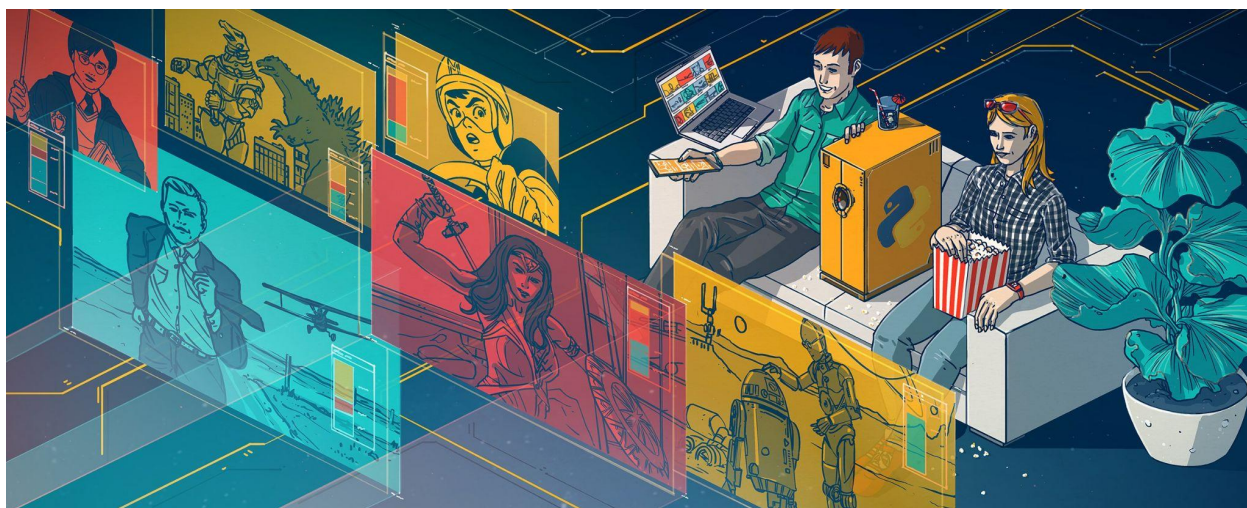


Sisteme de recomandare pentru filme



Bleoanca Diana Iulia

08.01.2022

IS anul 1

1. Problema studiata

În ziua de azi, unul din cele mai dezvoltate și căutate sisteme de recomandare este cel al filmelor, în care este necesar a fi luați câți mai mulți factori în considerare, cum ar fi: genul, directorul, descrierea, etc.

În prezenta lucrare ne propunem sa descriem diferite sisteme de recomandare a filmelor, construite pe baza unei baze de date de dimensiuni medii, dezvoltate astfel incat sa depindă de multipli factori, cu o comparatie amănunțită în diferite scenarii.

- **Descrierea algoritmilor**

Prima și cea mai simpla recomandare dezvoltată este reprezentată de un sistem bazat doar pe genuri ale filmelor. În acest sens, vom recomanda filme pe baza genului și a evaluării sau popularității filmului vizionat de majoritatea oamenilor.

In continuare, ca un al doilea caz, se dorește ca algoritmul să returneze primele 10 recomandări (filme) pe baza descrierii celui căutat, in care atat prezentarea generala cat si sloganul acestora sa balanseze rezultatele.

Avand in vedere ca nu întotdeauna exista o descriere atașată filmului, cat și faptul că acest lucru depinde foarte mult de modul prin care este prezentat un film (lucru care poate fi făcut în mod eronat sau nu foarte intuitiv / fără o legătură stransa cu mesajul filmului), în continuare am luat în considerare o adaptare al algoritmului propus pentru recomandarea în funcție de titlul unui film. Aceasta consta in analiza, interpretarea și compararea a unui set de date descriptive mai larg: cast, keywords, genuri, etc.

Mai mult decat atat, am mers mai departe, pentru a realiza un sistem de recomandare după cuvinte cheie. Astfel, utilizatorul va putea cauta dupa o secventa de cuvinte pe care-l interesează și să primească o lista de rezultate cu filme pe care l-ar putea interesa.

2. Metoda de rezolvare folosită

a. Limbaj de programare

Limbajul de programare pe care am ales sa îl folosesc este Python deoarece este unul dintre cele mai bine dezvoltate limbaje pentru machine learning, este foarte ușor de folosit și este mult mai apropiat de limbajele pe care deja le cunoaștem fata alte limbaje (precum R).

Unul dintre aspectele care fac din Python o alegere atât de populară în general, este abundența sa de biblioteci și framework-uri care facilitează codificarea și economisesc timp de dezvoltare. Învățarea automată și învățarea profundă sunt excepțional de bine abordate.

Cele mai importante pachete folosite pe care limbajul Python le pune la dispoziție și sunt în stransa legatura cu materia abordată:

- **Literal_eval**

literal_eval este folosit pentru a evalua literalele Python precum float, int, string etc. de exemplu: code = """(1, 2, {'foo': 'bar'})""" poate fi evaluat ca tuplu atunci când este trecut prin obiect literal_eval

- **Pandas**

Este o bibliotecă software scrisă pentru limbajul de programare Python pentru manipularea și analiza datelor. În special, oferă structuri de date și operațiuni pentru manipularea tabelor numerice și a seriilor de timp.

Avand in vedere ca seturile de date pe care urmează sa le procesez se afla în fișiere .csv, această librărie mi-a fost de mare ajutor

- **TfidfVectorizer si CountVectorizer**

Este un pachet care face conversi de colecții de documente brute într-o matrice de caracteristici TF-IDF sau de numărare.

TF-IDF este o abreviere pentru Term Frequency Inverse Document Frequency. Acesta este un algoritm foarte comun pentru a transforma textul într-o reprezentare semnificativă a numerelor, care este folosită pentru a se potrivi cu algoritmul mașinii pentru predicție.

Din necesitatea de a analiza descrierile filmelor și de alocare a unei ponderi per cuvânt, algoritmul TF-IDF a fost folosit pentru a transforma șirurile de caractere în șiruri de date care sa poată fi ulterior procesati pentru similitudini.

Pentru cazul datelor exacte (echipajul de producție, director, genuri), unde restul textului nu influențează ponderea cuvintelor, a fost suficient sa folosim un simplu counter.

- **Linear Kernel si Cosine Similarity de la Sklearn**

Funcția linear_kernel calculează nucleul polinom de gradul 1 între doi vectori. Nucleul polinom reprezintă asemănarea dintre doi vectori. Conceptual, nucleele polinomiale iau în considerare nu numai asemănarea dintre vectori sub aceeași dimensiune, ci și între dimensiuni. Când este utilizat în algoritmi de învățare automată, acest lucru permite să țină seama de interacțiunea cu caracteristicile.

Astfel, pentru a crea o matrice de similitudine, s-a realizat o matrice realizata din similitudinea intre setul de date si sine.

- **SnowballStemmer**

Libraria a fost folosită pentru a face stem la setul de keywords.

Stemming: este procesul de reducere a cuvântului la tulpina sa de cuvânt care se adaugă la sufixe și prefixe sau la rădăcinile cuvintelor cunoscute sub numele de leamă.

- Gensim
 - Models

Libraria de la Gensim ne pune la dispoziție algoritmul de LSI ce va fi folosit pentru recomandarea pe baza de cuvinte cheie. Prin acest algoritm se compune o matrice de similitudine pe baza descrierii filmelor.

- Similarities

Cu ajutorul acestui pachet putem sa calculăm lista de indecsi, că pentru fiecare căutare, extragerea filmelor dorite sa se proceseze în timp constant.

- corpora.Dictionary

Pentru modelul de LSI este necesar transformarea întregului set de cuvinte pe care îl avem într-un dicționar ce alcătuiește întregul nostru set de cunoștințe.

Mai mult decat atat, mi-a fost folositor limbajul și pentru a crea o minima interfața pentru prezentarea răspunsurilor procesate de algoritmi.

b. Set de date

Pentru analizarea și includerea unui set cât mai larg de filme (alături de datele acestora), s-a folosit o baza de date deja existentă, în care se afla detalii despre 45466 de filme.

Toate datele se afla în fișiere .csv, împărțite pe categorii de date ce urmează a fi consumate în funcție de etapele algoritmilor.

I. movies_metadata.csv

▲ belongs_to...	# budget	▲ genres	◀ homepage	🔗 id	▲ imdb_id	▲ original_la...
-----------------	----------	----------	------------	------	-----------	------------------

Din aceasta baza de date consumăm următoarele campuri: genres, vote_count, vote_average si release_date.

Cele dintai ne ajuta sa realizam recomandarea pe baza de gen bazat pe rating sau popularitate a celor mai privite filme. Din release date extragem ziua lansării pentru informație suplimentară.

Exemple de genuri: [Animation, Comedy, Family, Adventure, Fantasy, Romance];







Exemple de date de release: 2011-11-17, 2003-08-01, 1917-10-21, 2017-06-09;

Mai mult decat atat, pentru algoritmul ce are la baza descrierea și alte detalii despre filme, au fost necesare consumarea câmpurilor de tagline și overview.

Exemple de taglines: "A god incarnate. A city doomed.", "The band you know. The story you don't.", "Pokémon: Spell of the Unknown".







Exemplu de descriere: "When siblings Judy and Peter discover an enchanted board game that opens the door to a magical world, they unwittingly invite Alan -- an adult who's been trapped inside the game for 26 years -- into their living room. Alan's only hope for freedom is to finish the game, which proves risky as all three find themselves running from giant rhinoceroses, evil monkeys and other terrifying creatures."

II. links.csv

 movieid 	 imdbid 	 tmdbid 
---	--	--

Din aceasta tabela se consuma id-urile filmelor pentru a putea filtra informația pe care dorim sa o accesam.

III. credits.csv

 cast 	 crew 	 id 
--	--	--

Pentru cel de-al treilea caz, în care analizăm filmele pe baza de credite, ne intereseaza in mod explicit caracterele, actorii, directorul, s.a.m.d.

Exemplu de cast: [{**'cast_id': 14, 'character': 'Woody (voice)', 'credit_id': '52fe..', 'gender': 2, 'id': 31, 'name': 'Tom Hanks', 'order': 0, 'profile_path': '/pQFoyx7rp09CJTAb932F2g8NIho.jpg'**}, {**'cast_id': 15, 'character': 'Buzz Lightyear (voice)', 'credit_id': '52fe42..', 'gender': 2, 'id': 12898, 'name': 'Tim Allen', 'order': 1, 'profile_path': '/uX2xVf6pMmPepxnvFWyBtjexzgY.jpg'**}, {**'cast_id': 16, 'character': 'Mr. Potato Head (voice)', 'credit_id': '52fe42..', 'gender': 2, 'id': 7167, 'name': 'Don Rickles', 'order': 2, 'profile_path': '/h5BcaDMPRVLHLDzbQavec4xfSdt.jpg'**}, etc.]

Exemplu de crew: [{**'credit_id': '52fe44b...', 'department': 'Production', 'gender': 2, 'id': 511, 'job': 'Executive Producer', 'name': 'Larry J. Franco', 'profile_path': None**}, {**'credit_id': '52fe44b...', 'department': 'Writing', 'gender': 2, 'id': 876, 'job': 'Screenplay', 'name': 'Jonathan Hensleigh', 'profile_path': '/l1c4UFD3g0HVWj5f0CxXAvMAGiT.jpg'**}, {**'credit_id': '52fe44bfc3a36847f80a7cdd', 'department': 'Sound', 'gender': 2, 'id': 1729, 'job': 'Original Music Composer', 'name': 'James Horner', 'profile_path': '/oLOtXxXsYk8X4qq0ud4xVypXudi.jpg'**}, etc.]

IV. keywords.csv

 id 	 keywords 
--	--

Din aceasta tabela vom consuma cuvintele cheie pentru fiecare film, pentru a putea fi ulterior procesate spre a influența algoritmul de recomandare pe baza de detalii ale filmelor.

Exemple de cuvinte cheie:

['jealousy', 'toy', 'boy', 'friendship', 'friends', 'rivalry', 'boy next door', 'new toy', 'toy comes to life'],
['board game', 'disappearance', "based on children's book", 'new home', 'recluse', 'giant insect']

c. Preprocesarea datelor

Pentru ca datele pe care le avem sa fie relevante atât pentru masina, cat mai ales pentru algoritmi de învățare automată, acestea necesita a fi procesate. În afara eliminarii valorilor greșite sau incomplete, datele trebuiesc a fi pregatite într-un anumit format, compuse si alaturate.

Preprocesarea datelor este o etapă integrală în învățarea automată, deoarece calitatea datelor și informațiile utile care pot fi derivate din acestea afectează în mod direct capacitatea modelului nostru de a învăța; prin urmare, este extrem de important să ne preprocesăm datele înainte de a le introduce în modelul nostru.

Gestionarea valorilor nule

În orice set de date din lumea reală, există întotdeauna puține valori nule. Nu contează chiar dacă este o regresie, o clasificare sau orice alt tip de problemă, niciun model nu poate gestiona singur aceste valori NULL sau NaN, așa că trebuie să intervenim.

Exemple:

```
movie_dataset['genres'].fillna('[]')          - înlocuiește NaN cu [ ]
movie_dataset['vote_count'].notnull()         - considera doar valorile
care nu sunt nule
```

Standardizarea

Este un alt pas integral de preprocesare. În standardizare, ne transformăm valorile astfel încât media valorilor să fie 0 și abaterea standard să fie 1.

```
# Make directors into three stacked list in order to match the cast
members
smd['directors'] = smd['directors'].apply(lambda name: [name, name, name])
```

Pentru ca noi am ales sa pastram primi cei mai importanti actori, directorul va fi duplicat de același număr de ori pentru a nu influența ponderea actorilor fata de directorul filmului doar pentru ca este o lista mai vastă. Ne dorim ca ponderea acestuia descris anterior sa fie egala cu al altor personaje semnificative.

Evitarea multicoliniaritatii

Multicoliniaritatea apare în setul nostru de date atunci când avem caracteristici care sunt puternic dependente unele de altele. Ex - În acest caz avem caracteristici -

Dacă avem multicoliniaritate în setul nostru de date, atunci nu vom putea folosi vectorul nostru de greutate pentru a calcula importanța caracteristicii.

Folosim `drop_first=True` pentru a evita problema multicoliniarității.

```
movie_dataset_per_genre = movie_dataset.apply(lambda movie_data:
pd.Series(movie_data['genres']), axis = 1).stack().reset_index(level = 1,
drop = True)
```

Exemplu de genuri asignate filmelor:

["Comedy"]

["Drama", "Comedy", "Adventure"]

["Drama", "Family"]

Prin alocarea genurilor pe fiecare film pentru a putea fi preprocesate și după reacesare, dacă le-am crea câte o entitate pentru fiecare, fără a lua în considerare împărțirea, s-ar întâmpla următoarele:

45461	1	Family
45462	0	Drama
45463	0	Action
45463	1	Drama
45463	2	Thriller

Fiecărui gen i s-ar asigna un nou id, astfel am reuși să pierdem legătura între ele. După aplicare `drop`-ului, avem următorul rezultat:

45461	Family
45462	Drama
45463	Action
45463	Drama
45463	Thriller

d. Descrierea algoritmilor și valori intermediare

Recomandare în funcție de gen

În acest sens se recomandă filme pe baza generării și evaluării sau popularității filmului vizionat de majoritatea oamenilor.

În primul rând, scoatem valorile genurilor din dicționar, astfel încât numai numele din date să fie preluat și făcut în listă.

```
0    [Animation, Comedy, Family]
1    [Adventure, Fantasy, Family]
2          [Romance, Comedy]
3    [Comedy, Drama, Romance]
4          [Comedy]
```

Evaluarea TMBD este utilizată pentru evaluarea filmelor de top. Vom folosi evaluarea ponderată IMDB pentru a construi graficul.

Evaluare ponderată $(WR) = (vv+m.R)+(mv+m.C)$ unde,

- v este numărul de voturi pentru film m este numărul minim de voturi necesare pentru a fi enumerate în grafic
- R este evaluarea medie a filmului
- C este votul mediu în întregul raport

valoarea lui m este determinată luând procentul 80 ca limită. Adică luăm filme cu mai multe voturi de 80%.

```
min_votes = vote_count.quantile(0.80) # Minimum votes required, over 80%
print(min_votes)
50.0
```

Mai sus arătăm că pentru procentajul de 80%, numărul de voturi al filmelor trebuie să fie peste 50 și numai în acest caz, acele filme sunt luate în considerare.


```
movie_dataset['year'] = pd.to_datetime(movie_dataset['release_date'],
errors='coerce').apply(lambda date: str(date).split('-')[0]) # y-m-d format
```

Ne creăm propria coloana cu anul de lansare pe care o populăm doar cu informația aflată în locația în care este detectat anul.

```
qualified = movie_dataset[(movie_dataset['vote_count'] >= min_votes) &
(movie_dataset['vote_count'].notnull())][['title', 'year', 'vote_count',
'vote_average', 'popularity', 'genres']]
```

Creionăm noului set de date care conține numărul de voturi și apoi numărul de voturi procent de 80% ignorând valorile nule.

Putem vedea că numărul de voturi al filmului trebuie să fie peste 50, iar ratingul mediu va fi 5,244896612406511.

```
def weight_rating(data, min_votes, vote_average):
    m_votes_number = data['vote_count']
    m_average_rating = data['vote_average']
    return ( m_votes_number / ( m_votes_number + min_votes ) *
m_average_rating ) + ( min_votes / ( min_votes + m_votes_number ) *
vote_average )

qualified['weight_rating'] = qualified.apply(lambda movie_data:
weight_rating(movie_data, min_votes, vote_average), axis = 1)
```

	title	year	vote_count	vote_average	popularity \
0	Toy Story	1995	5415	7	21.946943
1	Jumanji	1995	2413	6	17.015539
2	Grumpier Old Men	1995	92	6	11.7129
4	Father of the Bride Part II	1995	173	5	8.387519
5	Heat	1995	1886	7	17.924927

	genres	weight_rating
0	[Animation, Comedy, Family]	6.983942
1	[Adventure, Fantasy, Family]	5.984671

2	[Romance, Comedy]	5.734119
4	[Comedy]	5.054910
5	[Action, Crime, Drama, Thriller]	6.954672

Mai sus am creat un sistem simplu de recomandare în care putem obține filmele recomandate pe baza evaluărilor și în funcție de gen. Dar cu acest sistem de recomandare nu putem recomanda un film pe baza alegerii personale.

De exemplu, dacă luăm în considerare “Inception” and “the Dark Knight”, amandoua sunt filme ale lui Christopher Nolan. Pe baza acestei informații, putem face o presupunere că utilizatorul care cauta după unul dintre aceste titluri prefera filmele lui Christopher Nolan. Dar el poate vedea doar filmele de acțiune dacă realizăm o simplă recomandare doar după gen, unde majoritatea au fost realizate de regizori diferiți și nu de Christopher Nolan.

Pentru a personaliza această recomandare avem nevoie de câteva caracteristici suplimentare pentru a forma o anumită măsurătoare pentru a recomanda filmul.

În continuare vom construi două sisteme bazate pe conținut.

Recomandare pentru un film pe baza descrierii

Recomandare bazată pe descrierea filmului are ca și suport atât tag line-urile, cât și overview-ul, pe care le alipim pentru a crea descrierea filmelor:

“A single and lonely woman finds the seemingly perfect man to date, but soon regrets it when his deranged and possessive other personality emerges and worst still, she cannot convince anyone else of his Jekyll/Hyde true nature.”

“Ray Liotta stars as a medical examiner who has been acquitted for his wife's murder but many still question his innocence. Obsessed with finding his wife's killer, a possible solution presents itself in an experimental serum designed by a neurobiology Linda Fiorentino which has the ability to transfer memories from one person to another, but not without consequences. Liotta driven to solve the case injects himself with the serum, bringing him closer and closer to finding her killer but bringing him closer to death. He loved her. He lost her. He won't let her memory die... until it tells him who killed her.”

Pentru că avem un set de date în care cuvintele sunt influențate de contextul lor, am ales să convertim colecția de documente în o matrice cu proprietăți TF-IDF.

TF-IDF înseamnă „Term Frequency — Inverse Document Frequency”. Aceasta este o tehnică de cuantificare a cuvintelor dintr-un set de documente. În general, calculăm un scor pentru fiecare cuvânt pentru a semnifica importanța acestuia în document și corpus. Această metodă este o tehnică utilizată pe scară largă în Information Retrieval și Text Mining.

Ne folosim de lista de cuvinte de stop pusa la dispozitie de catre biblioteca. Deoarece toate descrierile sunt în engleza, fără a exista cazuri de repetitii speciale pe care am vrea sa le inlaturam, lista implicita este suficienta.

Stopwords sunt cuvintele din fiecare limba care nu adaugă prea mult sens unei propoziții. Ele pot fi ignorate în siguranță, fără a sacrifica sensul propoziției. De exemplu, cuvinte precum the, he, have etc. Astfel de cuvinte sunt deja capturate în corpus.

```
tf = TfidfVectorizer(analyzer='word', ngram_range = (1,2), min_df = 0,
stop_words = 'english')
```

Aici analizatorul indică ceea ce trebuie analizat în document, cum ar fi cuvânt, număr, caractere. ngram_range este pentru a specifica gruparea în vectoriserul tf-idf - (1,2) indică gruparea literelor în forma unigramă și bigramă, unde, dacă luați în considerare un document „studiez NLP” pentru bigram, acesta va fi grupat ca „sunt”, „studiez” „studiez NLP”.

```
tfidf_mat = tf.fit_transform(smd['description'])
```

Codificam astfel descrierea în matricea TFid.

```
cos_sim = linear_kernel(tfidf_mat, tfidf_mat)
```

Vom folosi asemănarea Cosinusului pentru a găsi asemănarea dintre filme, astfel încât asemănarea cosinusului să fie înțeleasă prin reprezentarea grafică a datelor în graficul x y, unde x și y sunt valorile TF_IDF ale fiecărui document. Deoarece produsul punctual al asemănării cosinusului este deja găsit, vom folosi linear_kernal pentru a găsi asemănarea.

```
len(cos_sim[0])
45456
```

Din cele de mai sus putem observa că fiecare înregistrare a creat o listă de valori care conține asemănarea lor.

Pentru a putea rezolva pentru fiecare căutare indexul titlurilor și vice-versa, vom crea o mapare între cele 2:

```
indices = pd.Series(smd.index, index = smd['title']) # A map between
titles and movies index
```

Toy Story	0
Jumanji	1
Grumpier Old Men	2
Waiting to Exhale	3
Father of the Bride Part II	4

Astfel, dacă accesăm indexul, putem să cerem scorurile celorlalte filme preprocesate astfel încât să avem cele mai bune recomandări:

```
sim_scores = list(enumerate(cos_sim[idx]))
print(sim_scores[:5])
[(0, 0.0), (1, 0.007774131635450035), (2, 0.0), (3, 0.0), (4, 0.0)]
```

Pentru că pe noi ne interesează doar scorurile mari, următorul pas constă în sortarea listei în ordine descrescătoare și salvarea primelor n filme pe care ne interesează.

```
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True) # Sort
the movie based on the similarity score
sim_scores = sim_scores[1:31] #
Take first 30 movies (first one is itself)
```

Primul index este exclus deoarece reprezintă chiar indexul propriu, în care ponderea va avea scor maxim, și anume de 1.

Din cele de mai sus putem vedea un algoritm de recomandare, unde a găsit asemănarea bazată pe descriere și slogan. Această recomandare este bună mai ales atunci când vrem o continuare a filmului, dar să presupunem că dacă filmul nu mai are o continuare sau nu există o descriere potrivită, sau chiar lipsă. Atunci trebuie să recomandăm filme pe baza altor criterii, cum ar fi distribuția, echipajul, genurile și cuvintele cheie.

Recomandare pentru un film pe baza cuvintelor cheie și al plotului

Pentru această recomandare este necesar să consumăm date în plus din baza noastră de date, și anume detaliile despre echipaj și cuvinte cheie. Aceasta se va axa pe similitudinea dintre 2 filme în legătură cu actorii care fac parte din plot, de directori de filmari, de cuvinte cheie care ajută la înțelegerea contextului.

În primul rând, în echipajul filmului ni se furnizează un dicționar de persoane cu rolul lor și denumirea. Deoarece considerăm că directorul joacă unul dintre cele mai importante roluri într-un film, o să îl căutăm în mod special pentru a îi alocă un field separat.

```
def director_name(crew):
    for job in crew:
        if job['job'] == 'Director':
            return job['name']
    return np.nan
```

Din cast o sa pastram doar maximum primele 3 nume:

```
# Take the character names alone and make it as cast then take the first
three cast members
smd['cast'] = smd['cast'].apply(lambda cast_list: [cast['name'] for cast
in cast_list] if isinstance(cast_list, list) else []) # If list
smd['cast'] = smd['cast'].apply(lambda cast_list: cast_list[:3] if
len(cast_list) >= 3 else cast_list)
```

```
0          [Tom Hanks, Tim Allen, Don Rickles]
1      [Robin Williams, Jonathan Hyde, Kirsten Dunst]
2          [Walter Matthau, Jack Lemmon, Ann-Margret]
3      [Whitney Houston, Angela Bassett, Loretta Devine]
4          [Steve Martin, Diane Keaton, Martin Short]
```

Procesam și cuvintele cheie pentru a obține o lista insiruita de secvențe:

```
0      [jealousy, toy, boy, friendship, friends, rivalry, boy n...
1      [board game, disappearance, based on children's book, ne...
2          [fishing, best friend, duringcreditsstinger, old men]
3      [based on novel, interracial relationship, single mother...
4      [baby, midlife crisis, confidence, aging, daughter, moth...
```

Pentru ca nu dorim sa cantareasca diferit nume și prenume ale personajelor, pentru ca vrem o identificare unică a fiecărei persoane și vrem sa reducem din posibilitatea de nepotrivire din cauza micilor diferente de tipografie , procesam denumirile astfel incat sa fie alcătuite numai din litere mici și fără spații:

```
# Process director and cast names to lowercase and no space
smd['cast'] = smd['cast'].apply(lambda cast_names:
[str.lower(name.replace(" ", "")) for name in cast_names])
smd['directors'] = smd['directors'].astype('str').apply(lambda
director_name: str.lower(director_name.replace(" ", "")))
```

Pentru ca în momentul de fata avem o lista de personaje si doar un director, iar ponderea directorului trebuie sa fie la fel de semnificativă ca și cea a actorilor, o sa aplicăm o multiplicare a numelui acestuia intr-o stiva de aceeași dimensiune ca și cea a castului.

```
# Make directors into three stacked list in order to match the cast
members
smd['directors'] = smd['directors'].apply(lambda name: [name, name, name])
```

În continuare, ne ocupăm de cuvinte cheie.

```
# Process most important keywords to lowercase stem words with no
space
smd['keywords'] = smd['keywords'].apply(lambda keywords:
keep_most_important_keywords(keywords, keywords_movies))
smd['keywords'] = smd['keywords'].apply(lambda keywords:
[stemmer.stem(keyword) for keyword in keywords])
smd['keywords'] = smd['keywords'].apply(lambda keywords:
[str.lower(keyword.replace(" ", "")) for keyword in keywords])
```

Primul pas important este luarea celor mai importante corpus de cuvinte cheie separat și eliminarea înregistrărilor care au mai puține cuvinte cheie.

Acest lucru este datorat faptului ca nu dorim sa analizam cuvinte cheie care apar in mod unic sau care nu au o pondere așa semnificativă. Acest lucru influențează posibilitatea de a avea recomandări false pe baza cuvintelor cheie care nu se mai intalnesc și la alte filme.

Cel de-al doilea pas este extragerea tulpinii cuvintelor, un lucru des intalnit in NLP și care ajuta mult la reducerea plajei de posibile cuvinte intalnite și la formarea unori seturi de cuvinte corecte ("jucandu-se" si "joc" trebuie sa facă parte din aceeași plajă de cuvinte).

"Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language."

Cel din urma pas este reprezentat de procesarea secvențelor astfel incat cuvintele cheie sa fie alcătuite din litere mici și fără spatii între cuvinte.

```
0    [jealousi, toy, boy, friendship, friend, rivalri, boynex...
1    [boardgam, disappear, basedonchildren'sbook, newhom, rec...
2          [fish, bestfriend, duringcreditssting]
3    [basedonnovel, interracialrelationship, singlemoth, divo...
4    [babi, midlifecrisi, confid, age, daughter, motherdaught...
```

Toate informațiile procesate de până acum sunt alipite într-o listă de secvențe reprezentative pentru fiecare film.

```
smd['soup'] = smd['keywords'] + smd['cast'] + smd['directors'] +
smd['genres']
smd['soup'] = smd['soup'].apply(lambda x: " ".join(x))
```

Exemplu de cuvinte descriptive pentru unul dintre filme:

['jealousi', 'toy', 'boy', 'friendship', 'friend', 'rivalri', 'boynextdoor', 'newtoy', 'toycomestolif', 'tomhanks', 'timallen', 'donrickles', 'johnlasseter', 'johnlasseter', 'johnlasseter', 'Animation', 'Comedy', 'Family']

Pentru partea de analiza a cuvintelor, asemanator cu algoritmul descris anterior, vom folosi matricea de similitudine pentru a obține ponderi. Dar, de aceasta data, nu mai este necesara aplicarea algoritmului de TF-IDF deoarece cuvintele nu mai sunt influențate de context, astfel incat o simpla aplicare a unei numaratoare de apariții este suficientă.

```
count = CountVectorizer(analyzer = 'word', ngram_range = (1,2), min_df = 0, stop_words = 'english')
count_mat = count.fit_transform(smd['soup'])
cos_sim = cosine_similarity(count_mat, count_mat)
```

Acum, dacă aplicăm ca și în cazul precedent o simpla sortare, vom avea rezultate în care pot fi identificate elementele comune:

Recommendation for the 'The Dark Knight':

8031	The Dark Knight Rises
6218	Batman Begins
6623	The Prestige
2085	Following
7648	Inception
4145	Insomnia
3381	Memento
8613	Interstellar
7659	Batman: Under the Red Hood
1134	Batman Returns

din cele de mai sus putem observa că sistemul recunoaște Filmele Christopher Nolan în care putem recunoaște actori care se repeta.

Putem vedea că “Batman: Under the Red Hood” este similar cu “Dark Knight Rises” din cauza personajelor similare din ambele filme. Dar “Batman: Under the Red Hood” nu este un film atat de bine cotate, în comparație cu alte filme Batman. Pentru a rezolva ordinea în care ne sunt returnate filmele, putem sa refolosim logica aplicată la primul algoritm pentru a oferi o pondere cu ajutorul voturilor date de către alți utilizatori.

```
sim_score = list(enumerate(cos_sim[idx]))
sim_score = sorted(sim_score, key = lambda id_score: id_score[1], reverse = True)
sim_score = sim_score[1:31]
```

După ce enumerăm și salvăm cele mai similare n filme:

```
vote_count =  
movies_details[movies_details['vote_count'].notnull()][['vote_count']].astype('int')  
vote_averages =  
movies_details[movies_details['vote_average'].notnull()][['vote_average']].astype('int')  
vote_average = vote_averages.mean() # Votes average  
min_votes = vote_count.quantile(0.60) # Minimum votes required, over 60%
```

Din lista data aflăm media voturilor și găsim numărul de voturi minim astfel incat sa fie peste 60% din setul de voturi ca și număr.

```
quantifi['weight_rating'] = quantifi.apply(lambda movie_data:  
weight_rating(movie_data, min_votes, vote_average), axis = 1)
```

Și aplicăm un camp special în care afisam ponderea pe care o oferă algoritmul de aflare a greutatii medii pe baza de voturi.

Recomandare pe baza căutării după cuvinte cheie

După ce am realizat căutările în funcție de un gen sau un titlu, o sa mergem mai departe către o implementare de recomandare pe baza de cuvinte cheie. Astfel, utilizatorul nu va mai fi restrâns la ceva ce cunoaște deja, la un film precedent sau la un gen.

De aceasta data, ne vom folosi de toate datele pe care am reusit sa le procesam pana acum: atat descriere, cat și cuvinte cheie.

```
gathered_md = description_dataset.merge(keywords_dataset, on = 'id')
```

În momentul acesta avem alipită o descriere și o lista de cuvinte cheie. Dorim sa le transformăm pe toate într-o listă de cuvinte pentru a ne crea bagajul de cunoștințe și dicționarul.

```
def getWordList(x):  
    rough_wordList = re.sub("[^\w]", " ", x).split()  
    wordList = []  
    for word in rough_wordList:  
        if word not in stop_words:  
            wordList.append(word)  
  
    return wordList
```



```
gathered_md['dataset'] = gathered_md['description'].apply(lambda
description: getWordList(description))
gathered_md['dataset'] = gathered_md['dataset'] + gathered_md['keywords']
```

Pentru modelul nostru de antrenament, avem nevoie de bagaj de cunoștințe și de dicționar.

```
words_for_dictionary = gathered_md['dataset'].tolist()
dictionary = gensim.corpora.Dictionary(words_for_dictionary)
bow_corpus = [dictionary.doc2bow(doc) for doc in words_for_dictionary]
```

Acum avem întregul set de date transformat în structuri pe care modelul ce urmează să îl antrenăm reușește să le interpreteze astfel încât să transforme cuvintele în puncte de reper pentru dicționarul nostru.

Dictionary(41348 unique tokens: ['Afraid', 'Andy', 'But', 'Buzz', 'Led']...)

Bow_corpus: [[(0, 1), (1, 3), (2, 1), (3, 3), (4, 1), (5, 1), (6, 3), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), ..]

Bagajul de cunoștințe (bow_corpus) conține cuvântul id și frecvența acestuia în fiecare document. De aceea, un detaliu adăugat algoritmul este transformarea acestuia în spațiu TF-IDF.

```
# Transform bow_corpus in a tf-idf vector
tfidf = models.TfidfModel(bow_corpus)
corpus_tfidf = tfidf[bow_corpus]
```

Acum urmează antrenamentul propriu-zis și indexarea în forma matriceală a modelului antrenat:

```
lsi = models.LsiModel(corpus = corpus_tfidf, id2word = dictionary,
num_topics = 5)
# Compute a similarity matrix, which it's necessary later, for query
indexList = similarities.MatrixSimilarity(lsi[corpus_tfidf])
```

3. Rezultatele obținute și analiza

Recomandare în funcție de gen

Comedy:

	title	year	vote_count \
351	Forrest Gump	1994	8147
1225	Back to the Future	1985	6239
18465	The Intouchables	2011	5410
22841	The Grand Budapest Hotel	2014	4644
2211	Life Is Beautiful	1997	3643
732	Dr. Strangelove or: How I Learned to..	1964	1472
3342	Modern Times	1936	881
883	Some Like It Hot	1959	835
1236	The Great Dictator	1940	756
10309	Dilwale Dulhania Le Jayenge	1995	661

	vote_average	popularity	weight_rating
	9	34.457024	8.891981
351	8	48.307194	7.993442
1225	8	25.778509	7.991443
18465	8	16.086919	7.990136
22841	8	14.442048	7.988516
2211	8	39.39497	7.985378
732	8	9.80398	7.964101
3342	8	8.159556	7.940554
883	8	11.845107	7.937356
1236	8	9.241748	7.930978
10309			

Pentru a putea interpreta datele ne-am de recenzionarea realizata atat de către Google, cat si de IMDB, considerand media notelor oferite pe cele 2 platforme, cat și numărul de utilizatori care au participat la votare.

Titlu	Nota IMDB (x/10)	Nota Google (x/5)	Număr de voturi pe IMDB
Forrest Gump	8.8	4.8	1,944,380 votes
Back to the Future	8.5	4.9	1,134,050 votes
The Intouchables	8.5	4.8	811,954 votes

The Grand Budapest Hotel	8.1	4.7	761,772 votes
Life Is Beautiful	8.6	4.9	663,053 votes
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb	8.4	4.6	473,144 votes
Modern Times	8.5	4.7	232,350 votes
Some Like It Hot	8.2	4.7	257,792 votes
The Great Dictator	8.4	4.8	216,091 votes
Dilwale Dulhania Le Jayenge	8.1	4.7	67,212 votes

Action

15480	Inception	2010	14075
12481	The Dark Knight	2008	12269
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892
7000	The Lord of the Rings: The Return of the King	2003	8226
5814	The Lord of the Rings: The Two Towers	2002	7641
256	Star Wars	1977	6778
1154	The Empire Strikes Back	1980	5998
4135	Scarface	1983	3017
9430	Oldboy	2003	2000
1910	Seven Samurai	1954	892

	vote_average	popularity	weight_rating
15480	8	29.108149	7.993984
12481	8	123.167259	7.993101
4863	8	32.070725	7.990490
7000	8	29.324358	7.989723
5814	8	29.423537	7.988939
256	8	42.149697	7.987537
1154	8	19.470959	7.985924
4135	8	11.299673	7.972153
9430	8	10.616859	7.958202
1910	8	15.01777	7.907972

Titlu	Nota IMDB (x/10)	Nota Google (x/5)	Număr de voturi pe IMDB
Inception	8.8	4.8	2,215,072 votes
The Dark Knight	9	4.9	2,470,130 votes
The Lord of the Rings: The Fellowship of the Ring	8.8	4.8	1,760,653 votes
The Lord of the Rings: The Return of the King	8.9	4.9	1,739,698 votes
The Lord of the Rings: The Two Towers	8.7	4.8	1,572,159 votes
Star Wars: A New Hope (Episode IV)	8.6	4.6	1,296,104 votes
The Empire Strikes Back	8.7	4.8	1,224,476 votes
Scarface	8.3	4.8	792,254 votes
Oldboy	8.4	4.5	550,603 votes
Seven Samurai	8.6	4.9	333,431 votes

Concluzii:

Putem observa cu usurinta următoarele 2 aspecte esentiale:

- Toate notele oferite filmelor au scorurii acordate foarte bune, ceea ce indica faptul ca fac parte din categoria celor mai bune filme pentru aceste genuri
- Chiar dacă notele au o tenta descrescătoare, principalul factor de recomandare este numărul de voturi, observand ca exista doar o tenta descrescătoare.

Cele 2 recomandări pentru un titlu dat

Recommendation for “Toy Story”:

Recomandare pe baza de descriere		Recomandare pe baza de detalii	
10754	Luxo Jr.	3833	Monsters, Inc.
3024	Toy Story 2	7629	Toy Story 3
17551	Cars 2	2522	Toy Story 2
11074	Cars	8595	The Lego Movie
2262	A Bug's Life	6968	Horton Hears a Who!
22126	Toy Story of Terror!	3016	Chicken Run
15519	Toy Story 3	1832	Antz
3336	Creature Comforts	1662	One Hundred and One Dalmatians
4797	Monsters, Inc.	7404	Cloudy with a Chance of Meatballs
1738	Meet the Deedles	1883	A Bug's Life

Procent de repetare: 40%

Recommendation for “Harry Potter and the Prisoner of Azkaban”:

Recomandare pe baza de descriere		Recomandare pe baza de detalii	
5733	Harry Potter and the Chamber of Secrets	4265	Spirited Away
12036	Harry Potter and the Order of the Phoenix	3840	Harry Potter and the Philosopher's Stone
10649	Harry Potter and the Goblet of Fire	7921	Harry Potter and the Deathly Hallows: Part 2
4807	Harry Potter and the Philosopher's Stone	4366	Harry Potter and the Chamber of Secrets
1673	Great Expectations	8488	Gravity
14044	Harry Potter and the Half-Blood Prince	6354	Harry Potter and the Goblet of Fire
21791	Gravity	7742	Harry Potter and the Deathly Hallows: Part 1
5132	Y Tu Mamá También	6801	Harry Potter and the Order of the Phoenix
258	A Little Princess	7345	Harry Potter and the Half-Blood Prince
11462	Children of Men	521	Aladdin

Procent de repetare: 60%

Recommendation for “Interstellar”:

Recomandare pe baza de descriere	Recomandare pe baza de detalii
115651 Inception	115651 Inception
2486 Following	6981 The Dark Knight
11463 The Prestige	6623 The Prestige
12589 The Dark Knight	3381 Memento
4126 Memento	8031 The Dark Knight Rises
5302 Insomnia	6218 Batman Begins
18442 The Dark Knight Rises	8983 The Martian
5324 Silent Running	756 2001: A Space Odyssey
10210 Batman Begins	129 Apollo 13
30261 The Martian	8726 The Giver

Procent de repetare: 70%

Pentru validare, s-a realizat căutarea a unui număr de filme pe motorul de căutare Google și, realizând o comparație dintre lista răspunsurilor celor 2 algoritmi și lista acestora de recomandări, am reușit să extragem numărul de filme care coincid.

Titlu	Coincidenta algoritm 1	Coincidenta algoritm 2
Toy Story	7/10	7/10
Inception	6/10	6/10
The Polar Express	5/10	4/10
21 Jump Street	7/10	8/10
Harry Potter and the Prisoner of Azkaban	6/10	8/10
300	6/10	4/10
Matrix Revolutions	6/10	6/10
Interstellar	8/10	8/10
Rocky Balboa	7/10	7/10
Django Unchained	7/10	6/10
	8/10	

Concluzii:

Deși informația este diferită, procentul de repetare dintre răspunsuri este relativ mare (în medie de peste 50%), ceea ce întărește credibilitatea algoritmilor.

Mai mult decât atât, putem recunoaște un număr mare de recomandări care coincid cu cele ale platformei de căutare Google.

Recomandări pentru căutarea după cuvinte cheie

Răspunsuri pentru "Love":

2370	Rain	1932
5041	Last Tango in Paris	1972
563	The Superwife	1996
5872	L'eclisse	1962
1026	Annie Hall	1977
6443	Dinner with Friends	2001
8255	Take This Waltz	2011
1829	About Last Night...	1986
7648	Persuasion	2007
2230	Jules and Jim	1962

Răspunsuri pentru "Princess":

577	Barbarella	1968
9069	Insurgent	2015
6229	Ice Princess	2005
6786	Because I Said So	2007
74	The White Balloon	1995
730	Vive L'Amour	1994
8208	Gone	2012
8904	The Captive	2014
4013	Heidi	1937
7088	Nim's Island	2008

Răspunsuri pentru "Crime and mystery":

1195	Murder at 1600	1997
7146	Young People Fucking	2007
3246	She's Having a Baby	1988
2287	Tequila Sunrise	1988
3225	Men of Honor	2000
4952	Leap of Faith	1992
4127	Lucky Break	2001
5641	Two of a Kind	1983
1155	Star Trek II: The Wrath of Khan	1982
8724	RoboCop	2014

Validarea acestui algoritm este foarte dificilă atât timp cât nu avem un adevăr cu care putem să comparăm și depinde foarte mult de fiecare părere în parte. Astfel, singura validare pe care o putem în momentul face este cea manuală, în care, citind pentru fiecare descrierea, să încercăm să înclinăm o balanță spre 2 posibile cazuri: relevant sau nerelevant.

Concluzii generale

Recomandarea filmelor se poate realiza în foarte multe moduri. Există suficiente date pentru a reuși să analizezi fiecare film din mai multe perspective (după gen, după descriere, după actori, etc.). Cu toate acestea, cea mai importantă informație pe care nu o avem și ar fi putut deschide calea către multe îmbunătățiri este părerea utilizatorului. Acest lucru poate fi considerat o potențială abordare viitoare.

Dacă ar să vorbim despre o comparație între algoritmi, putem spune că primele recomandări descrise, atunci când utilizatorul are posibilitatea de a căuta după un gen sau după un titlu, sunt cei mai des întâlniți algoritmi pe orice platformă dedicată. Răspunsurile sunt destul de exacte, putem observa aceeași tendință pentru orice motor de căutare în care utilizatorul nu poate să-și lase amprenta (Google, IMDB, etc).

În schimb, algoritmul de căutare după cuvinte cheie nu este un subiect atât de abordat și poate fi considerat un prototip. Din cauza faptului că este foarte greu să aplici o validare pe acest algoritm, micile îmbunătățiri adăugate sunt la latitudinea noastră, după opinie personală, pentru a balansa răspunsurile mai mult spre o direcție sau alta.

În concluzie, există nenumărate feluri în care poți să consumi datele filmelor, să le procesezi, să creezi algoritmi de recomandare, însă comparația dintre ele sau acuratețea lor necesită implicarea utilizatorilor.

Bibliografie

[Recommendation Engines That Can Predict Your Movie Tastes](#)

[Building a movie recommender system with Python](#)

[Introduction to Data Preprocessing in Machine Learning](#)

[Build a Movie Recommendation System in Python using Machine Learning](#)

[Beginner Tutorial: Recommender Systems in Python](#)

[Latent Semantic Analysis using Python](#)

[Intuitive Understanding to Latent Semantic Indexing](#)

[Recommender System Based On Natural Language Processing](#)

[pandas.DataFrame](#)

[Gensim Tutorial – A Complete Beginners Guide](#)

[Text pre-processing: Stop words removal using different libraries](#)

[Stopwords: Important for the Language not so in NLP](#)

[Stemming and Lemmatization in Python](#)

[Cosine Similarity in Python](#)

Problema studiata	1
Descrierea algoritmilor	1
Metoda de rezolvare folosită	1
Limbaj de programare	1
Literal_eval	2
Pandas	2
TfidfVectorizer si CountVectorizer	2
Linear Kernel si Cosine Similarity de la Sklearn	2
SnowballStemmer	2
Gensim	3
Models	3
Similarities	3
corpora.Dictionary	3
Set de date	3
movies_metadata.csv	3
links.csv	4
credits.csv	4
Preprocesarea datelor	5
Gestionarea valorilor nule	5
Standardizarea	5
Descrierea algoritmilor și valori intermediare	7
Recomandare în funcție de gen	7
Recomandare pentru un film pe baza descrierii	9
Recomandare pentru un film pe baza cuvintelor cheie și al plotului	11
Recomandare pe baza căutării după cuvinte cheie	15
Rezultatele obținute și analiza	17
Recomandare în funcție de gen	17
Concluzii	19
Cele 2 recomandări pentru un titlu dat	20
Concluzii	22
Recomandări pentru căutarea după cuvinte cheie	22
Concluzii generale	23
Bibliografie	24

