RECOMMENDATION SYSTEMS

1. Introduction

Retails are constantly increasing the amount of products available but sometimes it can cause time wasting for finding what one might really need. Techniques to increase sales of desired products and saving searching time were developed, such as recommendation systems (RS).

This paper aims to apply this engine in different datasets.

2. Recommendation Systems

2.1 Types of Recommendation Systems and Similarity Scores

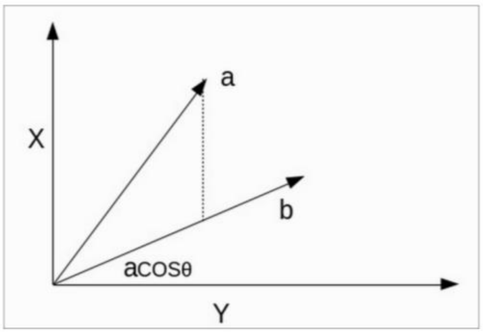
Recommendation Systems are used by companies' websites to suggest products for users by filtering the huge amount of options to only what they might like, resulting in time saving and, possibly, increase of profit. There are several of practical examples of it on people's daily use, such as Netflix recommending films and series that one might like (Sureshkumar Gorakala, 2016).

These systems can be applied on different bases, not being restricted to: content-based and collaborative filtering (Pramod Singh, 2019).

Content-based RS proposes products that are similar to the ones that a specific user has liked before. It is very accurate but fails in recommending products that do not relate to the content of what the user likes. On the other hand, collaborative filtering can be done on user-based, which makes recommendations based on what similar users like, or item-based, which uses the similarities between items' ratings. However, the "cold start" is a disadvantage of this system, once there is no data of new users/items, making recommendations poor (Sureshkumar Gorakala, (2016); Pramod Singh (2019)).

The neighbourhood of items and users is defined by their similarities scores, where a high score means that these points are similar to each other. It can be calculated by the Euclidean distance, Pearson correlation coefficient, Cosine or Jaccard similarity. For this project, Cosine similarity was used as it is the most common in RS. This method measures the cosine of the angle of two vectors and its result represent their similarity (Figure 1).

Figure 1: Cosine similarity.



Source: Sureshkumar Gorakala (2016).

2.2 Data Understanding and Preparation

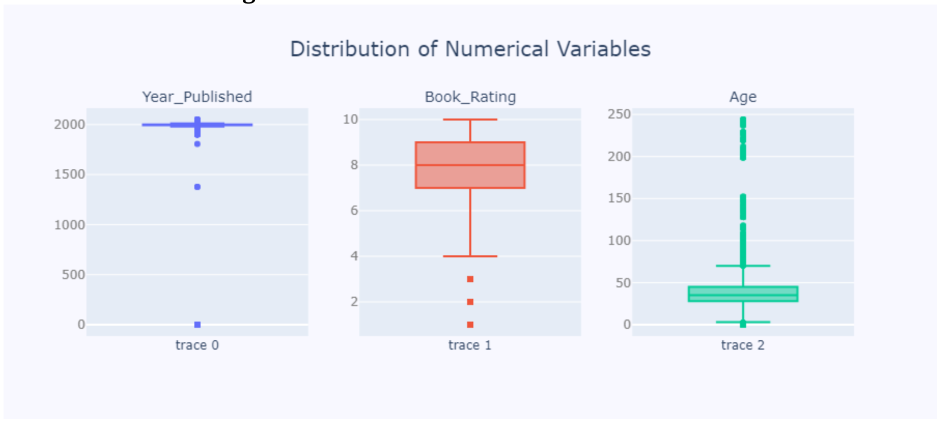
The data used for this analysis is a combination of three bases available at: "http://www2.informatik.uni-freiburg.de/~cziegler/BX/" and together they provide data of books with ratings and users' demographics. The final dataset contains 383,852 observations, 10 columns and there is some missing data, which was dropped to avoid creating synthetic data.

Table 1: Data Dictionary of Books Dataset.

Column Number	Column Name	Description	Data Type
0	ISBN	ID to identify books;	Object;
1	Book_Title	Titles of books;	Object;
2	Book_Author	Author(s) of books;	Object;
3	Publisher	Company that published the books;	Object;
4	Image1	URL to access the book covers;	Object;
5	User_ID	ID to identify users;	Integer;
6	Location	Where the user is from;	Object;
7	Year_Published	Year that books were published;	Integer;
8	Book_Rating	Rate gave by the user;	Integer;
9	Age	Age of the user.	Float.

Outliers were found and deleted because they represented inaccurate data (person aged as 250 years old, for instance).

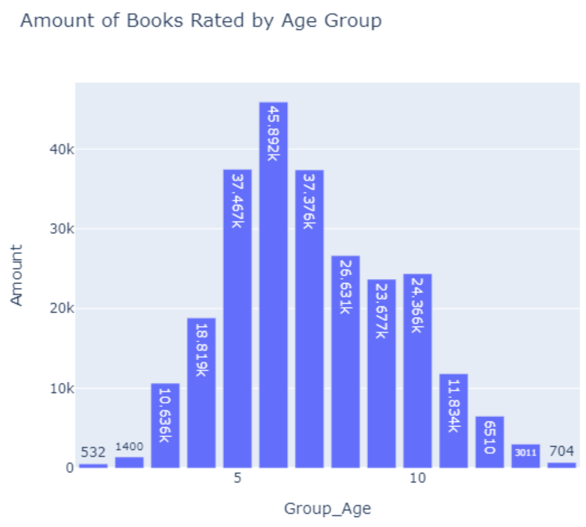
Figure 2: Distribution of Numerical Variables.

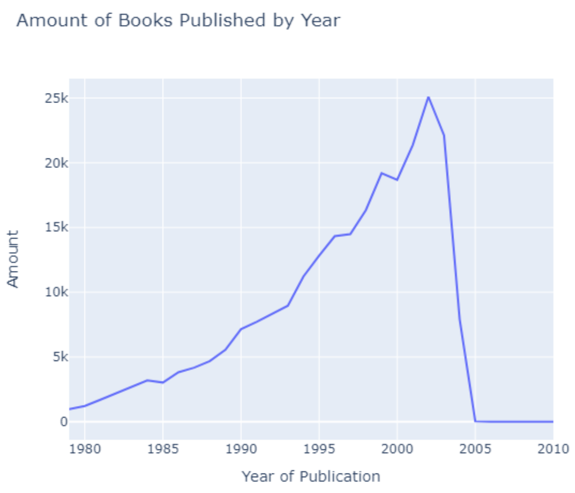


Source: Jupyter Notebook.

The most popular book on this data is “The Lovely Bones: A Novel”, Stephen King the most popular author and the year of publication varies from 1979 to 2010. The public that rates the most is between 23 and 38 years old and books published until the early 2000s were more likely to be rated than others.

Figure 3: Charts with data insights.

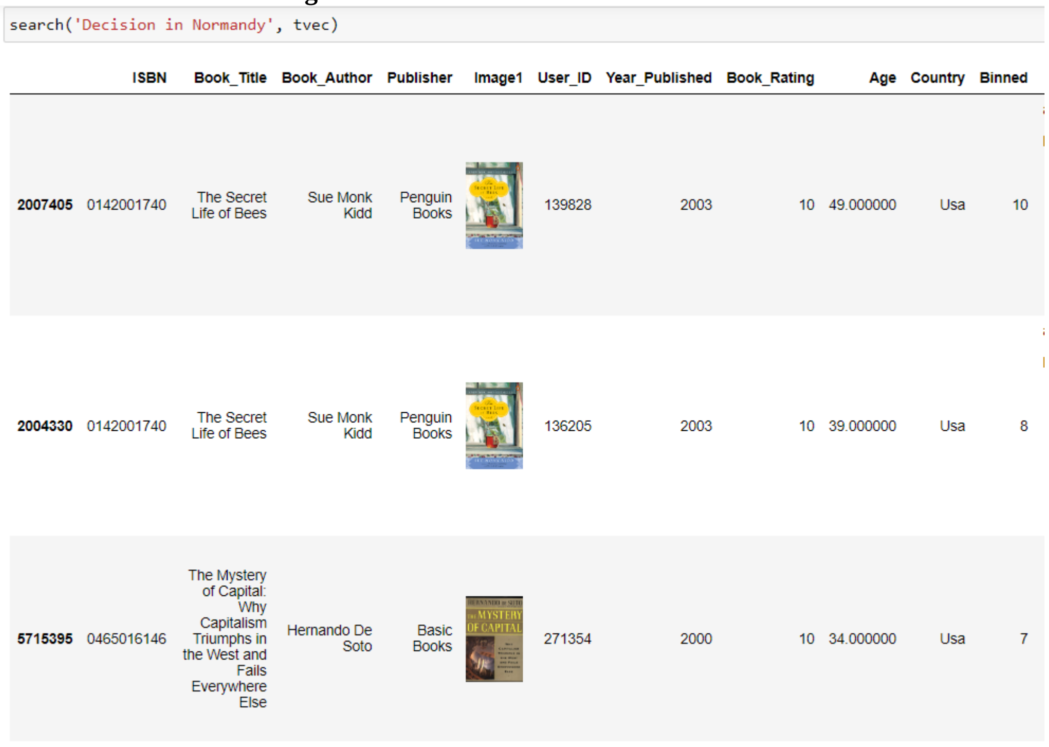




2.3 Modelling

The content-based recommendation was built by using the summary of each book, which was loaded from an external dataset, in order to suggest books with similar themes on top of just picking the titles and/or authors. They were prepared by removing initial spaces, punctuation and converting to lower string in order to make it more accessible for the TfidfVectorizer. This is an algorithm that helps machines to understand text by creating vectors of each document and transforming it into a matrix. Later, the cosine similarity of these vectors were calculated in order to find the most similar ones in terms of theme. When typing a book that you like on the search engine, it returns ones that you also might like.

Figure 4: Content-based Recommendations.



```
search('Decision in Normandy', tvec)
```

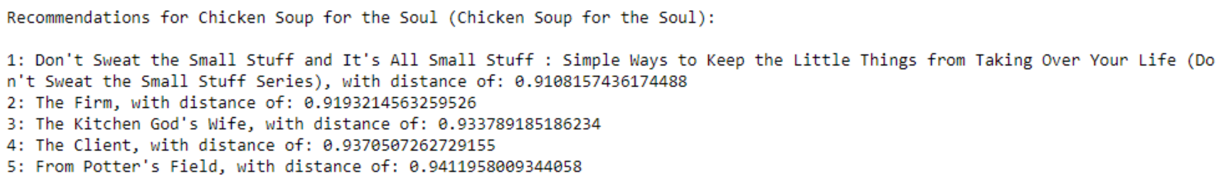
	ISBN	Book_Title	Book_Author	Publisher	Image1	User_ID	Year_Published	Book_Rating	Age	Country	Binned
2007405	0142001740	The Secret Life of Bees	Sue Monk Kidd	Penguin Books		139828	2003	10	49.000000	Usa	10
2004330	0142001740	The Secret Life of Bees	Sue Monk Kidd	Penguin Books		136205	2003	10	39.000000	Usa	8
5715395	0465016146	The Mystery of Capital: Why Capitalism Triumphs in the West and Fails Everywhere Else	Hernando De Soto	Basic Books		271354	2000	10	34.000000	Usa	7

Source: Jupyter Notebook.

On the other hand, collaborative filtering was based on ratings. First, only books rated at least 50 times were considered to avoid bias and to apply item-item. After that, a matrix to compare items, users and ratings was created and then transformed to sparse in order to reduce memory usage.

NearestNeighbors was the algorithm chosen for being known by good performances with recommender systems. It is an unsupervised learning method so that is not possible to calculate its accuracy.

Figure 5: Collaborative filtering recommendations.



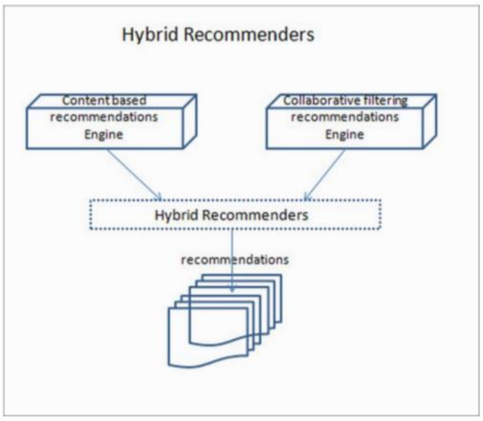
```
Recommendations for Chicken Soup for the Soul (Chicken Soup for the Soul):

1: Don't Sweat the Small Stuff and It's All Small Stuff : Simple Ways to Keep the Little Things from Taking Over Your Life (Do
n't Sweat the Small Stuff Series), with distance of: 0.9108157436174488
2: The Firm, with distance of: 0.9193214563259526
3: The Kitchen God's Wife, with distance of: 0.933789185186234
4: The Client, with distance of: 0.9370507262729155
5: From Potter's Field, with distance of: 0.9411958009344058
```

Source: Jupyter Notebook.

2.4 Deployment

It was concluded the possibility to recommend books similar to each other via two methods, even with their limitations. It is suggested as a further project to create a hybrid system which would overcome this issues and provide suggestions more meaningful and personalized to the users.

Figure 6: Example of Hybrid Recommender System.


Hybrid Recommenders
Content based recommendations Engine
Collaborative filtering recommendations Engine
Hybrid Recommenders
recommendations

Source: Sureshkumar Gorakala (2016).

3. Conclusion

This paper provided a practical application of RS and achieved its main goal, providing recommendations for stakeholders on how to increase their sales and which books they might like.

The datasets used provided a good amount of data for each purpose. Better results could be achieved by combining two types of RS which would overcome the limitations of current systems.

Overall, the results are good and were transmitted to stakeholders with appropriate design.

4. References

Ethem Alpaydin (2014). *Introduction to machine learning*. Cambridge, Massachusetts: The Mit Press.

Grus, J. and Media, O. (2017). *Data science from Scratch : [first principles with Python]*. [online] Bejing I Pozostałe: O’reilly. Available at: https://dl.acm.org/citation.cfm?id=2904392.

Osinga, D. (2018). *Deep Learning Cookbook*. ‘O’Reilly Media, Inc.’

Pramod Singh (2019). *Machine Learning with PySpark : With Natural Language Processing and Recommender Systems*. Berkeley, Ca: Apress.

Richert, W. and Luis Pedro Coelho (2013). *Building machine learning systems with Python : master the art of machine learning with Python and build effective machine learning systems with this* *intensive hands-on guide*. Birmingham: Packt Publishing.

Sureshkumar Gorakala (2016). *Building Recommendation Engines.* Packt.