Context

During the last few decades, with the rise of *Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).*

Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. As a proof of the importance of recommender systems, we can mention that, a few years ago, *Netflix organised a challenges (the "Netflix prize") where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win.*

By applying this simple dataset and related tasks and notebooks , we will evolutionary go through different paradigms of recommender algorithms . For each of them, we will present how they work, describe their theoretical basis and discuss their strengths and weaknesses.

Content

The Book-Crossing dataset comprises 3 files.

Users

Contains the users. Note that user IDs (User-ID) have been anonymized and map to integers. Demographic data is provided (Location, Age) if available. Otherwise, these fields contain NULL-values.

Books

Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (Book-Title, Book-Author, Year-Of-Publication, Publisher), obtained from Amazon Web Services. Note that in case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours (Image-URL-S, Image-URL-M, Image-URL-L), i.e., small, medium, large. These URLs point to the Amazon web site.

Ratings

Contains the book rating information. Ratings (Book-Rating) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

In [1]:
```
#import library
import pandas as pd
#for warning
```

```
import warnings
warnings.filterwarnings(action='ignore')
```

Item-Based Collaborative Filtering

Collaborative filtering methods are used to determine a user's level of interest in any product and to make recommendations by filtering products accordingly.

Product-based filtering, on the other hand, is a method that detects product similarities based on user votes. That is to say, for example, there are movies that show a similar liking structure with a movie that the person watches by being removed from being an object of the method. Similar movies can be found by finding similar reactions that other viewers collectively give to different movies. The movies with the highest correlation are selected and presented to the user as a recommendation.

In [2]:
```
# Loading the Datasets
df_book=pd.read_csv(r"D:\master of science\algorithm\project\book rec Sys\Books.csv"
df_user=pd.read_csv(r"D:\master of science\algorithm\project\book rec Sys\Users.csv"
df_rate=pd.read_csv(r"D:\master of science\algorithm\project\book rec Sys\Ratings.cs
```

In [3]:
```
#merge three DataFrame and create two columns
df1=df_book.merge(df_rate,how="left", on="ISBN")
df_merge=df1.merge(df_user,how="left", on="User-ID")
#show df_merge table
df_merge
```

Out[3]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.c
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.c
2	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.c
3	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.c
4	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.c
...	...	...	...	...	...	
1032340	0440400988	There's a Bat in Bunk Five	Paula Danziger	1988	Random House Childrens Pub (Mm)	http://images.amazon.c
1032341	0525447644	From One to One Hundred	Teri Sloat	1991	Dutton Books	http://images.amazon.c

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	
1032342	006008667X	Lily Dale : The True Story of the Town that Ta...	Christine Wicker	2004	HarperSanFrancisco	http://images.amazon.(
1032343	0192126040	Republic (World's Classics)	Plato	1996	Oxford University Press	http://images.amazon.(
1032344	0767409752	A Guided Tour of Rene Descartes' Meditations o...	Christopher Biffle	2000	McGraw-Hill Humanities/Social Sciences/Languages	http://images.amazon.(

1032345 rows × 12 columns

step 1 : data information

In [4]:
```
df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1032345 entries, 0 to 1032344
Data columns (total 12 columns):
 #   Column               Non-Null Count    Dtype  
---  ------               --------------    -----  
 0   ISBN                 1032345 non-null  object 
 1   Book-Title           1032345 non-null  object 
 2   Book-Author          1032344 non-null  object 
 3   Year-Of-Publication  1032345 non-null  object 
 4   Publisher            1032343 non-null  object 
 5   Image-URL-S          1032345 non-null  object 
 6   Image-URL-M          1032345 non-null  object 
 7   Image-URL-L          1032341 non-null  object 
 8   User-ID              1031136 non-null  float64
 9   Book-Rating          1031136 non-null  float64
 10  Location             1031136 non-null  object 
 11  Age                  753301 non-null   float64
dtypes: float64(3), object(9)
memory usage: 102.4+ MB
```

In [5]:
```
# Data null sum value
df_merge.isnull().sum()
```

Out[5]:
```
ISBN                      0
Book-Title                0
Book-Author               1
Year-Of-Publication       0
Publisher                 2
Image-URL-S               0
Image-URL-M               0
Image-URL-L               4
User-ID                1209
Book-Rating            1209
Location               1209
```

```
Age                  279044
dtype: int64
```

Step 2 : Delete non-existent values

(especially user rating which is very important in this algorithm)

In [6]:
```
#delete nan value
df1=df_merge.dropna(subset = ['User-ID' ,'Book-Rating','Location'])
#replace Nan Value with median
df=df1.interpolate(subset=['Age'])
```

step 3 : Data interference repair

In [7]:
```
#Converting User-ID and Age variable types to int
df['User-ID'] = df['User-ID'].astype('int')
df['Age'] = df['Age'].astype('int')
```

In [8]:
```
#Author of The Da Vinci Code written in two different ways
df["Book-Author"]=df["Book-Author"].astype("string")
df["Book-Author"]=df["Book-Author"].str.replace("DAN BROWN","Dan Brown")
```

In [9]:
```
#Author of Dreamcatcher book correcting incorrect entries
df["Book-Author"]=df["Book-Author"].str.replace("Audrey Osofsky","Stephen King")
df["Book-Author"]=df["Book-Author"].str.replace("Dinah McCall","Stephen King")
```

step 4 : Extracting and removing The trivial element

In [10]:
```
#Extracting Image URL from dataset
df.drop(columns=["Image-URL-S","Image-URL-M"],inplace=True)
```

In [11]:
```
#Removing books with zero ratings from the Datasets
df=df[df["Book-Rating"]>0]
df["Book-Rating"].describe()
```

Out[11]:
```
count    383842.000000
mean          7.626701
std           1.841339
min           1.000000
25%           7.000000
50%           8.000000
75%           9.000000
max          10.000000
Name: Book-Rating, dtype: float64
```

Counting : unique reader , unique books

In [12]:
```
#Unique reader count
df["User-ID"].nunique()
```

Out[12]:
```
68091
```

In [13]:
```
#Unique number of books
df["Book-Title"].nunique()
```

Out[13]:
```
135567
```

In [14]:
```
#We found how many books users read
df.groupby('User-ID')['Book-Title'].agg('count').sort_values()
```

Out[14]:
```
User-ID
138845       1
141631       1
141640       1
141641       1
141645       1
           ...
23902     1180
153662    1845
189835    1899
98391     5691
11676     6943
Name: Book-Title, Length: 68091, dtype: int64
```

In [15]:
```
#How many times have we read which book?
book_counts = pd.DataFrame(df["Book-Title"].value_counts())
```

In [16]:
```
#Most read books
book_counts.sort_values("Book-Title", ascending=False)
```

Out[16]:

	Book-Title
The Lovely Bones: A Novel	707
Wild Animus	581
The Da Vinci Code	494
The Secret Life of Bees	406
The Nanny Diaries: A Novel	393
...	...
Little angels	1
The Assassins of Tamurin	1
Dawn's Early Light (The Williamsburg Novels)	1
Celebration In Purple Sage	1
The Cultures of Native North Americans	1

135567 rows × 1 columns

step 5 : Removing rare books and found the widely read books

In [17]:
```
#We named the books with less than 100 reads as rare books.
rare_book = book_counts[book_counts["Book-Title"] <= 100].index
```

In [18]:
```
#Number of rarely read books
rare_book.nunique()
```

Out[18]:
```
135375
```

In [19]:
```
#By removing the rare books from the dataset, we found the widely read books
common_book = df[~df["Book-Title"].isin(rare_book)]
common_book
```

Out[19]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	
31	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com/images
32	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com/images
34	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com/images
36	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com/images
37	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com/images
...	...	...	...	...	...	
1009222	9506440298	Fahrenheit 451	Ray Bradbury	2003	Plaza & Janes Editores, S.A.	http://images.amazon.com/images
1011705	0385722206 0	Balzac and the Little Chinese Seamstress : A N...	DAI SIJIE	2002	Anchor	http://images.amazon.com/images
1013704	1565115392	The Secret Life of Bees	Sue Monk Kidd	2002	Penguin Highbridge	http://images.amazon.com/images
1013961	0891097686	The Summons	Dennis McCallum	1993	Navpress	http://images.amazon.com/images
1032154	0395647398	The Two Towers (The Lord of the Rings, Part 2)	J. R. R. Tolkien	1992	Houghton Mifflin	http://images.amazon.com/images

33641 rows × 10 columns

User-Book matrix

In [20]:
```
#User-Book matrix
user_book_df = common_book.pivot_table(index=["User-ID"], columns=["Book-Title"], va
user_book_df
```

Out[20]:

User-ID	Book-Title	1984	1st to Die: A Novel	2nd Chance	A Bend in the Road	A Child Called \It\": One Child's Courage to Survive"	A Heartbreaking Work of Staggering Genius	A Is for Alibi (Kinsey Millhone Mysteries (Paperback))	A Map of the World	A Painted House	Praye fc Owe Mean
16		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
26		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
32		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
51		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
91		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...		...	...	...	...	...	...	...	...	...	
278800		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
278836		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
278843		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
278844		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
278846		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

16397 rows × 192 columns

Approximately finished

Now choose a book (input)

In [21]:
```
#We chose a book
book_name="1984"
```

In [22]:
```
#We found the points given to the book
book_name=user_book_df[book_name]
book_name.sort_values(ascending=False)
```

Out[22]:
```
User-ID
275520    10.0
164008    10.0
61842     10.0
1706      10.0
207635    10.0
          ...
278800     NaN
```

```
278836     NaN
278843     NaN
278844     NaN
278846     NaN
Name: 1984, Length: 16397, dtype: float64
```

In [23]:
```
#found similar books and sort values
user_book_df.corrwith(book_name).sort_values(ascending=False).head(5)
```

Out[23]:
```
Book-Title
1984                              1.0
The Cider House Rules             1.0
I Know This Much Is True          1.0
Tara Road                         1.0
Seabiscuit: An American Legend    1.0
dtype: float64
```

Recommand book

In [24]:
```
# recommand book list
rec_book=user_book_df.corrwith(book_name).sort_values(ascending=False).head(5)
rec_book_list=list(rec_book.index)

rec_book_list
```

Out[24]:
```
['1984',
 'The Cider House Rules',
 'I Know This Much Is True',
 'Tara Road',
 'Seabiscuit: An American Legend']
```

Output

In [25]:
```
#Authors of 5 books we recommend
df_author=df[["Book-Title","Book-Author"]]
df_author.head()

df1 = df_author.loc[df_author["Book-Title"].isin(rec_book_list)]

df2=df1.drop_duplicates(subset=["Book-Author","Book-Author"], keep="first")
df2
```

Out[25]:

	Book-Title	Book-Author
4345	Seabiscuit: An American Legend	LAURA HILLENBRAND
63591	The Cider House Rules	John Irving
94370	1984	George Orwell
121773	I Know This Much Is True	Wally Lamb
178734	Tara Road	Maeve Binchy

In [26]:
```
df2.shape
```

Out[26]:
```
(5, 2)
```

Thank you for watch my code