Università degli studi "Aldo Moro" di Bari

Relazione progetto di Ingegneria della Conoscenza

Per l'anno accademico 2022/2023

"*BookOntology*": Una libreria digitale fornita di recommender system, un'ontologia con knowledge base interrogabile e un modello KNN per predizioni di una feature creata appositamente per rivalutare ogni libro.

A cura di:

Antonio De Rosis (mat. 703111)

Ivan Digioia (mat. 716685)

Email:

i.digioia3@studenti.uniba.it

a.derosis1@studenti.uniba.it

Sommario

Capitolo 1: Introduzione

In questo progetto, ci siamo impegnati a creare un applicativo in command line, che potesse essere d'aiuto a un qualunque amante della lettura in ricerca di nuovi libri da leggere. Il sistema permette l'esplorazione e lo studio di un dataset di libri, nello specifico costruendo un sistema per la raccomandazione di libri in base a dei dati di input dell'utente, una ontologia con knowledge base interrogabile per esplorare al meglio tutto ciò che riguarda le entità e le relazioni che le legano e infine un KNN che effettua predizioni su una feature costruita appositamente da noi per poter fornire all'utente un metrica di valutazione di ogni libro più oggettiva, mostrando una scala di gradimento della community differente dal semplice rating medio, facilmente influenzabile dal numero di recensioni.

1.1 Dataset e strumenti

Il dataset utilizzato è stato costruito tenendo in considerazione diversi altri file .csv ottenuti attraverso la piattaforma Kaggle, puntando ad avere tutte le feature che potessero essere utili al sistema e interessanti per l'utente, cercando anche di ottenere una percentuale di sparsità prossima allo zero. Tutto il codice è stato scritto in Python su PyCharm, utilizzando principalmente 4 librerie: **Numpy** e **Pandas** per la gestione e manipolazione dei dataframe contenenti i dati del dataset, **SciKit Learn** per i moduli di recommender system e KNN e Pytholog per la costruzione e interrogazione della base di conoscenza. Per l'editing e la rappresentazione dell'ontologia è stato utilizzato il software open-source **Protégé** (file .owl dell'ontologia presente nella cartella Documentazione).

Capitolo 2: Recommender System e KNN

2.1 Recommender System

Il recommender system è un sistema di information filtering che ha come scopo quello di fornire suggerimenti su degli specifici elementi in base alle preferenze dell'utente. Nello specifico, quello utilizzato in questa applicazione è un **Content-based Recommender System,** poiché il suggerimento è generato in base alle feature che descrivono ogni singola tupla del dataset, con l'unica differenza che non vi è un profilo utente, con informazioni sulle preferenze, ma viene scelto di far inserire manualmente all'utente informazioni sul tipo di libri che volesse leggere e su cui volesse ricevere suggerimenti.

2.1.1 Studio delle metriche di confronto

Nella costruzione di questo sistema, il primo passo è stato capire quale potesse essere la miglior metrica per il calcolo di similarità fra le informazioni inserite dall'utente e quelle fornite dal dataset. Le metriche considerate per il confronto sono il coefficiente di correlazione di Pearson, la similarità del coseno e la distanza euclidea:

Coefficiente di correlazione di Pearson – Questo indice esprime il tasso di correlazione lineare fra due vettori, attraverso un valora che è compreso tra -1 e 1 (corrisponde alla perfetta correlazione lineare positiva, corrisponde a un'assenza di correlazione lineare e corrisponde alla perfetta correlazione lineare negativa). Questo coefficiente è una variante del prodotto interno e rispetto alla similarità del coseno, viene applicata su una versione più centrata dei vettori e risulta indifferente al variare dei vettori

;

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Similarità del Coseno – Questa è una misura di similarità fra vettori definito in uno spazio del prodotto interno. Il calcolo della similarità è basato sul coseno dell'angolo tra i vettori, ignorando la loro dimensionalità. Anche in questo caso il valore può variare in un intervallo compreso fra -1 e 1 (dove con valore 1 si indica totale proporzionalità, e con -1 una totale

contrarietà). Risulta molto popolare come strategia nel Natural Language Processing e soprattutto nei casi in cui la sparsità del dataset è più elevata;

$$\frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2 \cdot \sum_{i=1}^{n} B_i^2}}$$

Distanza Euclidea - In matematica, la distanza euclidea è una distanza tra due punti, in particolare è una misura della lunghezza del segmento avente per estremi i due punti. In questo caso viene applicata ad uno spazio vettoriale per misurare la distanza fra due vettori, tuttavia risulta la meno efficace in quanto non viene applicata nel Natural Language Processing solitamente, ma più per valori continui.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

Per comprendere al meglio quale metrica fosse la più adeguata all'utilizzo abbiamo eseguito un confronto che misurasse il tempo di esecuzione e di generazione dei suggerimenti con le singole metriche.

```
Tempo di esecuzione per la similarità del coseno: 275.25687050819397
Tempo di esecuzione per il calcolo della coorelazione di Pearson: 47.22059369087219
Tempo di esecuzione per il calcolo della distanza euclidea: 284.2529122829437



Sparsità del dataset: 0.004366430879398919 %
```

Dai risultati ottenuti vi è un palese stacco del coefficiente di correlazione di Pearson, molto più veloce delle altre metriche. La velocità del calcolo è fondamentale ai fini di una esperienza utente godibile, quindi per via del fattore tempo e per le caratteristiche delle metriche in sé, la scelta finale ricade proprio sul **coefficiente di Pearson**.

2.1.2 Realizzazione

Alla base della realizzazione del Content-based Recommender System, ci sono gli strumenti forniti dalla libreria **SKLearn,** che hanno permesso non solo di calcolare e confrontare i diversi metodi visti sopra, ma fornisce anche una serie di algoritmi per il Natural Language Processing. Dopo aver scelto quali feature utilizzare per la generazione dei suggerimenti nel recommender system, ovvero le colonne: 'title', 'author', 'publisher', 'published_year', 'genre' e 'number_of_pages', è stato necessario ritrasformarne il contenuto affinchè fosse nella forma adeguata per il calcolo della similarità. Tale trasformazione avviene attraverso la formula del TF-IDF, che porta alla formazione di vettori di parole pregne di significati che identificano le caratteristiche dei libri.

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

1 Formula del TF-IDF

Viene in questo modo generata una matrice dove ogni colonna rappresenta una parola presa dalla concatenazione delle categorie scelte e allo stesso tempo rappresenterà il singolo elemento (un libro, nel nostro caso). Questa metrica misura in maniera adeguata il peso della parola specifica non solo all'interno del documento, ma nell'intera collezione di documenti, evitando che parole molto frequenti, ma in pochi documenti, abbiano importanza troppo superiore a parole presenti in tutta la collezione ma con meno frequenza all'interno dei singoli documenti.

```
# calcolo tf-idf e vettorizzazione delle feature utili al confronto
vectorizer = TfidfVectorizer(analyzer='word')
tfidf_matrix = vectorizer.fit_transform(books['comp_features'])
tfidf_matrix_array = tfidf_matrix.toarray()
```

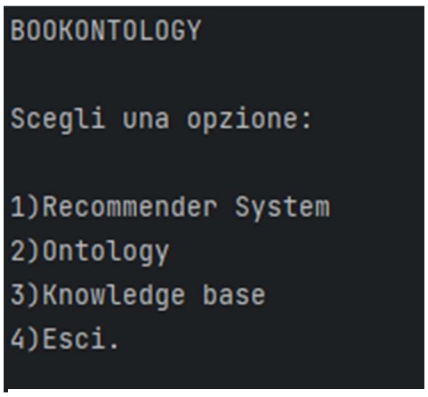
2 *Codice di implementazione G-idf*

Il passo successivo è calcolare la similarità con il coefficiente di Pearson, la cui formula è vista sopra, ottenendo una matrice di punteggi di similarità rispetto al singolo libro, che poi verrà ordinata e restituita come output del recommender system.

```
# calcolo correlazione di pearson per ogni libro all'interno del dataset rispetto al suggerimento
corr = []
for i in range(len(tfidf_matrix_array)):
    corr.append(pearsonr(tfidf_matrix_array[id], tfidf_matrix_array[i])[0])
corr = list(enumerate(corr))
sorted_corr = sorted(corr, reverse=True, key=lambda x: x[1])[1:6]
corr_index = [i[0] for i in sorted_corr]
correlated_books = \
    books[['title', 'author', 'publisher', 'published_year', 'genre', 'number_of_pages', 'average_rating']].iloc[
        corr_index]
return correlated_books, corr_index
```

3 Creazione matrice delle similarità

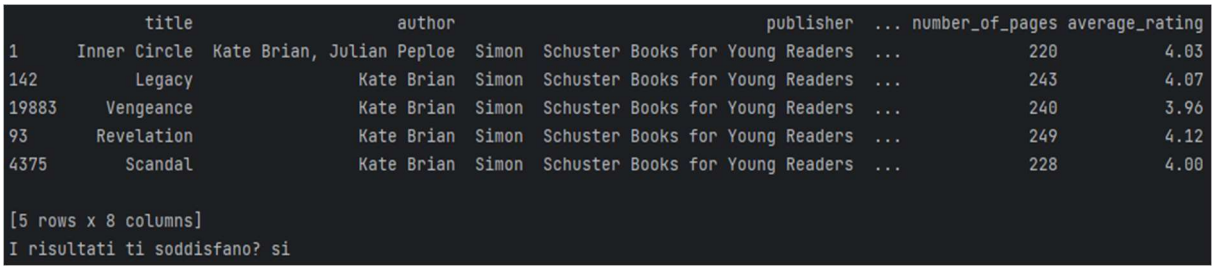
2.1.3 Esempio di utilizzo



5 Main Menu

```
**Books Recommender**
Inserire dati del libro

Attenzione: Per garantire una ricerca accurata, inserire piu' informazioni possibili (compilare almeno due campi).

Titolo: inner circle
Autore: kate brian
Casa editrice: Simon  Schuster Books for Young Readers
Data di pubblicazione: 2007
Genere: romance
Numero di pagine: 220

ricerca libri in corso...
```

4 Menu Inserimento dettagli per il suggerimento



```
              title                    author                              publisher  ... number_of_pages average_rating
1      Inner Circle  Kate Brian, Julian Peploe  Simon  Schuster Books for Young Readers  ...             220          4.03
142          Legacy                Kate Brian  Simon  Schuster Books for Young Readers  ...             243          4.07
19883     Vengeance                Kate Brian  Simon  Schuster Books for Young Readers  ...             240          3.96
93       Revelation                Kate Brian  Simon  Schuster Books for Young Readers  ...             249          4.12
4375        Scandal                Kate Brian  Simon  Schuster Books for Young Readers  ...             228          4.00

[5 rows x 8 columns]
I risultati ti soddisfano? si
```

6 Output del Recommender System

2.2 K-Nearest Neighbors (KNN)

Direttamente collegato al modulo del recommender system, c'è quello dedicato al KNN, che è stato addestrato al puro scopo di eseguire previsioni su una feature specifica creata da noi, quella salvata nella colonna 're_evaluation'.

2.2.1 Introduzione alla nuova feature per la previsione

Nel file denominato dataset.py, vi è il codice in cui il dataset originale viene arricchito con una feature studiata da noi per poter fornire un ulteriore tipo di valutazione per ogni libro, che possa essere più oggettiva rispetto al semplice rating medio, spesso influenzabile dal numero di recensioni ricevute. Portandone un esempio più concreto: se si dovesse stabilire l'attitudine della community su un determinato libro, non può essere sufficiente il rating medio contrassegnato, in quanto questo metterebbe in una posizione di superiorità quei libri che potrebbero avere un rating medio di 5 stelle, con ben poche recensioni, rispetto ad un libro con un rating di 4.5 stelle ma con centinaia di migliaia di recensioni in più. Il nostro scopo è stato definire una scala di apprezzamento basata su 5 valori: MOLTO NEGATIVO, NEGATIVO, NELLA MEDIA, POSITIVO e MOLTO POSITIVO, ovviamente rendendo questo nuovo valore ottenibile interrogando la base di conoscenza, andando dal menu principale nella sezione knowledge base e poi selezionando l'opzione 3.

Questa rivalutazione, funziona nel seguente modo:

1. Vengono calcolati i valori medi delle colonne 'average_rating' e 'rating_count';

```
book_df['re_evaluation'] = np.nan
avg_rating = book_df['average_rating'].mean()
avg_no_rating = book_df['rating_count'].mean()
```

2. Vengono definiti gli offset rispetto ai valori medi, per creare un intervallo il cui punto medio è la media stessa, affinché si possano definire i valori all'interno di tale intervallo come "NELLA MEDIA";

```
offset_noratings = 1500
offset_rating = 0.25
```

3. Nell'ultima fase vi è la rivalutazione dei libri basata sui valori descritti prima.

Se il valore di 'rating_count' del libro è maggiore del valore medio della colonna sommato all'offset specificato e il valore nella colonna average_rating è maggiore della media della colonna sommata all'offset specifico, risulta implicito che l'ottima valutazione è condivisa da un alto volume di lettori, quindi oggettivamente la valutazione della community è MOLTO POSITIVA.

Nel caso in cui il rating_count risulta maggiore della media sommata all'offset, ma il rating medio è inferiore alla media della colonna sottratto l'offset, risulta ovvio che nonostante l'alto volume di recensioni, l'opinione predominante è che il libro sia al di sotto delle aspettative, dunque per la community questo libro viene rivalutato in maniera MOLTO NEGATIVA.

Nel caso in cui il rating medio è inferiore alla media della colonna sottratto l'offset, ma il numero di rating_count risulta orbitare attorno alla media o è di poco al di sotto, l'esito della community, nonostante possa essere ritenuto ancora incerto, risulta tendente al NEGATIVO, ma è chiaramente soggetto a cambiamenti per via di possibili nuove valutazioni.

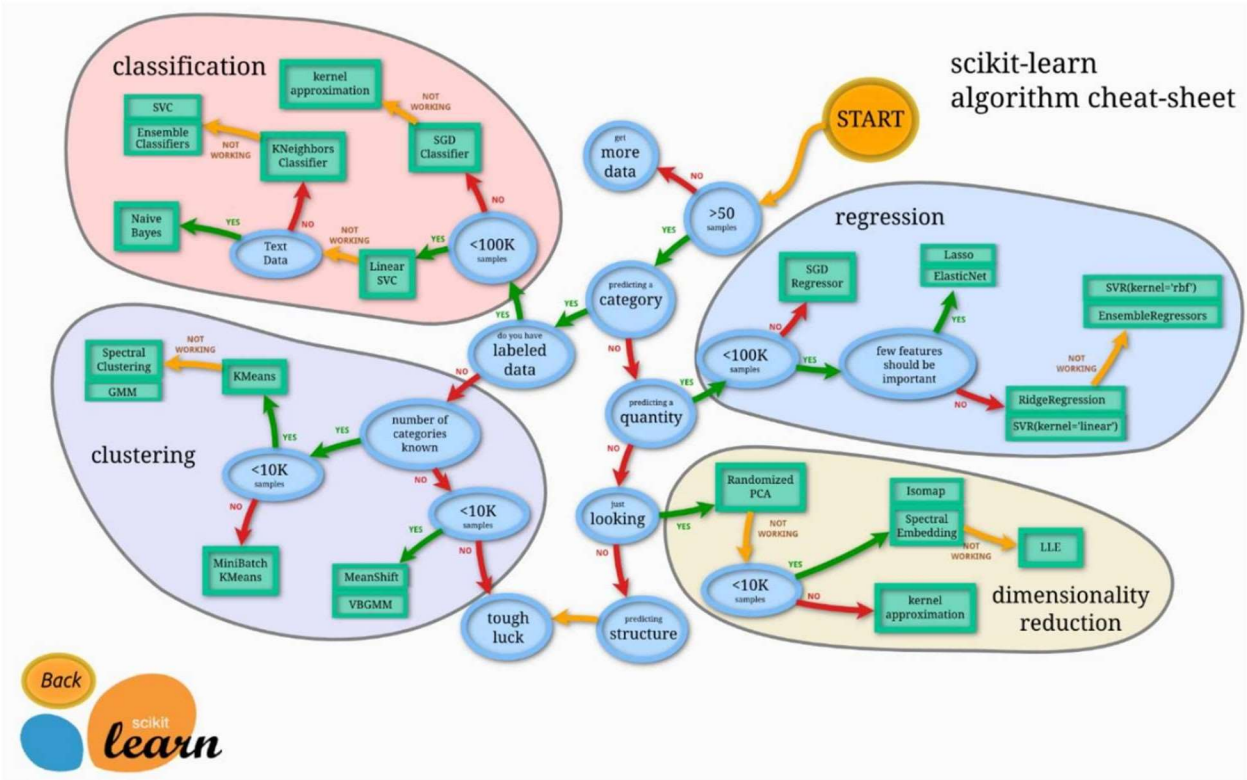
Se il valore di rating medio è maggiore alla media sommata all'offset, ma il valore di rating count orbita nell'intervallo vicino al valore medio, allora la valutazione della community a riguardo è POSITIVA, poiché per il basso numero di recensioni non vi può essere certezza e lo stesso valore potrebbe variare(vale lo stesso per il un rating nell'intervallo della media e per un numero di recensioni invece molto elevato).

Nel caso in cui sia rating_count che rating medio ricadano nell'intervallo medio, allora anche la valutazione della community sarà NELLA MEDIA. Di sotto la porzione di codice dedicata al controllo eseguito per ogni libro.

```
for i in range(book_df.shape[0]):

    if (float(book_df.average_rating[i]) >= avg_rating + offset_rating and
            float(book_df.rating_count[i]) >= float(avg_no_rating) + offset_noratings):

        book_df.re_evaluation[i] = "MOLTO POSITIVA"

    elif (float(book_df.average_rating[i]) <= float(avg_rating) - offset_rating and
          float(book_df.rating_count[i]) >= float(avg_no_rating) + offset_noratings):

        book_df.re_evaluation[i] = "MOLTO NEGATIVA"

    elif ((float(book_df.average_rating[i]) <= float(avg_rating) - offset_rating) and
          ((float(book_df.rating_count[i]) >= float(avg_no_rating)) or
          (float(book_df.rating_count[i]) <= float(avg_no_rating)))):
        book_df.re_evaluation[i] = "NEGATIVA"

    elif ((float(book_df.average_rating[i]) >= float(avg_rating) + offset_rating) and
          ((float(book_df.rating_count[i]) >= float(avg_no_rating)) or
          (float(book_df.rating_count[i]) <= float(avg_no_rating)))):

        book_df.re_evaluation[i] = "POSITIVA"

    elif ((float(book_df.rating_count[i]) >= float(avg_no_rating) + offset_noratings) and
          ((float(book_df.average_rating[i]) >= float(avg_rating)) or
          (float(book_df.average_rating[i]) <= float(avg_rating)))):

        book_df.re_evaluation[i] = "POSITIVA"

    else:
        book_df.re_evaluation[i] = "NELLA MEDIA"
```

2.2.2 Modello scelto

L'algoritmo **k-nearest neighbors**, noto anche come **KNN**, è un classificatore basato su apprendimento supervisionato non parametrico che utilizza la prossimità per effettuare classificazioni o previsioni sul raggruppamento di un singolo data point. Sebbene possa essere utilizzato sia per problemi di regressione che di classificazione, è tipicamente usato come algoritmo di classificazione, partendo dal presupposto che i punti simili possono essere trovati l'uno vicino all'altro. Va notato che l'algoritmo KNN fa anche parte di una famiglia di modelli di "apprendimento pigro", il che significa che memorizza solo un set di dati di addestramento invece di sottoporsi a una fase di addestramento. Ciò significa anche che tutti i calcoli vengono eseguiti quando viene effettuata una classificazione o una previsione. Poiché si affida fortemente alla memoria per memorizzare tutti i dati di addestramento, viene anche definito un metodo di apprendimento basato sull'istanza o sulla memoria. Un oggetto è classificato da un voto di pluralità dei suoi vicini, in quanto verrà assegnato alla classe più comune tra i suoi ***k*** elementi più vicini (***k*** è un numero intero positivo). Se ***k*** = ***1***, l'oggetto viene semplicemente assegnato alla classe dell'elemento più vicino.



7 Mappa concettuale per la scelta del modello

2.2.3 Suddivisione e preprocessing dataset

Per poter eseguire l'addestramento, è necessario suddividere il dataset in training set e test set, il primo contiene gli esempi che verranno utilizzati per addestrare il sistema, il secondo quelli in cui verrà testato. La suddivisione è del tipo 80% training set e 20% test set ed è stato importante settare i parametri random_state e stratify, per la funzione train_test_split di scikit learn, affinché si potessero prendere le giuste percentuali di ogni tipo di classe in maniera casuale, in modo da poter avere un addestramento migliore e dei risultati più affidabili.

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

Risulta fondamentale anche aver fatto label encoding della colonna target da predire, in questo caso re_evaluation, affinché i valori originali potessero essere etichettati con valori discreti piuttosto che stringhe, in quanto il knn non accetta questo tipo di dato per la previsione.

```
label_encoder = LabelEncoder()
book_df["reeval_int"] = label_encoder.fit_transform(book_df["re_evaluation"])
```

Dopo aver registrato l'output della previsione, si esegue reverse encoding delle etichette per poter stampare a schermo dei risultati sensati e ritornare ai valori originali.

```
corr["re_evaluation_prediction"] = label_encoder.inverse_transform(reeval_prediction)
```

2.2.4 Migliorare il modello: Hyperparameters Tuning

Inizialmente, abbiam testato il modello con dei parametri di default, per avere un punto di riferimento da cui partire per definire come migliorarne le performance.

```
knn = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', p=2, metric='minkowski',
                           metric_params=None, n_jobs=None)
```

Dopo aver fatto partire la prima fase di fitting e di testing, sono state calcolate le metriche su cui ci baseremo per poter notare i miglioramenti ottenuti durante il fine-tuning.

```
Valutazione del modello...

Classification report:
              precision    recall  f1-score   support

           0       0.12      0.05      0.07       130
           1       0.16      0.07      0.10       148
           2       0.22      0.22      0.22       827
           3       0.56      0.73      0.63      2427
           4       0.52      0.32      0.40      1465

    accuracy                           0.49      4997
   macro avg       0.32      0.28      0.28      4997
weighted avg       0.47      0.49      0.47      4997

ROC score:  0.6417823624615033

La nostra accuratezza è bassa, dobbiamo migliorare la qualità delle nostre predizioni
```

Queste metriche son state calcolate attraverso la funzione classification_report di sklearn, che ci permette di accedere a molteplici misurazioni e al ROC AUC score (misura dell'intera area bidimensionale sotto l'intera curva ROC), che inizialmente presenta un punteggio basso, ovvero 0.64. Subito dopo le prime predizioni, ha inizio il processo più importante per lo sviluppo di questo modello: la ricerca dei migliori parametri affinché possa migliorare non solo il ROC score, ma la prestazione generale del modello. Lo scopo è quello di implementare il random search utilizzando la cross validation (Repeated K-fold), implementando il metodo randomized_search.

```python
def randomized_search(hyperparameters, x_train, y_train):
    """
    Implementazione random search per fare finetuning degli iperparametri con cui addestrare il KNN
    """
    KNN = KNeighborsClassifier()

    # utilizzo della cross validation per trovare il numero di fold
    cvFold = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
    randomSearch = RandomizedSearchCV(estimator=KNN, cv=cvFold, param_distributions=hyperparameters)

    best_model = randomSearch.fit(x_train, y_train)

    return best_model
```

Nella random search vengono usate combinazione casuali di iperparametri per comprendere quale delle combinazioni esplorate potesse avere le prestazioni migliori ed è stata scelta rispetto al grid search proprio perché risulta più veloce. I possibili valori degli iperparametri sono i seguenti:

```
n_neighbors = list(range(1, 31))
weights = ['uniform', 'distance']
metric = ['euclidean', 'manhattan', 'hamming']
```

Dopo aver eseguito la ricerca per diverse volte, i risultati sono stati salvati in un dizionario da cui poi è stata presa la combinazione avente registrata l'incremento nelle prestazione più incisivo. L'esito della ricerca ha restituito la seguente combinazione:

```
Best weights: distance
Best metric: hamming
Best n_neighbors: 21
```

8 Miglior combinazione di parametri

Ricalcolando il ROC Score utilizzano il modello con gli iperparametri migliori, il miglioramento è evidente.

```
ROC score:  0.8122808180804864
```

Il miglioramento è visibile anche all'utilizzo, in quanto esegue la predizione in automatico per ogni libro e raccomandazione restituita dal recommender system.

```
Titolo: Harry potter and the philosopher's stone
Autore: j.k. Rowling
Casa editrice:
Data di pubblicazione: 1997
Genere: adventure
Numero di pagine: 345

ricerca libri in corso...

                                          title                  author  ... average_rating re_evaluation_prediction
8965          Harry Potter and the Philosopher's Stone  J.K. Rowling, Olly Moss  ...           4.47           MOLTO POSITIVA
5477                          Harry Potter: The Prequel            J.K. Rowling  ...           4.18                 POSITIVA
20659                          The End of Harry Potter?          David Langford  ...           4.40                 POSITIVA
2089                               The Hogwarts Library            J.K. Rowling  ...           4.46                 POSITIVA
269    Harry Potter Schoolbooks Box Set: Two Classic ...            J.K. Rowling  ...           4.40           MOLTO POSITIVA
```

9 Esempio di previsione

In conclusione, le prestazioni del modello sono più che soddisfacenti e potrebbero migliorare ancora con una maggiore espansione del dataset e un arricchimento di esempi per le classi che sono meno presenti.

Capitolo 3: Knowledge Base

Una knowledge base è una banca dati che riesce a fornire un supporto all'utente grazie alle informazioni presenti al suo interno, e rappresenta fatti sul mondo, un ragionamento su quei fatti e utilizza delle regole e altre forme di logica per dedurre nuovi fatti o evidenziare incongruenze. Quindi, la knowledge base (o KB) è definibile come un insieme di assiomi, cioè delle proposizioni che possono essere asserite essere vere che costituiscono dunque un ambiente volto a facilitare la raccolta, l'organizzazione e la distribuzione della conoscenza. In merito al nostro caso di studio, la KB viene utilizzata per consentire all'utente di porre domande inerenti al dominio approfondito.

3.1 Gestione della Knowledge Base

Per la gestione della knowledge base, abbiamo utilizzato l'estensione **Pytholog** del linguaggio Python basata sul linguaggio di programmazione logica Prolog.

Sulla base del Prolog, quindi, abbiamo iniziato con le funzioni di popolamento dei fatti della knowledge base, ogni dato del dataset verrà popolato tramite il proprio richiamo alla funzione, e infine per ogni popolamento abbiamo poi aggiunto la regola che permette l'interazione della KB con l'utente.

3.1 Fatti

I fatti rappresentano gli assiomi sempre veri della knowledge base, e sono la base per il funzionamento delle regole. Abbiamo 8 tipologie di fatti, ognuno rappresentante una relazione tra i libri e una delle possibili caratteristiche associate a questi. Tutti i popolamenti ti tipo Libro-caratteristica presentano all'interno della loro implementazione un controllo ovvero alcuni titoli dei libri contengono al loro interno una virgola e hanno necessità di un ulteriore fase di pre-processing che è eseguita tramite delle funzioni ovvero input_string (che esegue il pre-processing delle stringhe in input alla query) e output_string (che pulisce la stringa output della query affinchè sia gradevole alla lettura).

```python
def input_string(s):
    """
    Funzione per trasformare le stringhe date in input
    """
    if isinstance(s, float) or isinstance(s, int):
        s = str(s)
    s = s.lower()
    s = s.replace( _old: " ",  _new: "+")
    s = s.replace( _old: ",",  _new: "|")

    return s
```

```python
def output_string(s):
    """
    Funzione per ritrasformare le stringhe date in input alla funzione input_string
    """
    sx_offset = 10
    dx_offset = -2
    s = s.replace("|", ",")
    s = s.replace("+", " ")
    new_s = s[sx_offset:dx_offset]
    return new_s.title()
```

Le tipologie di fatti presenti nella KB sono le seguenti, e per ognuno è riportato un esempio:

1) “author(title, author)”
Rappresenta la relazione tra un libro e chi lo ha scritto;

2) “eval(title, re_evaluation)”
Rappresenta la relazione tra un libro e la sua rivalutazione in base al rating medio e numero di ratings;

3) “publisher(title, publisher)”
Rappresenta la relazione tra un libro e chi lo ha pubblicato;

4) “rating(title, average_rating)”
Rappresenta la relazione tra un libro e il rating medio;

5) “genre(title, genre)”
Rappresenta la relazione tra un libro e i suoi generi;

6) “published(title, published_year)”
Rappresenta la relazione tra un libro e l’anno di pubblicazione;

7) “pages(title, pages)”
Rappresenta la relazione tra un libro e il numero di pagine;

8) “no_of_ratings(title, rating_count)”
Rappresenta la relazione tra un libro e il numero di recensioni.

Oltre a queste 8 funzioni di popolamento sono state implementate ulteriori 3 funzioni che a differenza delle sopra citate non presentano il passaggio di pre-processing e sono solo re-implementazioni di funzioni “populate” già sviluppate ma con obiettivi diversi:

9) “rating (title, average_rating)”
Rappresenta la relazione tra un libro e il rating medio;

10) “author(title, author)”
Rappresenta la relazione tra un libro e chi lo ha autore;

11) “genre(title, genre)”
Rappresenta la relazione tra un libro e i generi.

3.3 Regole

Rappresentano delle domande che l'utente può rivolgere alla knowledge base senza il bisogno di conoscere la sintassi necessaria. Utilizzano i fatti che popolano la KB per poter mostrare all'utente i dati che questi ha richiesto.
Le regole presenti nella KB sono le seguenti, e per ognuna è riportato un esempio:

1) "is_written_by(X,Y) :- author(Y,X)"
Permette di ottenere informazioni su chi ha scritto un libro;

2) "re_evaluation(X,Y) :- eval(Y,X)"
Permette di ottenere informazioni sulla rivalutazione di un libro;

3) "published_by(X,Y) :- publisher(Y,X)"
Permette di ottenere informazioni su chi ha pubblicato un libro;

4) "rated(X,Y) :- rating(Y,X)"
Permette di ottenere informazioni sul rating medio di un libro;

5) "is_genre(X,Y) :- genre(Y,X)"
Permette di ottenere informazioni sui generi di un libro;

6) "is_published_in(X,Y) :- published(Y,X)"
Permette di ottenere informazioni sull'anno di pubblicazione di un libro;

7) "number_of_pages(X,Y) :- pages(Y,X)"
Permette di ottenere informazioni sul numero di pagine di un libro;

8) "rating_count(X,Y) :- no_of_ratings(Y,X)"
Permette di ottenere informazioni sul numero di rating;

9) "rated(X,Y) :- rating(Y,X)"
Permette di ottenere il titolo di un libro in base a un determinato rating;

10) "is_written_by(X,Y) :- author(Y,X)"
Permette di ottenere il titolo di un libro in base all'autore;

11) "is_genre(X,Y) :- genre(Y,X)"
Permette di ottenere il titolo di un libro in base a un genere.

3.4 Interazione con l'utente

L'utente appena entrato nella knowledge base avrà la possibilità di scegliere tra diversi tipi di ricerca o se preferisce tornare alla schermata iniziale uscendo dalla KB:

```
KNOWLEDGE BASE

Benvenuto, qui puoi eseguire ricerche sui libri e sulle loro caratteristiche

Ecco le ricerche che puoi eseguire:
1) Ricerche sulle caratteristiche di un libro
2) Confronti e ricerca di libri in base ad una caratteristica
3) Rivalutazione di un libro in base al rating medio e al suo numero di recensioni
4) Ritorna al main
```

10 Schermata principale knowledge base

1) Ricerche sulle caratteristiche di un gioco passato in input dall'utente, che, tramite le regole, permettono di ottenere la caratteristica che l'utente richiede:

```
Inserisci il numero corrispondente alla tua scelta: 1
Dammi il nome di un libro: Inner circle

Queste sono le caratteristiche che puoi cercare:
1) Chi lo ha scritto
2) Chi lo ha pubblicato
3) Qual'è il suo rating medio
4) Quanti rating ha ricevuto
5) Di che genere è
6) In che anno è uscito
7) Quante pagine ha
Puoi anche:
8) Cambia libro
9) Cambia tipo di ricerca
```

11 Menu opzione 1

Da qui, l'utente sceglie un tipo di informazione che vuole ricevere dalla KB selezionando un numero da 1 a 7 e al termine della richiesta ha la possibilità di fare una nuova ricerca con lo stesso libro, di sostituire il libro per la ricerca o cambiare tipo di ricerca.

```
Selezionane una: 1

 Inner Circle è stato scritto da: Kate Brian, Julian Peploe
```

2) Confronti di due libri scelti dall'utente in base alla qualità di questi, ricerca di libri che hanno un rating pari a quello inserito, ricerca di tutti i libri pubblicati da un autore e infine mostrare i primi 10 libri per rating di un genere specifico:

```
1) Confronto di qualità tra 2 libri
2) Ricerca libri con un determinato rating
3) Ricerca tutti i libri pubblicati da un autore
4) I migliori 10 libri di un genere specifico
Puoi anche:
5) Cambia tipo di ricerca
```

```
Selezionane una: 4

Di quale genere vuoi ricercare il miglior rating?: romance
[('The Strange Courtship of Abigail Bird', '5.0'), ('Grosvenor Square', '5.0'), ('When We Meet Again', '5.0'), ('Shadowed Love', '5.0'), ('The Immortal Lover'
, '4.93'), ('Binding Circumstance', '4.88'), ("Teapots And Attics: I'll Never Let Go", '4.83'), ("Siren's Song", '4.77'), ('Home Wrecker I', '4.77'), ('The Na
meless Encounter', '4.75')]
```

12 Esempio opzione 4

3) Rivalutazione di un libro confrontando il rating medio e il numero di recensioni:

La rivalutazione va da MOLTO POSITIVA (ovvero un libro con tante recensioni e una media di recensioni sopra la media o comunque molto vicina) fino a MOLTO NEGATIVA (ovvero un libro con molte recensioni e una media al di sotto o comunque molto vicina)

I libri che rientrano in POSITIVA o NEGATIVA sono libri di cui generalmente non si hanno troppi dati per poterli categorizzare ma che in entrambi i casi presentano un numero di recensioni molto vicino alla media ma con un rating tendenzialmente maggiore o inferiore alla media (o viceversa)

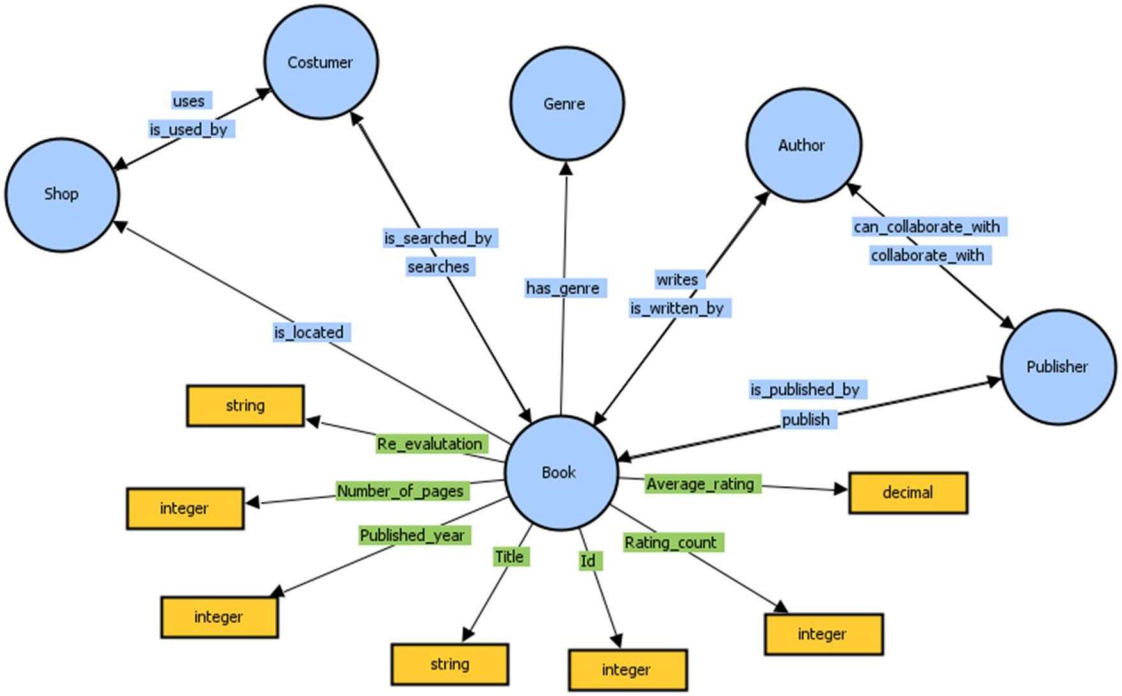
I libri che rientrano NELLA MEDIA sono libri su cui i dati rientrano perfettamente vicino alla media delle recensioni e del rating.

```
In questa sezione della knowledge base vengono rivalutati i libri,
non solo in base al rating medio ma anche in base al numero di recensioni

Inserire il libro da rivalutare: zoo city
Considerando il numero di recensioni di Zoo City, ovvero:
10007,
al momento molto sopra la media generale (7198) e considerando il rating medio ricevuto:
3.64
che risulta sotto la media (3.98), la valutazione finale della community è:
MOLTO NEGATIVA
```

Capitolo 4: Ontologia

L'ontologia, in informatica, è un modello di rappresentazione formale della realtà e della conoscenza. È una struttura di dati che consente di descrivere le entità (oggetti, concetti, ecc.) e le loro relazioni in un determinato dominio di conoscenza.

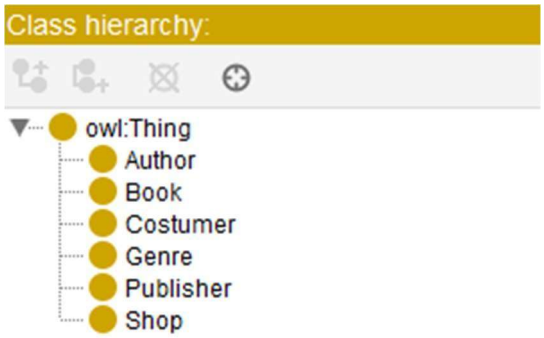
Abbiamo deciso di utilizzare e creare una ontologia che potesse rappresentare il dominio di un negozio di libri, nel caso in cui un lettore appassionato o un potenziale lettore, possano utilizzare la base di conoscenza e il sistema di raccomandazione per farsi consigliare un possibile libro in base ai propri gusti, senza l'aiuto di personale o ricerche su internet, che potrebbero o sviarlo dall'acquisto o consigliargli in modo errato



13 Struttura della Book-Ontolgy

4.1 Protégé

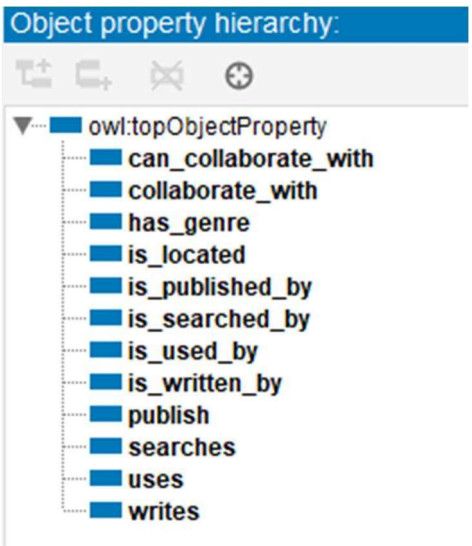
Per la realizzazione dell'Ontologia abbiamo deciso di utilizzare l'editor di ontologie open source Protégé. L'ontologia è organizzata in diverse classi, alcune di queste rappresentano una categoria legata al libro, che lo descrive nel dettaglio (Author, Publisher e Genre); altre invece sono rappresentati della modalità di acquisto di un libro (Costumer e shop)



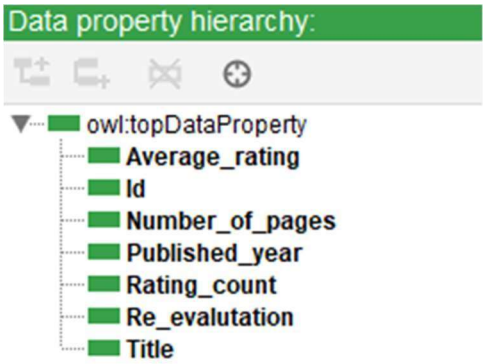
Classi dell'Ontologia

Oltre alla creazione delle classi, sono state create anche una serie di proprietà: Object-properties e Data-properties.

Queste proprietà aiutano a relazionare due individui di classi uguali o diverse (object-properties) oppure permettono di relazionare un individuo al loro tipo di dato primitivo (data-properties).



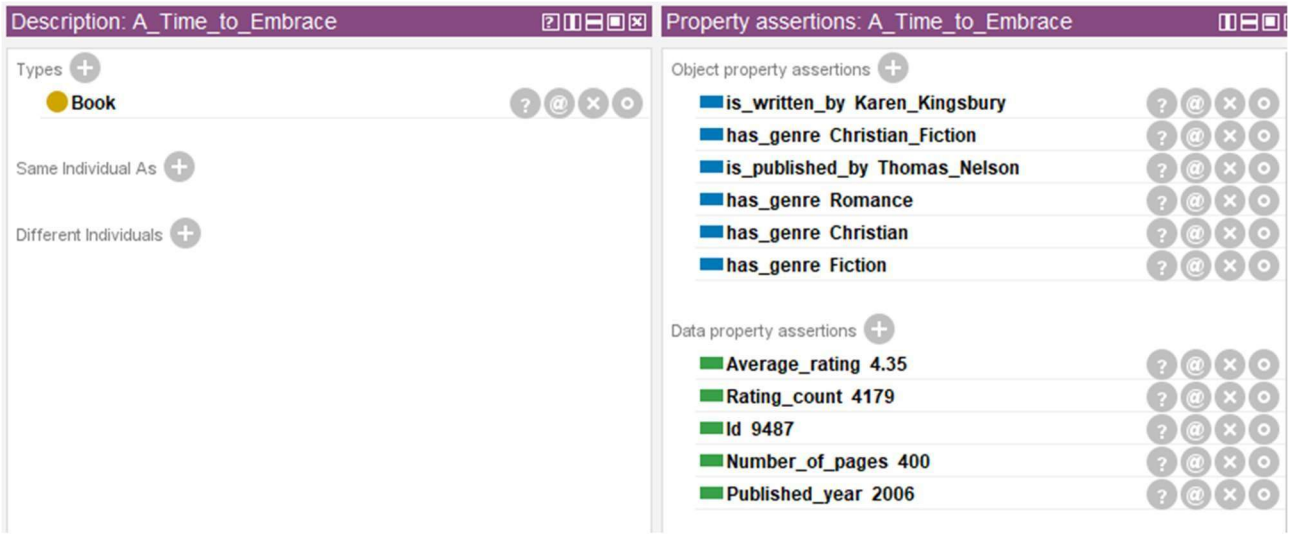
15 Object Properties



14 Data Properties

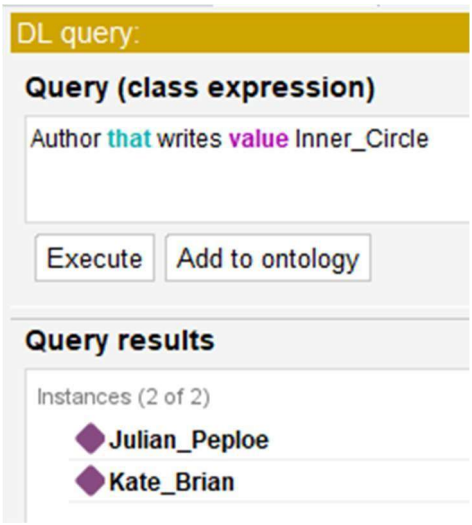
Successivamente si è passati alla creazione degli individui stessi che rappresentano alcuni esempi di libri presenti nel dataset.

Individui della classe Book, Author e Genre

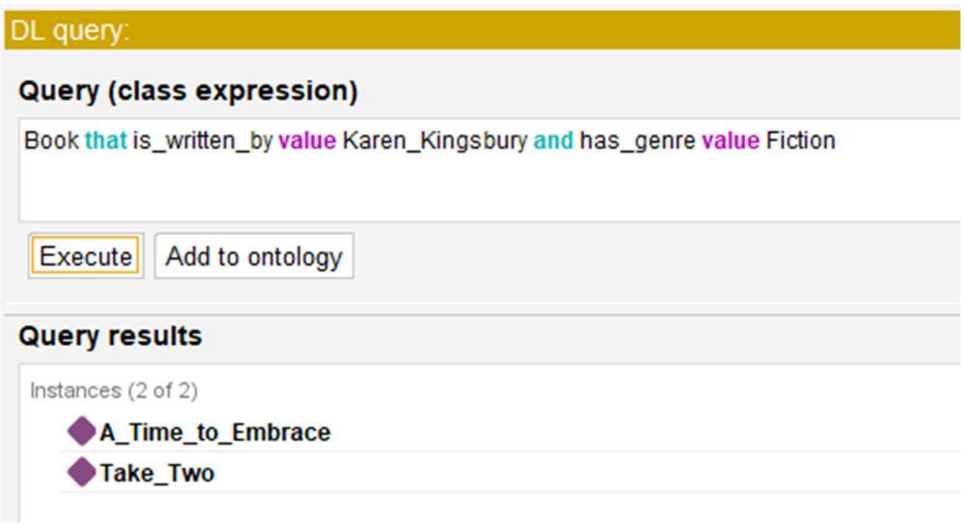


Esempio di individuo con properties annesse

È possibile anche interrogare l'ontologia tramite l'utilizzo di due tipologie di query, DL query e Snap SPARQL:



17 DL query con ricerca degli autori che hanno scritto "Inner circle"



16 DL query con ricerca di libri che hanno come autore "Karen Kingsbury" e come genere "Fiction"

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX table: <http://www.semanticweb.org/digio/ontologies/2023/5/untitled-ontology-11#>

SELECT ?Book ?Rating_count ?Average_rating
        WHERE {?Book table:Rating_count ?Rating_count.
        ?Book table:Average_rating ?Average_rating}
```

Execute

?Book	?Rating_count	?Average_rating
table:A_Time_to_Embrace	4179	4.35
table:Inner_Circle	7597	4.03
table:Reliquary	38382	4.01
table:Take_Two	6288	4.23

Snap SPARQL query con visualizzazione dei Libri, Numero di recensioni e Recensione media per libro

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX table: <http://www.semanticweb.org/digio/ontologies/2023/5/untitled-ontology-11#>

SELECT ?Book ?Number_of_pages ?Published_year
        WHERE {?Book table:Number_of_pages ?Number_of_pages.
                FILTER (?Published_year > 2000)
        ?Book table:Published_year ?Published_year}
```

Execute

?Book	?Number_of_pages	?Published_year
table:A_Time_to_Embrace	400	2006
table:Inner_Circle	220	2007
table:Take_Two	320	2009

Snap SPARQL query con visualizzazione dei Libri, Numero di pagine e Anno di pubblicazione dei libri pubblicati dopo il 2000

4.2 OwlReady2

Per la consultazione in Python dell'ontologia, è stata utilizzata la libreria Owlready2, attraverso la quale è stato possibile, interrogare l'ontologia precedentemente creata con il tool Protégé.

Nell'esecuzione del programma si potrà scegliere tra la visualizzazione delle classi, delle proprietà di oggetto, delle proprietà dei dati e della visualizzazione di alcune query d'esempio.

```
BENVENUTO NELLA BOOK-ONTOLOGY

Seleziona cosa vorresti esplorare:

1) Visualizzazione Classi
2) Visualizzazione proprietà d'oggetto
3) Visualizzazione proprietà dei dati
4) Visualizzazione query d'esempio
5) Exit Ontologia
```

Menù principale per l'esplorazione dell'Ontologia

```
Classi presenti nell'ontologia:

[BookOntology.owx.Author, BookOntology.owx.Book, BookOntology.owx.Costumer, BookOntology.owx.Genre, BookOntology.owx.Publisher, BookOntology.owx.Shop]
```

```
Vorresti esplorare meglio qualche classe in particolare?

1) Book
2) Author
3) Genre
4) Consumer
5) Publisher
6) Shop

Inserisci qui la tua scelta:    1

Lista di Libri presenti:

[BookOntology.owx.Book, BookOntology.owx.A_Time_to_Embrace, BookOntology.owx.Inner_Circle, BookOntology.owx.Reliquary, BookOntology.owx.Take_Two]
```

Visualizzazione delle Classi presenti nell'Ontologia

```
Proprietà d'oggetto presenti nell'ontologia:

[BookOntology.owx.can_collaborate_with, BookOntology.owx.collaborate_with, BookOntology.owx.has_genre, BookOntology.owx.is_located, BookOntology.owx.is_published_by, BookOntology.owx.is_searched_by, BookOntology.owx.is_used_by, BookOntology.owx.is_written_by, BookOntology.owx.publish, BookOntology.owx.searches, BookOntology.owx.uses, BookOntology.owx.writes]
```

Visualizzazione delle proprietà d'oggetto presenti nell'Ontologia

```
Proprietà dei dati presenti nell'ontologia:

[BookOntology.owx.Average_rating, BookOntology.owx.Id, BookOntology.owx.Number_of_pages, BookOntology.owx.Published_year, BookOntology.owx.Rating_count, BookOntology.owx.Re_evalutation, BookOntology.owx.Title]
```

Visualizzazione delle proprietà dei dati presenti nell'Ontologia

```
Query d'esempio:

-Lista di libri che presentano la categoria 'Romance':

[BookOntology.owx.A_Time_to_Embrace, BookOntology.owx.Inner_Circle, BookOntology.owx.Take_Two]


-Lista di libri che presentano la scrittrice 'Karen Kingsbury':

[BookOntology.owx.A_Time_to_Embrace, BookOntology.owx.Take_Two]


-Lista di libri che presentano la pubblicazione da 'Tor Books':

[BookOntology.owx.Reliquary]
```

Visualizzazione di qualche query d'esempio

```python
print("\nQuery d'esempio:")
print("\n-Lista di libri che presentano la categoria 'Romance':\n")
books = ontology.search(is_a=ontology.Book, has_genre=ontology.search(is_a=ontology.Romance))
print(books, "\n")
print("\n-Lista di libri che presentano la scrittrice 'Karen Kingsbury':\n")
books = ontology.search(is_a=ontology.Book, is_written_by=ontology.search(is_a=ontology.Karen_Kingsbury))
print(books, "\n")
print("\n-Lista di libri che presentano la pubblicazione da 'Tor Books':\n")
books = ontology.search(is_a=ontology.Book, is_published_by=ontology.search(is_a=ontology.Tor_Books))
print(books, "\n")
```

Esempio costruzione del codice per la formulazione delle query

Conclusioni

Il progetto rappresenta solo una piccola parte di ciò che potrebbe diventare con più risorse e dati. Uno dei possibili sviluppi futuri potrebbe essere ovviamente espandere il dataset notevolmente, affinché possano migliorare anche le capacita di predizione del knn oppure semplicemente allo scopo di poter offrire ancor più suggerimenti agli utenti. Un altro possibile miglioramento potrebbe essere l'implementazione di una interfaccia grafica e della possibilità di essere reindirizzati tramite link agli store online e filtrare per il prezzo più vantaggioso.

La repository con tutti i file del progetto: https://github.com/IvanDigioia/Progetto-Icon-2022-2023

La lista delle dipendenze si trova nel repository del progetto, nel file *requirements.txt*.

Il lavoro svolto è stato suddiviso equamente fra i due membri del gruppo.