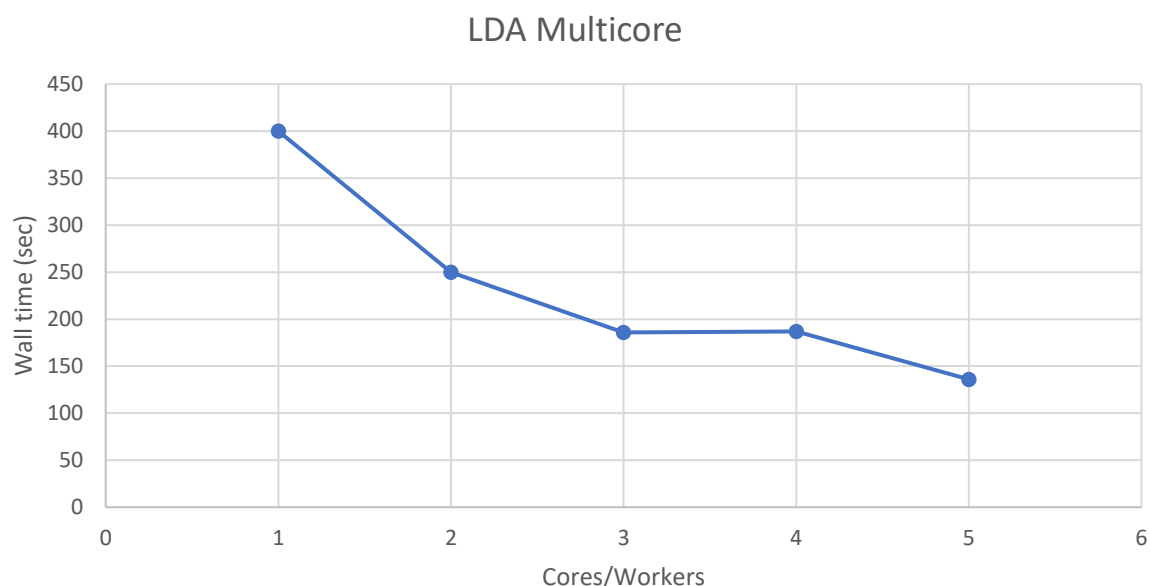


Chart showing time taken for LDA model on different cores.

Conclusion

I was able to merge all the json files together in the mongoDB database and get access to specific attributes based on the analysis question. The limitation which I could not resolve was even loading specific attribute information for the whole dataset took significant time and I could not speed up this process due to memory constraints. I could not load attributes like reviewText of the whole dataset as well, since that gave me memory errors as well limiting the analysis I could do on my PC.

I was able to use parallel processing to speedup the process of tokenization and LDA model used for topic modelling.

References

- <https://radimrehurek.com/gensim/models/ldamodel.html>
- <https://docs.mongodb.com/v3.4/tutorial/>
- https://www.packtpub.com/mapt/book/application_development/9781785287466/3/ch03lvl1sec43/creating-a-pandas-dataframe-from-a-mongodb-query
- https://rstudio-pubs-static.s3.amazonaws.com/79360_850b2a69980c4488b1db95987a24867a.html
- <https://github.com/cmchurch/python-multiprocessing/blob/master/multiprocess-pool-nltk.py>
- <https://datascience.blog.wzb.eu/2017/06/19/speeding-up-nltk-with-parallel-processing/>
- <https://pandas.pydata.org/pandas-docs/stable/>