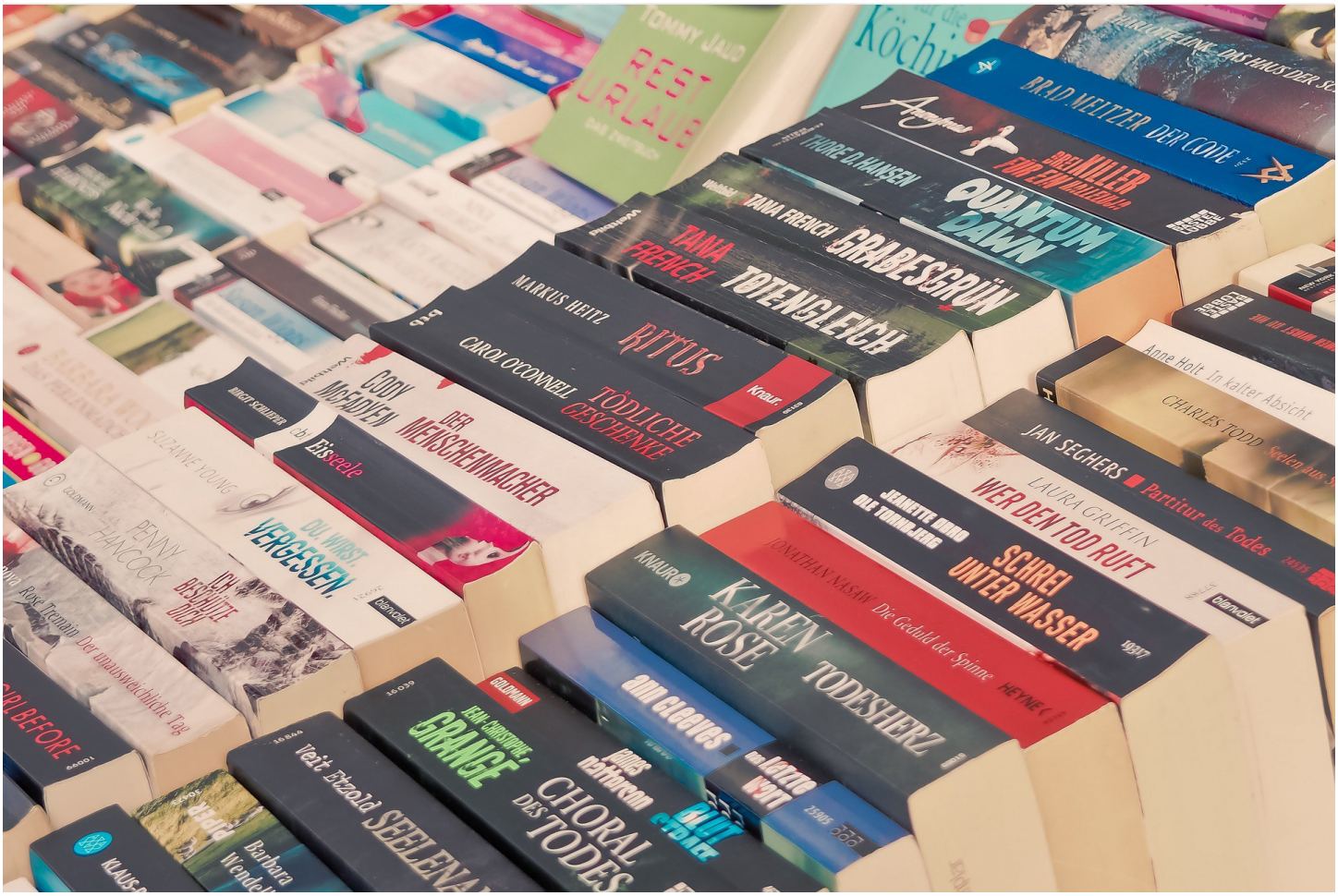

My Journey to building Book Recommendation System. . .

Chhavi Saluja Mar 23, 2018 · 7 min read

Recommendation systems have been keeping my mind occupied for quite a while, and owing to my inclination for reading books, exploring

Book Crossing dataset and very much engaging.

Online recommendation systems are the in thing to do for many e-commerce websites. A recommendation system broadly recommends products to customers best suited to their tastes and traits. For more details on recommendation systems, read my introductory post on Recommendation Systems and a few illustrations using Python.

My journey to building Book Recommendation System began when I came across Book Crossing dataset. This dataset has been compiled by Cai-Nicolas Ziegler in 2004, and it comprises of three tables for users, books and ratings. Explicit ratings are expressed on a scale from 1–10 (higher values denoting higher appreciation) and implicit rating is expressed by 0.

Before building any machine learning model, it is vital to understand what the data is, and what are we trying to achieve. Data exploration reveals the hidden trends and insights and data preprocessing makes the data ready for use by ML algorithms.

So, let's begin. . .

First, we load the dataset and check the shapes of books, users and ratings dataset as below:

```
users.columns = ['userID', 'Location', 'Age']
ratings = pd.read_csv('ratings.csv', sep=';', error_bad_lines=False, encoding="latin-1")
ratings.columns = ['userID', 'ISBN', 'bookRating']
```

```
print books.shape
print users.shape
print ratings.shape
```

```
(271360, 8)
(278858, 3)
(1149780, 3)
```

Books

Exploring each of these datasets one by one and beginning with books dataset, we can see that image URLs columns do not seem to be required for analysis, and hence these can be dropped off.

```
books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher	imageUrlS	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/019
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/000
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/006
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/037
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/039

```
books.drop(['imageUrlS', 'imageUrlM', 'imageUrlL'],axis=1,inplace=True)
```

```
books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company

display full text of columns.

```
books.dtypes
```

```
ISBN                 object
bookTitle            object
bookAuthor           object
yearOfPublication    object
publisher            object
dtype: object
```

```
pd.set_option('display.max_colwidth', -1)
```

yearOfPublication

Now we check the unique values for this attribute.

```
books.yearOfPublication.unique()
```

```
array([2002L, 2001L, 1991L, 1999L, 2000L, 1993L, 1996L, 1988L, 2004L,
       1998L, 1994L, 2003L, 1997L, 1983L, 1979L, 1995L, 1982L, 1985L,
       1992L, 1986L, 1978L, 1980L, 1952L, 1987L, 1990L, 1981L, 1989L,
       1984L, 0L, 1968L, 1961L, 1958L, 1974L, 1976L, 1971L, 1977L, 1975L,
       1965L, 1941L, 1970L, 1962L, 1973L, 1972L, 1960L, 1966L, 1920L,
       1956L, 1959L, 1953L, 1951L, 1942L, 1963L, 1964L, 1969L, 1954L,
       1950L, 1967L, 2005L, 1957L, 1940L, 1937L, 1955L, 1946L, 1936L,
       1930L, 2011L, 1925L, 1948L, 1943L, 1947L, 1945L, 1923L, 2020L,
       1939L, 1926L, 1938L, 2030L, 1911L, 1904L, 1949L, 1932L, 1928L,
       1929L, 1927L, 1931L, 1914L, 2050L, 1934L, 1910L, 1933L, 1902L,
       1924L, 1921L, 1900L, 2038L, 2026L, 1944L, 1917L, 1901L, 2010L,
       1908L, 1906L, 1935L, 1806L, 2021L, u'2000', u'1995', u'1999',
       u'2004', u'2003', u'1990', u'1994', u'1986', u'1989', u'2002',
       u'1981', u'1993', u'1983', u'1982', u'1976', u'1991', u'1977',
       u'1998', u'1992', u'1996', u'0', u'1997', u'2001', u'1974', u'1968',
       u'1987', u'1984', u'1988', u'1963', u'1956', u'1970', u'1985',
       u'1978', u'1973', u'1980', u'1979', u'1975', u'1969', u'1961',
       u'1965', u'1939', u'1958', u'1950', u'1953', u'1966', u'1971',
       u'1959', u'1972', u'1955', u'1957', u'1945', u'1960', u'1967',
       u'1932', u'1924', u'1964', u'2012', u'1911', u'1927', u'1948',
       u'1962', u'2006', u'1952', u'1940', u'1951', u'1931', u'1954',
       u'2005', u'1930', u'1941', u'1944', u'DK Publishing Inc', u'1943',
       u'1938', u'1900', u'1942', u'1923', u'1920', u'1933', u'Gallimard',
       u'1909', u'1946', u'2008', u'1378', u'2030', u'1936', u'1947',
       u'2011', u'2020', u'1919', u'1949', u'1922', u'1897', u'2024',
       u'1376', u'1926', u'2037'], dtype=object)
```

incorrectly loaded as yearOfPublication in dataset due to some errors in csv file. Also, some of the values are strings and same years have been entered as numbers at some places. We will make necessary correction for these rows and set the data type for yearOfPublication as int.

It can now be seen that yearOfPublication is of type int and it has values ranging from 0–2050. As this dataset was built in 2004, I am assuming all the years after 2006 are invalid keeping a margin of two years in case dataset may have been updated. For all the invalid entries

publisher

Coming to ‘publisher’ column, I have handled two NaN values by replacing them with ‘other’ as publisher name could not be inferred after some investigations (check jupyter notebook embed).

Users Dataset

Now we explore users dataset, firstly by checking its shape, first few columns and data types.

Age

Upon checking the unique values, userID looks correct. However, Age column has a NaN and some very high values. In my view ages below 5 and above 90 do not make much sense, and hence, these are being replaced with NaNs. All the NaNs are then replaced with mean value of Age, and its data type is set as int.

I am not doing any processing of Location column here. However, if you wish you can further split this into city, state, country and do some processing using text processing models.

Ratings Dataset

are quite less as compared to size of ratings matrix (number of users × number of books).

Now ratings dataset should have userID and ISBN which exist in respective tables, viz. users and books.

It is evident that, users have rated some of the books, which are not part of original books dataset. Sparsity of the dataset can be calculated as below:

The explicit ratings represented by 1–10 and implicit ratings represented by 0 will have to be segregated now. We will be using only explicit ratings for building our book recommendation system. Similarly, users are also segregated into those who rated explicitly and those whose implicit behavior was recorded.

A countplot of bookRating indicates that higher ratings are more common amongst users and rating 8 has been rated highest number of times.

Simple Popularity based Recommendation System

At this point, a simple popularity based recommendation system can be built based on count of user ratings for different books. It is evident that books authored by J.K. Rowling are quite popular.

Collaborative Filtering based Recommendation System

To cope up with computing power my machine has and to reduce the dataset size, I am considering users who have rated at least 100 books and books which have at least 100 ratings.

Next key step in building CF-based recommendation systems is to generate user-item ratings matrix from the ratings table.

Notice that most of the values in ratings matrix are NaNs indicating absence of ratings and hence sparsity of data. Also, note that only explicit ratings have been considered here. As most of the machine learning algorithms cannot handle NaNs, we replace them with 0, which now indicates absence of rating.

User-based CF

I will be reusing the functions from my post CF based Recommendation Systems Exemplified. The function **findksimilarusers** inputs userID and ratings matrix and returns similarities and indices of k similar users. (Read my previous stories to understand the concept and formulae of user/item based CF approaches)

The function **predict_userbased** predicts rating for specified user-item combination based on user-based approach.

The function **recommendItem** uses above functions to recommend books for user-based or item-based approach (based on selected approach and metric combination). Recommendations are made if the predicted rating for a book is greater than or equal to 6, and the books have not been rated already. You can select the similarity metric (**cosine/ correlation**) while calling this function.

Item-based CF

Similar functions have been written for Item-based CF to find k similar books and predict the user's ratings for every books. Same function **recommendItem** can be used to recommend books based on item-based approach and selected metric. Recommendations are made if the predicted rating for a book is greater than or equal to 6, and the books have not been rated already.

Wow!!! Check the top 10 book recommendations for user 4385 based on item-based CF approach. These are significantly different from those suggested by user-based approach.

worth exploring. Jupyter notebook for this code is embedded below.

Thanks for reading! I hope you liked this article. Please share your views in comments section below. Meanwhile, I will go and check some book recommendations for myself.

References:

1. https://towardsdatascience.com/how-did-we-build-book-recommender-systems-in-an-hour-the-fundamentals-dfee054f978e

2. https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

Machine Learning Data Science Recommendation System

Recommender Systems Collaborative Filtering