

**CISC7201 – INTRODUCTION TO DATA SCIENCE PROGRAMMING**

**1st Semester, 2019/2020**

Report of

**Amazon product data analysis**

**Recommendation System**

***Submitted by***:

Group: K

Wong Kit Kan - mb95513

Chio Chon Hou - mb95540

**INTRODUCTION**

In this project, we are going to build a recommendation system based on the products and rating detail of amazon China. In the recommendation systems, there are three common algorithms that is content-based filtering, neighborhood methods and latent factor models. And the latter two are collectively referred to as collaborative filtering. We decide using the transitional TF-IDF model as content-based filtering to rank the page (product name) and neighborhood methods to calculate the item-based and user-based ranking. An implementation of simple recommendation system and search engine.

**DATA SOURCE**

* From: Amazon China’s product data
* Data overview: (846 MB)
  + 20,000 products, 1,100 categories, 1.42 million users, 7.2 million reviews / rating data
* From: https://github.com/SophonPlus/ChineseNlpCorpus/blob/master/datasets/yf\_amazon/intro.ipynb

**DATA CLEANING**

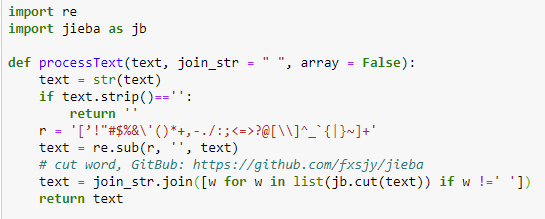
* Drop useless column
* Remove NaN product name/user id/rating data
* Remove punctuation(%$?) in product name
* Remove Extreme data (rating)
* Drop duplicate rating records

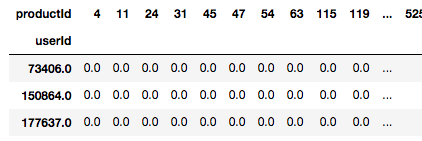
**DATA PROCESSING**

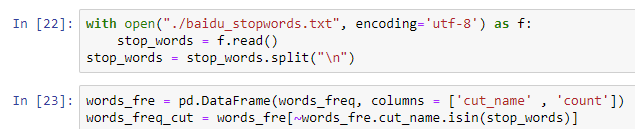
* Categories split
  + The original data in products csv, that is a column contain all the categories id level of the product. It is hard to analyze. So we split it into different column and join the categories csv and the result as below.



* Join table (join csv)
* Split the word (jieba)
  + In order to analyze the product name and the TFIDF model preparation, we need to separate the Chinese word. “jieba” library just helped us with this job. It has three segmentation mode, but our use case is the default model because of the TFIDF model. We don’t want to effect the term frequency. And the below function we are going to remove all the punctuation and split the Chinese word.

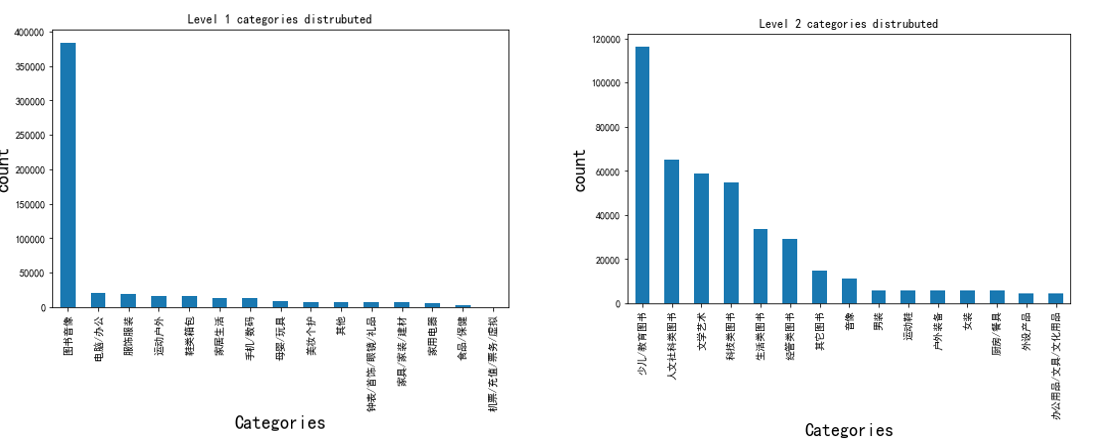


* Build user-items matrix
  + In order to build the recommend system, we based on the user-item-rating to find the similarity, so here we build the matrix: 
* Export the processed dataset (speed up)
  + We export the processed result to csv and avoid every time to do the calculation in memory.
* Cut stop word before the analyze (baidu stop word list)

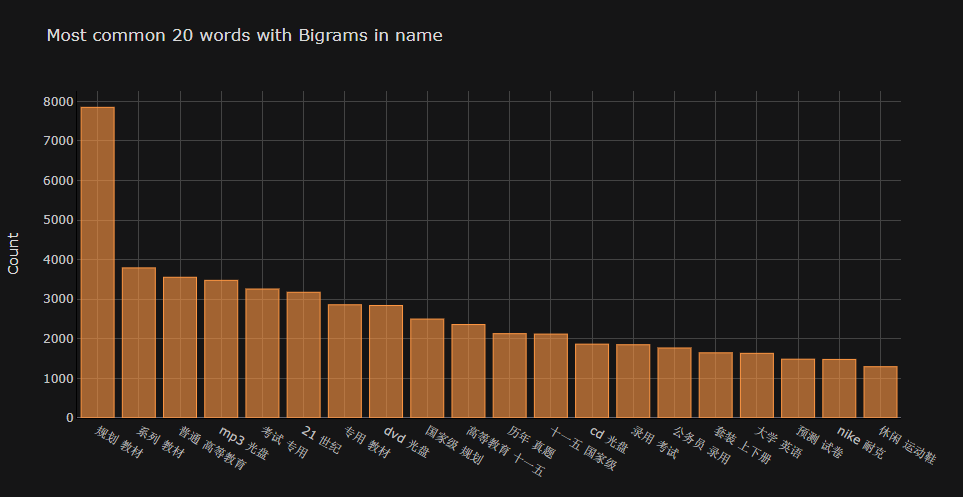


**DATA ANALYZE AND DATA VISUALIZSION**

* In our Amazon China product, almost product categories are Book. Second level categories mainly distributed in children and education book.

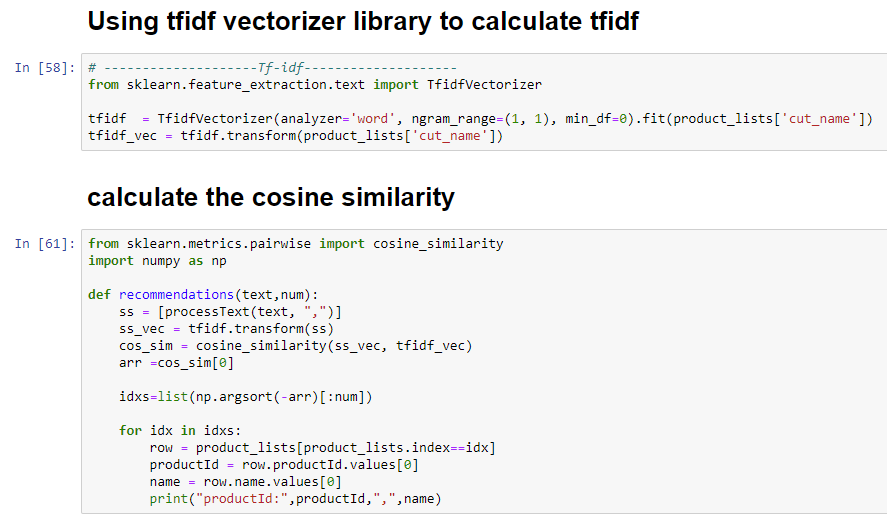


* Most common term calculate by Bigrams list:

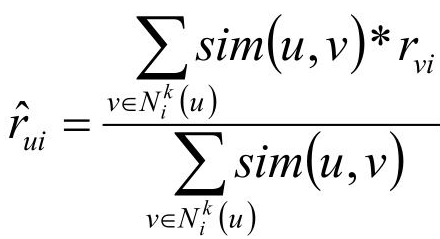
****

**DATA RECOMMENDATION**

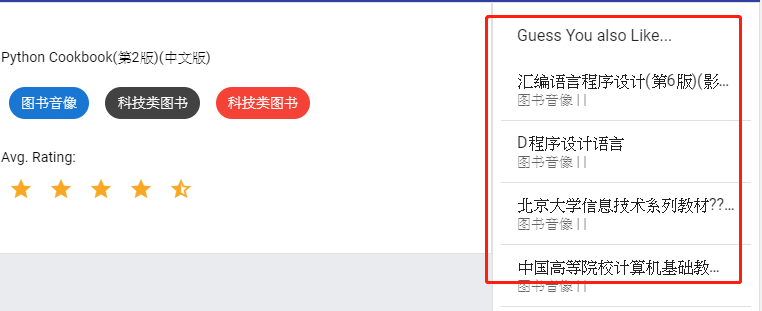
* In the recommendation system part one, we are using TF-IDF model to calculate the cosine similarly of the term and product name. After we use the “**jieba**” library help us segmentation, we will generate a new column name “cut\_name”. And then we use the **TfidfVectorizer** of **sklearn** library to use the TF-IDF model to calculate the vector. And also using the **cosine\_similarity** of the sklearn library to calculate the cosine similarity and sort it. Last will recommendation to the user.

* In Part two, we are using the user based collaborative filtering to find the recommend item. First we based to the user-item-rating matrix to build the 口relational tables between user and item. And we use the **sklearn** library to calculate the cosine similarity between users and find top-k similar users. Next, we use a simply weighted similar method1 to calculate the rating of the items. And we sort the answer. Finally we sort the rating and find the top-k item to recommend to the user.

………………………………………………………(1)

* In Part three, we are using the item based collaborative filtering to find the recommend item account to the searching item itself. The method is like used-based collaborative filtering. But we just transpose the user-item matrix. The row becomes item and the column becomes user. And we also use **sklearn** to find the top-k similar item. The result is visualize in our demo and presentation:



**DATA VISUALIZSION**