Long Tail Recommendations using Amazon Product Reviews

Final Project Report

Team: Pandas

Abstract: The project focuses on the products that belong to the long tail in the Amazon product dataset. Products generally end up in the long tail for various reasons - like adhering to the interests of limited users or not being popular due to improper marketing. Our main goal is to increase the popularity of products belonging to the long tail, by recommending them to the users as products that can be purchased. The products from the long tail which are similar to the queried product can be returned as recommendations so that the user is aware of the diverse range of products available to choose from, in a way increasing the popularity of those products.

1. *Introduction*:

To understand the problem statement it is important to understand the concept of *Long Tail*. The Long Tail theory states that our culture and economy is shifting away to focus on the huge number of niches in the tail from the relatively small number of hits which represent the mainstream products at the head of the demand curve[1]. The reduced cost of production and distribution specially with the advent of e-commerce, has allowed the industry to stop fitting customers in the same bucket. From the era of one-size-fits-all, online retailers have moved towards personalisation. Without the constraints of distribution bottlenecks and other constraints such as limited physical shelf space, narrowly-targeted products can be as economically attractive as mainstream ones. The aggregate sale of such products which might not be significant independently can rival the sale of goods that do cross that economic threshold.

Traditional retail economics emphasise that stores should only stock the most popular products, because storage is expensive. Storing a niche product may be more expensive than the profit obtained by selling that item. But online retailers can stock everything virtually and thus the number of available niche products outnumbers the hits by several orders of magnitude. Online stores can virtually carry an infinite amount of inventory and are not relegated to the limitations of a physical store's square footage.

Those millions of niches classified as products of the Long Tail, are generally neglected in favor of the short head of hits. Selling small volumes of hard-to-find products to many customers instead of only selling large numbers of popular items could be a better strategy generating more profits. People gravitate towards niches because they cater to personal interests better. In an ideal scenario, most of the stores would stock up the popular products, but the stores which have a combination of both would attract more customers. Customers would then land to those stores and buy the popular as well as the narrowly-targeted products. Amazon is one the largest e-retailers in the world covering a wide range of products, storing millions of niche products. Thus, modifying the recommender system to suggest products from the long tail would be a win-win for both the sellers and the customers.

Providing customers with the products that suit their narrow taste or products that are good but not popular, meet the requirements of diversity and best products for a particular price.

Efforts are being invested in finding the size of the long tail and the products that belong to it. Also, can we find out how the availability of niche products changes the shape of the demand curve[3]? There is also interest in finding out which tools and techniques will drive that shift, if there are any. This is where the main problem statement of boosting recommendations from the long tail of the Amazon product dataset, revolves around. We want to avoid the concept of rich getting richer i.e. popular products get recommended more. It also helps some of the amazing, yet not so popular products, to reach the head, providing users with diverse options to choose from.

2. *Background*:

The concept of Long Tail has been explored in various domains. From ranking social networking sites to recommending products online, the factor of long tail has been considered and its usefulness has been determined.

The paper, *Long tail and How to Leverage it*[3] studies the Long Tail problem for recommender systems. The items in the tail have few ratings or reviews and are therefore harder to recommend. Our approach divides the product set into head and tail and then clusters the tail items. These tail products are then rated based on the ratings of the cluster they belong to, thus, solving the problem of recommending them based on ratings.

The paper, *Shedding Light on the Dark Data in the Long Tail*[4], applies the concept to the troublesome class of data. The class of data which is not properly indexed or stored becomes invisible to its potential users and remains underutilised. It describes solutions based on learning the concept of long tail for better curation of such data.

The paper, *From Niches to Riches: Anatomy of the Long Tail*[5], examines the long tail of products from the angles of demand and supply. It explains how the suppliers cater to the varied taste by using the long tail properties. It also explains how the market would be affected by the addition of several niche products and how the curve demand curve would change.

These are a few selected implementations that we have chosen from a broad range of available ones. The tools used are:

Pyspark is the Spark API in Python which exposes the Spark programming model to the python language.

MlLib: It is the machine learning library for Spark that is highly scalable. It includes several algorithms for classification, regression, clustering, topic modelling etc.

Spark is the engine used for processing Amazon product review and metadata dataset.

Spark SQL is the Spark library for working with structured data.

Numpy package was used to create graphs and also for the use of array and list manipulations.

3. *Data Source*:

The data source is the Amazon Product Reviews from 1996 to 2014[5] which contains the ratings of products from different domains like electronics, video games, beauty products etc. The data also contains metadata about the products and user review data. The metadata is about 3 GB in size whereas the review data is around 12GB in size when several categories are combined summing up to 140 million reviews. The data set has several features like productID, title, related, categories, reviewerID, helpful, overall, summary, unixReviewTime etc.

4. *Methods*:

The data set had multiple files therefore the first step was to join the two files with the productID as the primary key. The SQL library was used to perform the join. This was the pre-processing step.

The next step was to count the reviews of the product to estimate its popularity and plot a scatter-plot to see how the products were distributed. The scatter plot resembled the long tail graph. The products were then categorised into head and tail based on a particular threshold. Products with a higher count than threshold belonged to the head and the rest belonged to the tail.

Since the data had a few features, we applied feature engineering to generate two more features for the data. Those features were namely the average rating and the product popularity. On getting the head and tail products, similarity between the products was calculated. For this, TF-IDF was applied to the list of categories each product belonged to. This yielded a sparse vector which was feeded as an input to the K- Means clustering algorithm to form clusters of products belonging to the same category.

The next step is the recommendation of products from the long tail. So for any random product in the head similar products from the tail are recommended. To to this first the cluster of that product is determined. To the description of these products, TF-IDF is applied after removing the stopwords. The vector is then passed to the cosine similarity function which then returns the similarity between two products. The products which are similar beyond a certain threshold are recommended to the user when he/she searches for that selected head product. The threshold was given by (avg + 0.3*std)
Where avg is the average and std is the standard deviation.

Therefore we used the following methods for achieving the goal of long tail product recommendations:

- Spark SQL library
- Numpy , Matplotlib
- Spark MlLib library
    - TF-IDF
    - K- means

5. Results / Evaluation:

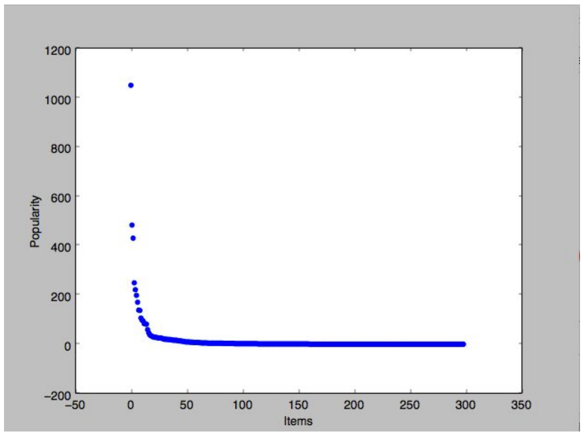


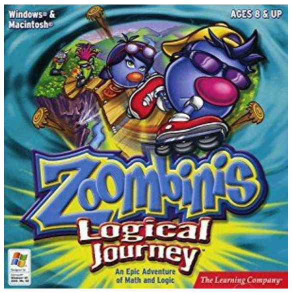
Figure 1: The Long Tail plot of video-games

We have plotted the graph which shows the head and tail products for Video Games. As we can see, there are few products in the head, while most of them lie in the tail. We started with this plot to visualize the concept of Long Tail.

After running our algorithm on the Video Game dataset, we have received the following results:

We have picked item "*Zoombinis*", a popular video game which has a lot of ratings. We are trying to find the video games from the tail which can be recommended along with this game.

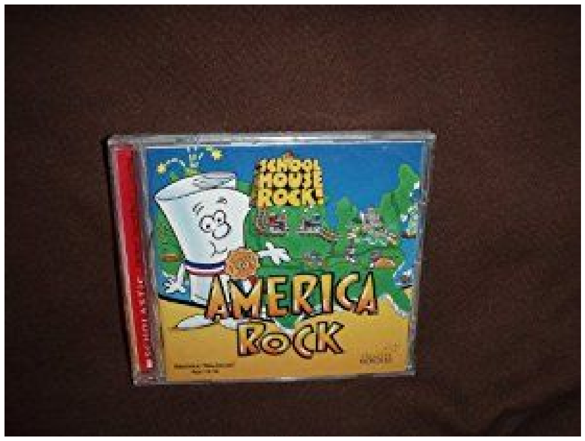
The output from our algorithm is:



The games recommended for the above video game are:

ASIN:0078764343 from TAIL

ASIN:'0545115507 from TAIL



ASIN: 0439394422 from TAIL

6. *Discussion*:

Our main problem statement is the recommendation of products of the long tail. Since we cannot test this online over a user base, we are comparing the productID of products which are initially in the long tail to see if they are now being recommended to customers or not. The results conclude that the long tail products are somewhere lost in the huge pile of products but with proper recommendation they can be made popular. The classification of products based on category finds clusters of products that belong to the same category, which is crucial for the recommendation. After this coarse level classification, the actual recommendation involves using cosine similarity calculation on the description feature of the two products getting compared hence giving better results.

7. *Conclusion:*

In this project, we have used the Amazon Product dataset and the Amazon Product Metadata dataset to find out segregate head and tail products. The threshold for determining the cut-off point is calculated based on popularity of the product. We then perform TF-IDF vectorization on the metadata of products, which helps in grouping products which are in the same category. Once we have performed vectorization, we perform K-Means clustering on these vectors. The clusters now contain both the head and tail items. The products are now grouped based on the cosine similarity of additional features like description of the product. We now pick one product from the head and try to find its similarity to other products in the cluster.

We have defined a threshold, above which products will be recommended. This threshold is based on the mean rating and the standard deviation of all the ratings for that product.

8. *References*:

[1] https://www.wired.com/2004/10/tail/

[2]http://www.twistimage.com/blog/archives/amazons-ever-growing-long-tail/

[3] Park, Yoon-Joo, and Alexander Tuzhilin. "The long tail of recommender systems and how to leverage it." *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008.

[4] Heidorn, P. Bryan. "Shedding light on the dark data in the long tail of science." *Library Trends* 57.2 (2008): 280-299.

[5] Brynjolfsson, Erik, Yu Jeffrey Hu, and Michael D. Smith. "From niches to riches: Anatomy of the long tail." *Sloan Management Review* 47.4 (2006): 67-71.

[5] http://jmcauley.ucsd.edu/data/amazon/

[6]http://dataconomy.com/an-introduction-to-recommendation-engines/