PythonProDataScience / 10 / lekce.ipynb

pesikj Rok 2024 2 months ago

4166 lines (4166 loc) · 168 KB

Preview Code Blame Raw

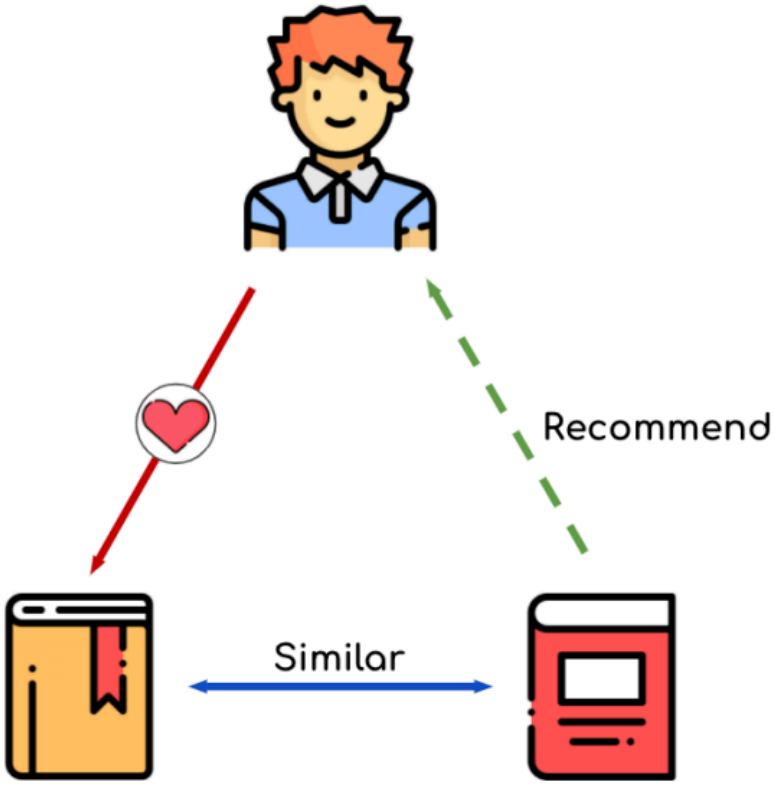
Lekce 10

Systémy na doporučování obsahu

Systémy na doporučování obsahu (*recommendation engine*) se snaží z dostupných možností vybrat tu nejzajímavější pro konkrétního uživatele. Systémy na doporučování obsahu jsou běžnou součástí řady služeb, například e-shopů, streamovacích služeb, sociálních sítí, pracovních portálů. Pomocí těchto systémů se poskytovatelé snaží udržet uživatele co nejdéle na jejich platformě (aby jim mohli zobrazit co nejvíce reklamy), nabídnout produkt, který si uživatel s největší pravděpodobností koupí atd.

Systémy na doporučování obsahu mohou fungovat na dvou principech.

Filtrování na základě obsahu (*content based filtering*) využívá data o jednotlivých položkách (např. filmech, knihách atd.). Jestliže se uživateli (uživatelce) líbí nějaký obsah, systém mu nabídne obsah, který je nejvíce podobný. Při vývoji tohoto systému tedy řešíme podobnost jednotlivých položek (např. na základě popisu).

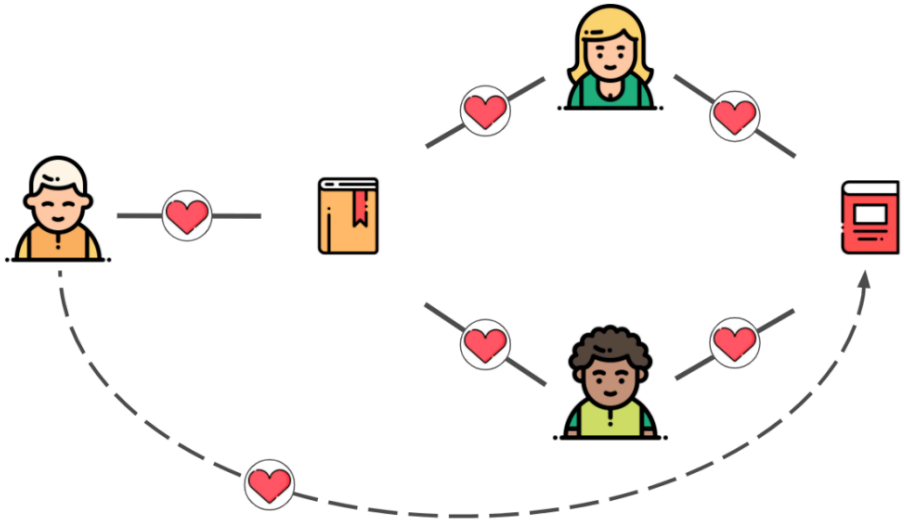


K vývoji nám stačí data o obsahu, tento přístup je tedy možné použít pro začínající službu. V některých případech může ale vést k "očividným" výsledkům, například pokud se někomu líbí první díl Star Wars, je mu nabídnutý druhý díl atd.

Kolaborativní filtrování (*collaborative filtering*) využívá data z chování uživatelů. Systém nějakému uživateli (uživatelce) vybírá takový obsah, který se líbil jemu podobným uživatelům (uživatelkám). Při vývoji tohoto systému tedy řešíme podobnost preferencí uživatelů (uživatelek) dané služby.

podobnost preferenci uživatelu (uživatelek) dané služby.

Pro vývoj je potřeba mít k dispozici dostatek dat o chování uživatelů. To může být problém například u nově vytvořených služeb, nových příspěvků na sociálních sítích, nových nebo nepřihlášených uživatelů/uživatelek atd. Pokud ale data máme k dispozici, může tento přístup generovat poměrně zajímavé výsledky.



Data mohou být založená na tzv. explicitním nebo implicitním feedbacku (*explicit feedback*, *implicit feedback*). Příkladem explicitního feedbacku je použití tlačítka "To se mi líbí", napsání pozitivní recenze atd. Příkladem implicitní vazby je dokoukání (nebo naopak nedokokání) filmu do konce, opětovné zhlédnutí atd.

Filtrování na základě obsahu

V této části si vyzkoušíme filtrování na základě popisu. Využijeme dataset s 5000 filmy ze serveru imdb.com. Pozor, jedná se o jiný dataset, než jaký jsme používali v minulé lekci. Dataset je uložený v souboru tmdb_5000_movies.csv. Budeme využívat sloupec `overview`, kde jsou uložené slovní popisy filmů. Filmy, které nemají slovní popis, z dat vyřadíme.

In [3]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

data = pd.read_csv("tmdb_5000_movies.csv")
data = data.dropna(subset=["overview"])
data.head()
```

Out[3]:

	title	budget	gen_Action	gen_Adventure	gen_Animation	gen_Comedy
0	Avatar	237000000	1.0	1.0	0.0	0.0
1	Pirates of the Caribbean:	300000000	1.0	1.0	0.0	0.0

	At World's End					
2	Spectre	245000000	1.0	1.0	0.0	0.0
3	The Dark Knight Rises	250000000	1.0	0.0	0.0	0.0
4	John Carter	260000000	1.0	1.0	0.0	0.0

5 rows × 40 columns

Podíváme se, kolik nám zbylo filmů.

In [4]:
```
data.shape
```

Out[4]:
```
(4800, 40)
```

Na sloupec `overview` opět použijeme algoritmus TF-IDF, který jsme již používali v minulé lekci. Tento algoritmus převede jednotlivé popisy (dokumenty) na číselnou matici. Každé slovo (nebo dvojice slov) budou mít svůj sloupeček a číslo v něm bude reflektovat přítomnost slova v dokumentu a přítomnost slova ve všech dokumentech. Nově přidáme dva parametry.

- `min_df` je minimální počet výskytů slova v dokumentech, aby nebylo vyřazeno. Potřebujeme slova, která se nachází alespoň ve dvou dokumentech, unikátní slovo nám nijak nepomůže propojit dokument s jiným dokumentem.
- `max_df` je maximální počet výskytů slova (zadáváme jako desetinné číslo, tj. 70 %). Slova, která jsou ve většině dokumentů, nám taky nepomůžou vybrat vhodný obsah.

Pro oba parametry platí následující pravidla:

- Pokud zadáme hodnotu v intervalu 0 až 1, je hodnota braná jako procentuální výskyt.
- Pokud zadáme hodnotu vyšší než 1, je hodnota braná jako výskyt v absolutní hodnotě.

Parametry `stop_words` a `ngram_range` jsme si již ukázali minule a jich význam je stále stejný. Dále použijeme metodu `fit_transform()` a na výsledná data se podíváme.

In [5]:
```
X = data["overview"]

vec = TfidfVectorizer(min_df=2, max_df=0.7, stop_words="english", ngram_ran
data_tfidf = vec.fit_transform(X)
```

```
data_tfidf = vec.fit_transform(X)
# Převod na pole
data_tfidf.toarray()
```

Out[5]:
```
array([[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.]])
```

Jednotlivá slova, kterým čísla patří, získáme pomocí metody `get_feature_names_out()` .

In [6]:
```
vec.get_feature_names_out()
```

Out[6]:
```
array(['00', '00 agent', '000', ..., 'zookeeper', 'zoologists', 'zorro'],
      dtype=object)
```

Abychom se na data mohli podívat v lepším formátu, můžeme si z nich opět sestavit pandas tabulku. Při vytváření tabulky řekneme, že popisky sloupců je možné získat pomocí metody `get_feature_names_out()` . Protože počet ani pořadí řádků se nijak nezměnily, načteme index tabulky ze sloupce `title` z původní tabulky s daty.

In [7]:
```
data_tfidf = pd.DataFrame(data_tfidf.toarray(), columns=vec.get_feature_nam
data_tfidf.head()
```

Out[7]:

title	00	00 agent	000	000 000	000 feet	000 foot	000 years	007	10	10 million	...	zomb outbrea
Avatar	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
Pirates of the Caribbean: At World's End	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
Spectre	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
The Dark Knight Rises	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
John Carter	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0

5 rows × 17468 columns

Nyní si vybereme jeden film, například film The Fugitive (Uprchlík) z roku 1993 s Harrísonem Fordem v hlavní roli. Seřadíme si hodnoty od nejvyšší po nejmenší.

Vidíme, že především slovo "Richard" může být celkem matoucí, protože jde o jméno hlavní postavy filmu, ale k ději nemá žádnou vazbu. Naopak slova jako "killer", "wrongfully" nebo "pursuing team" můžou být poměrně důležitá a můžou nám pomoci najít dějově podobný film. (Film Uprchlík je o chirurgovi Richardovi Kimblovi, který byl nespravedlivě odsouzen za vraždu své ženy. Při převozu do vězení se mu podaří uprchnout. Poté hledá skutečného vraha a je při tom pronásledován skupinou policitů.)

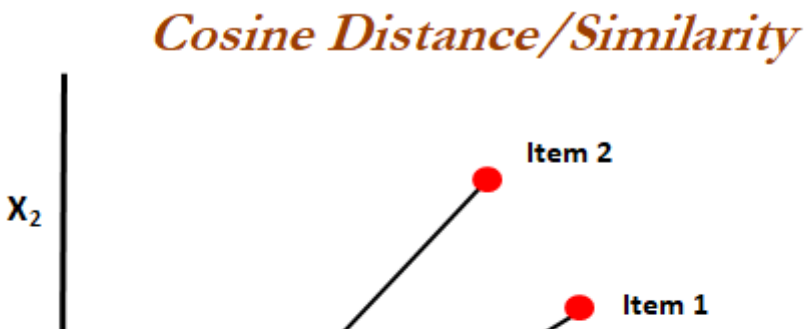
In [8]:
```
data_tfidf.loc["The Fugitive"].sort_values(ascending=False).head(20)
```

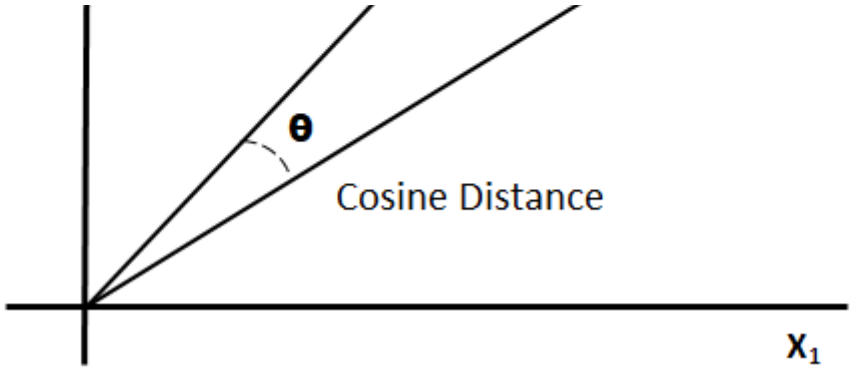
Out[8]:
```
richard              0.399296
killer               0.216403
discovers secrets    0.186469
wrongfully           0.186469
kimble               0.186469
death struggles      0.186469
pursuing team        0.186469
struggles expose     0.186469
team                 0.185769
gerard               0.180066
marshals             0.180066
wife                 0.177323
accused murdering    0.171041
samuel               0.167610
intricate            0.167610
chases               0.164638
deputy               0.162017
pursuing             0.159672
murdering            0.153832
expose               0.149211
Name: The Fugitive, dtype: float64
```

Abychom dokázali změřit podobnost mezi jednotlivými řádky tabulky `data_tfidf` , použijeme kosinovou podobnost.

Kosinová vzdálenost a podobnost

Kosinová vzdálenost (*cosine distance*) a kosinová podobnost (*cosine similarity*) jsou způsoby měření vzdálenosti, se kterým jsme se zatím nesetkali. Ukažme si, jak kosinová vzdálenost a podobnost fungují v dvourozměrném prostoru, tj. jak by změřila vzdálenost dvou hodnot z tabulky se dvěma sloupci. Body vidíme na obrázku níže. Kosinová podobnost je vypočítaná jako kosinus úhlu θ, tj. úhlu, který svírají vektory, které směrují do obou bodů. Pokud bychom měli dva naprosto totožné body, úhel mezi jejich vektory by měl velikost 0 stupňů. Kosinová vzdálenost těchto bodů by byla 0, kosinová podobnost by byla 1.





Kosinovou vzdálenost a podobnost je možné spočítat i ve vícerozměrném prostoru a i v něm platí, že čím jsou si vektory podobnější, tím větší je hodnota kosinové podobnosti a menší hodnota kosinové vzdálenosti. My budeme využívat kosinovou podobnost. Kosinová podobnost je v modulu `scikit-learn` implementovaná ve funkci `cosine_similarity()` .

Funkci můžeme dát dvě hodnoty (v našem případě dva filmy) a ona spočítá kosinovou podobnost, tj. podobnost jejich popisů. Pokud funkci dáváme dva filmy, můžeme je vybrat s využitím indexu (a vlastnosti `loc`). Funkce ale očekává dvourozměrné pole.

Jednorozměrné pole můžeme ze série získat s využitím vlastnosti `.values` .

In [9]:
```
data_tfidf.loc["The Fugitive"].values
```

Out[9]:
```
array([0., 0., 0., ..., 0., 0., 0.])
```

Abychom získali dvourozměrnou matici, můžeme použít metodu `reshape()` . Té dáme parametry `1` a `-1` , čímž získáme dvourozměrné pole.

Hodnota `1` znamená, že chceme mít pole, které má jeden řádek. A hodnota `-1` udělá pole dvourozměrné.

In [10]:
```
data_tfidf.loc["The Fugitive"].values.reshape(1, -1)
```

Out[10]:
```
array([[0., 0., 0., ..., 0., 0., 0.]])
```

Spočítejme nyní podobnost například mezi filmem The Fugitive a Witness

In [11]:
```
cosine_similarity(data_tfidf.loc["The Fugitive"].values.reshape(1, -1), dat
```

Out[11]:
```
array([[0.01564247]])
```

Podobnost je sice poměrně nízká (hodnota je blízká 0), absolutní hodnota nemusí být příliš důležitá. Zkusme ještě spočítat podobnost mezi filmy The Fugitive a Titanic.

In [12]:
```
cosine_similarity(data_tfidf.loc["The Fugitive"].values.reshape(1, -1), dat
```

Out[12]:
```
array([[0.00693092]])
```

Vidíme, že pro tyto fimy je podobnost větší. Náš systém by tedy nabídl uživateli, kterému se líbil film The Fugitive, film Witness a nikoli film Titanic.

V celé datové sadě se ale může nacházet ještě vhodnější film. Abychom získali obecná data pro doporučení libovolného filmu, použijeme funkci `cosine_similarity` na celou tabulku `cosine_similarity()`. Tento příkaz spočítá kosinovou podobnost pro každou dvojici filmů. Získáme tedy čtvercovou matici, kde se počet řádků a počet sloupců rovná počtu filmů v původní tabulce.

In [13]:
```
cosine_similarity_array = cosine_similarity(data_tfidf)
cosine_similarity_array
```

Out[13]:
```
array([[1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 1.        , 0.        , ..., 0.02190503, 0.        ,
        0.        ],
       [0.        , 0.        , 1.        , ..., 0.01450648, 0.        ,
        0.        ],
       ...,
       [0.        , 0.02190503, 0.01450648, ..., 1.        , 0.01549051,
        0.00614697],
       [0.        , 0.        , 0.        , ..., 0.01549051, 1.        ,
        0.01001404],
       [0.        , 0.        , 0.        , ..., 0.00614697, 0.01001404,
        1.        ]])
```

Pro snazší práci s daty opět provedeme převod na tabulku. Názvy sloupců i řádků (index) jsou dané jmény filmů.

In [14]:
```
cosine_similarity_df = pd.DataFrame(cosine_similarity_array, index=data_tfi
cosine_similarity_df.head()
```

Out[14]:

title	Avatar	Pirates of the Caribbean: At World's End	Spectre	The Dark Knight Rises	John Carter	Spider-Man 3	Tangled
title							
Avatar	1.000000	0.000000	0.0	0.020459	0.000000	0.024428	0.000000
Pirates of the Caribbean: At World's End	0.000000	1.000000	0.0	0.000000	0.034509	0.000000	0.000000
Spectre	0.000000	0.000000	1.0	0.000000	0.000000	0.000000	0.000000
The Dark Knight	0.020459	0.000000	0.0	1.000000	0.009002	0.003669	0.011148

Rises							
John Carter	0.000000	0.034509	0.0	0.009002	1.000000	0.000000	0.009827

5 rows × 4800 columns

Matice je symetrická. Pokud nás zajímají podobnosti jednotlivých filmů s filmem The Fugitive, můžeme vyjmut řádek (pomocí `.loc`) nebo sloupec (s využitím hranatých závorek). Jednodušší bude práce se sloupcem, protože budeme chtít data seřadit.

In [15]:
```
distances_the_fugitive = cosine_similarity_df["The Fugitive"]
distances_the_fugitive
```

Out[15]:
```
title
Avatar                                      0.000000
Pirates of the Caribbean: At World's End    0.000000
Spectre                                     0.000000
The Dark Knight Rises                       0.018202
John Carter                                 0.000000
                                              ...
El Mariachi                                 0.048717
Newlyweds                                   0.000000
Signed, Sealed, Delivered                   0.032423
Shanghai Calling                            0.000000
My Date with Drew                           0.000000
Name: The Fugitive, Length: 4800, dtype: float64
```

Zobrazíme si tedy 10 nejvíce podobných filmů. Nejvíce "podobný" je film "The Rocket: The Legend of Rocket Richard", který ale dějově nemá s filmem The Fugitive nic společného. Podobnost je daná čistě tím, že hlavní postavy obou filmů se jmenují Richard. To bychom mohli vyřešit například tím, že bychom zohlednili zánr filmu. Naopak následující dva filmy The Iceman a The Life of David Gale lze považovat za dobré tipy.

In [16]:
```
distances_the_fugitive.sort_values(ascending=False).head(10)
```

Out[16]:
```
title
The Fugitive                                1.000000
The Rocket: The Legend of Rocket Richard    0.207967
The Iceman                                  0.171206
The Life of David Gale                      0.165951
Married Life                                0.157624
The Work and The Story                      0.153640
A Time to Kill                              0.112665
Austin Powers in Goldmember                 0.109479
Somewhere in Time                           0.108948
High Anxiety                                0.100456
Name: The Fugitive, dtype: float64
```

Čtení na doma: Výběr na základě uživatelského profilu

Pokud uživatel používá nějakou službu delší dobu, budeme určitě vědět o více produktech (např. filmech), které se mu líbily. V takovém případě nebudeme hledat podobnost mezi dvěma produkty, ale mezi uživatelským profilem a produkty, které

budou pro uživatele nové (např. filmy, které ještě neviděl). Uvažujme například uživatele (uživatelku), které se líbily firmy Notting Hill, Titanic, The Great Gatsby a The Lovers. To, že se filmy líbily, můžeme poznat podle explicitního (uživatel použil tlačítko "To se mi líbí") nebo implicitního (uživatel zhlédl celý film) feedbacku.

Vytvoříme tedy proměnnou `data_user_profile`, která bude reprezentovat preference uživatele (uživatelky). Ty získáme tak, že spočítáme průměrné hodnoty jednotlivých slov (a dvojic slov) pro výše uvedené filmy. Abychom měli data ve dvourozměrné matici, oipět je třeba použít metodu `reshape()`.

In [17]:
```
favorite_movies = ["Notting Hill", "Titanic", "The Great Gatsby", "The Love
data_user_profile = data_tfidf.loc[favorite_movies]
data_user_profile = data_user_profile.mean()
data_user_profile = data_user_profile.values.reshape(1, -1)
data_user_profile
```

Out[17]:
```
array([[0., 0., 0., ..., 0., 0., 0.]])
```

Nyní zkusme najít filmy, které budou nejvíce podobného uživatelskému profilu. Nejprve z původní tabulky vyřadíme filmy ze seznamu `favorite_movies`. Poté spočítáme kosinové vzdálenosti mezi uživatelským profilem a jednotlivými filmy. Vznikne matice s jedním sloupcem, který pro každý film udává kosinovou vzdálenost mezi popisem filmu a uživatelským profilem.

In [18]:
```
data_tfidf_not_seen = data_tfidf.drop(favorite_movies, axis=0)
cosine_similarity_array_not_seen = cosine_similarity(data_tfidf_not_seen, c
cosine_similarity_array_not_seen
```

Out[18]:
```
array([[0.02648553],
       [0.02318485],
       [0.        ],
       ...,
       [0.01260339],
       [0.00843268],
       [0.02732532]])
```

Data opět převedeme na tabulku, abychom viděli názvy filmů, které náš systém danému uživateli doporučí. Jako indexy použijeme názvy filmů z tabulky `data_tfidf_not_seen`. Nakonec data seřadíme. Vidíme, že náš systém by doporučil filmy Romance & Cigarettes, Four Weddings and a Funeral a Rent.

In [19]:
```
user_recommendation = pd.DataFrame(cosine_similarity_array_not_seen, index=
user_recommendation.sort_values(by="score", ascending=False).head()
```

Out[19]:

	score
title	
Romance & Cigarettes	0.130770
Four Weddings and a Funeral	0.128872

Rent	0.121126
The Little Prince	0.112538
Dil Jo Bhi Kahey...	0.100109

Čtení na doma: Kolaborativní filtrování

U kolaborativního filtrování se řídíme preferencemi jednotlivých uživatelů a uživatelek. Využijeme data ze seouboru user_ratings.csv, který obsahuje uživatelská hodnocení fimů. U hodnocení víme ID uživatele (`userId`), bodové hodnocení filmu (sloupec `rating`) a název filmu (sloupec `title`).

In [20]:
```
data_ratings = pd.read_csv("user_ratings.csv")
data_ratings = data_ratings[["userId", "rating", "title"]].drop_duplicates(
data_ratings.head()
```

Out[20]:

	userId	rating	title
0	1	4.0	Toy Story (1995)
1	5	4.0	Toy Story (1995)
2	7	4.5	Toy Story (1995)
3	15	2.5	Toy Story (1995)
4	17	4.5	Toy Story (1995)

Pro další práci potřebujeme mít trochu jinou strukturu tabulky. Měli bychom mít názvy filmů jako sloupce, takže jeden řádek tabulky bude představovat všechna hodnocení jednoho uživatele (uživatelky). K tomu můžeme využít funkce `pivot()` . Pozor, nejedná se o funkci `pivot_table()` ! Rozdíl mezi `pivot()` a `pivot_table()` je v tom, že `pivot()` neprovádí žádnou agregaci, ale pouze "přeskládá" hodnoty z jednoho sloupce do více sloupců. Z toho důvodu zadáváme funkci `pivot` pouze tři parametry: `index` (názvy řádků - index), `columns` (názvy sloupců) a `values` (hodnoty). Nezadáváme agregační funkci.

In [21]:
```
data_ratings_pivot = data_ratings.pivot(index="userId", columns="title", va
data_ratings_pivot
```

Out[21]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	I S
userId									

1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
606	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
607	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
608	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
609	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
610	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

610 rows × 9719 columns

Podívejme se nyní na hodnocení uživatele (nebo uživatelky) s id 2. Jeho hodnocení zjistíme jako řádek s id 2. Pozor na to, že sloupce v původní tabulce jsou čísla, musíme tedy do `loc` vložit hodnotu 2 jako číslo, tj. bez uvozovek. Pokud bychom zadali číslo s uvozovkami, modul `pandas` by hledal řetězec 2 a nic by nenašel, protože řetězec 2 je jiná hodnota než číslo 2.

Vidíme, že máme celkem 29 hodnocení a jsou zastoupena jak vysoká, tak nízká hodnocení.

In [22]:
```
data_ratings_pivot.loc[2].dropna().sort_values()
```

Out[22]:
```
title
The Drop (2014)                                      2.0
Girl with the Dragon Tattoo, The (2011)              2.5
Zombieland (2009)                                    3.0
Shawshank Redemption, The (1994)                     3.0
Interstellar (2014)                                  3.0
Exit Through the Gift Shop (2010)                    3.0
Collateral (2004)                                    3.5
Django Unchained (2012)                              3.5
Dark Knight Rises, The (2012)                        3.5
Ex Machina (2015)                                    3.5
Whiplash (2014)                                      4.0
Tommy Boy (1995)                                     4.0
Talladega Nights: The Ballad of Ricky Bobby (2006)   4.0
Shutter Island (2010)                                4.0
Louis C.K.: Hilarious (2010)                         4.0
Kill Bill: Vol. 1 (2003)                             4.0
Departed, The (2006)                                 4.0
Inception (2010)                                     4.0
Gladiator (2000)                                     4.0
Inglourious Basterds (2009)                          4.5
Good Will Hunting (1997)                             4.5
Town, The (2010)                                     4.5
Dark Knight, The (2008)                              4.5
Mad Max: Fury Road (2015)                            5.0
```

```
Inside Job (2010)                                                5.0
Step Brothers (2008)                                             5.0
The Jinx: The Life and Deaths of Robert Durst (2015)             5.0
Warrior (2011)                                                   5.0
Wolf of Wall Street, The (2013)                                  5.0
Name: 2, dtype: float64
```

Velkým problémem dat tohoto typu je, že v nich je obrovské množství prázdných hodnot. Pro taková data se používá označení řídká (*sparse*). My si ukážeme velmi jednoduchý přístup k řešení tohoto problému. Nemůžeme nyní ale jen nahradit chybějící hodnoty 0, protože tím bychom zkresili průměrná hodnocení filmu. Využijeme následující postup.

1. Pro každý film spočteme průměrné hodnocení.
2. Od každého hodnocení odečteme průměrné hodnocení.
3. Nyní můžeme prázdné hodnoty nahradit 0, aniž bychom ovlivnili průměrné hodnocení filmů.

In [23]:
```
avg_ratings = data_ratings_pivot.mean()
avg_ratings
```

Out[23]:
```
title
'71 (2014)                                    4.000000
'Hellboy': The Seeds of Creation (2004)       4.000000
'Round Midnight (1986)                        3.500000
'Salem's Lot (2004)                           5.000000
'Til There Was You (1997)                     4.000000
                                                ...
eXistenZ (1999)                               3.863636
xXx (2002)                                    2.770833
xXx: State of the Union (2005)                2.000000
¡Three Amigos! (1986)                         3.134615
À nous la liberté (Freedom for Us) (1931)     1.000000
Length: 9719, dtype: float64
```

In [24]:
```
data_ratings_pivot_standardized = data_ratings_pivot.sub(avg_ratings, axis=
data_ratings_pivot_standardized = data_ratings_pivot_standardized.fillna(0)
data_ratings_pivot_standardized
```

Out[24]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	9
userId									
1	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	(
3	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	(

5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
...	...	...	...	...	...	...	...	...	
606	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
607	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
608	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
609	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
610	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-(

610 rows × 9719 columns

Dopočítání hodnot

Podívejme se třeba na hodnocení filmu Pulp Fiction od uživatele s id 1 a na něm si ukážeme, jak vznikly hodnoty v tabulce `data_ratings_pivot_standardized`. Film dostal od uživatele hodnocení 3 body. Nyní vidíme, že má hodnocení -1.197068. Kde se toto číslo vzalo? Rozeberme si postup výpočtu.

In [25]:
```
data_ratings_pivot.loc[1, "Pulp Fiction (1994)"]
```

Out[25]:
```
3.0
```

V tabulce `data_ratings_pivot_standardized` je hodnocení filmu Pulp Fiction -1.197068403908795.

In [26]:
```
data_ratings_pivot_standardized.loc[1, "Pulp Fiction (1994)"]
```

Out[26]:
```
-1.197068403908795
```

Podívejme se na průměrné hodnocení tohoto filmu v původních tabulce `avg_ratings`. To má hodnotu 4.197068403908795.

In [27]:
```
avg_ratings.loc["Pulp Fiction (1994)"]
```

Out[27]:
```
4.197068403908795
```

K číslu se dopočítáme, pokud vezmeme hodnocení 3 a odečteme od něj průměrné hodnocení filmu. Tím získáme hodnotu -1.197068403908795, což je stejná hodnota, jaká byla v

In [28]:
```
3 - 4.197068403908795
```

Out[28]:
```
-1.197068403908795
```

Podobnost uživatelů

Naším cílem nyní bude vyhledat k našemu uživateli č. 1 uživatel a uživatelky, kteří

jsou mu (jí) nejvíce podobní. K výpočtu podobnosti opět můžeme využít kosinovou vzdálenost. Jednou z jejích hodnot je, že jí nevadí obrovské množství chybějících hodnot, jimi ovlivněná není. Pro výpočet podobnosti uživatelů využijeme tabulku se standardizovanými hodnoceními, tj. tabulku `data_ratings_pivot_standardized`. Výsledná data opět převedeme na tabulku.

In [29]:
```
similarities = cosine_similarity(data_ratings_pivot_standardized)
similarities = pd.DataFrame(similarities, index=data_ratings_pivot_standard
similarities.head()
```

Out[29]:

userId	1	2	3	4	5	6	7
userId							
1	1.000000	0.005147	-0.045052	0.005082	0.018204	0.009896	-0.011553
2	0.005147	1.000000	0.000000	-0.018459	0.088285	-0.009396	0.008564
3	-0.045052	0.000000	1.000000	-0.013393	-0.034685	0.015053	0.000000
4	0.005082	-0.018459	-0.013393	1.000000	-0.059430	0.013483	0.070708
5	0.018204	0.088285	-0.034685	-0.059430	1.000000	-0.001382	0.003460

5 rows × 610 columns

Zobrazíme si, kteří uživatelé do 1 nejvíce podobných uživatelů patří. Nemáme žádný klíč k tomu, kolik uživatelů vybrat, zobrazíme tady 10 nejbližších. Do nich se počítá i sám uživatel 1, protože pro něj je kosinová podobnost 1 (tj. nejvyšší možná podobnost).

In [30]:
```
n_users = 10
user_1_similarities = similarities.loc[2]
user_1_similarities = user_1_similarities.sort_values(ascending=False)
user_1_similarities.head(n_users)
```

Out[30]:
```
userId
2      1.000000
189    0.174249
145    0.170194
524    0.120279
468    0.104534
564    0.103556
379    0.093442
5      0.088285
329    0.087958
378    0.086594
Name: 2, dtype: float64
```

Pomocí `iloc[]` nyní vyberu `n_users` nejvíce podobných uživatelů, samotného uživatele 1 už vyřadíme.

In [31]:
```
similar_users = user_1_similarities.iloc[1:n_users+1].index
```

```
similar_users
```

Out[31]:
```
Index([189, 145, 524, 468, 564, 379, 5, 329, 378, 272], dtype='int64', name
='userId')
```

V tabulce níže máme hodnocení filmů od 10 vybraných uživatelů. Aby bylo hodnocení relevantní, vybereme pouze filmy, které byly ohodnoceny alespoň od 3 těchto uživatelů. K tomu využijeme metodu `dropna()`, kterou již známe, ale tentokrát ji použijeme poměrně nezvykle.

- Jako hodnotu parametru `axis` použijeme `1`, tj. chceme, aby metoda odebírala sloupečky, nikoli řádky. My totiž z výběru chceme vyřadit filmy, které mají málo hodnocení, a filmy jsou v tabulce jako sloupce.
- Nechceme ale ponechat pouze filmy hodnocené od všech vybraných uživatelů, to by nám v tabulce moc filmů nezbylo. Proto použijeme parametr `tresh=3` (zkratka od anglického výrazu Treshold), který ponechá sloupečky, které mají alespoň tři hodnoty.

In [32]:
```
data_ratings_pivot_similar = data_ratings_pivot.loc[similar_users]
data_ratings_pivot_similar = data_ratings_pivot_similar.dropna(axis=1, thre
data_ratings_pivot_similar
```

Out[32]:

title	Ace Ventura: Pet Detective (1994)	Aladdin (1992)	Apollo 13 (1995)	Batman (1989)	Batman Forever (1995)	Beauty and the Beast (1991)	Braveheart (1995)	Clear and Present Danger (1994)
userId								
189	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145	2.0	4.0	3.0	3.0	3.0	4.0	NaN	NaN
524	3.0	4.0	5.0	3.0	3.0	NaN	3.0	5.0
468	3.0	3.0	5.0	3.0	3.0	5.0	4.0	2.0
564	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
379	2.0	NaN	5.0	3.0	3.0	NaN	5.0	4.0
5	3.0	4.0	3.0	3.0	3.0	5.0	4.0	3.0
329	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
378	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
272	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

10 rows × 36 columns

Nakonec spočítáme průměrná hodnocení pro jednotlivé filmy a seřadíme je. U těchto filmů tedy můžeme předpokládat větší šanci, že by se mohly líbit i uživateli 2.

In [33]:

In [33]:
```
average_ratins_by_similar_users = data_ratings_pivot_similar.mean().sort_va
average_ratins_by_similar_users.head(20)
```

Out[33]:
```
title
Dances with Wolves (1990)                                    4.750000
Beauty and the Beast (1991)                                  4.666667
Fugitive, The (1993)                                         4.666667
Seven (a.k.a. Se7en) (1995)                                  4.333333
Toy Story (1995)                                             4.300000
Schindler's List (1993)                                      4.250000
Apollo 13 (1995)                                             4.200000
Forrest Gump (1994)                                          4.000000
Braveheart (1995)                                            4.000000
Terminator 2: Judgment Day (1991)                            3.875000
Clueless (1995)                                              3.833333
Crimson Tide (1995)                                          3.750000
Aladdin (1992)                                               3.750000
Mask, The (1994)                                             3.666667
Outbreak (1995)                                              3.666667
Silence of the Lambs, The (1991)                             3.625000
Pulp Fiction (1994)                                          3.500000
Lion King, The (1994)                                        3.500000
Clear and Present Danger (1994)                              3.500000
Lord of the Rings: The Fellowship of the Ring, The (2001)    3.500000
dtype: float64
```

Čtení na doma: Odebrání filmů, které uživatel již viděl

Většině uživatelů asi nemá smysl doporučovat filmy, které už viděli, protože dají přednost něčemu novému. Můžeme tedy z našeho seznamu odebrat filmy, které už uživatel 2 ohodnotil. Nejprve si zjistíme, které to jsou. Pokud vybereme z tabulky hodnocení uživatele 2 a odebereme prázdné hodnoty, pak indexy vybrané série budou představovat filmy, které uživatel již viděl. Výsledek si uložíme do proměnné `seen_by_user_2` .

In [34]:
```
seen_by_user_2 = data_ratings_pivot.loc[2].dropna().index
seen_by_user_2
```

Out[34]:
```
Index(['Collateral (2004)', 'Dark Knight Rises, The (2012)',
       'Dark Knight, The (2008)', 'Departed, The (2006)',
       'Django Unchained (2012)', 'Ex Machina (2015)',
       'Exit Through the Gift Shop (2010)',
       'Girl with the Dragon Tattoo, The (2011)', 'Gladiator (2000)',
       'Good Will Hunting (1997)', 'Inception (2010)',
       'Inglourious Basterds (2009)', 'Inside Job (2010)',
       'Interstellar (2014)', 'Kill Bill: Vol. 1 (2003)',
       'Louis C.K.: Hilarious (2010)', 'Mad Max: Fury Road (2015)',
       'Shawshank Redemption, The (1994)', 'Shutter Island (2010)',
       'Step Brothers (2008)',
       'Talladega Nights: The Ballad of Ricky Bobby (2006)', 'The Drop (201
4)',
       'The Jinx: The Life and Deaths of Robert Durst (2015)',
       'Tommy Boy (1995)', 'Town, The (2010)', 'Warrior (2011)',
       'Whiplash (2014)', 'Wolf of Wall Street, The (2013)',
       'Zombieland (2009)'],
      dtype='object', name='title')
```

Nakonec tyto filmy odebereme z filmů, které chceme doporučit. Pomocí dotazu