Documentazione di progetto

Corso: Ingegneria della conoscenza

“Che film vedo stasera?”

Gruppo di lavoro

Bitetti Andrea 778423 a.bitetti16@studenti.uniba.it

Gatti Giovanni 778546 g.gatti15@studenti.uniba.it

URL REPOSITORY GITHUB

https://github.com/andreabitetti/che-film-vedo-stasera

AA 2024-2025

Sommario

Elenco argomenti di interesse

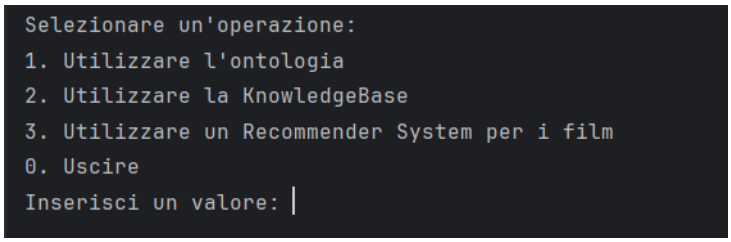
- Tipi di errore (cap 7.2.3)
- Rappresentazione e ragionamento proposizionale (cap. 4)
- Ontologie e condivisione della conoscenza (cap 6.3)
- Individui ed identificatori (cap 6.1.2)
- Classi e Proprietà (cap 6.2)

Introduzione

Lo streaming di film e serie TV è diventato un elemento essenziale della nostra quotidianità. Le piattaforme di streaming rappresentano oggi uno dei mezzi più diffusi per accedere a contenuti multimediali, offrendo una vastissima scelta di titoli.
Tuttavia, l'abbondanza di contenuti e la crescente competizione tra le piattaforme rendono necessario un approccio più strutturato per comprendere le preferenze degli utenti e migliorare la loro esperienza.
Quante volte ci siamo chiesti "Che film posso vedere stasera?" e soltanto dopo diverse ore siamo riusciti a trovare quello adatto alle esigenze della serata. Partendo da questa esigenza, abbiamo sviluppato un sistema che aiuti l'utente nella selezione dei contenuti da guardare, consigliando un film sulla base delle sue richieste, o nella scelta della piattaforma di streaming più adatta alle sue esigenze.
Il nostro progetto si articola in tre componenti principali:

- Un **sistema di raccomandazione**, progettato per suggerire film e serie TV sulla base delle preferenze personali degli utenti.
  Il sistema propone contenuti in linea con i gusti individuali, facilitando la scelta tra l'ampia offerta disponibile.
  A questo sistema si aggiunge un modello di classificazione dei rating, capace di prevedere la categoria delle valutazioni assegnate dagli utenti ai vari contenuti. Questo strumento aiuta a comprendere meglio i criteri di valutazione e consente di affinare ulteriormente le raccomandazioni.
- Una **knowledge base** interattiva, che fornisce agli utenti informazioni dettagliate su film e serie TV, permettendo una consultazione approfondita dei contenuti disponibili.
- Un'**ontologia** del dominio, che struttura e descrive il mondo dello streaming, chiarendo il significato di simboli e concetti utilizzati all'interno del sistema.

L'utente, una volta avviato il programma, potrà interagire in qualsiasi momento con queste funzionalità, passando facilmente tra il sistema di raccomandazione, la knowledge base e l'ontologia, in modo da trovare la soluzione più adatta alle proprie esigenze.



Visualizzazione pagina iniziale dell'interfaccia

Ambiente di sviluppo e librerie utilizzate

E' stato utilizzato Python come linguaggio di programmazione.
Per l'hosting, è stato selezionato GitHub, dove è ospitata la repository del progetto.
Le librerie e le versioni utilizzate sono specificate nel file *requirements.txt*.
In particolare:

- NumPy (numpy): gestione di array multidimensionali e operazioni matematiche
- Pandas: manipolazione e analisi dei dati strutturati (dataset in formato CSV)
- Matplotlib: creazione di grafici
- Seaborn: visualizzazione avanzata di dati statistici
- Plotly: creazione di grafici interattivi
- Ipywidgets: creazione di widget interattivi in Jupyter Notebook
- Owlready2: lavorare con ontologie nel web semantico
- PySWIP: interfaccia tra Python e Prolog
- Scikit-learn: machine Learning in Python

Dataset utilizzato

Un dataset è un insieme strutturato di dati organizzati in tabelle, utile per analisi e modellazione. Ogni riga rappresenta un'osservazione (un film), mentre ogni colonna contiene una specifica caratteristica (titolo, anno di uscita, genere, ecc.).
Il dataset utilizzato è stato scaricato dalla piattaforma Kaggle, una community online sotto Google LLC. Kaggle consente di trovare e pubblicare dataset, esplorare e creare modelli in un ambiente di data science basato sul Web.
Il dataset contiene 22 colonne, ognuna con informazioni sui film/serie TV. In particolare, le più rilevanti sono:

- id: identificativo unico del film/serie TV
- title: Titolo del film/serie TV
- type: indica se è un "MOVIE" o una "SERIE"
- description: trama del film/serie TV
- release_year: anno di uscita
- runtime: durata in minuti
- genres: generi del film
- production_countries: paesi di produzione
- seasons: numero di stagioni (se è una serie)
- imdb_votes: numero di voti su IMDb
- tmdb_popularity: indice di popolarità su TMDb
- tmdb_score: punteggio medio su TMDb
- streaming_service: servizio di streaming disponibile
- name: lista degli attori principali
- budget: budget del film
- primaryName: nome del regista o principale responsabile
- subscription_cost: costo dell'abbonamento al servizio di streaming

Analisi del Dataset e Pre-Processing

La prima fase di lavoro riguarda lo studio del dataset per comprenderne la struttura e le caratteristiche principali.
Il nostro obiettivo è stato quello di individuare eventuali problemi, come celle vuote o dati inconsistenti, e risolverli, oltre ad analizzare la correlazione tra la colonna target e le altre variabili presenti.

Per far ciò è stato creato un file **Jupyter Notebook** (.ipynb), scritto in **Python**.
Jupyter Notebook è un sistema interattivo utilizzato per l'analisi dati.
Questa fase è stata eseguita una volta sola, al fine di migliorare il dataset ed utilizzarlo per le operazioni successive. Quindi ad ogni successiva esecuzione del programma non viene effettuato nuovamente analisi e pre-processing del dataset.

In particolare sono state effettuate le seguenti operazioni:

- Analisi della struttura dei dati
- Identificazione di problemi come dati mancanti o inconsistenti
- Pulizia e trasformazione dei dati per analisi successive
- Studio della correlazione tra variabili

Per iniziare, abbiamo importiamo le librerie necessarie e caricato il dataset.
In questa fase eseguiamo una prima lettura dei dati per osservare la loro struttura e tipologia.

```
movies = pd.read_csv('../../originale/dataset.csv')
movies.head()
[37]
```

5 rows × 22 columns — Static Output

id	title	type	description	release_year	runtime	genres
tm100001	The Lucky Texan	MOVIE	Jerry Mason, a young Texan, and Jake Benson, a…	1934	61	['western', 'ac
tm1000022	Boonie Bears: The Wild Life	MOVIE	Bear brothers Briar and Bramble set off on an …	2021	99	['scifi', 'anim
tm1000169	Bad Cupid	MOVIE	Archie is a God on a mission to ensure that tr…	2021	81	['romance', 'cc
tm1000186	Carol's Christmas	MOVIE	Scrooge encounters the ghost of her late busin…	2021	70	['drama', 'thri
tm1000203	Digging to Death	MOVIE	David Van Owen moves into a mysterious house a…	2021	96	['horror', 'thr

```
movies.describe()
[38]
```

8 rows × 9 columns — Static Output

	Unnamed: 0	release_year	runtime	seasons	imdb_score	imdb_votes	tmdb_popularity	tmdb_score	budget
count	18719.00000	18719.000000	18719.000000	18719.0	18719.000000	1.871900e+04	18719.000000	18719.000000	6.335000e+0
mean	9359.00000	2001.818847	95.312944	0.0	6.011720	2.674374e+04	13.118934	6.083898	1.416556e+0
std	5403.85418	25.499656	26.308532	0.0	1.255525	1.108678e+05	63.821552	1.240955	3.478821e+0
min	0.00000	1912.000000	0.000000	0.0	0.500000	5.000000e+00	0.000153	0.500000	0.000000e+0
25%	4679.50000	1996.000000	82.000000	0.0	5.300000	2.180000e+02	1.803500	5.500000	0.000000e+0
50%	9359.00000	2014.000000	93.000000	0.0	6.100000	1.027000e+03	3.790000	6.100000	0.000000e+0

Individuati i problemi principali, si procede alla fase di pulizia e trasformazione del dataset.
Il fine ultimo è di rendere il sistema più veloce ed efficiente, evitare l'utilizzo di dati ridondanti o superflui.
Inoltre, per scelta progettuale, si è deciso di utilizzare solo film e quindi di eliminare eventuali tuple riguardanti le serie tv.

Abbiamo rimosso le colonne che crediamo non diano valore al processo. In particolare:

- 'type': il dataset è composto esclusivamente da film, quindi la distinzione non è necessaria
- 'seasons': non sono presenti serie TV, quindi stagioni
- 'imdb_id': identificativo non utile al nostro scopo
- 'genres': è necessario il solo genere principale
- 'budget': questo campo non è utile alla nostra ricerca e presenta un grande numero di valori nulli
- 'nconst': identificativo non utile al nostro scopo
- 'primaryName': non necessario per la ricerca

```
columns_to_drop = ['type', 'seasons', 'production_countries', 'imdb_id', 'imdb_votes', 'tmdb_popularity', 'genres', 'budget', 'nconst',
 'primaryName']
movies = movies.drop(columns=columns_to_drop, errors='ignore')
movies.head()
[40]
```

5 rows × 12 columns

	Unnamed: 0	id	title	description	release_year	runtime	imd
0	0	tm100001	The Lucky Texan	Jerry Mason, a young Texan, and Jake Benson, a…	1934	61	
1	1	tm1000022	Boonie Bears: The Wild Life	Bear brothers Briar and Bramble set off on an …	2021	99	
2	2	tm1000169	Bad Cupid	Archie is a God on a mission to ensure that tr…	2021	81	
3	3	tm1000186	Carol's Christmas	Scrooge encounters the ghost of her late busin…	2021	70	
4	4	tm1000203	Digging to Death	David Van Owen moves into a mysterious house a…	2021	96	

Eseguiamo alcune operazioni di pulizia come la rimozione di prefissi e simboli non necessari nelle colonne id, description e name

```
df = pd.DataFrame(movies)

df['id'] = df['id'].str.replace('^tm', '', regex=True)
df['description'] = df['description'].str.replace('"', '', regex=False)
df['name'] = df['name'].str.replace("'", "", regex=False)

df.to_csv('standard_dataset.csv', index=False)
```

Arrotondiamo le valutazioni 'imdb_score' e 'tmdb_score' alla prima cifra decimale per maggiore uniformità dei dati

```
df = pd.read_csv('standard_dataset.csv')
df['imdb_score'] = df['imdb_score'].round(1)
df['tmdb_score'] = df['tmdb_score'].round(1)

print(df[['imdb_score', 'tmdb_score']].to_string(index=False))

df.to_csv('renamed_dataset.csv', index=False)
[43]
```

```
 5.3         6.3
 5.3         4.7
 6.7         6.5
 7.8         6.0
 5.8         5.9
 6.9         6.9
 7.6         7.8
 6.1         5.4
 2.3         6.0
 6.4         8.1
 5.6         6.0
 6.1         6.3
```

Per rendere i nomi più chiari, rinominiamo le colonne 'main_genre', 'name' e 'subscription_cost' rispettivamente con 'genre', 'actors' e 'monthly_subscription_cost'.

```
df = pd.read_csv('renamed_dataset.csv')
df.rename(columns={'subscription_cost':'monthly_subscription_cost'}, inplace=True)
df['monthly_subscription_cost'] = df['monthly_subscription_cost'].replace(to_replace=r'\D', value='', regex=True).astype(float)
df['monthly_subscription_cost'] = df['monthly_subscription_cost'] / 100
df.to_csv('pre_processed_dataset.csv', index=False)

print(df['monthly_subscription_cost'])
[44]
0        14.99
1        14.99
2        14.99
3        14.99
4        14.99
         ...
18714     4.99
18715     4.99
18716     4.99
18717     4.99
18718     4.99
Name: monthly_subscription_cost, Length: 18719, dtype: float64
```

Dall'analisi risulta non ci siano valori nulli da gestire, quindi possiamo procedere con l'analisi dei dati.

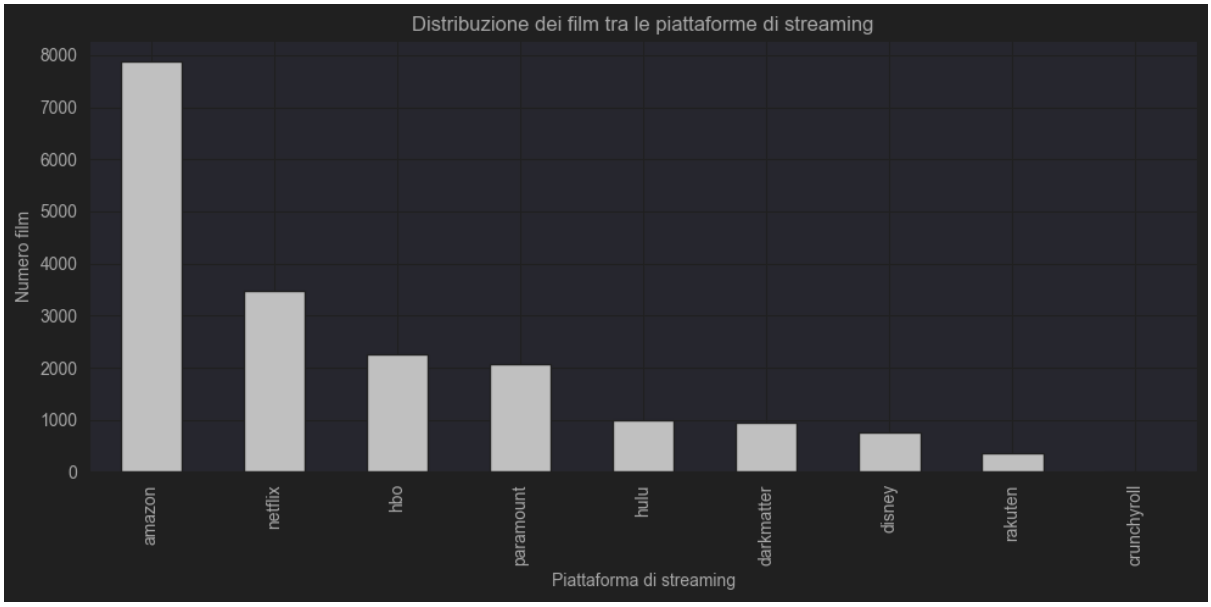
```
movies.isnull().sum()
[45]
id                  0
title               0
description         0
release_year        0
runtime             0
imdb_score          0
tmdb_score          0
streaming_service   0
main_genre          0
name                0
subscription_cost   0
dtype: int64
```

Studio dei dati e rappresentazioni con grafici

Per ottenere varie informazioni significative, utilizziamo diversi grafici esplorativi.

Istogramma che rappresenta la distribuzione dei film tra le piattaforme di streaming:

```
df = pd.DataFrame(movies)
movie_distribution = df['streaming_service'].value_counts()
plt.figure(figsize=(10, 5))
movie_distribution.plot(kind='bar', color='black')
plt.title('Distribuzione dei film tra le piattaforme di streaming')
plt.xlabel('Piattaforma di streaming')
plt.ylabel('Numero film')
plt.grid(True)
plt.tight_layout()
```



La maggior parte dei dati rilevati dall'istogramma sono legati alle piattaforme 'Amazon' e 'Netflix'. Questo ci fa credere che le raccomandazioni saranno maggiormente indirizzate su queste piattaforme, rispetto ad altre che hanno minore rilevanza.

Istogramma che rappresenta la distribuzione dei generi dei film per ciascuna piattaforma di streaming:

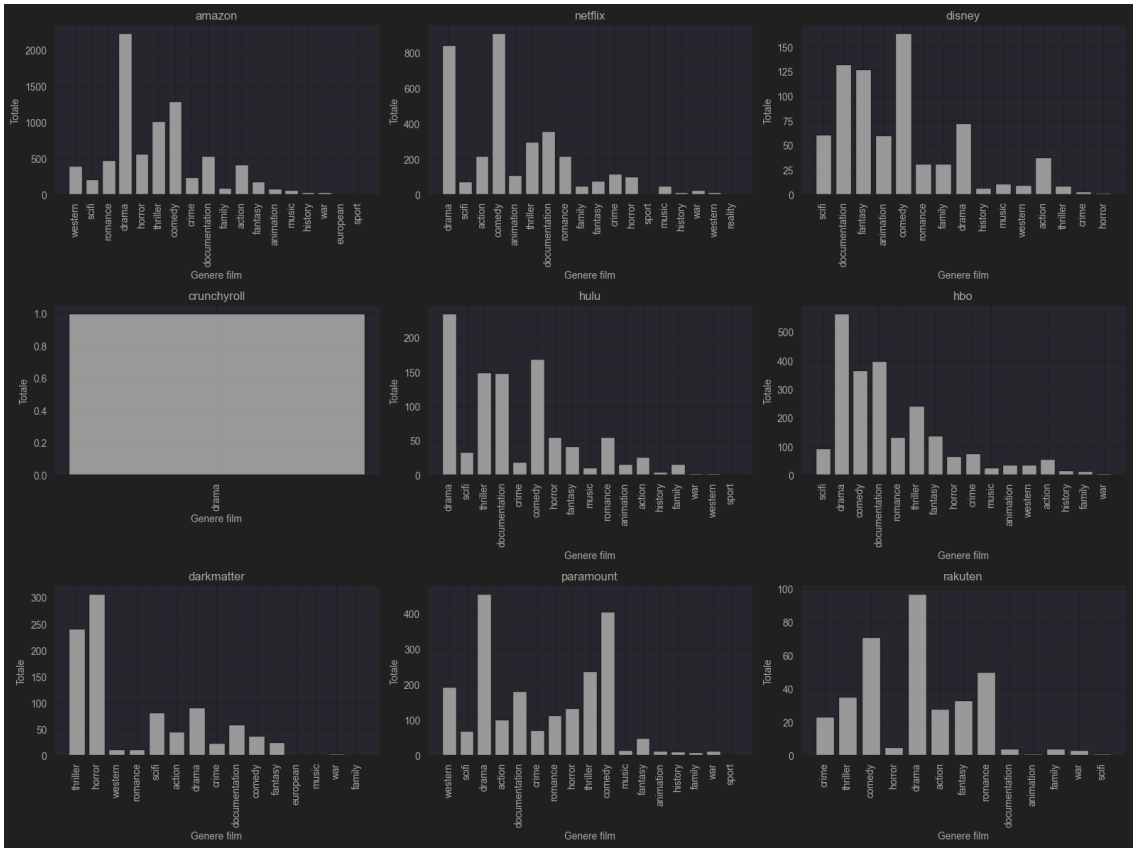
```python
import math

df = pd.DataFrame(movies)
streaming_services = df['streaming_service'].unique()
df.replace([float('inf'), -float('inf')], pd.NA, inplace=True)
num_services = len(streaming_services)
num_cols = 3
num_rows = math.ceil(num_services / num_cols)

plt.figure(figsize=(16, num_rows * 4))
for i, service in enumerate(streaming_services, 1):
    plt.subplot(num_rows, num_cols, i)
    subset = df[df['streaming_service'] == service]
    sns.histplot(data=subset, x='main_genre', discrete=True, shrink=0.8, color='black')

    plt.title(f'{service}')
    plt.xlabel('Genere film')
    plt.ylabel('Totale')
    plt.xticks(rotation=90)
    plt.grid(True)

plt.tight_layout()
plt.show()
```

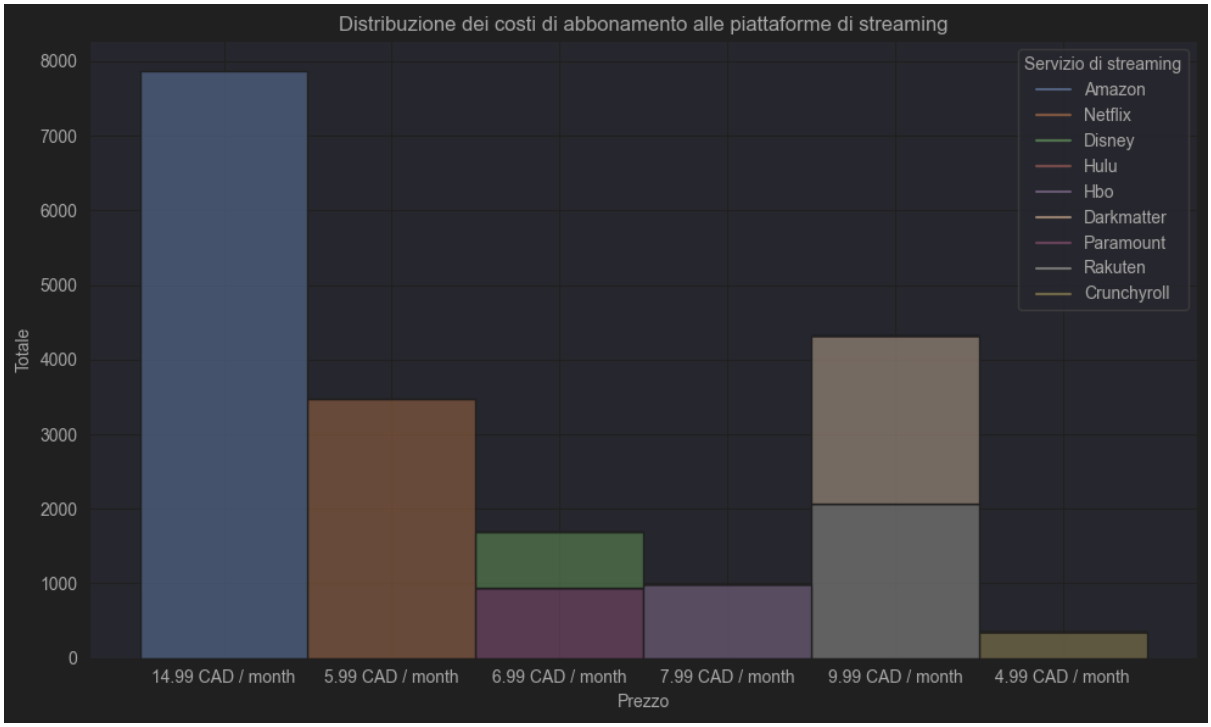


Istogramma che rappresenta la distribuzione dei prezzi tra le varie piattaforme di streaming:

```
df = pd.DataFrame(movies)
streaming_services=['Amazon', 'Netflix', 'Disney', 'Hulu', 'Hbo', 'Darkmatter', 'Paramount', 'Rakuten', 'Crunchyroll']

plt.figure(figsize=(10, 6))
hist = sns.histplot(data=df, x='subscription_cost', hue='streaming_service', multiple='stack', palette='muted')
plt.title('Distribuzione dei costi di abbonamento alle piattaforme di streaming')
plt.xlabel('Prezzo')
plt.ylabel('Totale')
plt.grid(True)

handles, labels = hist.get_legend_handles_labels()
if not handles:
    palette = sns.color_palette('muted', len(streaming_services))
    color_dict = dict(zip(streaming_services, palette))
    for service in streaming_services:
        handles.append(plt.Line2D([], [], color=color_dict[service], label=service))
plt.legend(handles=handles, title='Servizio di streaming')
plt.tight_layout()
plt.show()
[48]
```



Dal grafico possiamo studiare la distribuzione dei costi di abbonamento alle piattaforme di streaming presenti nel dataset. Si può notare come Amazon, nonostante sia il più costoso, rappresenta la maggioranza degli abbonamenti sottoscritti.

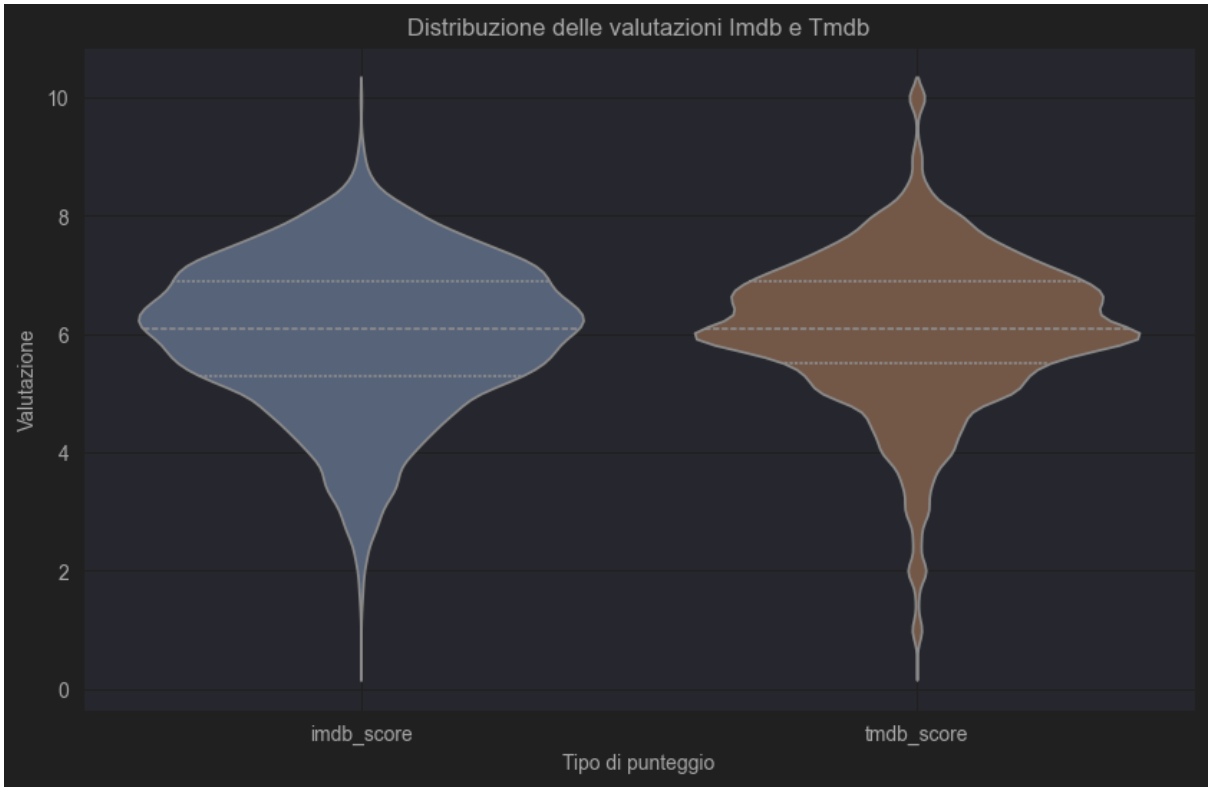
Grafico a violino che rappresenta la distribuzione delle valutazioni 'imdb_score' e 'tmdb_score:

```python
df = pd.DataFrame(movies)

df_long = pd.melt(df, id_vars=['title'], value_vars=['imdb_score', 'tmdb_score'], var_name='score_type', value_name='score')
plt.figure(figsize=(10, 6))
sns.violinplot(x='score_type', y='score', data=df_long, inner='quartile', hue='score_type', palette='muted', legend=False)

plt.title('Distribuzione delle valutazioni Imdb e Tmdb')
plt.xlabel('Tipo di punteggio')
plt.ylabel('Valutazione')
plt.grid(True)

plt.show()
```



Dal grafico emerge la distribuzione delle votazioni per ogni livello di valutazione.
Ogni grafico a forma "violino" illustra la variazione delle valutazioni assegnate ai singoli titoli sulle due piattaforme (IMDB e TMDB).
Dall'analisi di questi grafici, si osserva che la maggior parte delle valutazioni tende a concentrarsi attorno al valore di 6.

Recommender System

Introduzione

Un sistema di raccomandazione o motore di raccomandazione è un software di filtraggio dei contenuti che crea delle raccomandazioni personalizzate specifiche per l'utente così da aiutarlo nelle sue scelte.

I sistemi di raccomandazione sono spesso usati da siti e-commerce per fornire all'utente suggerimenti su prodotti che potrebbero interessargli e che non ha ancora visto. Ma possono essere applicati a diversi campi, come libri, musica, film, video, notizie e social media.
Nel nostro caso è stato utilizzato per effettuare classificazioni e raccomandazioni per film.

Esistono tre tipi di approccio utilizzato per creare raccomandazioni:

- Approccio collaborativo
  L'approccio collaborativo crea dei suggerimenti utilizzando la similarità tra gli utenti. Il presupposto è che utenti simili abbiano probabilmente gusti simili. L'approccio collaborativo si basa sul tipo di elementi visualizzati dall'utente in precedenza e su eventuali voti che l'utente ha attribuito all'elemento. Partendo dal principio che utenti simili potrebbero attribuire allo stesso elemento una valutazione simile, l'algoritmo fornisce all'utente gli elementi votati da altri utenti con le valutazioni più alte.
- Approccio basato sul contenuto
  L'approccio basato sui contenuti incrocia il contenuto di un elemento e il profilo di un utente.
  Il contenuto di un elemento è costituito dalla sua descrizione, attributi, parole chiave e etichette. Questi dati vengono messi a confronto con il profilo utente che racchiude le preferenze dell'utente, costruite analizzando gli elementi visualizzati da un utente durante la navigazione.
  L'approccio basato sul contenuto utilizza sistemi per creare delle stime probabili e affidabili. Viene spesso usato il classificatore Bayesiano.
- Approccio ibrido
  L'approccio ibrido è ottenuto dalla combinazione tra l'approccio collaborativo e l'approccio basato sul contenuto. In questo caso, non si può constatare come il sistema crea delle raccomandazioni perché dipende dalla combinazione degli altri due approcci. I metodi utilizzabili sono i seguenti:
  Netflix utilizza l'approccio ibrido per raccomandare prodotti all'utente. Per creare delle raccomandazioni, vengono utilizzate le valutazioni degli utenti simili, sfruttando l'approccio collaborativo, e il contenuto dell'elemento incrociato al profilo dell'utente, l'approccio basato sul contenuto.

Decisioni di progetto

Abbiamo scelto di adottare l'approccio basato sul contenuto in quanto il dataset permette di accedere ad una vasta gamma di informazioni dettagliate sui film, organizzate in categorie. Il sistema analizza le caratteristiche dei film (genere, attori, regista, trama, ecc.) e suggerisce film simili a quelli preferiti dall'utente.
Nel nostro caso si è deciso non incrociare i dati con il profilo dell'utente ma di permettere all'utente stesso di definire quali sono i suoi interessi e quindi ricevere una raccomandazione in relazione a quelle informazioni.

In particolare, una volta avviato il sistema, l'utente definisce qual è il titolo su cui è interessato, il genere e l'anno di uscita. Il sistema inizia la ricerca del film e raccomanderà 5 film con caratteristiche simili.

Implementazione

Per realizzare il recommender system, abbiamo fatto uso della libreria open source python **SKLearn**, che ci ha permesso di utilizzare una serie di algoritmi appartenenti al Natural Language Processing.

Per calcolare le raccomandazioni, abbiamo utilizzato tre categorie del dataset: "title", "genre" e "release_year".

All'utente è stato infatti richiesto di definire questi tre parametri tramite linea di comando.
I parametri vengono salvati in una struttura più complessa delle variabili, ossia un DataFrame.

```
def get_recommendation():  2 usages
    print("\nInserire i dati del film di cui si vuole avere una raccomandazione.")
    title = input("Inserisci il titolo: \n")
    main_genre = input("Inserisci il genere: \n")
    year = input("Inserisci l'anno di uscita: \n")
    user_data = pd.DataFrame( data: {'title': title, 'main_genre': main_genre, 'year': year}, index=[0])
    movie_index = construct_recommendation( filename: '../dataset/pre-processato/pre_processed_dataset.csv', user_data)
    return movie_index
```

```
Selezionare un'operazione:
1. Utilizzare l'ontologia
2. Utilizzare la KnowledgeBase
3. Utilizzare un Recommender System per i film
0. Uscire
Inserisci un valore: 3

Inserire i dati del film di cui si vuole avere una raccomandazione.
Inserisci il titolo:
Attack
Inserisci il genere:
action
Inserisci l'anno di uscita:
1980
```

Poiché i dati di ciascuna categoria non erano nel formato adatto per confrontare al similarità tra diversi film, è stato necessario trasformare queste tre categorie in vettori di parole. Questi vettori rappresentano una versione vettoriale delle parole presenti in un "documento", in cui ogni vettore trasmette un significato semantico associato a ciascuna parola.

```
movie_data['all_content'] = (movie_data['title'].astype(str) + ';' + movie_data['release_year'].astype(str) + ';' +
    movie_data['runtime'].astype(str) + ';' + movie_data['main_genre'].astype(str))

tfidf_matrix = vectorize_data(movie_data)
tfidf_matrix_array = tfidf_matrix.toarray()
```

```
def vectorize_data(movie_data):  1 usage
    vectorizer = TfidfVectorizer(analyzer='word')
    tfidf_matrix = vectorizer.fit_transform(movie_data['all_content'])
    return tfidf_matrix
```

Con la tecnica TF-IDF (term frequency-inverse document frequency), vengono calcolati i vettori e viene generata una matrice in cui ogni colonna rappresenta una parola ottenuta dalla concatenazione delle categorie selezionate e, allo stesso tempo, rappresenta il film stesso.

Il punteggio TF-IDF misura la frequenza di una parola in un documento, ponderata in base al numero di documenti in cui quella parola compare. Questo approccio serve a diminuire l'importanza delle parole che appaiono spesso, riducendo così il loro impatto nel calcolo finale.

La matrice è stata generata utilizzando la classe TFIdfVectorizer' della libreria SKLearn.

Successivamente è stata calcolata la similarità. Si è deciso di utilizzare come metrica principale la Correlazione di Pearson.

La correlazione è una misura statistica che indica la tendenza che hanno due variabili (X e Y) a variare insieme. Il concetto di correlazione, però, non ipotizza un nesso di causa-effetto ma è legato piuttosto al rapporto che esiste tra due variabili quantitative. La correlazione, infatti, ci permette di affermare se tra due variabili c'è una relazione lineare, ma non che una variabile sia la causa dell'altra. Questa misura indica sia la forza sia la direzione di una relazione lineare.

I valori che può assumere variano da-1 a +1. Un valore pari a 1 di questi estremi indica una correlazione lineare perfetta, rispettivamente positiva o negativa.

Viene quindi creata una successiva matrice in cui verranno riportati i punteggi di similarità tra ciascun film e tutti gli altri. In altre parole, ogni film sarà rappresentato nella matrice da una colonna-vettore, in cui ogni elemento indicherà il punteggio di similarità rispetto ad ogni altro film del dataset.

Infine vengono selezionati i cinque film più simili a quello inserito dall'utente e vengono mostrati a video.

```
                          title  release_year main_genre streaming_service  \
6631         Attack of the Demons          2019     horror            amazon
3659             The Ghazi Attack          2017   thriller            amazon
461    Gamera 2: Attack of Legion          1996      drama            amazon
14913         All Monsters Attack          1969    fantasy               hbo
7691                  The Intruder          1956      drama            amazon
```

Oltre al recommender system, viene effettuata una classificazione del dataset secondo una categoria aggiuntiva detta “star”. La categoria “star” si basa su categorie esistenti che valutano i film ossia imdb_score e tmdb_score.
Quindi il suo punteggio viene calcolato in base alla media dei due rispettivi valori per un film.

```
movie_data['star'] = ((movie_data['imdb_score'] + movie_data['tmdb_score']) / 2)
movie_data.loc[(movie_data['star'] >= 7.5), ['star']] = 5
movie_data.loc[(movie_data['star'] < 7.5) & (movie_data['star'] >= 5), 'star'] = 4
movie_data.loc[(movie_data['star'] < 5) & (movie_data['star'] >= 3), 'star'] = 3
movie_data.loc[(movie_data['star'] < 3) & (movie_data['star'] >= 1.5), 'star'] = 2
movie_data.loc[(movie_data['star'] < 1.5)] = 1
knn_data = movie_data[['id', 'runtime', 'star', 'imdb_score', 'tmdb_score']].copy()
```

Infine sono state effettuate delle operazioni di valutazione del modello e di ottimizzazione dello stesso. Il modello di classificazione utilizzato è il k-NN, ossia un algoritmo impiegato nel riconoscimento di pattern per classificare oggetti in base alle caratteristiche dei loro vicini (dove k sono il numero di vicini selezionati).

Per rendere il dataset più efficiente e quindi adatto all’uso del modello selezionato, esso è stato diviso in due parti: training e test. La sezione di training, pari all’80% del dataset, verrà utilizzata per addestrare il modello, mentre la restante, ossia il 20% del dataset, verrà utilizzato per il test.

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: x, y, test_size=0.2, random_state=1, stratify=y)
```

Per far sì che le operazioni siano significative è importante dire che gli elementi del dataset devono essere assegnata in maniera casuale, assegnando al parametro “random-state” un numero intero qualsiasi.

Infine, il modello viene allenato sfruttando le funzioni messe a disposizione dalla libreria SKLearn.

Knowledge Base

Introduzione

Una **Knowledge Base** (KB), ovvero una base di conoscenza, è un sistema organizzato che raccoglie, memorizza e gestisce informazioni, dati e conoscenze in un formato strutturato, che può essere facilmente interrogato e utilizzato per risolvere problemi o rispondere a domande. Le KB sono progettate per rappresentare e immagazzinare la conoscenza in un dominio specifico, consentendo agli utenti o ai sistemi di accedere rapidamente alle informazioni rilevanti.
In generale, una Knowledge Base si basa su:

1. **Fatti**: informazioni di base e dettagliate su entità specifiche.

Nel nostro contesto, la KB conteniene dati sui film (titolo, anno di uscita, attori, genere, ecc.).

2. **Regole**: connessioni logiche tra le entità o regole che descrivono come le entità interagiscono tra loro. Ad esempio, una regola che lega il film alla sua descrizione o ad una sua valutazione.

Una KB non si limita a memorizzare le informazioni: è progettata per essere interattiva e capace di supportare sistemi intelligenti (come motori di ricerca o sistemi di raccomandazione) nell'elaborare risposte a domande o problemi complessi, attingendo ai dati e alle regole memorizzate al suo interno.

Operazioni effettuabili

Nel nostro contesto le operazioni che si è deciso di effettuare sono le seguenti:

1. Ricerca di film per periodo di uscita, genere e durata

- Selezione del periodo di uscita
- Selezione del genere
- Selezione della durata

Risultato: Il sistema restituisce l'elenco degli ID e dei titoli dei film che corrispondono ai criteri specificati. Se nessun film corrisponde ai criteri, viene visualizzato un messaggio.

```
Scegliere quale funzione eseguire:
1. Trova un film dato il periodo d'uscita, il genere e la durata
2. Dato il genere ed una tipologia di valutazione, determina i film migliori secondo l'indice IMDD
3. Trovare la miglior piattaforma di streaming (devi aver eseguito 1.)
0. Esci
Inserisci un valore: 1

Inserisci il periodo relativo all'anno di uscita del film
1. recente
2. tra 2000 e 2010
3. fino al 2000
2
```

```
Inserisci il genere cui vorresti appartenga il film
1. western  2. scifi    3. romance
4. drama    5. horror   6. thriller
7. comedy   8. crime    9. documentation
10. family  11. action  12. fantasy
13. animation   14. music   15. history
16. war 17. european    18. sport
19. reality
1
Inserisci la durata del film
1. breve
2. media
3. lunga
3
I film trovati sono:
130980 Doc West
145587 Open Range
92451 Red Hill
129730 And Starring Pancho Villa as Himself
21423 All the Pretty Horses
72682 Bury My Heart at Wounded Knee
79639 3:10 to Yuma
103563 Meeks Cutoff
```

2. Ricerca dei migliori film per genere e valutazione

- Selezione del genere
- Selezione della valutazione

Risultato: il sistema restituisce i primi 10 film con il punteggio IMDb più alto, che corrispondono al genere e alla valutazione selezionati. Se nessun film corrisponde ai criteri, viene visualizzato un messaggio

```
Scegliere quale funzione eseguire:
1. Trova un film dato il periodo d'uscita, il genere e la durata
2. Dato il genere ed una tipologia di valutazione, determina i film migliori secondo l'indice IMDD
3. Trovare la miglior piattaforma di streaming (devi aver eseguito 1.)
0. Esci
Inserisci un valore: 2
Inserisci il genere di cui vorresti trovare il miglior film
1. western  2. scifi    3. romance
4. drama    5. horror   6. thriller
7. comedy   8. crime    9. documentation
10. family  11. action  12. fantasy
13. animation   14. music   15. history
16. war 17. european    18. sport
19. reality
4
```

```
Inserisci la valutazione del film
1. bassa
2. buona
3. eccellente
1
I migliori film trovati:
Film ID: 1000619, IMDb Score: 6.9
Film ID: 366562, IMDb Score: 6.1
Film ID: 1011235, IMDb Score: 6.0
Film ID: 1034522, IMDb Score: 6.0
Film ID: 1035258, IMDb Score: 6.0
Film ID: 104793, IMDb Score: 6.0
Film ID: 105139, IMDb Score: 6.0
Film ID: 105196, IMDb Score: 6.0
Film ID: 106911, IMDb Score: 6.0
Film ID: 107305, IMDb Score: 6.0
```

3. Selezione della piattaforma di streaming
   Questa operazione può essere eseguita solo dopo aver effettuato una ricerca di film (operazione 1)

- Selezione del prezzo dell'abbonamento

Risultato: il sistema restituisce la piattaforma di streaming con il maggior numero di film che corrispondono alla tua ricerca e al filtro di prezzo, insieme al numero di film disponibili su quella piattaforma. Se nessuna piattaforma corrisponde ai criteri, viene visualizzato un messaggio.

```
Scegliere quale funzione eseguire:
1. Trova un film dato il periodo d'uscita, il genere e la durata
2. Dato il genere ed una tipologia di valutazione, determina i film migliori secondo l'indice IMDD
3. Trovare la miglior piattaforma di streaming (devi aver eseguito 1.)
0. Esci
Inserisci un valore: 3
Seleziona il costo dell'abbonamento:
1. economico
2. medio
3. costoso
2
La piattaforma consigliata è: hbo
Numero di film disponibili su hbo: 4
```

Strumenti utilizzati

Per la gestione della nostra Knowledge Base, abbiamo utilizzato l'estensione **Pyswip** del linguaggio di programmazione Python, che integra le funzionalità del linguaggio logico **Prolog**. Prolog è un linguaggio di programmazione dichiarativo basato sulla logica formale, noto per la sua capacità di rappresentare conoscenza in forma di **fatti** e **regole** e di inferire nuove informazioni attraverso un processo di **risoluzione logica**.
L'estensione Pyswip ci ha permesso di sfruttare la potenza di Prolog direttamente all'interno di Python, integrando la logica del linguaggio con la flessibilità di Python per la gestione dei dati.

Struttura

La KB è composta da fatti e regole che rappresentano un database di film, le loro caratteristiche, i servizi di streaming, etc.
I fatti, come affermato precedentemente, sono asserzioni che descrivono le proprietà dei film e sono memorizzati come predicati Prolog.
Questi includono:

- title(ID, Titolo): titolo del film con l'ID corrispondente
- description(ID, Descrizione): descrizione del film con l'ID corrispondente
- release_year(ID, Anno): anno di uscita del film con l'ID corrispondente
- runtime(ID, Durata): durata del film in minuti con l'ID corrispondente
- main_genre(ID, Genere): genere principale del film con l'ID corrispondente
- name(ID, Nome): nome associato al film con l'ID corrispondente
- imdb_score(ID, Punteggio): punteggio IMDB del film con l'ID corrispondente
- streaming_service(ID, Servizio): servizio di streaming che offre il film con l'ID corrispondente
- monthly_subscription_cost(ID, Costo): costo dell'abbonamento mensile del servizio di streaming che offre il film con l'ID corrispondente

Le regole sono definite per inferire nuove informazioni a partire dai fatti esistenti.
Queste regole includono:
Classificazione del prezzo di abbonamento:

- prezzo_economy(ID): Vero se il costo dell'abbonamento è inferiore a 9.99
- prezzo_medio(ID): Vero se il costo dell'abbonamento è uguale a 9.99
- prezzo_costoso(ID): Vero se il costo dell'abbonamento è maggiore di 9.99

Classificazione dell'anno di uscita:

- film_recente(ID): Vero se il film è uscito dopo il 2010
- film_tra_2000_2010(ID): Vero se il film è uscito tra il 2000 e il 2010 inclusi
- film_pre_2000(ID): Vero se il film è uscito prima del 2000

Definizione del genere del film:

- film_genre(ID, Genere): Vero se il film con ID ha il genere specificato

Classificazione della durata del film:

- film_breve_durata(ID): Vero se la durata del film è inferiore o uguale a 60 minuti
- film_media_durata(ID): Vero se la durata del film è compresa tra 60 e 90 minuti
- film_lunga_durata(ID): Vero se la durata del film è maggiore di 90 minuti

Classificazione della valutazione del film:

- film_valutazione_bassa(ID): Vero se il punteggio IMDB del film è inferiore o uguale a 6
- film_valutazione_buona(ID): Vero se il punteggio IMDB del film è compreso tra 6.1 e 8

- film_valutazione_eccellente(ID): Vero se il punteggio IMDB del film è compreso tra 8.1 e 10

Raggruppamento della durata:

- film_durata(ID, breve): Vero se il film ha una durata breve
- film_durata(ID, media): Vero se il film ha una durata media
- film_durata(ID, lunga): Vero se il film ha una durata lunga

Combinazioni di anno e genere:

- recente_genre(ID, Genre): Vero se il film è recente e di un determinato genere
- tra_2000_2010_genre(ID, Genre): Vero se il film è uscito tra il 2000 e il 2010 ed è di un determinato genere
- pre_2000_genre(ID, Genre): Vero se il film è uscito prima del 2000 ed è di un determinato genere

Combinazioni di anno, genere e durata:

- recente_genre_durata(ID, Genre, DurataCat): Vero se il film è recente, di un determinato genere e di una determinata durata
- tra_2000_2010_genre_durata(ID, Genre, DurataCat): Vero se il film è uscito tra il 2000 e il 2010, di un determinato genere e di una determinata durata
- pre_2000_genre_durata(ID, Genre, DurataCat): Vero se il film è uscito prima del 2000, di un determinato genere e di una determinata durata

Decisioni di progetto

Le decisioni di progetto ed i vincoli imposti alla base di conoscenza sono i seguenti:

- Pulizia dei dati: la funzione string_cleaning rimuove i caratteri non ASCII e le virgolette dai testi per garantire la compatibilità con Prolog
- Predicati discontigui: le direttive :- discontiguous permettono di definire i predicati in più parti del file
- Categorie di durata: la durata è divisa in tre categorie: breve (<= 60 minuti), media (tra 60 e 90 minuti), e lunga (> 90 minuti)
- Categorie di valutazione: la valutazione è divisa in tre categorie: bassa (<=6), buona (tra 6.1 e 8), eccellente (>= 8.1)
- Ricerca per periodo, genere e durata: l'utente seleziona il periodo di uscita (recente, tra 2000 e 2010, pre 2000), il genere e la durata del film per ottenere i risultati
- Ricerca del miglior film per genere e valutazione: l'utente seleziona il genere e la valutazione per trovare i film migliori
- Selezione della piattaforma di streaming: l'utente può selezionare il costo dell'abbonamento desiderato (economico, medio, costoso) per filtrare le piattaforme che offrono i film ricercati in precedenza
- Limitazione dei risultati: la ricerca dei migliori film per valutazione restituisce al massimo i primi 10 risultati

Ontologia

Introduzione

L'ontologia è una rappresentazione strutturata della conoscenza in un dominio specifico, che definisce concetti, relazioni e proprietà degli oggetti presenti in quel dominio.
Essa viene utilizzata per organizzare e interpretare dati in modo significativo, consentendo l'inferenza automatica e il ragionamento logico.
Un'ontologia è composta da:

- Classi: rappresentano le entità del dominio (es. "Film", "Genere", "Piattaforma di Streaming")
- Proprietà d'oggetto: descrivono le relazioni tra le classi (es. "un film è distribuito da una piattaforma di streaming")
- Proprietà dei dati: attribuiscono caratteristiche alle entità (es. "un film ha una durata")
- Individui: esempi specifici delle classi (es. "Inception" come istanza della classe "Film")

Decisioni di Progetto

La scelta dell'utilizzo di un'ontologia è dovuta all'ottima rappresentazione dei dati.
In particolare si è deciso di rappresentare i film, correlati alle loro caratteristiche, ed i diversi servizi di streaming.

Classi: L'ontologia contiene le seguenti classi:

- Film
- Genere
- Studio di produzione
- Anno di rilascio
- Piattaforma di streaming

Proprietà degli oggetti: l'ontologia include proprietà degli oggetti che descrivono le relazioni tra le classi. In particolare:

- has_been_released_in: descrive la relazione tra un film e la data di rilascio
- has_genre: descrive la relazione tra un film ed il suo genere
- is_distribuited_by: descrive la relazione tra un film e la piattaforma di streaming
- is_realized_by: descrive la relazione tra un film e lo studio di produzione

Proprietà dei dati: l'ontologia include proprietà dei dati che descrivono le caratteristiche delle classi. In particolare:

- description
- imdb_score
- platform
- runtime
- subscription_cost
- title
- tmdb_score

Operazioni effettuabili

Nel nostro contesto si è deciso che le operazioni effettuabili fossero le seguenti:

```
Selezionare un'operazione:
1. Visualizzare le classi
2. Visualizzare le proprietà d'oggetto
3. Visualizzare le proprietà dei dati
4. Eseguire una query
0. Uscire
```

L'utente può selezionare una classe specifica per visualizzare gli elementi appartenenti a quella classe.

```
Inserire scelta: 1
Le classi dell'ontologia sono:
- Ontology.owx.Film
- Ontology.owx.Genre
- Ontology.owx.ProductionStudios
- Ontology.owx.ReleaseYear
- Ontology.owx.Runtime
- Ontology.owx.StreamingPlatform

Selezionare una delle seguenti classi:
1. Film
2. Genere
3. Studio di produzione
4. Anno di rilascio
5. Piattaforma di streaming
0. Uscita
```

```
Inserire scelta: 1

Lista dei film:
- Ontology.owx.Film
- Ontology.owx.Attack
- Ontology.owx.Chehre
- Ontology.owx.Doors
- Ontology.owx.Kombu
- Ontology.owx.Lost_in_the_Stratosphere
- Ontology.owx.Midnight
```

L'utente può visualizzare le proprietà d'oggetto

```
Selezionare un'operazione:
1. Visualizzare le classi
2. Visualizzare le proprietà d'oggetto
3. Visualizzare le proprietà dei dati
4. Eseguire una query
0. Uscire

Inserire scelta: 2

Proprietà d'oggetto presenti nell'ontologia:
- Ontology.owx.has_been_released_in
- Ontology.owx.has_genre
- Ontology.owx.is_distribuited_by
- Ontology.owx.is_realized_by
```

L'utente può visualizzare le proprietà dei dati

```
Selezionare un'operazione:
1. Visualizzare le classi
2. Visualizzare le proprietà d'oggetto
3. Visualizzare le proprietà dei dati
4. Eseguire una query
0. Uscire

Inserire scelta: 3

Proprietà dei dati presenti nell'ontologia:
- Ontology.owx.description
- Ontology.owx.imdb_score
- Ontology.owx.platform
- Ontology.owx.runtime
- Ontology.owx.subscription_cost
- Ontology.owx.title
- Ontology.owx.tmdb_score
```

L'utente può effettuare delle query prefissate sull'ontologia in base a criteri specifici

- L'utente può visualizzare i film presenti su Amazon.

```
Selezionare un'operazione:
1. Visualizzare le classi
2. Visualizzare le proprietà d'oggetto
3. Visualizzare le proprietà dei dati
4. Eseguire una query
0. Uscire

Inserire scelta: 4

1) Lista dei film presenti su 'Amazon'
2) Lista film di genere 'comedy'
3) Film di durata superiore a 70 min
0) Torna indietro

Inserire un valore: 1
Film presenti su Amazon:
- Ontology.owx.Attack
- Ontology.owx.Chehre
- Ontology.owx.Doors
- Ontology.owx.Kombu
- Ontology.owx.Lost_in_the_Stratosphere
- Ontology.owx.Midnight
```

- L'utente può visualizzare i film di genere comedy.

```
1) Lista dei film presenti su 'Amazon'
2) Lista film di genere 'comedy'
3) Film di durata superiore a 70 min
0) Torna indietro

Inserire un valore: 2
Film di genere comedy:
- Ontology.owx.Lost_in_the_Stratosphere
```

- L'utente può visualizzare i film con durata superiore a 70 minuti

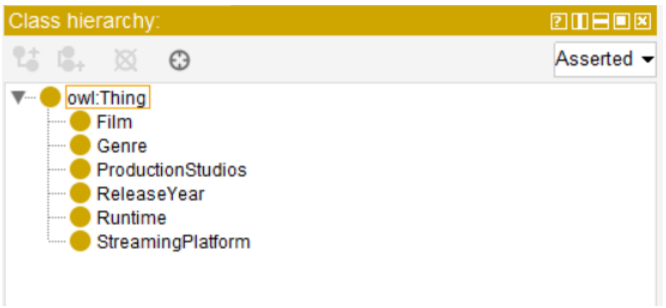
```
1) Lista dei film presenti su 'Amazon'
2) Lista film di genere 'comedy'
3) Film di durata superiore a 70 min
0) Torna indietro

Inserire un valore: 3
Film di durata superiore a 70 minuti:
- Ontology.owx.Attack
- Ontology.owx.Chehre
- Ontology.owx.Doors
- Ontology.owx.Kombu
- Ontology.owx.Midnight
```

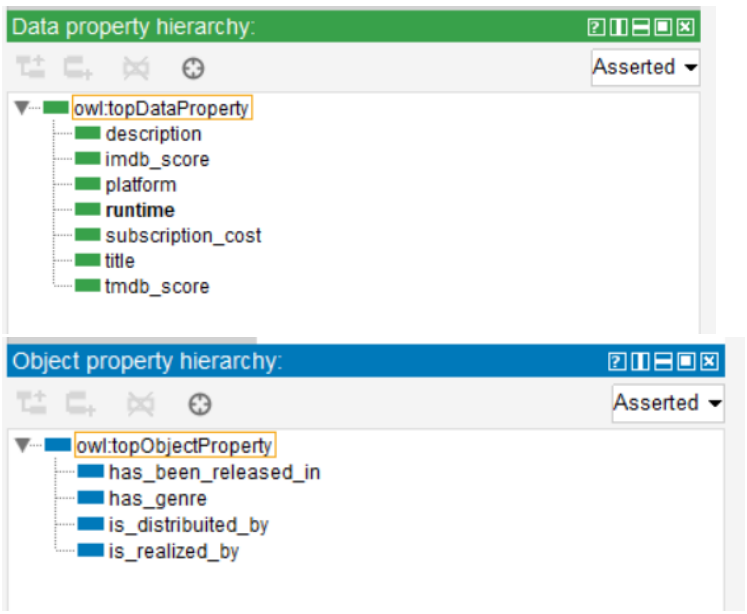
Creazione dell'ontologia

Per la realizzazione dell'ontologia abbiamo scelto di utilizzare l'editor di ontologie open source **Protégé**. Essendo un software con un'apposita interfaccia grafica, è stato più semplice ed intuitivo costruire l'ontologia.
L'applicativo infine consente di esportare il file in formato ".owx", utilizzato nel progetto per poter interagire in Python.
E' necessario inizialmente creare le classi, che costituiscono la base portante del sistema.

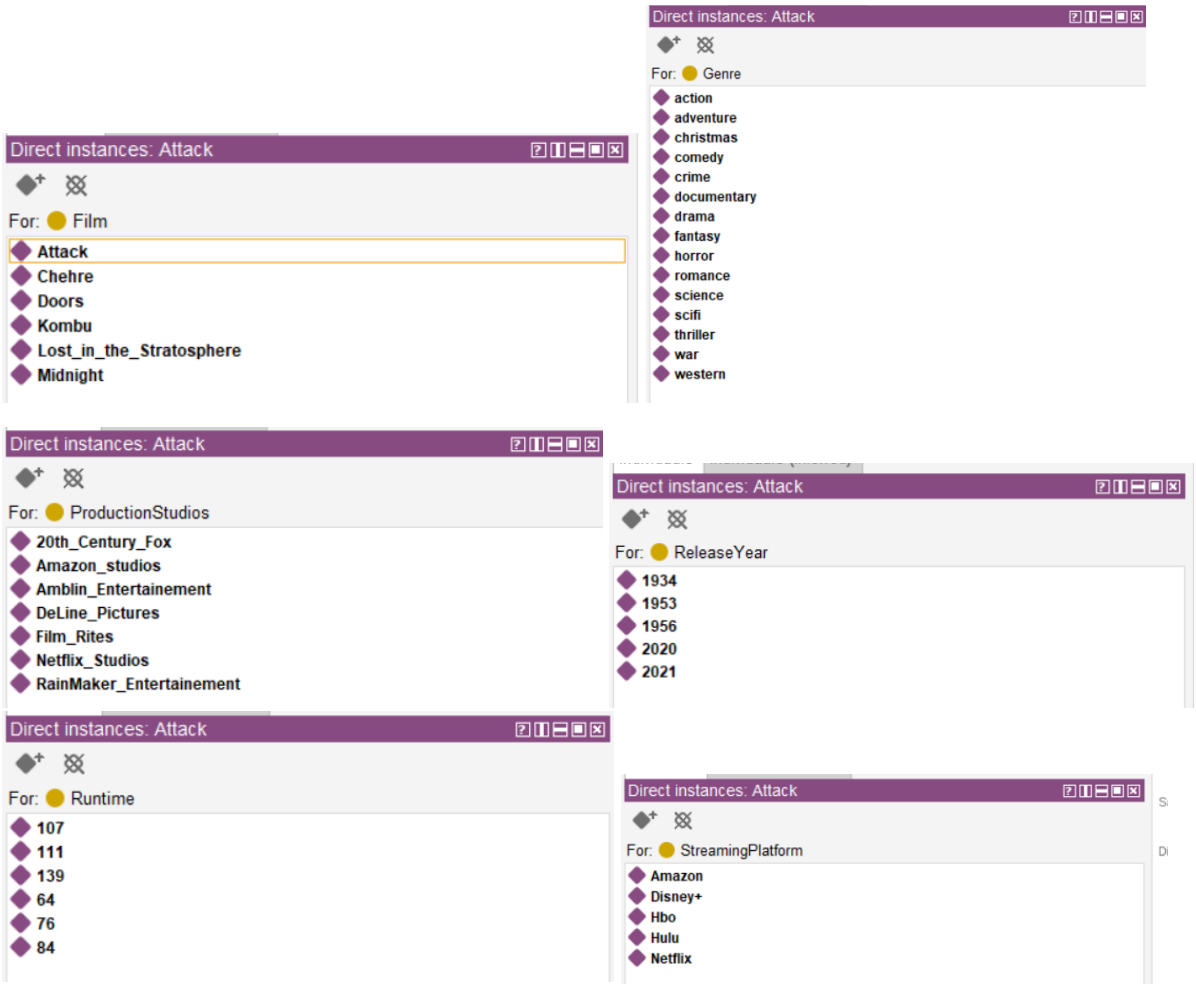
L'ontologia prevede la presenza di diverse classi, come prescritto precedentemente.

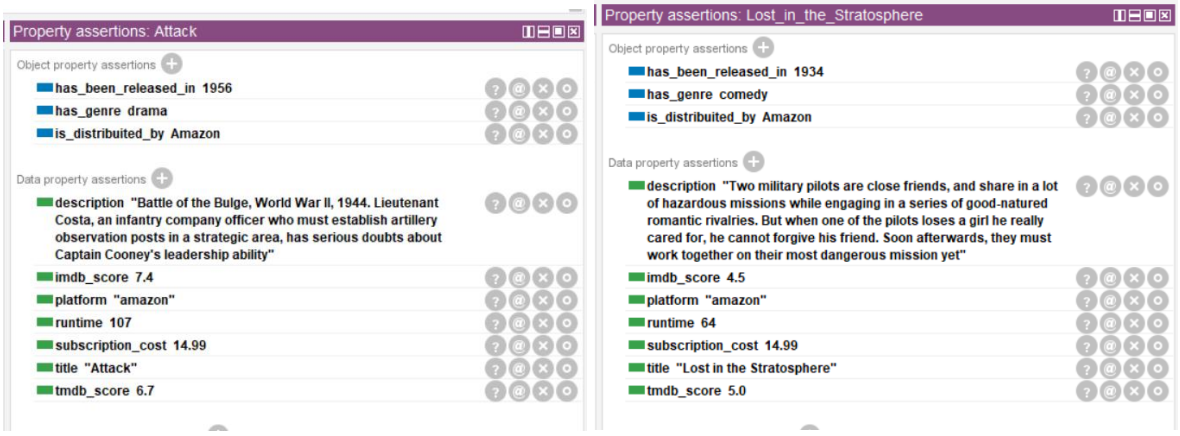


Dopo aver creato le classi abbiamo creato, inoltre, una serie di proprietà: object-properties e dataproperties. Queste servono a creare una relazione tra due individui appartenenti a classi uguali o differenti (object properties) o a collegare un individuo a un tipo di dato primitvo (data properties).



Successivamente siamo passati alla realizzazione degli individui. Essendo il database molto estato è stato necessario selezionare alcuni esempi di film presenti nel dataset che fossero quanto più rappresentativi del sistema.





Implementazione dell'ontologia

Il file esportato dal progetto Protege, in formato ".owx" è stato poi importato nel progetto al file di poter interrogare l'ontologia utilizzando il linguaggio Python, in particolare implementando la libreria Owlready2.
Utilizzando la libreria precedentemente citata è stato possibile selezionare e visualizzare a video le classi, le proprietà d'oggetto e dei dati, oltre a poter effettuare le query, che permettono di effettuare le operazioni precedentemente descritte.

Visualizzazione delle classi:

```
    if class_answer == '1':
        print("\nLista dei film: ")
        film = ontology.search(is_a=ontology.Film)
        for item in film:
            print(f"- {item}")
    elif class_answer == '2':
        print("\nGeneri: ")
        genre = ontology.search(is_a=ontology.Genre)
        for item in genre:
            print(f"- {item}")
    elif class_answer == '3':
        print("\nStudi di produzione: ")
        film_production_studio = ontology.search(is_a=ontology.ProductionStudios)
        for item in film_production_studio:
            print(f"- {item}")
    elif class_answer == '4':
        print("\nAnni di rilascio: ")
        year = ontology.search(is_a=ontology.ReleaseYear)
        for item in year:
            print(f"- {item}")
    elif class_answer == '5':
        print("\nLista piattaforme di streaming: ")
        streaming_service = ontology.search(is_a=ontology.StreamingPlatform)
        for item in streaming_service:
```

Esecuzione delle query:

```
while True:
    print("\n1) Lista dei film presenti su 'Amazon'\n""2) Lista film di genere 'comedy'\n"
          "3) Film di durata superiore a 70 min\n" "0) Torna indietro\n")

    query_choice = input("Inserire un valore: ")
    if query_choice == '1':
        print("Film presenti su Amazon: ")
        amazon_films = ontology.search(is_a=ontology.Film,
                                       is_distribuited_by=ontology.search(is_a=ontology.Amazon))
        for item in amazon_films:
            print(f"- {item}")
    elif query_choice == '2':
        print("Film di genere comedy:")
        scifi_films = ontology.search(is_a=ontology.Film, has_genre=ontology.search(is_a=ontology.comedy))
        for item in scifi_films:
            print(f"- {item}")
    elif query_choice == '3':
        print("Film di durata superiore a 70 minuti:")
        films = [film for film in ontology.Film.instances() if
                 hasattr(film, "runtime") and film.runtime[0] > 70]
        for item in films:
            print(f"- {item}")

    elif query_choice == '0':
        break
```

Conclusioni

Il progetto rappresenta una base per la creazione di un sistema di raccomandazione di film.
Con il sistema implementato l'utente potrà avere sicuramente una risposta alla domanda posta inizialmente "Che film vedo stasera?", ma vi sono ampi margini di miglioramento e numerose estensioni del progetto possono essere effettuate.
Innanzitutto potrebbe essere ampliato il dataset, che faccia riferimento ad altre piattaforme di streaming o che usi gli stessi dataset delle piattaforme, al fine di essere sempre aggiornato sulle nuove uscire.
Successivamente potrebbe essere implementato un sistema che dia all'utente maggiore possibilità di scelta per quanto riguarda i filtri dei film, specificandone di più dettagliati e caratteristici.
Sarebbe possibile effettuare un'integrazione con assistenti vocali, quali Alexa o Google Assistant, che in tempo reale forniscono una raccomandazione sulla base delle parole chiave che impartisci al sistema.
Infine potrebbe essere implementato un sistema che possa valutare le emozioni ed i sentimenti dell'utente e possa raccomandare film specifici.