# Building a Book Recommender system using Collaborative Filtering

Ankita Vinod Ahir, Abhinaya Krishna Mandepudi, Saikrishna Kanukuntla, Michael Mugo
Department of Computer Science, University of Texas at Dallas,
axa165030@utdallas.edu,axm163231@utdallas.edu,sxk160432@utdallas.edu,mkm140630@utdallas.edu

**Abstract— Today, many Websites are using recommendation systems to provide relevant suggestions regarding the product interest of their customers. Various parameter contributes to the recommendation system such as the item popularity, past purchase behavior of the top customers, user characteristics such as geographical location or demographic information. They are widely used to recommend products to the end users that are most appropriate. Online book selling websites has also adopted the recommendation system approach to enhance the user experience. It is one of the stronger tools to increase profit and retaining buyer. The book recommendation system must recommend books that are of buyer's interest. Our project is on book recommendation system based on ALS collaborative filtering**

**Index Terms— Collaborative Technique, Recommendation Engine, User's Interest, Spark, PySpark, Flask, CherryPy**

## I. INTRODUCTION

A Recommendation Engine, in actual definition can be referred to as a system that can run on clustered / non-clustered environment taking user online footprint as one of its input set and generating a probable footprint for the user thereby providing its users a prediction closer to reality. Recommendation Engines require a large dataset and a fast computing system that can perform analytics on the same within fraction of seconds. Recommendation Engines, in simpler terms are programs that are data intensive and involve complex pattern matching on a set of predefined parameters and they become efficient with the increase in the size of the content being fed to hem. Recommender systems represent user preferences for suggesting items to purchase or examine. They have become fundamental applications in electronic commerce and information access, providing suggestions that effectively prune large information spaces so that users are directed toward those items that best meet their needs and preferences. A variety of techniques have been proposed till today for performing recommendations. The techniques such as content-based, collaborative, knowledge-based and demographic are used for recommendations. Sometimes, the features of these techniques are combined in hybrid recommenders to improve the performance of recommendation engine.

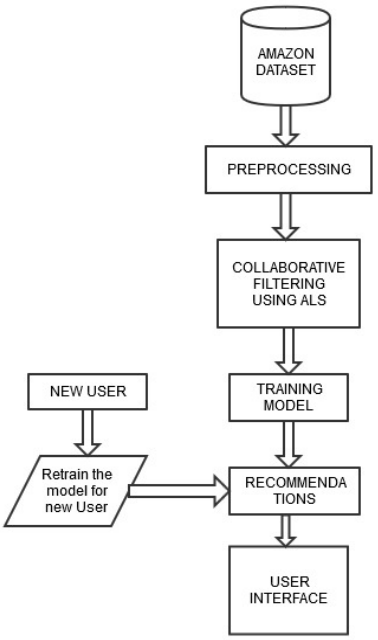
## 2. PROPOSED SYSTEM

A) Preprocessing of the dataset:

In this step, we process the book dataset into a structured dataset. In Mllib's collaborative filtering package, the format of the input is expected in the form UserId (Int), BookId (Int), and Rating (Int/Float). Data which we are using has the format of String (User ID) String, (Book ID) String and Rating (Int/Float). We mapped each string to a unique integer so that the package can accept the data without any issues. Secondly, in metadata of the books, there are some entries related to faulty books which had the title "204". The data was cleaned accordingly.

B) Collaborative Filtering:

We are using ALS Collaborative filtering technique for the recommendation of books. With the help of the rating given by the users, we try to fit in and predict the unknown rating in the user rating matrix. We build a training model from the collaborative filtering and based on the model we recommend the books to the user. The training model is retrained using the collaborative filtering and recommendations are given to the user.




## 2. DATASET

Our recommendation system is built on the Amazon Product Data. This dataset contains product reviews and

metadata from Amazon including reviews spanning from May 1996-July 2014.
The datasets was divided into different parts as follows:
We've used all of those for building our model.
**K-cores** (i.e., dense subsets): These data have been reduced to extract the k-core, such that each of the remaining users and items have k reviews each.
**Ratings only:** These datasets include no metadata or reviews, but only (user, item, rating, timestamp) tuples. (22,507,155 ratings)
**Metadata**: Metadata includes descriptions, price, sales-rank, brand info, and co-purchasing links: (2,370,585 products)

Metadata:


```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-
amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
```


k-core:


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the
piano.  He is having a wonderful time playing these old hymns.
The music  is at times hard to read because we think the book
was published for singing from more than playing from.  Great
purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


## 3. METHODS AND TECHNOLOGIES USED

### 1. Collaborative Filtering:

Collaborative recommendation is probably the most familiar, most widely implemented and most mature of the technologies. Collaborative recommender systems aggregate ratings or recommendations of objects, recognize commonalities between users based on their ratings, and generate new recommendations based on inter-user comparisons. A typical user profile in a collaborative system consists of a vector of items and their ratings, continuously augmented as the user interacts with the system over time. Some systems used time-based discounting of ratings to account for drift in user interests.

In our project, we are using the MLLib's collaborative filtering using alternating least squares(ALS) for building the recommendation system. The Als algorithm upon a certain number of iteration predicts the users ratings for the product which was previously unrated.

### 2. Spark

Apache Spark is an open source project that has gained attention from analytics experts. In contrast to Hadoop MapReduce's two-stage, disk-based paradigm, Spark's multi-stage in-memory primitives provides performance up to 100 times faster for certain applications. Since it is possible to load data into a cluster's memory and query it repeatedly, Spark is commonly used for iterative machine learning algorithms at scale. Furthermore, Spark includes a library with common machine learning algorithms, MLlib, which can be easily leveraged in a Spark application. Spark succeeds where traditional MapReduce approach fails, making it easy to develop and execute iterative algorithms. Many ML algorithms are based on iterative optimization, which makes Spark a great platform for implementing them.

### 3. Flask

Flask is a web micro framework for Python. It is very easy to start up a web API, by just importing it in our script and using some annotations to associate our service end-points with Python functions. In our case, we will wrap our Recommendation Engine methods around some of these end-points and interchange JSON formatted data with the web client. Flask was used to API-like

### 4. CherryPy

The CherryPy framework features a reliable, HTTP/1.1-compliant, WSGI thread-pooled webserver. It is also easy to run multiple HTTP servers (e.g. on multiple ports) at once. All this makes it a perfect candidate for an easy to deploy production web server for our on-line recommendation service.

## 4. RESULTS

We trained the data by splitting the data in two parts 70 -30 splits for training and testing. And then calculated the root mean square error (RMSE).

In our model upon trying different values for regularization factor and rank (Latent Factors) we got the best value for rank 11 and Regularization factor 0.1. The **RMSE** for the model with the above values is **2. 20291336772**.

The model was then used to build a basic web application for displaying the top recommendations for given users. CherryPy and Flask frameworks were used to deploy the web application.

The home page displays the top 50 recommendations. The application can display the results specific to users and can also take the input of *k* for top *k* results. The application can also handle the requests for predicting the rating for a specific book for a user.

## 5. CONCLUSION AND FUTURE SCOPE

Apart from just the traditional Collaborative Filtering or content based filtering, many modern techniques are being exploited nowadays. The hybrid algorithms are a mixture of many techniques which is the present trend. Demographic filtering helps give more personalized recommendations. With the advancement of the web, its use in the process of recommendation can help improve the efficiency. The web application can have more interactive interface for adding new

users by displaying 10 random popular movies and asking the user to rate to give recommendations. As of now the data is static, making use of streaming data for constantly updating model may also yield better results.

## II. References

[1] R. He, J. McAuley. Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW, 2016

[2] J. McAuley, C. Targett, J. Shi, A. van den Hengel. Image-based recommendations on styles and substitutes. SIGIR,2015

[3] http://jmcauley.ucsd.edu/data/amazon/links.html

[4] https://aws.amazon.com/blogs/big-data/building-a-recommendation-engine-with-spark-ml-on-amazon-emr-using-zeppelin/

[5] https://github.com/sridharswamy/Book-Recommender-System/blob/master/Conference%20Paper.pdf

[6] https://spark.apache.org/docs/2.1.0/mllib-collaborative-filtering.html