Московский государственный технический университет им. Н.Э.Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №4 по дисциплине

«Методы машинного обучения» на тему

«Создание рекомендательной модели. »

Выполнил:
Студент группы ИУ5-21М Ся Бэйбэй

____

Москва — 2022г.

1. Цель лабораторной работы

Изучение разработки рекомендательных моделей.

2. Задание

1.Выбрать произвольный набор данных (датасет), предназначенный для построения рекомендательных моделей.

2.Опираясь на материалы лекции, сформировать рекомендации для одного пользователя (объекта) двумя произвольными способами.

3.Сравнить полученные рекомендации (если это возможно, то с применением метрик).

Отчет по лабораторной работе должен содержать:

1.титульный лист;

2.описание задания;

3.текст программы;

экранные формы с примерами выполнения программы.

3. Ход выполнения работы

Подключим все необходимые библиотеки и настроим отображение графиков :

```
In [6]:
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import plotly.graph_objects as go
from scipy.sparse import csr_matrix
import numpy as np
from sklearn.neighbors import NearestNeighbors
from fuzzywuzzy import fuzz
```

Для выполнения данной лабораторной работы возьмём набор данных по приложениям:

```
In [7]:
books= pd.read_csv('./books.csv')
```

```
In [10]:
ratings = pd.read_csv('./ratings.csv')
```

```
In [11]:
tags = pd.read_csv('./book_tags.csv')
```

```
In [12]:
btags = pd.read_csv('./tags.csv')
```

Посмотрим на эти наборы данных:

In [8]:

```
books.head()
```

Out[8]:

	id	book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	...	ratings_count	work_ra
0	1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008.0	The Hunger Games	...	4780653	
1	2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	Harry Potter and the Philosopher's Stone	...	4602479	
2	3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005.0	Twilight	...	3866839	
3	4	2657	2657	3275794	487	61120081	9.780061e+12	Harper Lee	1960.0	To Kill a Mockingbird	...	3198671	
4	5	4671	4671	245494	1356	743273567	9.780743e+12	F. Scott Fitzgerald	1925.0	The Great Gatsby	...	2683664	

5 rows × 23 columns

```
ratings.head()
```

Out[10]:

	book_id	user_id	rating
0	1	314	5
1	1	439	3
2	1	588	5
3	1	1169	4
4	1	1185	4

```
tags.tail()
```

Out[11]:

	goodreads_book_id	tag_id	count
999907	33288638	21303	7
999908	33288638	17271	7
999909	33288638	1126	7
999910	33288638	11478	7
999911	33288638	27939	7

In [12]:

```
btags = pd.read_csv('./tags.csv')
btags.tail()
```

Out[12]:

	tag_id	tag_name
34247	34247	Childrens
34248	34248	Ｆａｖｏｒｉｔｅｓ
34249	34249	Ｍａｎｇａ
34250	34250	ＳＥＲＩＥＳ
34251	34251	ｆａｖｏｕｒｉｔｅｓ

3.1. Обработка данных

Очистка данных, удаление дубликатов.

```
In [13]:  ratings=ratings.sort_values("user_id")
          ratings.shape

Out[13]:  (981756, 3)

In [14]:  ratings.drop_duplicates(subset =["user_id","book_id"],
                                   keep = False, inplace = True)
          ratings.shape

Out[14]:  (977269, 3)

In [15]:  print(books.shape)
          books.drop_duplicates(subset='original_title',keep=False,inplace=True)
          print(books.shape)

          (10000, 23)
          (9151, 23)

In [16]:  print(btags.shape)
          btags.drop_duplicates(subset='tag_id',keep=False,inplace=True)
          print(btags.shape)

          (34252, 2)
          (34252, 2)

In [17]:  print(tags.shape)
          tags.drop_duplicates(subset=['tag_id','goodreads_book_id'],keep=False,inplace=True)
          print(tags.shape)

          (999912, 3)
          (999896, 3)

In [18]:  fillnabooks= books.fillna('')
```

Content Based.

По следующим факторам: Заголовок,Авторы,Средний рейтинг

Очистка данных - перевод всех слов в нижний регистр. Извлечение только признаков из заданных данных.

```
In [18]:  fillnabooks= books.fillna('')

In [19]:  def clean_data(x):
                  return str.lower(x.replace(" ", ""))

In [20]:  features=['original_title','authors','average_rating']
          fillednabooks=fillnabooks[features]

In [21]:  fillednabooks = fillednabooks.astype(str)
          fillednabooks.dtypes

Out[21]:  original_title    object
          authors           object
          average_rating    object
          dtype: object

In [22]:  for feature in features:
              fillednabooks[feature] = fillednabooks[feature].apply(clean_data)

          fillednabooks.head(2)

Out[22]:
```

	original_title	authors	average_rating
0	thehungergames	suzannecollins	4.34
1	harrypotterandthephilosopher'sstone	j.k.rowling,marygrandpré	4.44

Создание "soup" или для всех строк.

```
In [23]:  def create_soup(x):
              return x['original_title']+ ' ' + x['authors'] + ' ' + x['average_rating']

In [24]:  fillednabooks['soup'] = fillednabooks.apply(create_soup, axis=1)
```

Импорт векторизатора количества терминов.

```
In [25]:
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(fillednabooks['soup'])

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

```
In [26]:
fillednabooks=fillednabooks.reset_index()
indices = pd.Series(fillednabooks.index, index=fillednabooks['original_title'])
```

Рекомендация:

```
In [27]:
def get_recommendations_new(title, cosine_sim=cosine_sim2):
    title=title.replace(' ','').lower()
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return list(books['original_title'].iloc[movie_indices])
```

```
In [29]:
l=get_recommendations_new('The Hobbit', cosine_sim2)
fig = go.Figure(data=[go.Table(header=dict(values=l,fill_color='orange'))
                     ])
fig.show()
```

Next	The Hobbit or There and Back Again	The Fellowship of the Ring	The Two Towers	The Return of the King	The Lord of the Rings	City of Heavenly Fire	The Tenth Circle	Luckiest Girl Alive	The Hobbit and The Lord of the Rings

```
In [30]:
l=get_recommendations_new('Harry Potter and The Chamber of Secrets', cosine_sim2)
fig = go.Figure(data=[go.Table(header=dict(values=l,fill_color='orange'))
                     ])
fig.show()
```

Harry Potter and the Order of the Phoenix	Harry Potter and the Goblet of Fire	Harry Potter and the Deathly Hallows	Harry Potter Boxed Set Books 1-4	Harry Potter and the Philosopher's Stone	Harry Potter and the Prisoner of Azkaban	Harry Potter and the Half-Blood Prince	The Casual Vacancy	Shadow Kiss	The Tales of Beedle the Bard

Collaborative Filtering

Удалить пустые данные.

In [31]:
```
usecols=['book_id', 'original_title']
books_col=books[usecols]
```

In [32]:
```
books_col.dropna()
```

Out[32]:

	book_id	original_title
0	2767052	The Hunger Games
1	3	Harry Potter and the Philosopher's Stone
3	2657	To Kill a Mockingbird
4	4671	The Great Gatsby
5	11870085	The Fault in Our Stars
...	...	...
9995	7130616	Bayou Moon
9996	208324	Means of Ascent
9997	77431	The Mauritius Command
9998	8565083	Cinderella Ate My Daughter: Dispatches from th...
9999	8914	The First World War

9151 rows × 2 columns

Создание матрицы

In [33]:
```
# pivot ratings into movie features
df_book_features = ratings.pivot(index='book_id',columns='user_id',values='rating').fillna(0)
mat_book_features = csr_matrix(df_book_features.values)
```

In [34]:
```
df_book_features.head()
```

Out[34]:

user_id book_id	1	2	3	4	5	6	7	8	9	10	...	53415	53416	53417	53418	53419	53420	53421	53422	53423	53424
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 53380 columns

Здесь алгоритм K ближайших соседей используется для поиска ближайшей книги с наименьшим доступным расстоянием.

In [35]:
```

model_knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)


num_users = len(ratings.user_id.unique())
num_items = len(ratings.book_id.unique())
print('There are {} unique users and {} unique movies in this data set'.format(num_users, num_items))
```

```
There are 53380 unique users and 10000 unique movies in this data set
```

In [36]:
```
ratings=ratings.dropna()
```

In [37]:
```
df_ratings_cnt_tmp = pd.DataFrame(ratings.groupby('rating').size(), columns=['count'])
df_ratings_cnt_tmp.head(10)
```

Out[37]:

rating	count
1	19485
2	63010
3	247698
4	355878
5	291198

In [38]:
```python
total_cnt = num_users * num_items
rating_zero_cnt = total_cnt - ratings.shape[0]

df_ratings_cnt = df_ratings_cnt_tmp.append(
    pd.DataFrame({'count': rating_zero_cnt}, index=[0.0]),
    verify_integrity=True,
).sort_index()
df_ratings_cnt
```

Out[38]:

	count
0.0	532822731
1.0	19485
2.0	63010
3.0	247698
4.0	355878
5.0	291198

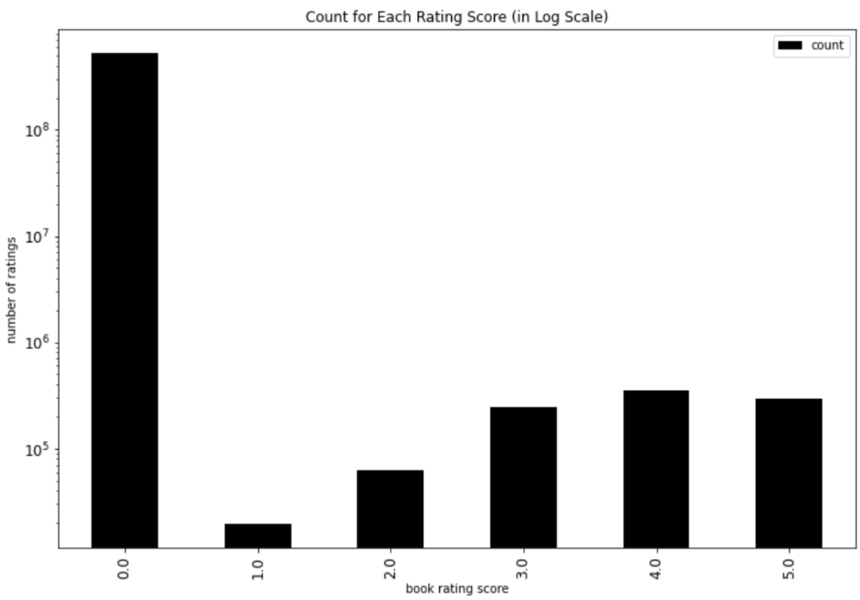
После подсчета всех оценок видно, что большое количество книг имеют рейтинг 0 или не имеют рейтинга.

In [39]:
```python
df_ratings_cnt['log_count'] = np.log(df_ratings_cnt['count'])
df_ratings_cnt

import matplotlib.pyplot as plt


get_ipython().run_line_magic('matplotlib', 'inline')
ax = df_ratings_cnt[['count']].reset_index().rename(columns={'index': 'rating score'}).plot(
    x='rating score',
    y='count',
    kind='bar',
    figsize=(12, 8),
    title='Count for Each Rating Score (in Log Scale)',
    logy=True,
    fontsize=12,color='black'
)
ax.set_xlabel("book rating score")
ax.set_ylabel("number of ratings")
```

Out[39]: Text(0, 0.5, 'number of ratings')



На графике четко видно, что многие данные неактуальны и могут быть удалены.

In [40]:
```python
df_books_cnt = pd.DataFrame(ratings.groupby('book_id').size(), columns=['count'])
df_books_cnt.head()
```

Out[40]:

book_id	count
1	100
2	100
3	100
4	100
5	100

In [41]:
```python
popularity_thres = 60
popular_movies = list(set(df_books_cnt.query('count >= @popularity_thres').index))
df_ratings_drop = ratings[ratings.book_id.isin(popular_movies)]
print('shape of original ratings data: ', ratings.shape)
print('shape of ratings data after dropping unpopular movies: ', df_ratings_drop.shape)
```

```
shape of original ratings data:  (977269, 3)
shape of ratings data after dropping unpopular movies:  (975605, 3)
```

In [42]:
```python
# get number of ratings given by every user
df_users_cnt = pd.DataFrame(df_ratings_drop.groupby('user_id').size(), columns=['count'])
df_users_cnt.head()
```

Out[42]:

user_id	count
1	3
2	3
3	2
4	3
5	5

In [43]:
```python
ratings_thres = 50
active_users = list(set(df_users_cnt.query('count >= @ratings_thres').index))
df_ratings_drop_users = df_ratings_drop[df_ratings_drop.user_id.isin(active_users)]
print('shape of original ratings data: ', ratings.shape)
print('shape of ratings data after dropping both unpopular movies and inactive users: ', df_ratings_drop_users.shape)
```

```
shape of original ratings data:  (977269, 3)
shape of ratings data after dropping both unpopular movies and inactive users:  (417687, 3)
```

In [44]:
```python
book_user_mat = df_ratings_drop_users.pivot(index='book_id', columns='user_id', values='rating').fillna(0)
book_user_mat
```

Out[44]:

user_id book_id	7	35	41	75	119	143	145	153	158	173	...	53245	53279	53281	53292	53293	53318	53352	53366	53373	53381
1	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	5.0	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	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	5.0	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	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

9886 rows × 4892 columns

In [45]:

```python
book_user_mat_sparse = csr_matrix(book_user_mat.values)
```

In [46]:

```python
book_user_mat_sparse
```

Out[46]:

```
<9886x4892 sparse matrix of type '<class 'numpy.float64'>'
	with 417687 stored elements in Compressed Sparse Row format>
```

In [47]:

```python
model_knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)
# fit
model_knn.fit(book_user_mat_sparse)
```

Out[47]:

```
NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=20)
```

In [52]:

```python
def fuzzy_matching(mapper, fav_book, verbose=True):
    """
    return the closest match via fuzzy ratio.

    Parameters
    ----------
    mapper: dict, map movie title name to index of the movie in data
    fav_movie: str, name of user input movie

    verbose: bool, print log if True
    Return
    ------
    index of the closest match
    """
    match_tuple = []
    # get match
    for title, idx in mapper.items():
        ratio = fuzz.ratio(title.lower(), fav_book.lower())
        if ratio >= 60:
            match_tuple.append((title, idx, ratio))
    # sort
    match_tuple = sorted(match_tuple, key=lambda x: x[2])[::-1]
    if not match_tuple:
        print('Oops! No match is found')
        return
    if verbose:
        print('Found possible matches in our database: {0}\n'.format([x[0] for x in match_tuple]))
    return match_tuple[0][1]
```

In [53]:

```python
def make_recommendation(model_knn, data, mapper, fav_book, n_recommendations):
    """
    return top n similar book recommendations based on user's input book
    Parameters
    ----------
    model_knn: sklearn model, knn model
    data: book-user matrix
    mapper: dict, map book title name to index of the book in data
    fav_book: str, name of user input book
    n_recommendations: int, top n recommendations
    Return
    ------
    list of top n similar book recommendations
    """
    # fit
    model_knn.fit(data)
    # get input movie index
    print('You have input book:', fav_book)
    idx = fuzzy_matching(mapper, fav_book, verbose=True)

    print('Recommendation system starting to make inference')
    print('......\n')
    distances, indices = model_knn.kneighbors(data[idx], n_neighbors=n_recommendations+1)

    raw_recommends =         sorted(list(zip(indices.squeeze().tolist(), distances.squeeze().tolist())), key=lambda x: x[1])[:0:-1]
    # get reverse mapper
    reverse_mapper = {v: k for k, v in mapper.items()}
    # print recommendations
    print('Recommendations for {}:'.format(fav_book))
    rec=[]
    for i, (idx, dist) in enumerate(raw_recommends):
        if idx not in reverse_mapper.keys():
            continue
        print('{0}: {1}, with distance of {2}'.format(i+1, reverse_mapper[idx], dist))
        rec.append(reverse_mapper[idx])
    return rec
```

In [54]:
```python
my_favorite = 'To Kill a Mockingbird'
indices = pd.Series(books_col.index, index=books_col['original_title'])
```

In [55]:
```python
make_recommendation(
    model_knn=model_knn,
    data=book_user_mat_sparse,
    fav_book=my_favorite,
    mapper=indices,
    n_recommendations=10)
```

```
You have input book: To Kill a Mockingbird
Found possible matches in our database: ['To Kill a Mockingbird', 'Mockingbird', 'Stolen Songbird']

Recommendation system starting to make inference
......

Recommendations for To Kill a Mockingbird:
1: Lord of the Flies , with distance of 0.45598309432313877
2: Little Women, with distance of 0.4526896099993938
3: Nineteen Eighty-Four, with distance of 0.4396460119625992
4: Memoirs of a Geisha, with distance of 0.43283216907946764
5: Animal Farm: A Fairy Story, with distance of 0.4252435075403517
6: Pride and Prejudice, with distance of 0.4251608152166305
7: Of Mice and Men , with distance of 0.4204446294803902
8: Harry Potter and the Philosopher's Stone, with distance of 0.3892592020883805
9: The Catcher in the Rye, with distance of 0.3699905318987523
10: The Great Gatsby, with distance of 0.2966652339964868
```

Out[55]:
```
['Lord of the Flies ',
 'Little Women',
 'Nineteen Eighty-Four',
 'Memoirs of a Geisha',
 'Animal Farm: A Fairy Story',
 'Pride and Prejudice',
 'Of Mice and Men ',
 "Harry Potter and the Philosopher's Stone",
 'The Catcher in the Rye',
 'The Great Gatsby']
```

In [56]:
```python
make_recommendation(
    model_knn=model_knn,
    data=book_user_mat_sparse,
    fav_book='Harry Potter and the Chamber of Secrets',
    mapper=indices,
    n_recommendations=10)
```

```
You have input book: Harry Potter and the Chamber of Secrets
Found possible matches in our database: ['Harry Potter and the Chamber of Secrets', 'Harry Potter and the Goblet of Fire', 'Harry Potter
and the Chamber of Secrets: Sheet Music for Flute with C.D', 'Gregor and the Marks of Secret', 'Harry Potter and the Half-Blood Prince',
'Harry Potter and the Order of the Phoenix', 'Harry Potter and the Prisoner of Azkaban', "Harry Potter and the Philosopher's Stone", 'Har
ry Potter and the Deathly Hallows', "James Potter and the Hall of Elders' Crossing ", 'Harry Potter and the Cursed Child, Parts One and T
wo', 'Haroun and the Sea of Stories']

Recommendation system starting to make inference
......

Recommendations for Harry Potter and the Chamber of Secrets:
1: The Return of the King, with distance of 0.5137453857083071
2: Mockingjay, with distance of 0.484811069871498
3: The Da Vinci Code, with distance of 0.48437188831920774
4: Catching Fire, with distance of 0.466786667832629206
5: Harry Potter and the Philosopher's Stone, with distance of 0.4454417431428892
6: Harry Potter and the Deathly Hallows, with distance of 0.2774345523014743
7: Harry Potter and the Half-Blood Prince, with distance of 0.21458444953407796
8: Harry Potter and the Order of the Phoenix, with distance of 0.17345094201226208
9: Harry Potter and the Goblet of Fire, with distance of 0.1489778170737216
10: Harry Potter and the Prisoner of Azkaban, with distance of 0.1395682125920943
```

Out[56]:
```
['The Return of the King',
 'Mockingjay',
 'The Da Vinci Code',
 'Catching Fire',
 "Harry Potter and the Philosopher's Stone",
 'Harry Potter and the Deathly Hallows',
 'Harry Potter and the Half-Blood Prince',
 'Harry Potter and the Order of the Phoenix',
 'Harry Potter and the Goblet of Fire',
 'Harry Potter and the Prisoner of Azkaban']
```

Список литературы

[1] Netflix Vs Books-Recommender, Analysis, EDA // Kaggle. — 2022. — Access mode: https://www.kaggle.com/code/niharika41298/netflix-vs-books-recommender-analysis-eda/notebook#Recommendation-System (online; accessed:19.04.2022).