UNIVERSIDAD PRIVADA-DE-TACNA

INGENIERIA DE SISTEMAS

TITULO:

Sistema de Recomendación de Libros de Biblioteca

CURSO:

INTELIGENCIA DE NEGOCIOS

DOCENTE(ING):

Patrick Cuadros Quiroga

Integrantes:

Acosta Ortiz, Orlando Antonio	(2015052775)
Zegarra Reyes, Roberto	(2010036175)
Catari Cabrera, Yofer Nain	(2017059289)
Mamani Maquera, Jorge Luis	(2016055236)
Rivas Rios, Marko Antonio	(2016054461)
Cruz Escalante, Richard Manuel	(2013047247)

Índice

1. RESUMEN

El incremento en el número de libros, y otros hace que los sistemas tradicionales de búsqueda de literatura sobre algún tema en particular sean complejos y lentos, no siempre obteniendo buenos resultados. Peor aún si se trata de recomendar algún libro en particular, basado en el conocimiento de la calidad de su contenido.
Por ello se propone diseñar e implementar un sistema de recomendación de libros. Se diseñó un modelo basado en lenguaje de programación Python para la recomendación de libros con el objetivo de la recomendación de varios tipos de libros del interés del usuario, mientras la recomendación de libros busca incrementar el conocimiento de los usuarios, este sistema ayuda a seleccionar un tema de investigación o encontrar una referencia bibliográfica ajustada al tema de investigación del usuario.

2. ¿Qué es Colap?

Convertir datos en información es, hoy en día, una ventaja competitiva que las empresas deben comenzar a explotar. Optimizar sus procesos, entender su entorno o adelantarse a futuras tendencias son solo algunas de las posibilidades que brindan las herramientas de análisis de información. Colap es una herramienta de análisis y visualización de información con la particularidad que está diseñada para personas que no son del área IT. Se consulta escribiendo en español (similar a como se hace una búsqueda en Google), permite crear reportes personalizados y además incluye la posibilidad de compartir consultas con otros miembros de la empresa.

3. INTRODUCCION

Con la finalidad de tener una biblioteca con información actualizada y brindar información rápida diferentes medios bibliográficos de la biblioteca, además de poder realizar reservaciones desde el sistema.
Este sistema será de mucha utilidad para ubicar un libro y otros medios de la biblioteca rápidamente, nos facilitará conocer el status de los libros y préstamos, la adquisición de nuevos libros y los procesos técnicos por ejemplo catalogación y clasificación de los ejemplares.

Los beneficios sociales que un proyecto serio y estructurado de un sistema para la administración de Biblioteca, en el cual se involucren diferentes sectores de un ente académico, privado, de carácter estatal o del gobierno, son simples y fácilmente demostrables.

La finalidad principal de un proyecto de este tipo es la generación, administración y disposición de conocimiento para una comunidad determinada. Los beneficios académicos que recibirá la institución es automatizar estos procesos con un sistema de Inventario y préstamos de libros haciendo la tarea más sencilla para los estudiantes y el administrador de la misma.
Con la implementación de esta herramienta se obtendrá la información al instante de los libros, revista, editoriales entre otros. Se podrá obtener una lista de todos los libros en stock, editoriales, etc., y buscar en cualquier momento en base a varios reglas de filtrado. Se podrá organizar la biblioteca por editoriales, autores entre otros.

4. MARCO TEORICO

4.1. Python

Python es un lenguaje sencillo y rápido de aprender. Su sintaxis es parecida a escribir cualquier texto en inglés, pero con la potencia de sus principales competidores en el BackEnd. Es un placer de leer y redactar. Python predica que un código debe ser escrito por humanos para humanos. Después de todo lo que programas va a ser leído por ti y por el resto del equipo. Si escribes para máquinas, solo te entenderán máquinas.
Además, viene con "Pilas incluidas". Eso quiere decir que posee su propio gestor de paquetes, sin necesidad de instalar aplicaciones externas. Simplificando tareas de instalación o actualización.
Otro punto a su favor es que no necesita un ecosistema para ejecutarse, como puede ser Xampp, Vangrant, Docker... Python solo requieres Python. Lanzando un comando en el terminal estará ejecutándose su propio servidor Web, consiguiendo que su puesta en producción sea sorprendentemente rápida. Y por si fuera poco, es el segundo lenguajes que mejor esta pagado por las empresas. Por detrás de Ruby.

4.2. Google Colab: Python y Machine Learning en la nube

En este veremos qué es y cómo utilizar Google Colab, la herramienta de Google en la nube para ejecutar código Python y crear modelos de Machine Learning a través de la nube de Google y con la posibilidad de hacer uso de sus GPU . Sí, has leído bien: con sus GPU y en la nube.

4.3. Frameworks Web

Entre sus numerosos y fantásticos Frameworks, nos podemos encontrar unas bestias: Django y Flask (que no confundir que el zombie Adobe Flash). Django sería lo más cercano a Laravel en PHP o Ruby on Rails para Ruby. Un marco de trabajo completo y eficiente para desarrollar Aplicaciones Web de una gran complejidad con un mínimo esfuerzo. Casi cualquier cosa que necesites posiblemente estará integrada.
Para desarrollos altamente personalizados o con unos tiempos cortos, nos encontramos a Flask. Autodenominado microframework, pero con funcionalidades sencillas e inteligentes para construir cualquier sitio que se te pase por la cabeza. Uno no sustituye al otro. Merece la pena experimentarlos y ver sus diferentes enfoques.

4.4. Ventajas de programar en Python

- Simplificado y rápido: Este lenguaje simplifica mucho la programación, es un gran lenguaje para scripting.

- Elegante y flexible: El lenguaje ofrece muchas facilidades al programador al ser fácilmente legible e interpretable.

- Programación sana y productiva: Es sencillo de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y fomenta la productividad.

– Ordenado y limpio: es muy legible y sus módulos están bien organizados.

– Portable: Es un lenguaje muy portable. Podemos usarlo en prácticamente cualquier sistema de la actualidad.

– Comunidad: Cuenta con un gran número de usuarios. Su comunidad participa activamente en el desarrollo del lenguaje.

4.5. Futuro

Las previsiones son muy buenas. Las versiones son constantes y compatibles con todas las plataforma. Su creador, Guido van Rossum, es denominado como "Benevolente dictador vitalicio" por dejar que la comunidad tomen las decisiones. Tan solo dejó 4 directrices:

– Python debería ser fácil, intuitivo y tan potente como sus principales competidores.

– El proyecto sería de Código Abierto para que cualquiera pudiera colaborar.

– El código escrito en Python sería tan comprensible como cualquier texto en inglés.

– Python debería ser apto para las actividades diarias permitiendo la construcción de prototipos en poco tiempo.

5. EJEMPLO

5.1. Ejemplos estadisticos

Se utilizo estos ejemplos para desarrollar el sistema de recomendacion de libros.

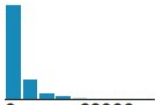
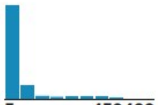
ratings_small.csv (2.33MB)

	# userId	# movieId	# rating	# timestamp
	1 671	1 163949	0.5 5	789652...147664...
1	1	31	2.5	1260759144
2	1	1029	3.0	1260759179
3	1	1061	3.0	1260759182
4	1	1129	2.0	1260759185
5	1	1172	4.0	1260759205
6	1	1263	2.0	1260759151
7	1	1287	2.0	1260759187
8	1	1293	2.0	1260759148
9	1	1339	3.5	1260759125
10	1	1343	2.0	1260759131
11	1	1371	2.5	1260759135
12	1	1405	1.0	1260759203
13	1	1953	4.0	1260759191
14	1	2105	4.0	1260759139
15	1	2150	2.0	1260759194

tmdb_5000_credits.csv (38.19MB)

	# movie_id	A title	A cast	A crew
	5 459488	4800 Unique Values	4761 Unique Values	4776 Unique Values
1	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gender": 2, "id": 65731, "name": "Sam Worthington", "order": 0}, {"cast_id": 3, "character": "Neytiri", "credit_i...	[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor", "name": "Stephen E. Rivkin"}, {"credit_id": "539c47ecc3a36810e3001f87", "department": "Art...
2	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Sparrow", "credit_id": "52fe4232c3a36847f800b50d", "gender": 2, "id": 85, "name": "Johnny Depp", "order": 0}, {"cast_id": 5, "character": "Will Turner", "cre...	[{"credit_id": "52fe4232c3a36847f800b579", "department": "Camera", "gender": 2, "id": 120, "job": "Director of Photography", "name": "Dariusz Wolski"}, {"credit_id": "52fe4232c3a36847f800b4fd", "depar...
3	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "credit_id": "52fe4d22c3a368484e1d8d6b", "gender":	[{"credit_id": "54805967c3a36829b5002c41", "department": "Sound", "gender": 2,
4	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Batman", "credit_id": "52fe4781c3a36847f8139869", "gender": 2, "id": 3894, "name": "Christian Bale", "order": 0}, {"cast_id": 8, "character": "Alfred Pennyw...	[{"credit_id": "52fe4781c3a36847f81398c3", "department": "Sound", "gender": 2, "id": 947, "job": "Original Music Composer", "name": "Hans Zimmer"}, {"credit_id": "52fe4781c3a36847f8139899", "departmen...
5	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "credit_id": "52fe479ac3a36847f813ea75", "gender": 2, "id": 60900, "name": "Taylor Kitsch", "order": 0}, {"cast_id": 20, "character": "Dejah Thoris", "credi...	[{"credit_id": "52fe479ac3a36847f813eaa3", "department": "Writing", "gender": 2, "id": 7, "job": "Screenplay", "name": "Andrew Stanton"}, {"credit_id": "52fe479ac3a36847f813ea65", "department": "Direc...
6	559	Spider-Man 3	[{"cast_id": 30, "character": "Peter Parker / Spider-Man", "credit_id": "52fe4252c3a36847f80151c7", "gender": 2, "id": 2219, "name": "Tobey Maguire",	[{"credit_id": "52fe4252c3a36847f80151a5", "department": "Production", "gender": 1, "id": 6410, "job": "Casting", "name": "Francine

tmdb_5000_movies.csv (5.43MB)

	# budget	A genres	A homepage	# id	A keywords	A original_language	A original_title
	 0 38000...	[{"id": 18, "nam... 8% [{"id": 35, "na... 6% Other (1173) 86%	1691 Unique Values	 5 459488	[] 9% [{"id": 10183, "n... 1% Other (4220) 90%	en 94% fr 1% Other (35) 5%	4801 Unique Values
1	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"}, {"id": 3388, "name": "space colony"}, {"id": 3679, "name": "society"}, {"id": 3801, "name": "...	en	Avatar
2	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 28, "name": "Action"}]	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "name": "drug abuse"}, {"id": 911, "name": "exotic island"}, {"id": 1319, "name": "east india trading company"}, {"id": 2038, "name": "love of one's life"}, ...	en	Pirates of the Caribbean: At World's End
3	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 80, "name": "Crime"}]	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name": "based on novel"}, {"id": 4289, "name": "secret agent"}, {"id": 9663, "name": "sequel"}, {"id": 14555, "name": "mi6"}, {"id": 156095,	en	Spectre

6. ANALISIS

6.1. Datos Estadisticos del sistema de recomendacion de libros

Se utilizo estos ejemplos para desarrollar el sistema de recomendacion de libros.

PARTE 1 : collaborative filtering

In [0]:
```
# IMPORTAR LIBRERIAS
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

In [0]:
```
# IMPORTAR DATA
books=pd.read_csv('./books.csv')
ratings=pd.read_csv('./ratings.csv')
tags=pd.read_csv('./tags.csv')
book_tags=pd.read_csv('./book_tags.csv')
```

In [0]:
```
# Titulos nulos
books['original_title'].isnull().sum()
```

Out[0]:
```
585
```

In [0]:
```
# Id's nulos
books['book_id'].isnull().sum()
```

Out[0]:
```
0
```

In [0]:
```
# Registros Rating nulos
ratings.apply(lambda x: x.isnull().sum(),axis=0)
```

Out[0]:
```
book_id    0
user_id    0
rating     0
dtype: int64
```

In [0]:
```
# Registros Libros nulos
books.apply(lambda x:x.isnull().sum(), axis=0)
```

Out[0]:
```
id                            0
book_id                       0
best_book_id                  0
work_id                       0
books_count                   0
isbn                        700
isbn13                      585
authors                       0
original_publication_year    21
original_title              585
title                         0
language_code              1084
average_rating                0
ratings_count                 0
work_ratings_count            0
work_text_reviews_count       0
ratings_1                     0
ratings_2                     0
ratings_3                     0
ratings_4                     0
ratings_5                     0
image_url                     0
small_image_url               0
dtype: int64
```

In [0]:
```
# Obtener data frame con las columnas que deseamos mostrar
books_dataset = pd.DataFrame(books, columns=['book_id', 'authors', 'title', 'average_rating'])
```

In [0]:
```
# Ordenar los libros por ID
books_dataset = books_dataset.sort_values('book_id')
```

In [0]:
```
# Verificar que ordeno
books_dataset['book_id']
```

Out[0]:
```
26            1
20            2
1             3
17            5
23            6
3274          8
3752         10
53           11
336          13
373          21
1459         24
1975         25
2320         26
2278         27
1448         28
4078         29
963          30
188          33
18           34
4228         36
387          50
3503         67
638          93
7682         98
2816        105
1104        106
1261        112
2490        117
3229        119
1596        122
          ...

          ...
3986    29632984
2331    29639736
3884    29780253
4851    29868610
9391    29906980
8582    29925715
7702    29975458
2272    29981261
8204    29991719
5218    30002998
6442    30008702
4640    30065028
7447    30226723
9412    30253700
4575    30253864
7093    30314465
9815    30364931
1538    30555488
5883    30831912
9547    30839185
5110    31140847
5295    31176886
8712    31194270
7442    31538614
6427    31538635
7522    31538647
4593    31845516
9568    32075671
9579    32848471
8891    33288638
Name: book_id, Length: 10000, dtype: int64
```

In [0]:
```
# Unir libros con rating por el book_id
books_data = pd.merge(books_dataset, ratings, on='book_id')
```

In [0]:
```
each_book_rating = pd.pivot_table(books_data, index='user_id', values='rating', columns='title', fill_valu
e=0)
```

In [0]:
```
# Unir libros con rating por el book_id
books_data = pd.merge(books_dataset, ratings, on='book_id')
```

In [0]:
```
each_book_rating = pd.pivot_table(books_data, index='user_id', values='rating', columns='title', fill_valu
e=0)
```

In [0]:
```
each_book_rating
```

Out[0]:

title	'Salem's Lot	'Tis (Frank McCourt, #2)	1421: The Year China Discovered America	1776	1984	A Bend in the River	A Bend in the Road	A Brief History of Time	A Briefer History of Time	A Case of Need	A Christmas Carol	A Christmas Carol and Other Christmas Writings	A Fine Balance
user_id													
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
53357	0	0	0	0	0	0	0	0	0	0	0	0	0
53364	0	0	0	0	0	0	0	0	0	0	0	0	0
53366	0	0	0	0	0	0	0	0	0	3	0	0	0
53371	0	0	0	0	0	0	0	0	0	0	0	0	0
53372	0	0	0	0	0	0	0	3	0	0	0	0	0
53373	0	0	0	0	0	0	0	0	0	0	0	0	0
53374	0	0	0	0	0	0	0	0	0	0	0	0	0
53377	0	0	0	0	0	0	0	0	0	0	0	0	0
53378	0	0	0	0	0	0	0	0	0	0	0	0	0
53381	0	0	0	0	0	0	0	0	0	0	0	0	0
53382	0	0	0	0	0	0	0	0	0	0	0	0	0
53388	0	0	0	0	0	0	0	0	0	0	0	0	0
53389	0	0	0	0	0	0	0	0	0	0	0	0	0
53390	0	0	0	0	0	0	0	0	0	0	0	0	0
53391	0	0	0	0	0	0	0	0	0	0	0	0	0
53393	0	0	0	0	0	0	0	0	0	0	0	0	0
53398	0	0	0	0	0	0	0	0	0	0	0	0	0
53400	0	0	0	0	0	0	0	0	0	0	0	0	0
53401	0	0	0	0	0	0	0	0	0	0	0	0	0
53403	0	0	0	0	0	0	0	0	0	0	0	0	0

53398	0	0	0	0	0	0	0	0	0	0	0	0	0
53400	0	0	0	0	0	0	0	0	0	0	0	0	0
53401	0	0	0	0	0	0	0	0	0	0	0	0	0
53403	0	0	0	0	0	0	0	0	0	0	0	0	0
53404	0	0	0	0	0	0	0	0	0	0	0	0	0
53406	0	0	0	0	0	0	0	0	0	0	0	0	0
53408	0	0	0	0	0	0	0	0	0	0	0	0	0
53409	0	0	0	0	0	0	0	0	0	0	0	0	0
53416	0	0	0	0	0	0	0	0	0	0	0	0	0
53419	0	0	0	0	0	0	0	0	0	0	0	0	0
53420	0	0	0	0	0	0	0	0	0	0	0	0	0
53422	0	0	0	0	0	0	0	0	0	0	0	0	0
53423	0	0	0	0	0	0	0	0	0	0	0	0	0
53424	0	0	0	0	0	0	0	0	0	0	0	0	0

28906 rows × 812 columns

In [0]:
```
book_corr = np.corrcoef(each_book_rating.T)
```

In [0]:
```
book_corr.shape
```

Out[0]:
```
(812, 812)
```

In [0]:
```
book_list=  list(each_book_rating)
book_titles =[]
for i in range(len(book_list)):
    book_titles.append(book_list[i])
```

In [0]:
```
book_titles
```

Out[0]:
```
["'Salem's Lot",
 "'Tis (Frank McCourt, #2)",
 '1421: The Year China Discovered America',
 '1776',
 '1984',
 'A Bend in the River',
 'A Bend in the Road',
 'A Brief History of Time',
 'A Briefer History of Time',
 'A Case of Need',
 'A Christmas Carol',
 'A Christmas Carol and Other Christmas Writings',
 'A Fine Balance',
 'A Great and Terrible Beauty (Gemma Doyle, #1)',
 'A Heartbreaking Work of Staggering Genius',
 'A History of God: The 4,000-Year Quest of Judaism, Christianity, and Islam',
 'A History of the World in 6 Glasses',
 'A Home at the End of the World',
 'A House for Mr Biswas',
 'A Lesson Before Dying',
 'A Little Princess',
 'A Living Nightmare (Cirque Du Freak, #1)',
 'A Man Without a Country',
 'A Map of the World',
 "A Midsummer Night's Dream",
 'A Million Little Pieces',
 'A Modest Proposal and Other Satirical Works',
 'A Moveable Feast',
 'A Painted House',
 "A People's History of the United States",
 'A Portrait of the Artist as a Young Man',
 'A Prayer for Owen Meany',
 'A Raisin in the Sun',
 'A Room with a View',
 'A Separate Peace',
 'A Short History of Nearly Everything',
 'A Son of the Circus',
 'A Spot of Bother',
 "A Supposedly Fun Thing I'll Never Do Again:  Essays and Arguments",
 'A Tale of Two Cities',
 'A Virtuous Woman',
 'A Walk in the Woods',
 'A Walk to Remember',
 'A Widow for One Year',
 'A Woman of Substance (Emma Harte Saga #1)',
 'About a Boy',
 'Agamemnon (Oresteia, #1)',
 "Ahab's Wife, or The Star-Gazer",
 'Airframe',
 'All Families are Psychotic',
 "All the King's Men",
 'All the Names',
 'All-of-a-Kind Family (All-of-a-Kind Family, #1)',
 'Allies of the Night (Cirque du Freak, #8)',
```

"America (The Book): A Citizen's Guide to Democracy Inaction",
'American Gods (American Gods, #1)',
'Amsterdam',
'An Ideal Husband',
'Anansi Boys',
'Angels & Demons (Robert Langdon, #1)',
"Anil's Ghost",
'Animal Farm',
'Animal Farm / 1984',
'Anne Frank Remembered: The Story of the Woman Who Helped to Hide the Frank Family',
'Anne Frank: Beyond the Diary - A Photographic Remembrance',
'Anne of Green Gables (Anne of Green Gables, #1)',
'Another Bullshit Night in Suck City',
'Another Roadside Attraction',
'Anthem',
'Antigone (The Theban Plays, #3)',
'As the Crow Flies',
'Assassination Vacation',
'Atlas Shrugged',
'Atonement',
'Awakening the Buddha Within: Tibetan Wisdom for the Western World',
'Bagombo Snuff Box',
'Baltasar and Blimunda',
'Barrel Fever: Stories and Essays',
'Bel Canto',
'Beloved',
'Betsy-Tacy (Betsy-Tacy, #1)',
'Birdsong',
'Black Beauty',
'Black and Blue',
'Bleach, Volume 01',
'Bleach, Volume 15',
'Bleachers',
'Blind Willow, Sleeping Woman',
'Blindness',
'Blink',
'Blue Like Jazz: Nonreligious Thoughts on Christian Spirituality',
'Blue Ocean Strategy: How To Create Uncontested Market Space And Make The Competition Irrelevant',
'Bluebeard',
'Book of the Dead (Kay Scarpetta, #15)',
'Boy: Tales of Childhood',
'Brave New World',
'Brave New World / Brave New World Revisited',
'Brave New World Revisited ',
'Breakfast of Champions',
'Breaking the Spell: Religion as a Natural Phenomenon',
'Breath, Eyes, Memory',
'Bridge to Terabithia',
'Brief Interviews with Hideous Men',
'Brokeback Mountain',
'Built to Last: Successful Habits of Visionary Companies',
'Burmese Days',
'By the River Piedra I Sat Down and Wept',
'By the Shores of Silver Lake (Little House, #5)',

```
'Cane River',
'Cannery Row',
"Carrie / 'Salem's Lot / The Shining",
'Carter Beats the Devil',
'Casino Royale (James Bond, #1)',
'Cause of Death (Kay Scarpetta, #7)',
'Chapterhouse: Dune (Dune Chronicles #6)',
'Charlie and the Chocolate Factory (Charlie Bucket, #1)',
'Charlie and the Great Glass Elevator (Charlie Bucket, #2)',
'Children of Dune (Dune Chronicles #3)',
'City of Glass (The New York Trilogy, #1)',
'City of the Beasts (Eagle and Jaguar, #1)',
'Cloudy With a Chance of Meatballs',
'Code to Zero',
'Collapse: How Societies Choose to Fail or Succeed',
'Comfort Me with Apples: More Adventures at the Table',
'Complete Works of Oscar Wilde',
"Complications: A Surgeon's Notes on an Imperfect Science",
'Confessions of a Shopaholic (Shopaholic, #1)',
'Confessions of an Economic Hit Man',
'Congo',
'Consider the Lobster and Other Essays',
"Corelli's Mandolin",
'Cover Her Face (Adam Dalgliesh #1)',
'Cradle and All',
'Cradle to Cradle: Remaking the Way We Make Things',
'Crime and Punishment',
'Crossing to Safety',
'Crow Lake',
'Cry, the Beloved Country',
'Cryptonomicon',
'Daniel Deronda',
'Danny the Champion of the World',
'Darkness',
"Darwin's Dangerous Idea: Evolution and the Meanings of Life",
'Deadeye Dick',
'Dear John',
'Deception Point',
'Deerskin',
'Demons',
'Desert Flower',
'Diamonds Are Forever (James Bond, #4)',
'Digging to America',
"Dirk Gently's Holistic Detective Agency (Dirk Gently #1)",
'Disclosure',
'Disgrace',
'Dispatches',
'Do Androids Dream of Electric Sheep?',
'Doctor No (James Bond, #6)',
'Don Quixote',
"Don't Make Me Think: A Common Sense Approach to Web Usability",
"Dr. Seuss's ABC: An Amazing Alphabet Book! (Bright and Early Board Books)",
'Dragonfly in Amber (Outlander, #2)',
'Dreamland',
'Drowning Ruth',
```

```
'Dune Messiah (Dune Chronicles #2)',
'East of Eden',
'Eaters of the Dead',
'Eats, Shoots & Leaves: The Zero Tolerance Approach to Punctuation',
'Economics in One Lesson: The Shortest & Surest Way to Understand Basic Economics',
'Eiger Dreams: Ventures Among Men and Mountains',
'Eleven Minutes',
'Embroideries',
'Emily of New Moon (Emily, #1)',
'Emma',
'Enchantment',
"Ender's Shadow (Ender's Shadow, #1)",
'Enduring Love',
'Endymion (Hyperion Cantos, #3)',
'Ethan Frome',
'Even Cowgirls Get the Blues',
'Everyday Italian: 125 Simple and Delicious Recipes',
'Execution: The Discipline of Getting Things Done',
'Extremely Loud and Incredibly Close',
'Fahrenheit 451',
'Fall on Your Knees',
'Falling Angels',
'Fantastic Mr. Fox',
'Farewell, My Lovely (Philip Marlowe, #2)',
'Farmer Boy (Little House, #3)',
'Fast Food Nation: The Dark Side of the All-American Meal',
'Fear and Loathing in Las Vegas',
"Fear and Loathing on the Campaign Trail '72",
'Fear of Flying',
'Fever Pitch',
'Fierce Invalids Home from Hot Climates',
'Fight Club',
'Fire Sea (The Death Gate Cycle, #3)',
'First They Killed My Father: A Daughter of Cambodia Remembers',
'Five Little Peppers and How They Grew',
'Forever Amber ',
'Forever in Blue: The Fourth Summer of the Sisterhood (Sisterhood, #4)',
'Founding Brothers: The Revolutionary Generation',
'Four Blondes',
'Four to Score (Stephanie Plum, #4)',
'Franny and Zooey',
'Freak the Mighty (Freak The Mighty, #1)',
'Freakonomics: A Rogue Economist Explores the Hidden Side of Everything (Freakonomics, #1)',
'Fried Green Tomatoes at the Whistle Stop Cafe',
'From Beirut to Jerusalem',
"From Potter's Field (Kay Scarpetta, #6)",
'From Russia With Love (James Bond, #5)',
'From the Mixed-Up Files of Mrs. Basil E. Frankweiler',
'Fullmetal Alchemist, Vol. 1 (Fullmetal Alchemist, #1)',
'Galápagos',
'Gates of Fire: An Epic Novel of the Battle of Thermopylae',
'Generation X: Tales for an Accelerated Culture',
'Genome: the Autobiography of a Species in 23 Chapters',
'Getting Things Done: The Art of Stress-Free Productivity',
'Ghostwritten',
```

In [0]:
```
def get_recommendation(books_list):
    book_similarities = np.zeros(book_corr.shape[0])

    for book in books_list:
#         print(book)
        book_index = book_titles.index(book)
#         print(book_index)
        book_similarities += book_corr[book_index]
    book_preferences = []
    for i in range(len(book_titles)):
        book_preferences.append((book_titles[i],book_similarities[i]))

    return sorted(book_preferences, key= lambda x: x[1], reverse=True)

#     return book_preferences
```

In [0]:
```
my_fav_books = ['The Alchemist','The Adventures of Sherlock Holmes','The Great Gatsby','To Kill a Mockingb
ird','The Da Vinci Code (Robert Langdon, #2)','The Fellowship of the Ring (The Lord of the Rings, #1)']
```

In [0]:
```
book_recommendations = get_recommendation(my_fav_books)
print('The books you should like')
print('-'*25)
i=0
cnt=0
while cnt < 9:
    book_to_read = book_recommendations[i][0]
    i += 1
    if book_to_read in my_fav_books:
        continue
    else:
        print(book_to_read)
        cnt += 1
```

```
The books you should like
-------------------------
The Plot Against America
The New York Trilogy
Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
The Lord of the Rings (The Lord of the Rings, #1-3)
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings
The Ultimate Hitchhiker's Guide to the Galaxy
The Body Farm (Kay Scarpetta, #5)
Perfume: The Story of a Murderer
Hatchet (Brian's Saga, #1)
```

PARTE 2 : BASADO EN POPULARIDAD

In [0]:
```
# demographic recomendation
C = books['average_rating'].mean()
C
```

Out[0]: 4.002191000000001

In [0]:
```
m= books['ratings_count'].quantile(0.9)
m
```

Out[0]: 94103.10000000003

In [0]:
```
q_books = books.copy().loc[books['ratings_count'] >= m]
q_books.shape
```

Out[0]: (1000, 23)

In [0]:
```
def weighted_rating(x, m=m, C=C):
    v = x['ratings_count']
    R = x['average_rating']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [0]:
```
# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_books['score'] = q_books.apply(weighted_rating, axis=1)
q_books = q_books.sort_values('score', ascending=False)
```

In [0]:
```
q_books[['id','book_id','original_title','ratings_count','average_rating','score']].head(10)
```

Out[0]:

	id	book_id	original_title	ratings_count	average_rating	score
24	25	136251	Harry Potter and the Deathly Hallows	1746574	4.61	4.578926
26	27	1	Harry Potter and the Half-Blood Prince	1678823	4.54	4.511454
17	18	5	Harry Potter and the Prisoner of Azkaban	1832823	4.53	4.504224
23	24	6	Harry Potter and the Goblet of Fire	1753043	4.53	4.503111
421	422	862041	Complete Harry Potter Boxed Set	190050	4.74	4.495659
134	135	62291	A Storm of Swords	469022	4.54	4.450127
191	192	186074	The Name of the Wind	400101	4.55	4.445690
20	21	2	Harry Potter and the Order of the Phoenix	1735368	4.46	4.436452
1	2	3	Harry Potter and the Philosopher's Stone	4602479	4.44	4.431228
160	161	18512	The Return of the King	463959	4.51	4.424371

In [0]:
```
## colaborating https://www.kaggle.com/jmy666/book-recommendation-collaborative-filtering
books=pd.read_csv('./books.csv')
booksC = books[['book_id','authors','title']]
booksC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
book_id    10000 non-null int64
authors    10000 non-null object
title      10000 non-null object
dtypes: int64(1), object(2)
memory usage: 234.5+ KB
```

In [0]:
```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 981756 entries, 0 to 981755
Data columns (total 3 columns):
book_id    981756 non-null int64
user_id    981756 non-null int64
rating     981756 non-null int64
dtypes: int64(3)
memory usage: 22.5 MB
```

In [0]:
```
ratings['rating'].unique()
```

Out[0]:
```
array([5, 3, 4, 1, 2])
```

In [0]:
```
books_data = pd.merge(booksC, ratings, on='book_id')
```

Instalamos surprise

In [0]:
```
!pip install surprise
```

```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6fdda95693468ed061945493
dea2dd37c5618b/surprise-0.1-py2.py3-none-any.whl
Collecting scikit-surprise (from surprise)
  Downloading https://files.pythonhosted.org/packages/4d/fc/cd4210b247d1dca421c25994740cbbf03c5e980e31881f
10eaddf45fdab0/scikit-surprise-1.0.6.tar.gz (3.3MB)
    |████████████████████████████████| 3.3MB 2.8MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-surpris
e->surprise) (0.13.2)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-packages (from scikit-surpri
se->surprise) (1.16.4)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surpris
e->surprise) (1.3.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise
->surprise) (1.12.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/ec/c0/55/3a28eab06b53c220015063ebbdb81213cd3dcbb72c088251ec
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.0.6 surprise-0.1
```

In [0]:
```python
from surprise import Reader, Dataset, SVD, evaluate,accuracy
from surprise.model_selection import train_test_split
from surprise.model_selection import KFold
reader = Reader(rating_scale=(1,5))


data = Dataset.load_from_df(ratings[['book_id', 'user_id', 'rating']], reader)
trainset, testset = train_test_split(data, test_size=.25)

#kf = KFold(n_splits=3)

algo = SVD()
algo.fit(trainset)
predictions = algo.test(testset)
accuracy.rmse(predictions, verbose=True)
#for trainset, testset in kf.split(data):

    # train and test algorithm.
    #algo.fit(trainset)
    #predictions = algo.test(testset)

    # Compute and print Root Mean Squared Error
    #accuracy.rmse(predictions, verbose=True)
```

```
RMSE: 0.8453
```

Out[0]:
```
0.8452914927765088
```

In [0]:
```
def recommendation(user_id):
    user = booksC.copy()
    already_read = books_data[books_data['user_id'] == user_id]['book_id'].unique()
    user = user.reset_index()
    user = user[~user['book_id'].isin(already_read)]
    user['Estimate_Score']=user['book_id'].apply(lambda x: algo.predict(user_id, x).est)
    user = user.drop('book_id', axis = 1)
    user = user.sort_values('Estimate_Score', ascending=False)
    print(user[['index','title','Estimate_Score']].head(10))
```

In [0]:
```
recommendation(2)
```

	index	title	Estimate_Score
467	467	Their Eyes Were Watching God	5.0
6303	6303	Remember Me	5.0
6652	6652	Elric of Melniboné (Elric, #1)	5.0
3790	3790	The Crimson Petal and the White	5.0
6500	6500	Crooked House	5.0
8607	8607	Darkest Fear (Myron Bolitar #7)	5.0
1552	1552	Circus of the Damned (Anita Blake, Vampire Hun...	5.0
2525	2525	All-Star Superman, Vol. 1	5.0
6315	6315	The Anti-Christ	5.0
6298	6298	Falling Angels	5.0

7. CONCLUSIONES

En este sistema de recomendacion de Biblioteca se utilizaron diferentes datos para obtener datos estadisticos de los libros para realizar validaciones en cada uno de los módulos implementados.

Se establecieron varias metodologías para desarrollar los módulos de libros de manera que ofrezcan procesos eficientes y una interfaz amigable al usuario.

8. BIBLIOGRAFIA

- https://dockertips.com/volumenes
- https://cerebro-digital.com/panel/knowledgebase/64/ExportarorImportar-contenedor-de-Docker-via-archivo-TAR.html
- https://www.docker.com/
- https://www.campusmvp.es/recursos/post/los-beneficios-de-utilizar-docker-y-contenedores-a-la-hora-de-programar.aspx
- https://www.colap.io/