DTI5125[EG] Data Science Applications: Book RecommendationSystem

Kareem Atif Mohamed Bakli[1], Mostafa Mahmoud Nofal[2], Gehad Hisham Hassanr[3], and Kareem Khaled Moner Waly[4]

[1]**kbakl031@uOttawa.ca**
[2]**mnofa091@uottawa.ca**
[3]**gsoma101@uOttawa.ca**
[4]**kwaly008@uOttawa.ca**

- PROBLEM FORMULATION

A book recommendation system is a type of recommendation system where we have to recommend similar books to the reader based on his interest. The books recommendation system is used by online websites which provide eBooks like google play books, open library,
good Read's, etc. recommendation systems ought to increase profit from product sales. To achieve this, recommendations need to be relevant, novel, and diverse.

- INTRODUCTION:

Our work:

1- Data Preparation

2-Feature Engineering

3-Data Analysis and Visualization

4-Modeling (Classification and Clustering).

5-Evaluation and Error Analysis.

6-INNOVATIVENESS

7-Integration with Dialog flow fulfillment

Dataset Explanation:

- **Dataset Contain three files which are:**

i. Users_dataset.

- User-ID (unique for each user)
- Location (contains city, state and country separated by commas)
- Age

ii. Books_dataset:

- ISBN (unique for each book)
- Book-Title
- Book-Author
- Year-Of-Publication
- Publisher

iii. Ratings_dataset:

- User-ID
- ISBN
- Image-URL-S
- Image-URL-M
- Image-URL-L
- Book-Rating

1- Data Preparation

1) Handling Missing Values

- firstly, we search for the null and missing values in the whole dataset then fill the missing values with 'Unknown'

```
df_books['Book-Author'].isnull().values.any()
```

```
True
```

```
df_books['Book-Author'].fillna('Unknown',inplace=True)
```

```
df_books['Book-Author'].isnull().values.any()
```

```
False
```

```
Missing Data Count
age_bins        248343
Age             245274
Image-URL-L          4
Publisher            2
dtype: int64
----------------------------------------
```

-Handle missing value in Age column with random number between

(mean-standatrddeviation , mean+standard_deviation)

```
random_age = np.random.randint(median - std, median + std, size = nulls)
age = df_merged['Age'].copy()
age[pd.isnull(age)] = random_age
df_merged['Age'] = age
df_merged['Age'] = df_merged['Age'].astype(int)
```

After applying

```
# Null values in age column
nulls = sum(df_merged['Age'].isnull())
print('Null Values in Age Column: ',nulls)
```

```
Null Values in Age Column:  0
```

2) Define a function to combine two columns namely title and author.

Before

Book-Title	Book-Author
Classical Mythology	Mark P. O. Morford
Clara Callan	Richard Bruce Wright

After

new_title
Classical Mythology by Mark P. O. Morford
Clara Callan by Richard Bruce Wright

3) Get rid of duplication

Before

```
df_books.shape
```

```
(271360, 8)
```

After

```
df_books.shape
```

```
(251185, 9)
```

4) Merge three dataset

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
2	2313	0679745580	8	In Cold Blood (Vintage International)	TRUMAN CAPOTE	1994	Vintage
3	2313	0060173289	9	Divine Secrets of the Ya-Ya Sisterhood : A Novel	Rebecca Wells	1996	HarperCollins

5) Information of the Features:

```
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   User-ID              916933 non-null  int64
 1   ISBN                 916933 non-null  object
 2   Book-Rating          916933 non-null  int64
 3   Book-Title           916933 non-null  object
 4   Book-Author          916933 non-null  object
 5   Year-Of-Publication  916933 non-null  int64
 6   Publisher            916931 non-null  object
 7   Image-URL-S          916933 non-null  object
 8   Image-URL-M          916933 non-null  object
 9   Image-URL-L          916929 non-null  object
 10  new_title            916933 non-null  object
 11  Location             916933 non-null  object
 12  Age                  671659 non-null  float64
```

6) Get Data Description:

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	
count	916933.00000	916933	916933.000000	916933	916933	916933.000000	916931	
unique	NaN	250075	NaN	241061	101587	NaN	16542	
top	NaN	0971880107	NaN	Wild Animus	Nora Roberts	NaN	Ballantine Books	h
freq	NaN	2502	NaN	2502	7645	NaN	30011	
mean	140202.83165	NaN	2.825417	NaN	NaN	1968.353922	NaN	
std	80804.41894	NaN	3.848183	NaN	NaN	230.251189	NaN	
min	2.00000	NaN	0.000000	NaN	NaN	0.000000	NaN	
25%	69697.00000	NaN	0.000000	NaN	NaN	1991.000000	NaN	
50%	140410.00000	NaN	0.000000	NaN	NaN	1997.000000	NaN	
75%	211426.00000	NaN	7.000000	NaN	NaN	2001.000000	NaN	
max	278854.00000	NaN	10.000000	NaN	NaN	2099.000000	NaN	

7) Clean up data (Fix Inconsistent Data)

Before

```
df_merged['Year-Of-Publication'].unique()
```

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

After

```
df_merged['Year-Of-Publication'].unique()
```

```
array([2002, 1994, 1996, 1998, 2001, 1987, 1984, 1997, 1970, 1978, 1993,
       1989, 1995, 1990, 1992, 1950, 1991, 1999, 1954, 1988, 2003, 2004,
       2000, 1983, 1985, 1982, 1956, 1979, 1986, 1975,    0, 1976, 1977,
       1980, 1981, 1974, 1957, 1958, 1960, 1963, 1969, 1972, 1961, 1971,
       1953, 1968, 1973, 1967, 1962, 1937, 1959, 1955, 2005, 1945, 1965,
       1964, 1927, 2020, 2050, 1920, 1966, 1952, 1930, 1940, 1942, 1947,
       1925, 1923, 2030, 1951, 1936, 1946, 1943, 2099, 1928, 1941, 2011,
       1948, 1901, 1932, 1939, 1944, 1938, 1902, 1929, 1900, 1911, 1949,
       1926, 1933, 1922, 1897, 2026, 1906, 1806, 1935, 2006, 2037, 1921,
       2024, 2038, 1924, 1378, 2008, 1934, 1909, 1931, 1904, 1917, 2012,
       1914, 1376, 1908, 1919])
```

2-FEATURE ENGINEERING:

1. Defining a function to extract the country names Data Before applying extract country function

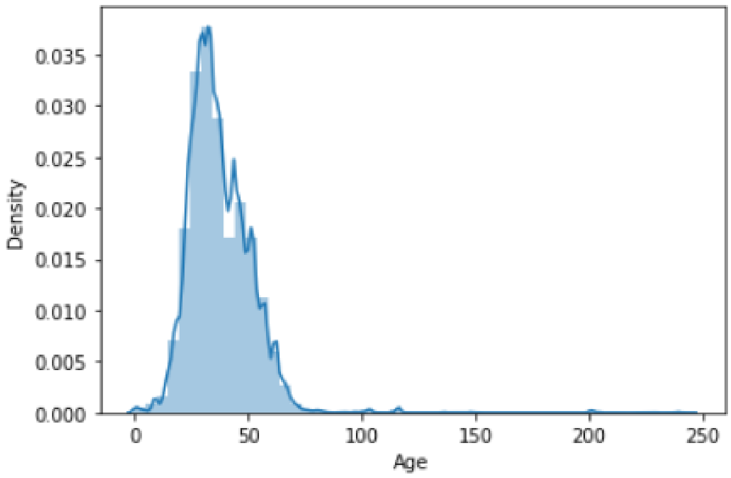
Before Applying:

	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

After Applying:

User-ID	Location	Age	country
1	nyc, new york, usa	NaN	usa
2	stockton, california, usa	18.0	usa
3	moscow, yukon territory, russia	NaN	russia
4	porto, v.n.gaia, portugal	17.0	portugal

3-Creating bins for the age column.



3-Calculating the Rating Count and Rating Mean for each Book-Title

Location	Age	age_bins	Rating-Count	Rating-Mean
usa	44	NaN	60	2.93
usa	23	Adult	60	2.93
usa	34	Adult	60	2.93

4-Using IMDB Formula to calculate the Weighted Rating for our books using Rating count and Rating Mean that calculated to calculate Weighted Average.

$$W = \frac{Rv + Cm}{v + m}$$

where:

W = Weighted Rating

R = average for the movie as a number from 0 to 10 (mean) = (Rating)

v = number of votes for the movie = (votes)

m = minimum votes required to be listed in the Top 250 (currently 3000)

C = the mean vote across the whole report (currently 6.9)

5-Using TF-IDF Tranformation to convert description data which will use in content base as input to cosine similarity model.

```
tfidf = TfidfVectorizer(min_df=5)
tfidf_mat = tfidf.fit_transform(df_descriptions['description'])
tfidf_mat
```

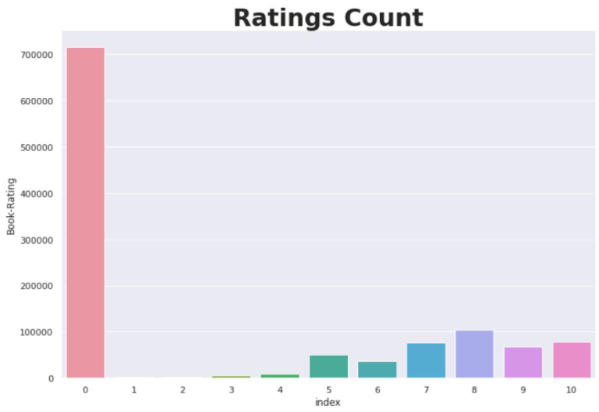
```
<7021x6443 sparse matrix of type '<class 'numpy.float64'>'
	with 152705 stored elements in Compressed Sparse Row format>
```

4- DATA ANALYSIS AND VISUALIZATION:

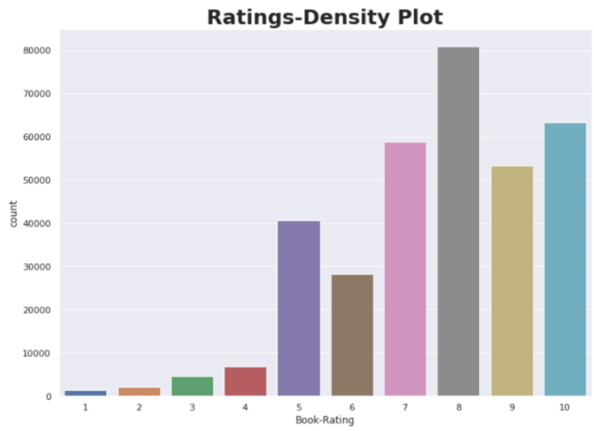
1) Splitting the Dataset **into Two Based on the Explicit and Implicit Ratings**

- In the first photo we have the dataset with both explicit and implicit ratings is highly skewed toward the value of zero. Implicit means that it contains zero ratings. In the explicit ratings only, the skewness perishes after we remove the implicit ratings.

Explicit - Implicit Ratings



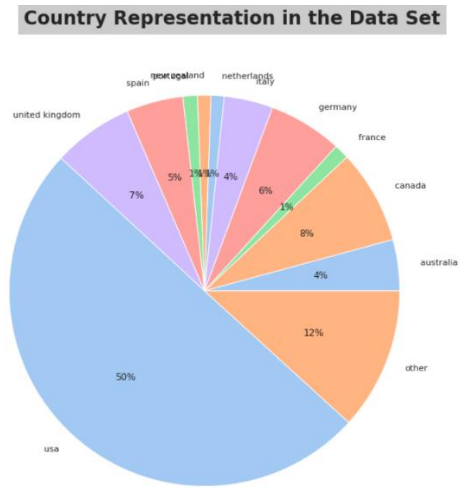
Explicit Ratings

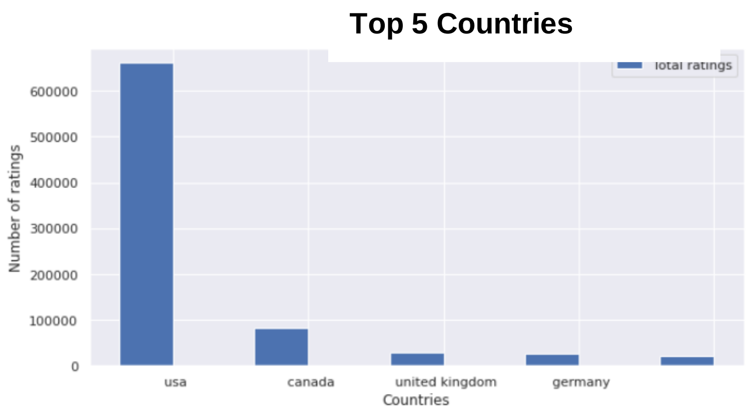


2) Location

- Most customers are from United states of America, followed by Canada, United Kingdom and Germany.

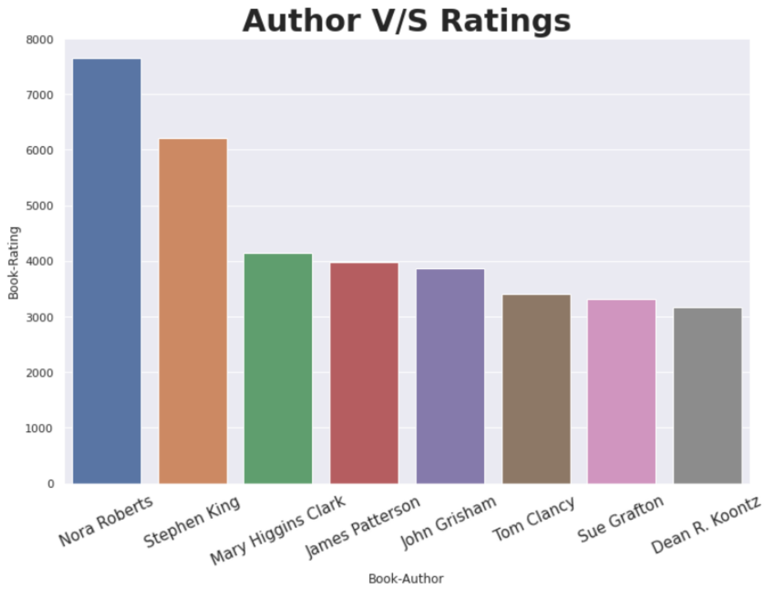
- Countries with less than 1% customers are labeled as other





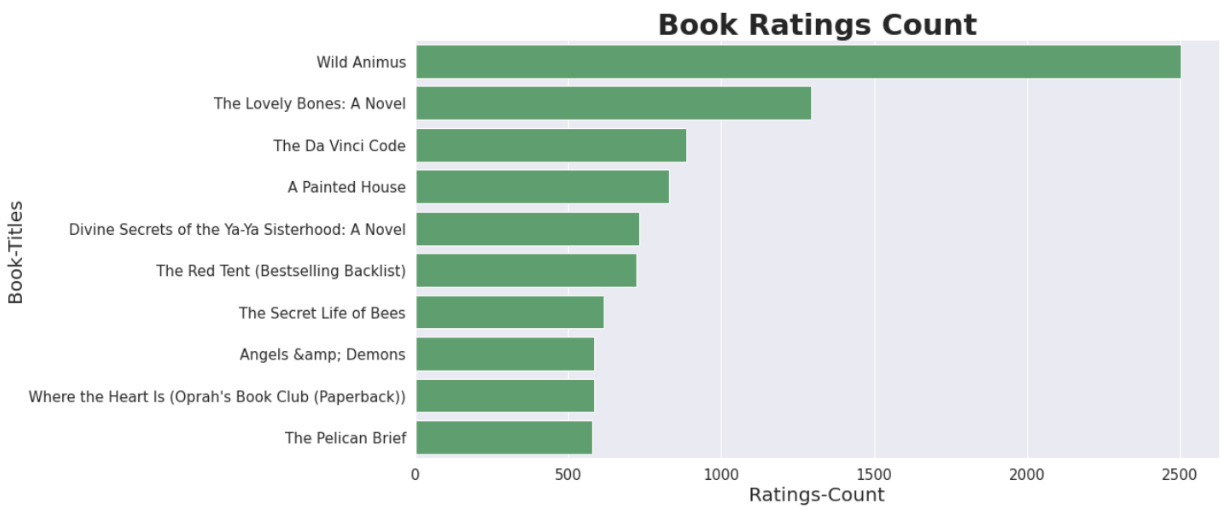
3) Author V/S Ratings

- Here, we can observe, most frequently rated Authors. Most frequently rated author is Nora Roberts, followed by Stephen King.



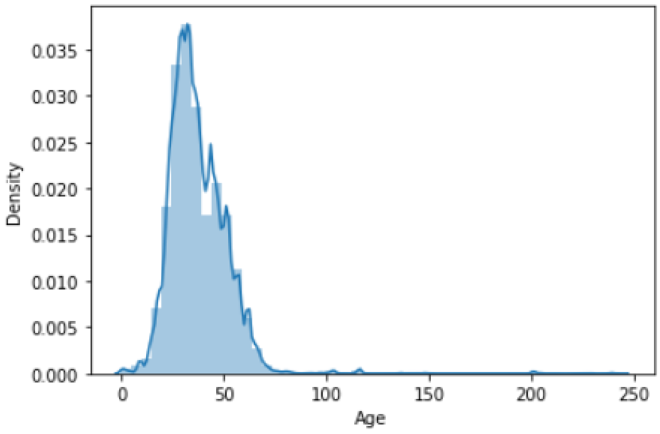
4) Book Ratings Counts

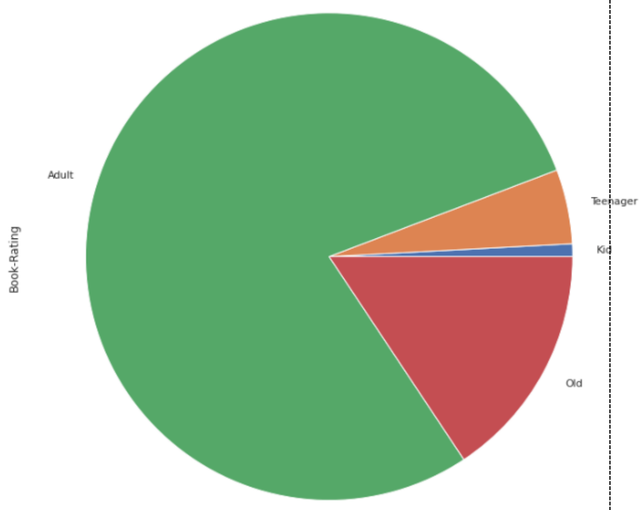
- For book ratings counts we are able to observe, most frequently rated books by the users. Most frequently rated book, happens to be Wild Animus



5) Age Distribution of users

- Most customers are Adults who are between 20 and 50 years. The second most represented age group is for boomers who are older than 50 years.





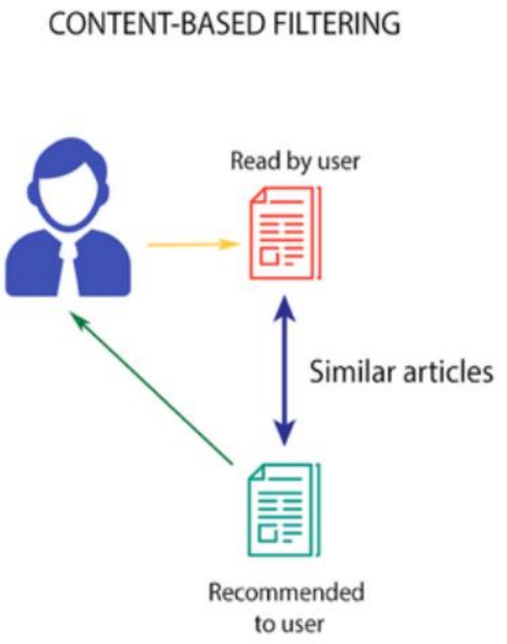
4- MODELING (CLASSIFICATION AND CLUSTERING)

1) Content Based

- The algorithm recommends a book that is similar to the reader based on his interest. In simple words, Inthis algorithm, we try to find finding item look alike. For example, a person likes to watch history movies,so he may like reading history books too because the two items have similar tags.
- In our previous Dataset we didn’t have text description, so we used Google Books API to extract description of books based on its ISBN and exported it as csv and this was our output. Then we will merge it with our original dataset which contained the user id and book id and consider this as initiative part to transform data from can’t apply content based on to can apply.

	A	B	C
1		ISBN	description
2	3	4.51E+08	A high-tech submarine thriller follows Admiral Michael Pacino as he emerges from retire
3	4	3.73E+08	The Flower And The Sword by Jacqueline Navin released on Jul 24, 1998 is available now
4	23	6.71E+08	In late-fifteenth-century Spain, the indomitable and passionate Isabel Valderocas, living
5	25	08217661	Mountain man Smoke Jensen comes to the aid of a sharpshooter who decides to reform
6	27	3.81E+08	Stephanie promises to help the ghost of her new home's former inhabitant and is sent ba
7	33	3.81E+08	A former member of the Manhattan homicide prosecution team profiles fourteen of the
8	34	4.4E+08	A collection of time-saving techniques enables readers to measure their reading speeds
9	36	4.51E+08	Revealing an appalling true story, the gripping tale of a murderous mother reconstructs
10	47	5.54E+08	Posing as the devoted wife to a powerful army commander, Kitty Radachek is unable to
11	56	1.4E+08	Half aborigine and half white, Jimmie Blacksmith is unable to fit into either culture and,
12	58	5.53E+08	When her job at an antique store sends her to old Miss Watson's mansion to pick up an
13	59	20434901	As a slag heap, the result of strip mining, creeps closer to his house in the Ohio hills, fifte
14	60	5.15E+08	Murdered in 1848 in his bed by a lover's protective father, seductive rogue Valentine Tre
15	70	3.73E+08	High-Society Bachelor by Krista Thoren released on Dec 25, 2001 is available now for pu
16	79	15516661	Three lively tales of romance and adventure center around the Lassiter Detective Agenc
17	85	4.51E+08	Three ambitious executives--one woman and two men--embark on a retreat in the wilds
18	96	3.45E+08	Rose, an amateur photographer, and Robert, a cynical scriptwriter witness a rooftop kid
19	136	8.87E+08	Nineteen tales of horror and the supernatural encompass works by such acclaimed write
20	173	4.49E+08	When a childhood schoolmate is suspected of murdering a crooked judge with a priceles
21	192	61098035	An accidental discovery in the basement of FBI headquarters at Quantico nearly costs ag
22	217	05535696	P.I. Phoebe Siegel, of Billings, Montana, balks at investigating the strange behavior of a t
23	254	4.49E+08	Champion golfer Kate O'Brien, a big draw on the women's exhibition tour, is murdered, a

- We used cosine similarity and descriptions of the books were transformed into a keyword-based vector-space representation using Tf-IDF. Then cosine similarity was computed between books. Finally, the calculated distances are used to filter previous recommendations on items for every user in the entire user-item dataset.



Recommendation Results:

```
#Recommendations based on -> 0749322179: MARRANOS
recommendations('0749322179')
```

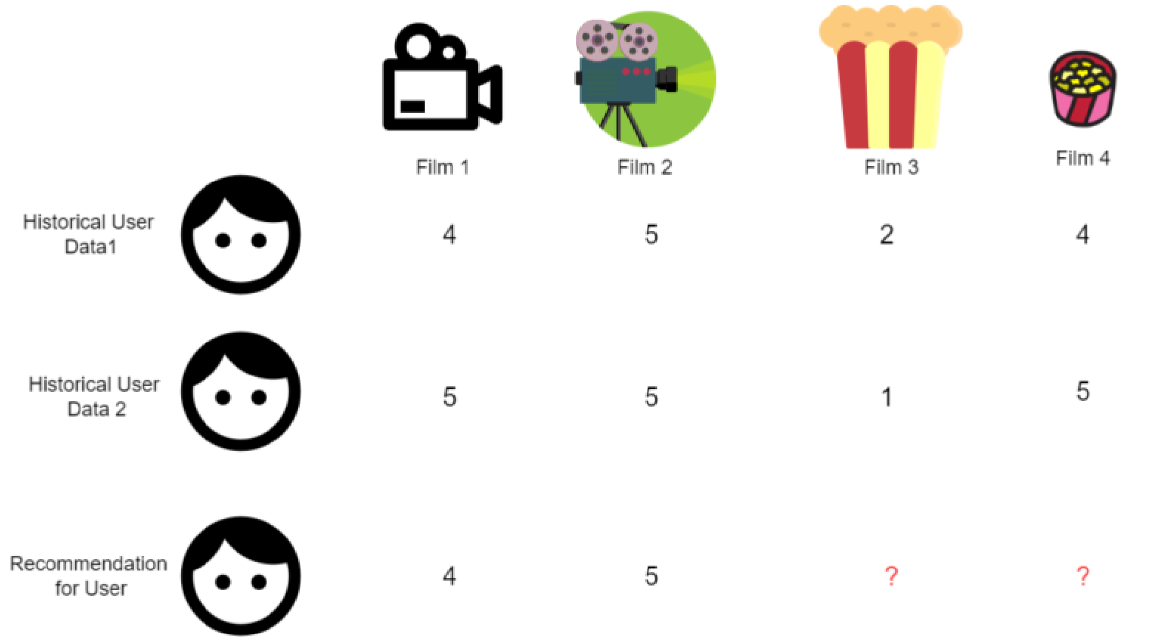
```
['Waltz in Time (An Avon Romantic Treasure)',
 'Pressure Points',
 'Dead Duck (Sam and Hollis Mystery)',
 'Burn Factor',
 'MARRANOS',
 '21st Century Guide to Increasing Your Reading Speed (21st Century Reference)',
 'Barracuda Final Bearing',
 'In the Midnight Hour (Haunting Hearts)',
 "The Year's Best Horror Stories: Series XIV",
 'Most Wanted']
```

```
#Recommendations based on ->2211021662: Dead Duck (Sam and Hollis Mystery)
recommendations('2211021662')
```

```
['Sleep My Child, Forever (Onyx True Crime)',
 'M. C. Higgins, the Great',
 'Burn Factor',
 'Dead Duck (Sam and Hollis Mystery)',
 "The Year's Best Horror Stories: Series XIV",
 'Roofworld',
 'Pressure Points',
 'Most Wanted',
 'High - Society Bachelor (Harlequin American Romance, No. 908)',
 'In the Midnight Hour (Haunting Hearts)']
```

2) Collaborative Filtering

-Memory-based collaborative filtering
Memory-Based Collaborative Filtering approaches can be divided into two main sections: **user-item filtering and item-item filtering**. A user-item filtering takes a particular user, find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked.



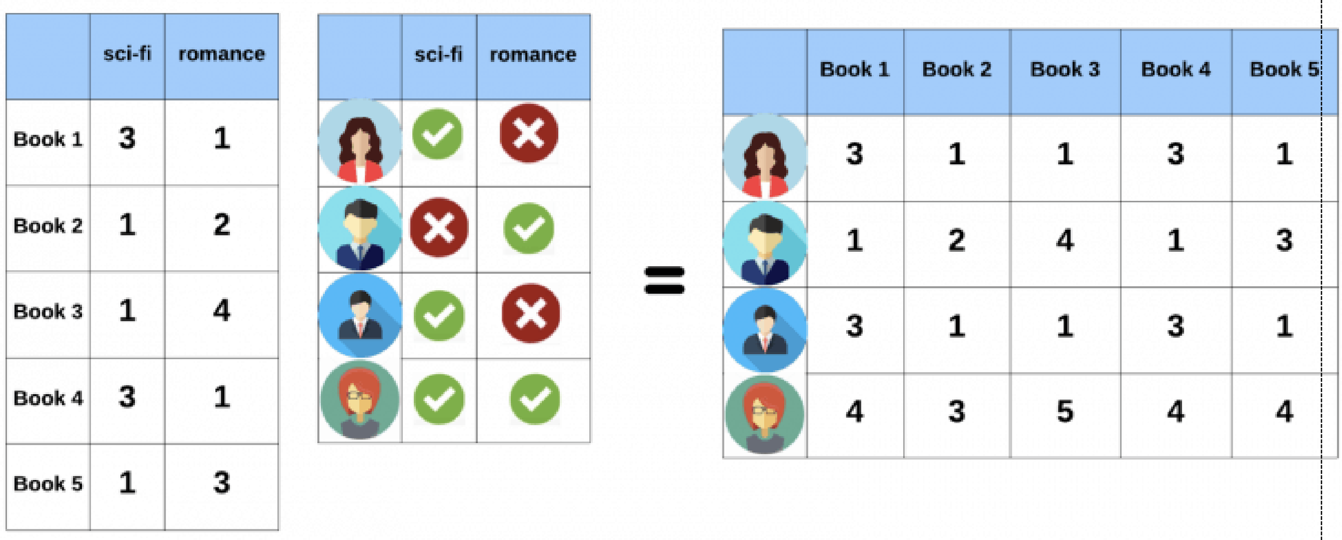
-As you can see, the system has identified users who have a similar preference to the third user, Since the third user has not watched Films 3 and 4, we find the aggregate of ratings of similar users.
We can understand from the image that Film 4 will be recommended to the user based on historical user data.

We use Three Model in Memory-based (Classification):

1-K-NN Based Algorithms
2-KNN with Means
3-KNN with ZScore

3) Matrix Factorzation Collaborative Filtering:

Matrix factorization is a collaborative filtering method to find the relationship between items' and users' entities



Matrix factorization methods are used to find a set of latent factors and determine user preferences using these factors. Latent Information can be reported by analyzing user behavior

We use Three Model in Matrix Factorization (classification):

1-SVD

2-SVDpp

3-NMF

----

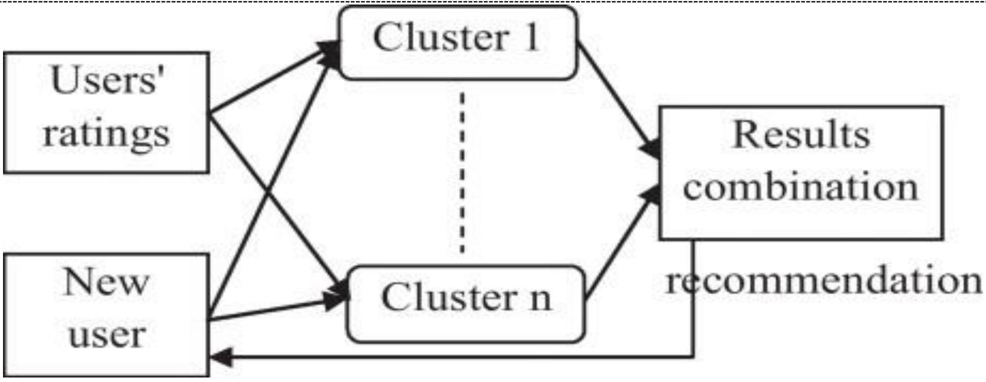
4) Collaborative Filtering Algorithms (Clustering)

A collaborative filtering algorithm based on co-clustering.

This is a straightforward implementation of [George:2005].

Basically, users and items are assigned some clusters Cu, Ci, and some co-clusters.

Clusters are assigned using a straightforward optimization method, much like k-means.



We use one Model (Clustering):

1-CoClustering

------

5) Other Collaborative Filtering Algorithms used:

1- SlopeOne.

SlopeOne is a straightforward implementation of the SlopeOne algorithm.

5- EVALUATION AND ERROR ANALYSIS:

1- Memory-based collaborative filtering (Classification Models):

Our best model used is KNN Basic

Algorithm	RMSE	MAE
KNN Basic	0.2071	0.0462
KNN Means	0.3462	0.2535
KNN ZScore	0.3625	0.2659

2- Matrix Factorization Collaborative Filtering

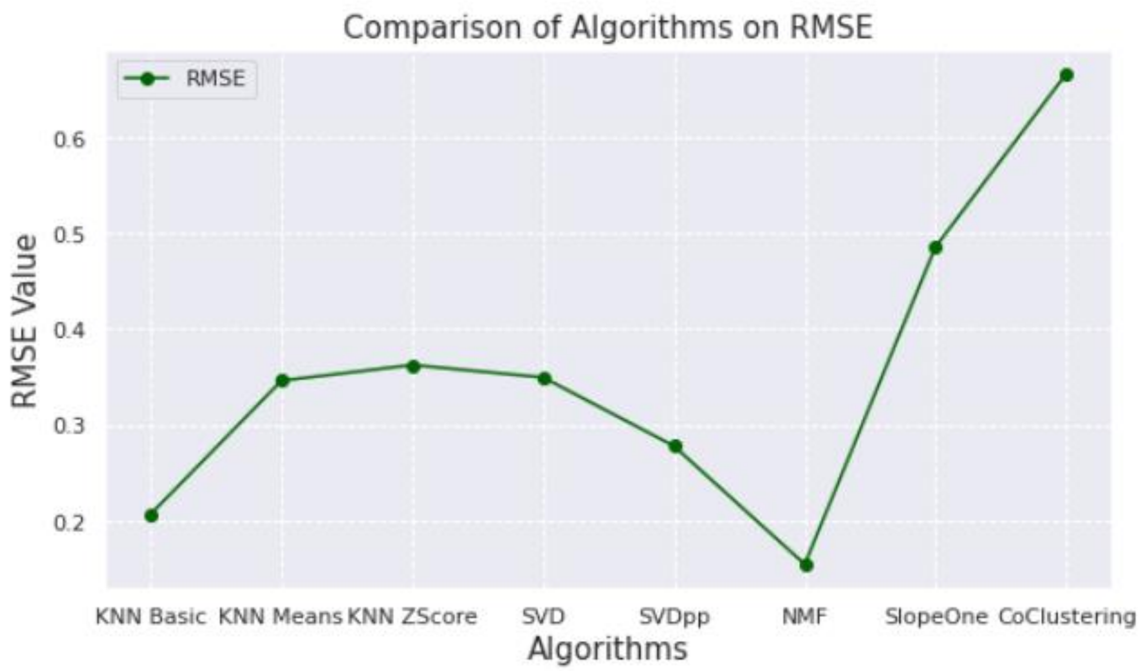
Best model in Matrix Factorization is NMF with

Algorithm	RMSE	MAE
SVD	0.3496	0.2529
SVDpp	0.2781	0.1907
NMF	0.1548	0.0923

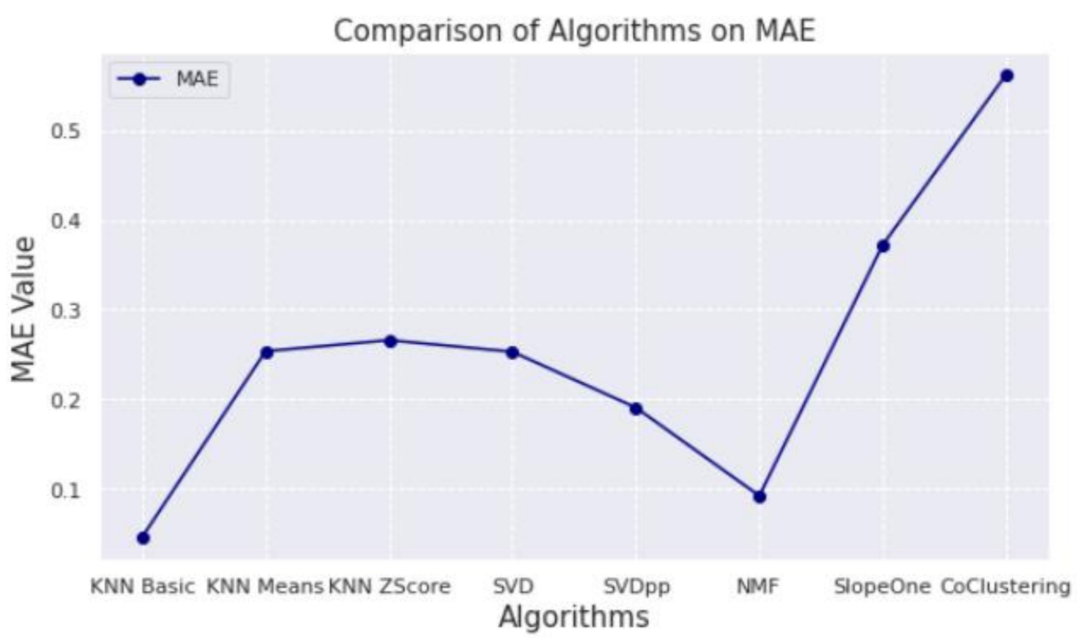
3- Other Collaborative filtering CoClustering and Slopeone

Algorithm	RMSE	MAE
SlopeOne	0.4856	0.3723
CoClustering	0.6662	0.5623

-Comparison of all algorithms used: RMSE



-Comparison of all algorithms used: MAE



- **Our Champion model is KNNBasic() with RMSE and MAE are 0.2, 0.04**

Error Analysis using Hit Rate

- We need to make sure that the results of our recommendation system are related to the original books that our users rated. So first to compute the hit rate, we need to get the top 5 predictions for the users.

```
[ ]   1 top_n
```

```
defaultdict(list,
            {2313: [('Bad Business by Robert B. Parker', 5),
              ('The Sinner by TESS GERRITSEN', 5),
              ('Warchild by Karin Lowachee', 5),
              ('Harry Potter and the Order of the Phoenix (Book 5) by J. K. Rowling',
               5),
              ('Harry Potter and the Goblet of Fire (Book 4) by J. K. Rowling',
               5)],
             10314: [('Bad Business by Robert B. Parker', 5),
              ('The Sinner by TESS GERRITSEN', 5),
              ('Warchild by Karin Lowachee', 5),
              ('Harry Potter and the Order of the Phoenix (Book 5) by J. K. Rowling',
               5),
              ('Harry Potter and the Goblet of Fire (Book 4) by J. K. Rowling',
               5)],
             77480: [('Bad Business by Robert B. Parker', 5),
              ('The Sinner by TESS GERRITSEN', 5),
              ('Warchild by Karin Lowachee', 5),
              ('Harry Potter and the Order of the Phoenix (Book 5) by J. K. Rowling',
               5),
              ('Harry Potter and the Goblet of Fire (Book 4) by J. K. Rowling',
               5)],
             98391: [('Harry Potter and the Order of the Phoenix (Book 5) by J. K. Rowling',
```

- After that we will do cross validation and remove one of the books from the training data. Then we will increase the hit rate by 1 if there is a similarity between the removed item and the predictions.

- The number of hits divided by the number of test users represents the system's overall hit rate. A higher value indicates better results.

- If we have an extremely low hit rate means that we simply do not have enough data to work with.

- Our Hit Rate Output

```
Hit Rate:  0.007955077211043519
```

6-INNOVATIVENESS

-Data we found just contain:

Users_dataset.	2-Books_dataset:	3-Ratings_dataset:
● User-ID (unique for each user) ● Location (contains city, state and country separated by commas) ● Age	● ISBN (unique for each book) ● Book-Title ● Book-Author ● Year-Of-Publication ● Publisher	● User-ID ● ISBN ● Image-URL-S ● Image-URL-M ● Image-URL-L ● Book-Rating

1-**There is no information regarding user description about book which read.**

We use Google API to collect description of books according to their ISTPN.

Create and save new csv file which called description which we used in Content based Recommendation System with Cosine Similarity.

We consider this as innovative work change data form can't applying content based to can apply.

2-**Optimum Book reader**

We can't take every user's rating at facevalue because if the user is a novice reader with only an experience of reading a couple of books, his/her ratings might not be much relevant for finding similarity among books.

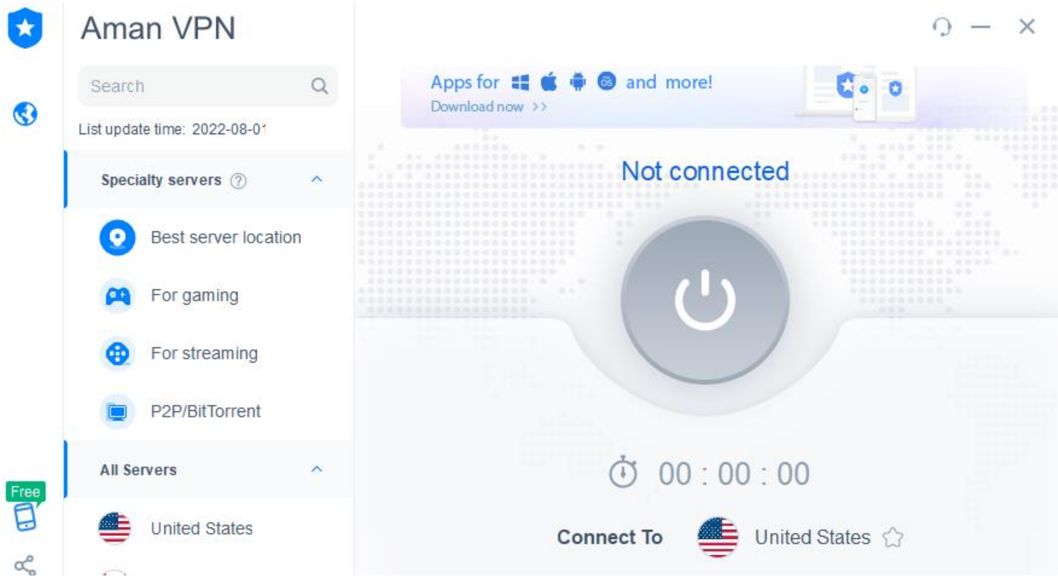
Therefore as a general rule of thumb we choosing only those User's who have rated atleast 10 Books for builing the recommendation system we consider it as innovative work .

```
df_merged_updated= df_merged_updated[df_merged_updated['User-ID'].isin(counts1[counts1 >=10].index)].reset_index()
df_merged_updated.drop(columns='index', inplace=True)
```

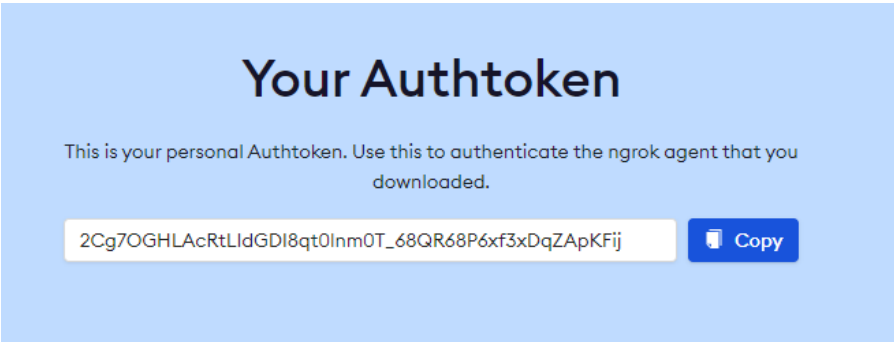
7-GRAPHICAL INTUITION AND ANALYSIS

Integration With Dialog Flow Fulfillment

1-We use AMAN VPN to give us access to use negrock.



2- Use Ngrock to create static IP that Dialog can connect With Dialog flow.



2-Copy Authtoken above to notebook (code).

```
! ngrok authtoken "2Cg7OGHLAcRtLIdGDI8qt0lnm0T_68QR68P6xf3xDqZApKFij"
```

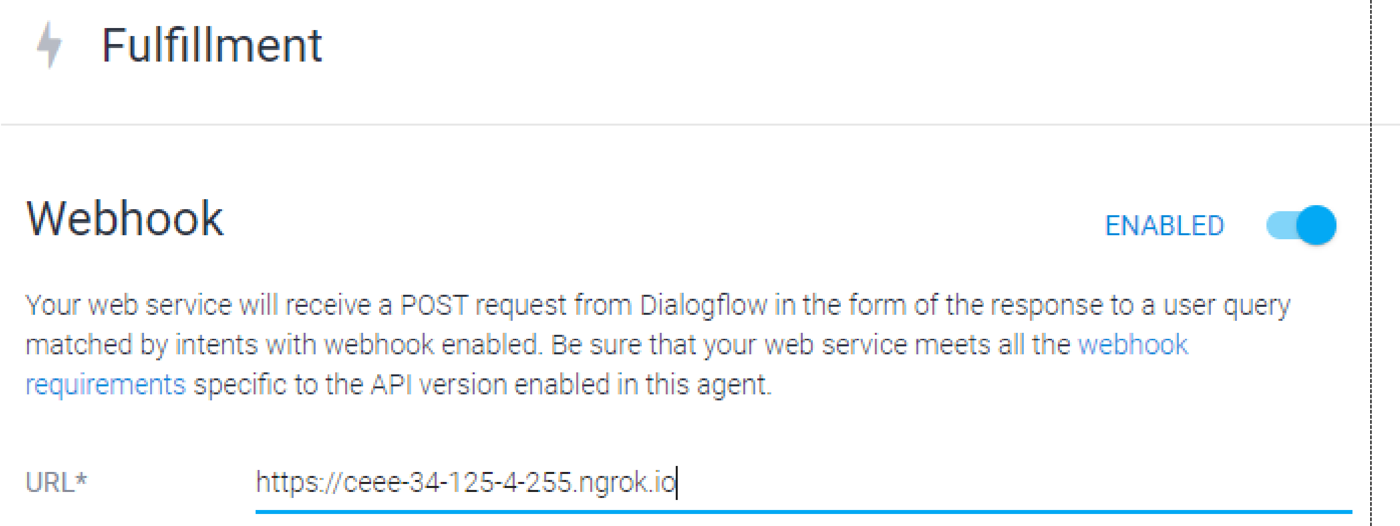
```
Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
```

3- After that run this Cell create link paste it to dialog flow fulfillment.

```
ngrok_tunnel = ngrok.connect(8000)
print('Public URL:', ngrok_tunnel.public_url)
nest_asyncio.apply()
uvicorn.run(app, port=8000)
```

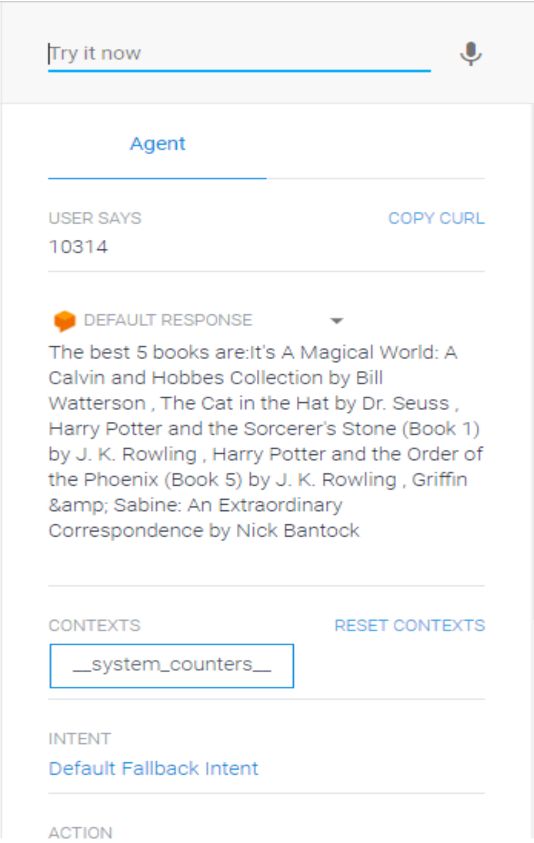
```
Public URL: http://ceee-34-125-4-255.ngrok.io
INFO:     Started server process [59]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)

2313
INFO:     66.249.83.58:0 - "POST / HTTP/1.1" 200 OK

10314
INFO:     66.249.83.208:0 - "POST / HTTP/1.1" 200 OK
```

4-Paste Public URL Dialog flow Fulfillment

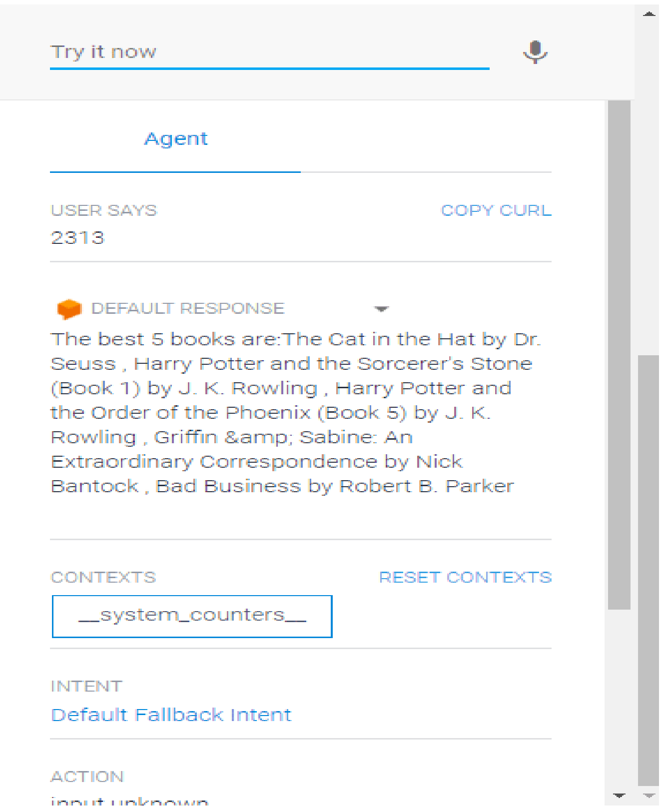


Results:

-When enter user id 10314 get prediction of



- Another Example with user id 2313



-These Predictions based on our Champion Model we can switch in code to Cosine Similarity model and predict top 10 similar books to user interest.

```
async def home(info: Request):
    req_info = await info.json()
    print()
    intent_NAme = req_info["queryResult"]["intent"]["displayName"]
    input = req_info["queryResult"]["queryText"]

    text = "The best 5 books are:"
    if(intent_NAme=="hello" or  intent_NAme == "Default Fallback Intent" or intent_NAme=="user" ):
      print(input)
      x = [x[0] for x in get_predictions(int(input))]
      text+= ' , '.join(x)
    elif (intent_NAme=="cant understand"):
       text ="None"

    return {"fulfillmentMessages": [
      {
        "text": {
          "text": [
            text
          ]
        }
      }
```