Book Recommendations Engine

Final Project – DAT_SF_13

Mandip Shah

May 20, 2015

Recommender systems have become fundamental part of sales of online services and content delivery; Books are one of the interesting problems!

There are millions of books!

- There are approximately 130 MM books globally!
- There are many more millions articles, papers, journals, etc. are available for consumers!

It takes a long time read one!

- Selection is important. On an average a reader takes around 6 hours to read a book, which is much higher than shopping or movies
- Only 38% of readers finish a book they started

It is quite a bit choice driven!

- Of course!

Great product to sale online!

- Only Amazon's Book sales is over $6 BN a year!
- Very cost effective to manage inventory
- Easy to deliver at home or online

Building a recommender system for books is an interesting machine learning problem

There are three components of the Book-Crossing data sets – Books, Users and Ratings; Additional data was pulled through API calls and web scrapping

Books*

Attributes:

- ISBN
- Book Title
- Book Author
- Year-of-publication
- Publisher
- Cover Image URL

Sample size:

- Total - 271K

Users*

Attributes:

- User ID
- Age
- Location

Sample size:

- Total - 278K

Ratings*

Attributes:

- User ID
- ISBN
- Rating (Scale 1-10)

Sample size:

- Total – 1.14MM

API Pulls

Amazon Product API:

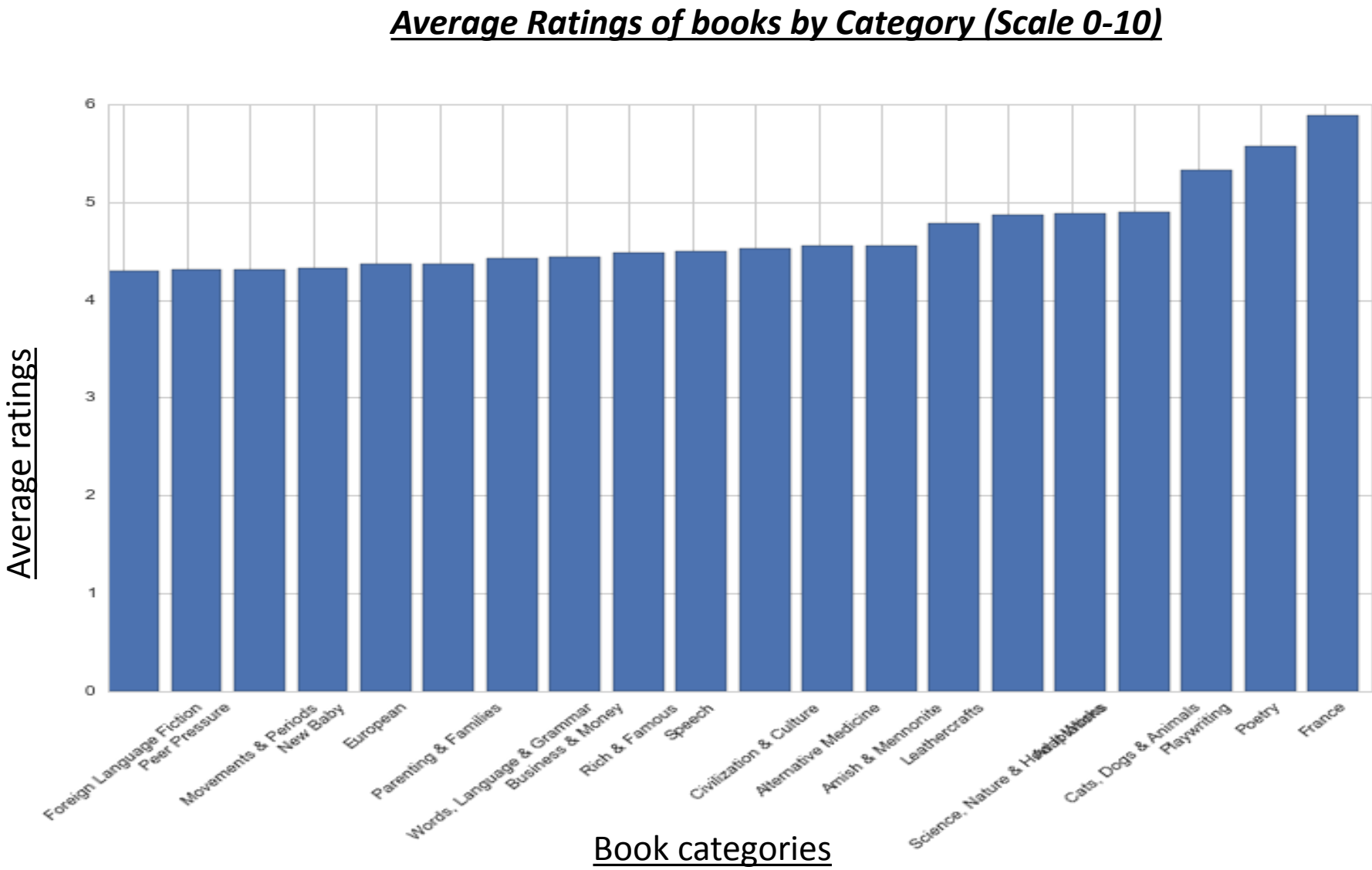
- Category/Genre of books

Amazon Website:

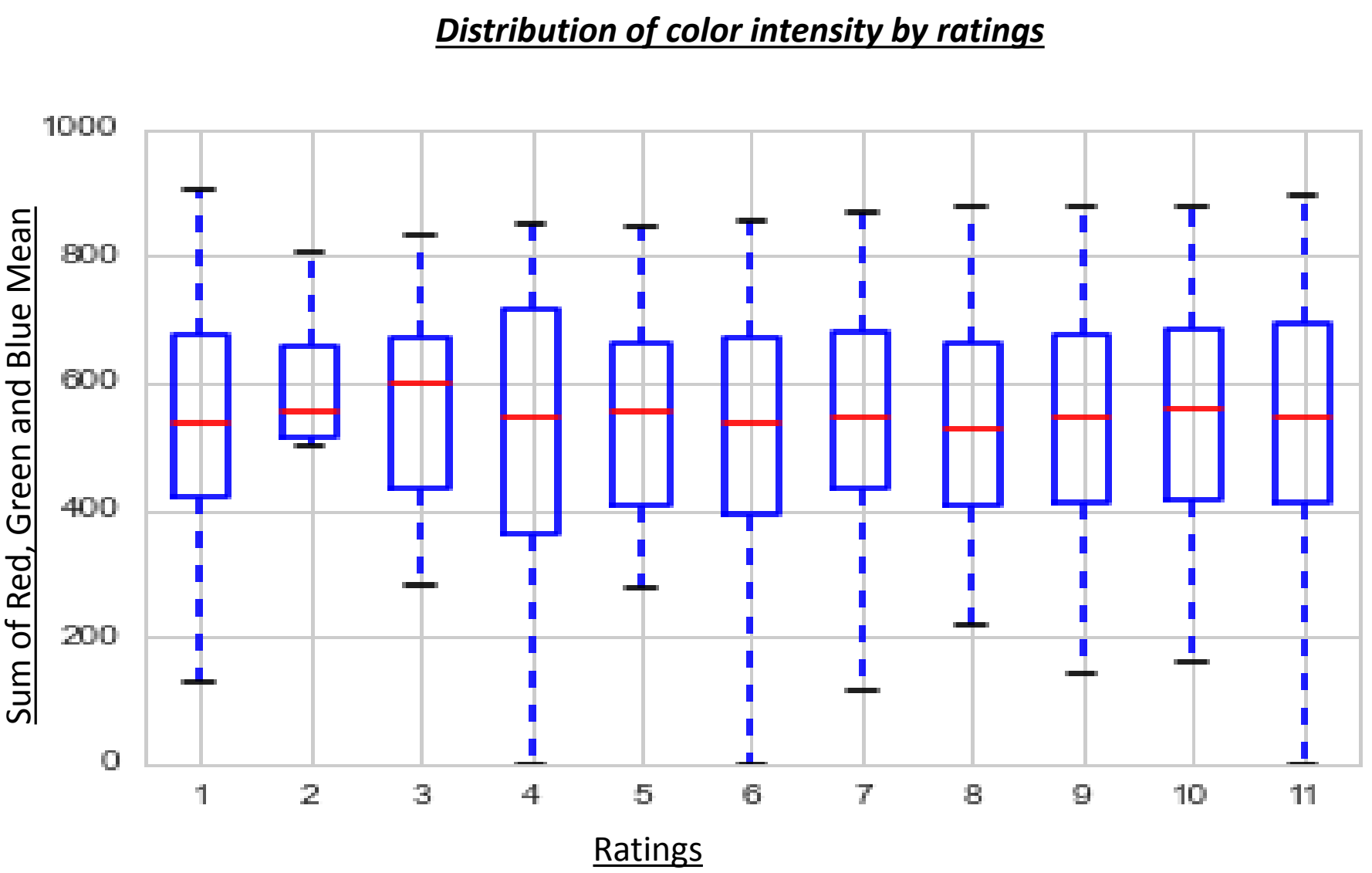
- Image

*Collected by Cai-Nicolas Ziegler in a 4-week crawl (August / September 2004) from the Book-Crossing community with kind permission from Ron Hornbaker, CTO of Humankind Systems.

Book Genre/Categories seem to have solid influence on how the books might be rated; It seems content based filtering might also work



Initial results suggest that Book cover page colors do not have much influence on the book ratings which confirms that image may not help recommendations



Collaborative filtering and Hybrid of Collaborative/Content based filtering applied to minimize "Root Mean Squared Error (RSME)" on the test sample

Collaborative Filtering

Step1: Solve SVD with these books and users directly

Books User	Book1	Book2	Book3	Book I	Book 5
U1	5	8		7	8
U2	10		1		
U3	2		10	9	9
U4		2	9	9	10
U5	1	5			1

Book_1 book will also be recommended to folks who liked Book_5

Hybrid of Collaborative and Content Based Filtering

Step1: Develop clusters of books based on book level features (mix of numeric and categorical values) e.g. category, author, publisher, year of publication, etc.

Step2: Solve SVD with these clusters instead of books

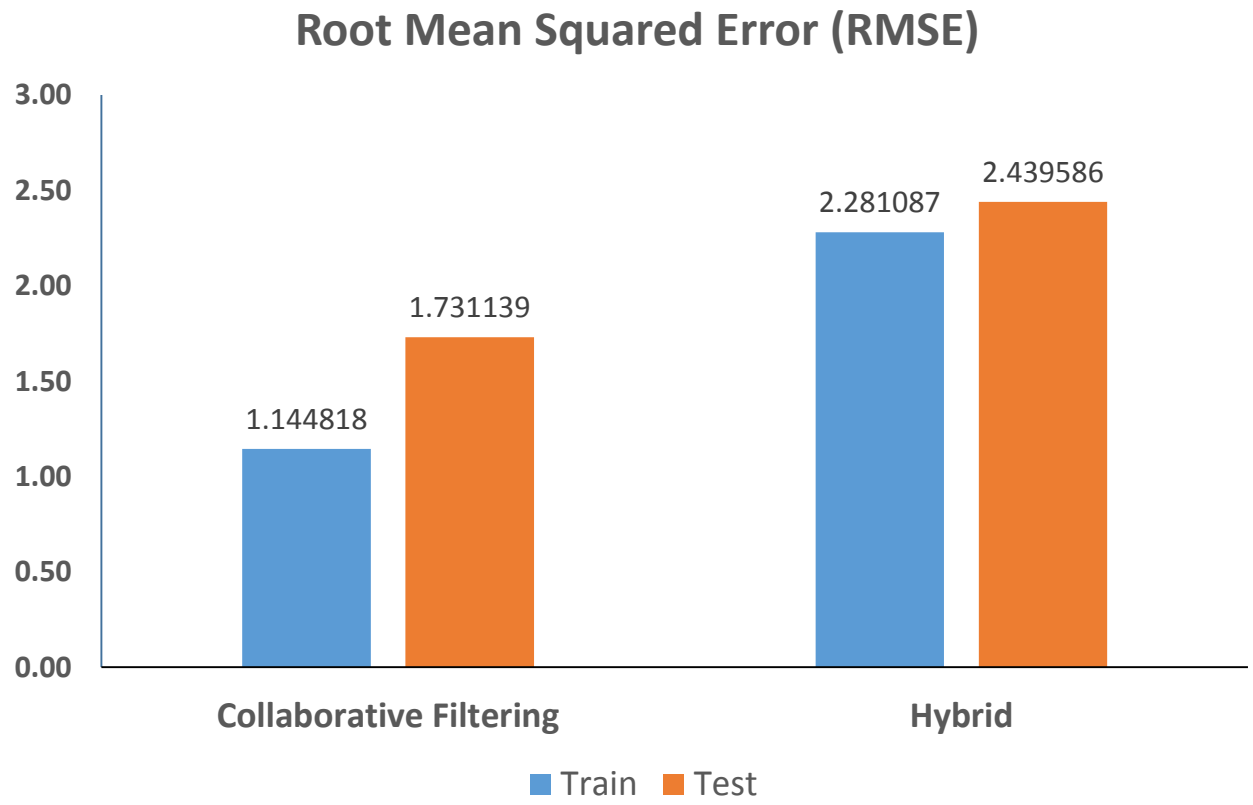
Books User	Clust1	Clust2	Clust9
U1	5.5	8	6.5
U2	9		8.1
U3	4.8		9
U4		2	4.5
U5	1	5	1

Cluster_9 books will also be recommended to Cluster_1 folks

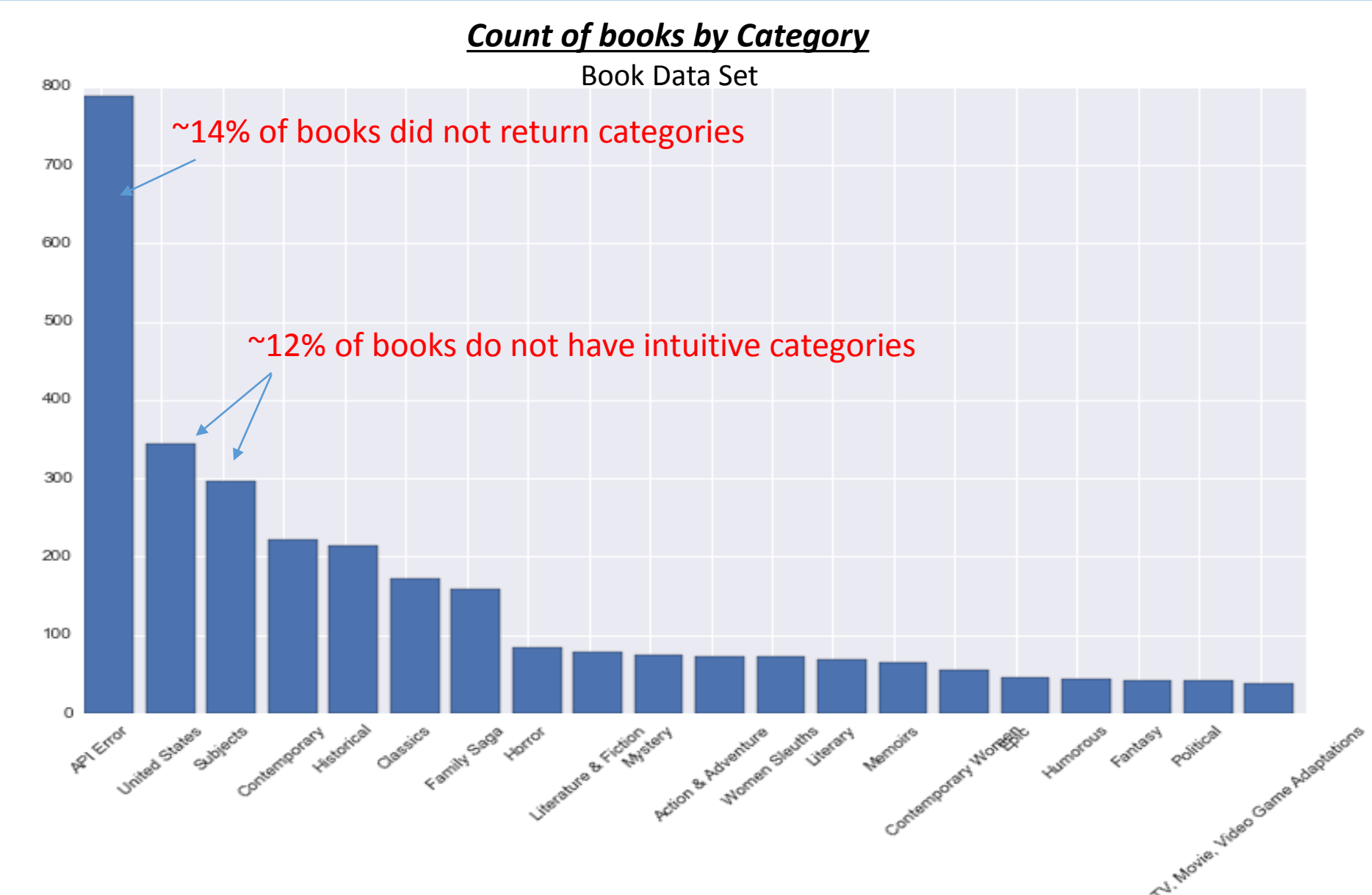
Collaborative filtering performed better than Hybrid approach; Hybrid approach is intuitive, has a reasonable accuracy and can give a good cold start

For all books RMSE is calculated on the Train and Test samples where users have rated those books.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2. \qquad RMSE = \sqrt{MSE}$$



Content based filtering may not produced sound results because of lack of quality data; Category fields do have “Errors” or “Unintuitive values”



Packages and tools used to build book recommendation system

Tools:

- IPythonNotebook (Anaconda) and Spyder

Packages:

- Basic python packages for DS – *Numpy, Pandas, Scikit-learn*
- Plotting – *Matpotlib, PyPlot, Seaborn*
- Clustering – *kmodes*
- Image processing – *Scikit-Image*
- Recommender systems – *python-recsys, divisi2, csc-pysparse, networkx*
- API calls – *python-amazon-product*
- Misc. – *random, math, datetime, urllib*