RELAZIONE PROGETTO DI INGEGNERIA DELLA CONOSCENZA DI:

MARCO CAPPIELLO (722686)

ALBERTO DICEMBRE (715990)

MANUEL FASANELLA (717519)

UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

GESTIONE DI UNA VIDEOTECA: MOVIE RECOMMENDER (KNN), ONTOLOGY, CSP SOLVER, PATHFINDER

INDICE

INTRODUZIONE 2
ONTOLOGIA 4
KNN RECOMMENDER SYSTEM 7
RICERCA SU GRAFO 10
CSP 15
CONCLUSIONI 16

INTRODUZIONE

Nella seguente documentazione verrà descritta la ***relazione tecnica*** del progetto di **Ingegneria della Conoscenza Anno Accademico 2021-2022**.

Il *progetto* svolto consiste nella realizzazione di un recommender system di film basato su KNN prendendo come dataset: https://grouplens.org/datasets/movielens/.
L'idea è nata da un interessamento generale da parte del gruppo nel mondo dei recommender system per poi svilupparsi e allargarsi alla gestione di una videoteca.
Per questo progetto si è utilizzato il linguaggio Python (versione 3.10) e PyCharm come IDE.
Il nostro primo compito è stato quello di progettare una planimetria, individuare i vari percorsi e dividere i percorsi in nodi intermedi come mostrato in Figura 1.

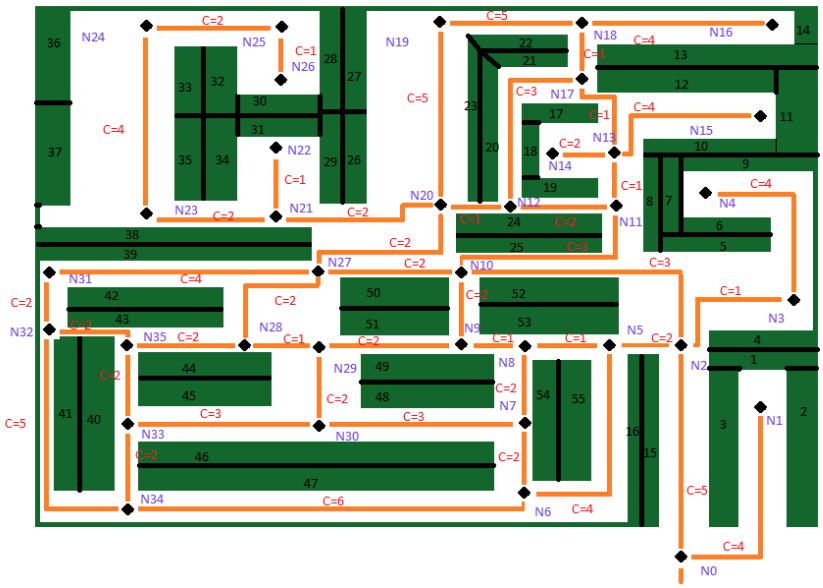

Figura 1. Planimetria Videoteca

Successivamente è stato assegnato un costo ad ogni arco sulla base della distanza in centimetri tra i due nodi collegati.
Dopodiché abbiamo convertito la planimetria in un grafo, individuando la radice nel nodo 0 come in Figura 2.

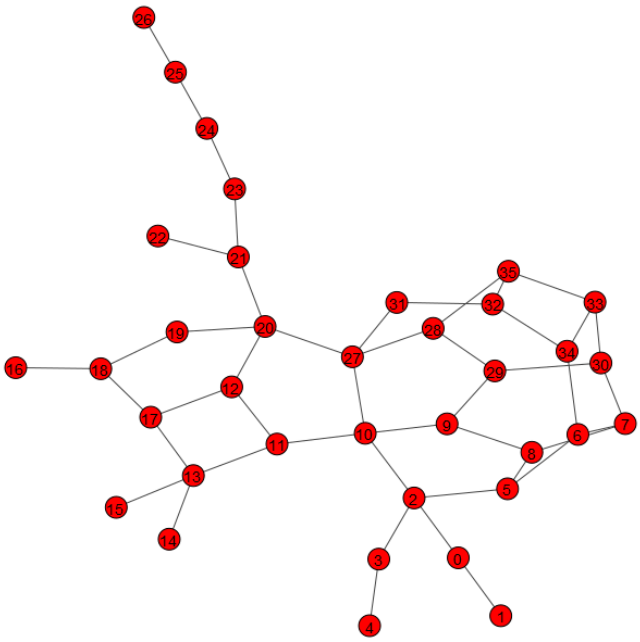

Figura 2. Grafo

Per simulare un sistema di prenotazione automatica abbiamo utilizzato un sistema di vincoli (film, ora, data), selezionato 24 film e programmato la disponibilità dei film nelle varie fasce orarie come in Figura 3.

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì
8	58559-1217-260-1196-1210-2571-3114-1240-589-597-587-356	41556-10-2-168492-115210-85788-858-1221-2023-1-2021-1213	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	58559-1217-260-1196-1210-2571-3114-1240-589-597-587-356
9					
10	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	41556-10-2-168492-115210-85788-858-1221-2023-1-2021-1213	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571
11					
12	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	41556-10-2-168492-115210-85788-858-1221-2023-1-2021-1213	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	58559-1217-260-1196-1210-2571-3114-1240-589-597-587-356
13					
14	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356
15					
16	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	41556-10-2-168492-115210-85788-858-1221-2023-1-2021-1213	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571
17					
18	58559-1217-260-1196-1210-2571-3114-1240-589-597-587-356	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	858-1221-2023-1-2021-1213-3114-1240-589-597-587-356	41556-10-2-168492-115210-85788-58559-1217-260-1196-1210-2571	41556-10-2-168492-115210-85788-858-1221-2023-1-2021-1213
19					

Figura 3. Programmazione Film

Come ultimo step preliminare abbiamo partizionato tutti i film presenti nel dataset (circa 193 mila) in 55 scaffali sparsi per la videoteca e abbiamo associato un nodo più vicino per ogni scaffale.
Siamo partiti con il progettare un'ontologia su "Protègè" contenente le classi "Movie", "Customer", "Salesperson", "Shelf", " Shop" e le loro relative proprietà ed è completata con gli individui per simulare la videoteca come mostrato in Figura 4.

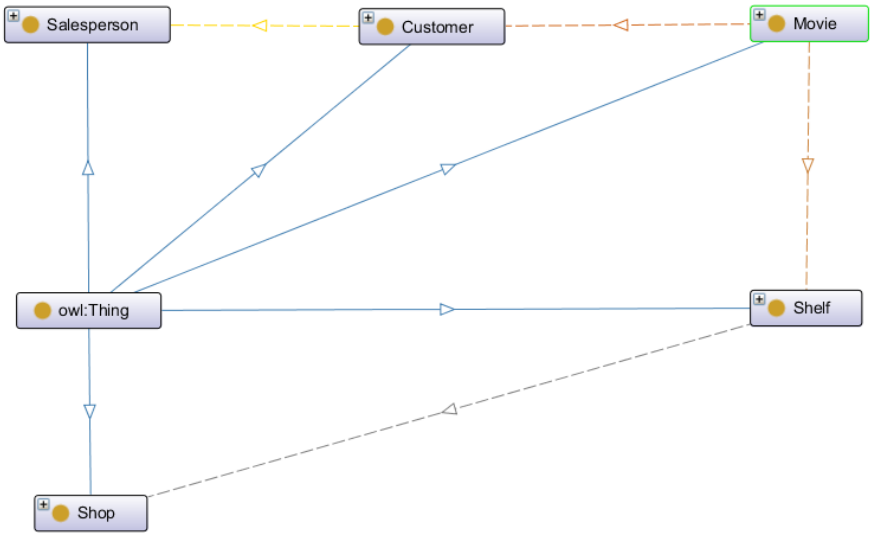

Figura 4. Ontologia su Protègè

ONTOLOGIA

Un’ontologia è una specifica del significato dei simboli usato in un sistema d'informazione, in particolare cosa viene modellato e il vocabolario utilizzato nel sistema. Grazie alla specifica formale si garantisce l'interoperabilità semantica, ovvero l’abilità da parte di diverse basi di conoscenze di collaborare a livello semantico, nel rispetto del significato dei simboli. L’ontologia è un insieme di conoscenze anche detta concettualizzazione di oggetti, concetti e relazioni fra di essi in una determinata area d’interesse.

Per creare l’ontologia si è usato il programma Protègè (https://protege.stanford.edu/). Oltre alla creazione delle varie classi e sotto-classi, sono state aggiunte proprietà di oggetto e proprietà di dati con un dominio e un relativo range. Inoltre è possibile interrogare l’ontologia effettuando delle query sia usando il tab “DL Query” che “SPARQL Query”. Il reasoner utilizzato è quello di default di Protègè ovvero HermiT.

Le varie entità definite nell’ontologia sono messe in relazione attraverso proprietà sia di oggetto che di dati. Di seguito (Figura 5) si riportano le proprietà di oggetto, ovvero delle proprietà in grado di mettere in relazione due individui di stessa classe oppure diversa.

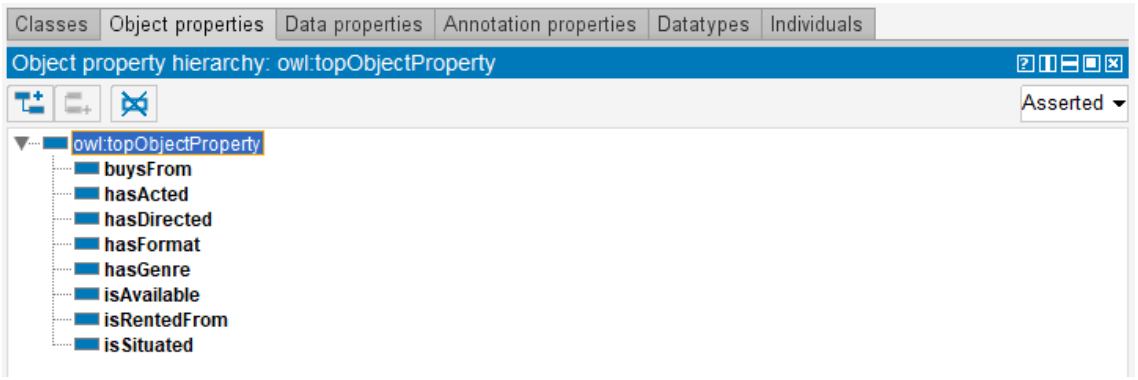

Figura 5. Object property

Oltre alle proprietà di oggetto, sono state anche definite le proprietà sui dati che mettono in relazione un individuo con il suo valore primitivo (int, string, etc), come mostrato in Figura 6.

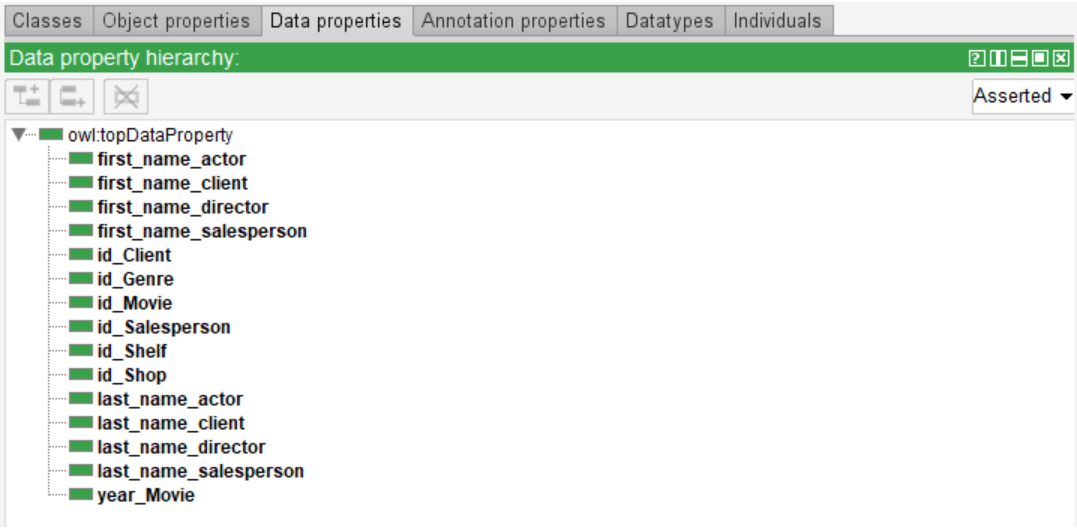

Figura 6. Data property

Successivamente si è passati a creare delle vere e proprie istanze per le entità. Ad esempio abbiamo le istanze per i film come “The Dark Knight” oppure “Terminator”, per gli attori vi è “Christian_Bale” oppure “Demi_Moore”, per individuare i generi abbiamo ad esempio “Action” oppure “Adventure” come mostrato in Figura 7.

Figura 7. Istanze per Actor, Genre e Format

Dopo aver creato gli individui, si è passati a formulare delle query sia in DL che in SPARQL per visualizzare la correttezza degli output. Di seguito si allegano in Figura 8 le query in DL e in Figura 9 quelle in SPARQL.

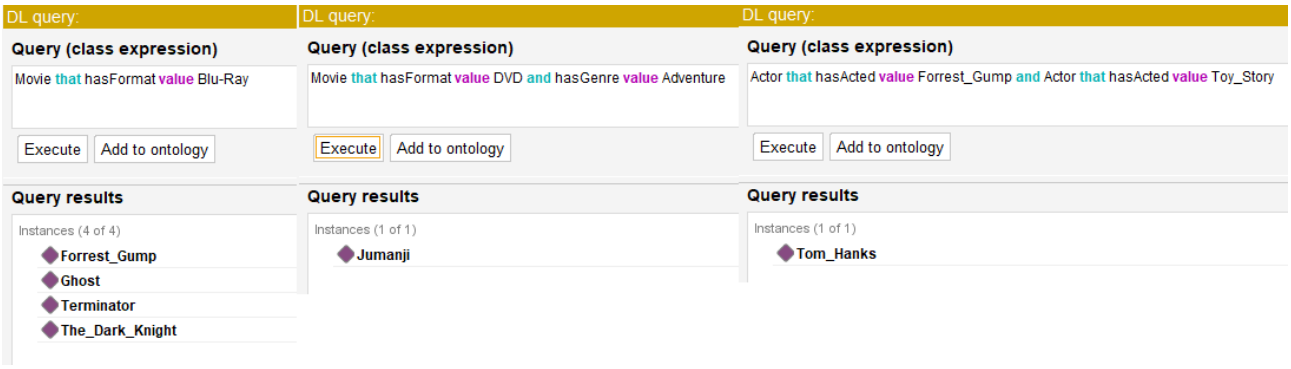

Figura 8. DL query

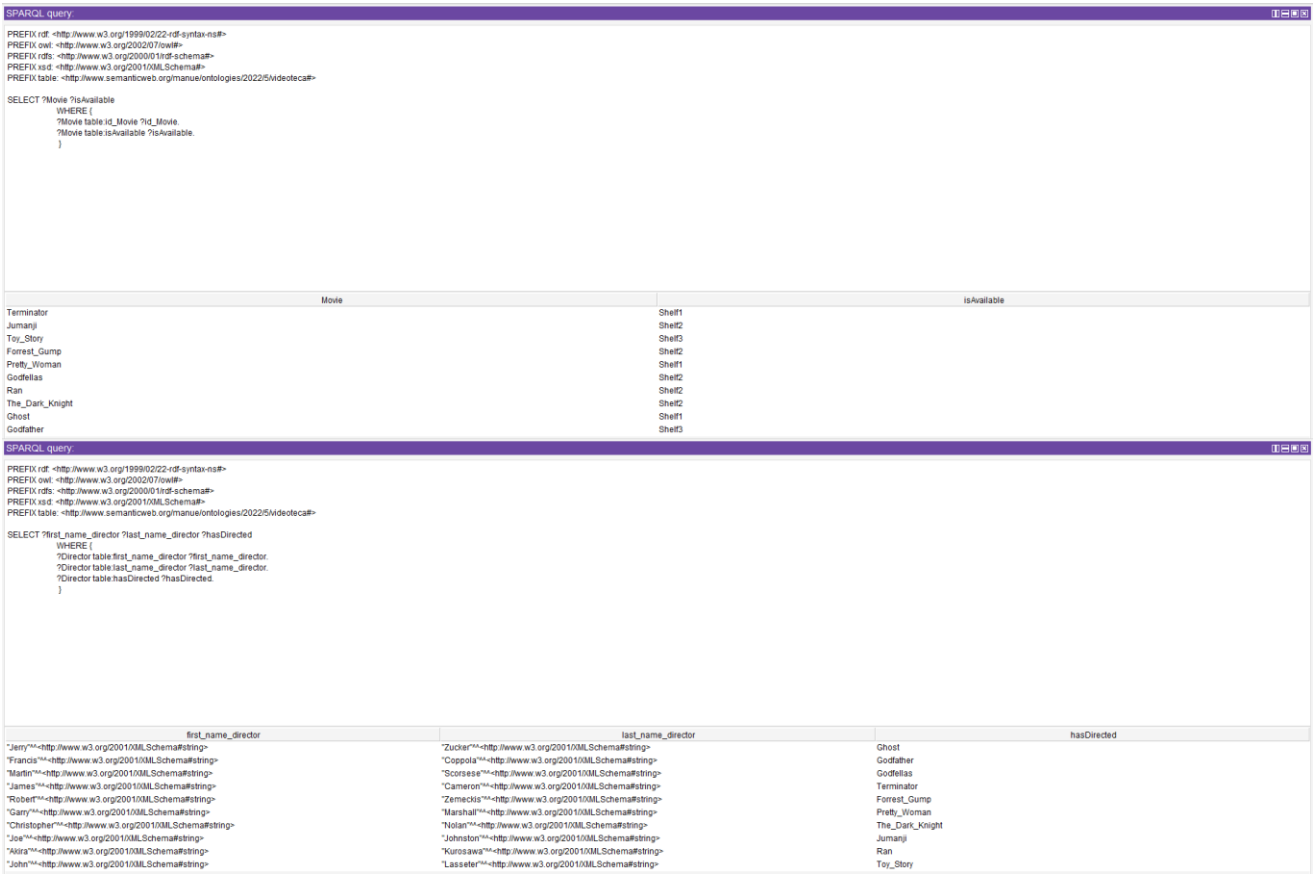

Figura 9. SPARQL query

Inoltre abbiamo creato il file “ontology” con il quale è possibile consultare l’ontologia direttamente in Python grazie alla libreria OwlReady2 e si sono svolte anche due query, come mostrato in Figura 10 e Figura 11.

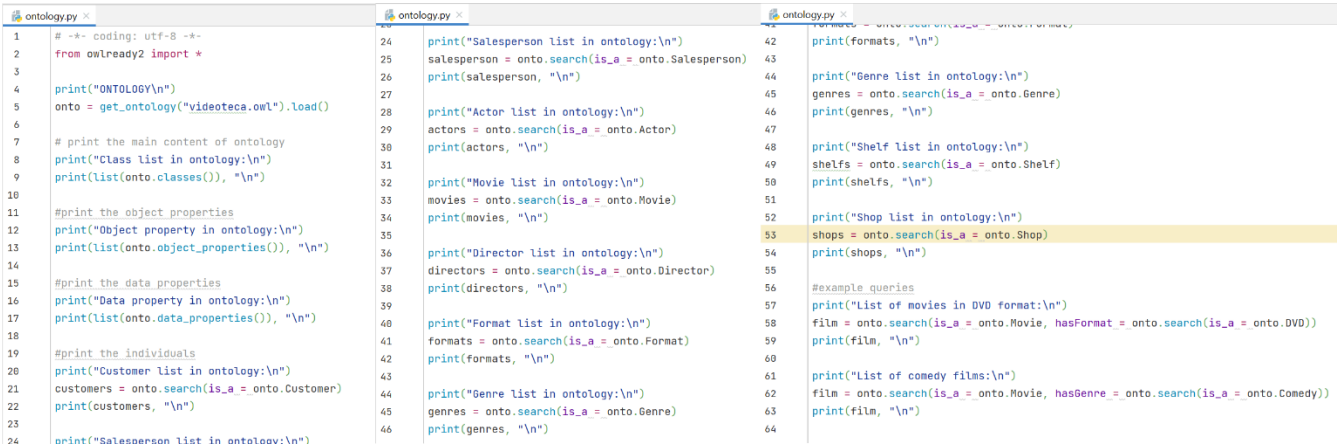

Figura 10. Codice Ontologia

```
ONTOLOGY

Class list in ontology:

[videoteca.Customer, videoteca.Salesperson, videoteca.Actor, videoteca.Movie, videoteca.Director, videoteca.Format, videoteca.Genre, videoteca.Sh

Object property in ontology:

[videoteca.buysFrom, videoteca.hasActed, videoteca.hasDirected, videoteca.hasFormat, videoteca.hasGenre, videoteca.isAvailable, videoteca.isRente

Data property in ontology:

[videoteca.first_name_actor, videoteca.first_name_client, videoteca.first_name_director, videoteca.first_name_salesperson, videoteca.id_Client, v

Customer list in ontology:

[videoteca.Customer, videoteca.Alberto_Dicembre, videoteca.Manuel_Fasanella, videoteca.Marco_Cappiello]

Salesperson list in ontology:

[videoteca.Salesperson, videoteca.James_Rumbaugh]
```

Figura 11. Output del file “ontology.py”

KNN RECOMMENDER SYSTEM

Per apprendimento si intende quella capacità di sfruttare l'esperienza passata per migliorare il comportamento.
L’apprendimento supervisionato genera un output a partire da un set di dati d'addestramento etichettati da usare come base per inferire nuova conoscenza.
Nel nostro progetto abbiamo utilizzato il KNN:
Il KNN o lazy learning è un algoritmo di apprendimento supervisionato che consiste nell'individuare i K esempi più vicini ad un dato in input da classificare.
Il dataset è diviso in due file in formato .csv, un file contenente una lista di film con corrispettivo Id e un file contenente una lista di id Utenti, l'id di un film, e il voto che l'utente ha assegnato al film.
L'algoritmo prenderà in input il valore di K da usare nel KNN e film preferito dall'utente e cercherà una corrispondenza nel dataset, attraverso un metodo di matching approssimato di stringhe della libreria “fuzzywuzzy”, e ne estrarrà l’id, come mostrato in figura 12.

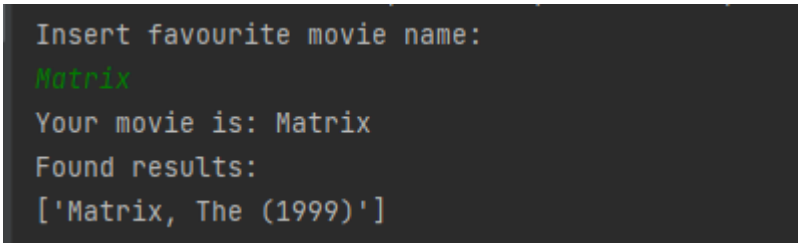

Figura 12. Corrispondenza Input-Dataset

Attraverso l'id del film vengono estratti tutti gli utenti che hanno valutato il film, omettendo quelli che hanno votato meno di 10 film in passato.
Attraverso la funzione della libreria skLearn.neighbors.NearestNeighbors:

`kneighbors`(*Film_Preferito*, *n_neighbors=K*)

Verranno inferiti i K utenti con gusti simili a quelli dell'utente attivo.
La similarità tra l'utente attivo e gli utenti nel dataset viene calcolata attraverso la similarità del coseno (Figura 13):

$$\frac{\sum_{k=1}^{n} A(k)B(k)}{\sqrt{\sum_{k=1}^{n} A(k)^2}\sqrt{\sum_{k=1}^{n} B(k)^2}}$$

Figura 13. Formula similarità del coseno

Gli utenti sono rappresentati come vettori di lunghezza N (N numero di film nel dataset) con voto da 1 a 5 in corrispondenza del film in posizione i, come mostrato in figura 14.

User-item rating matrix

	m1	m2	m3
u1	2	?	3
u2	5	2	?
u3	3	3	1
u4	?	2	2

Figura 14. Matrice user-item

La similarità è dunque il coseno dell'angolo compreso tra questi vettori (Figura 15):

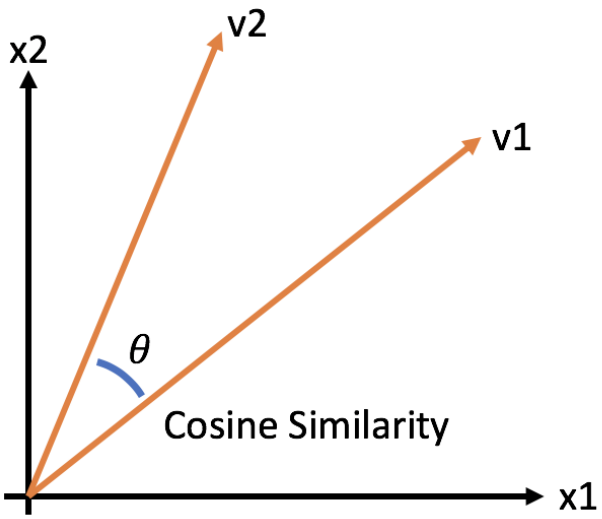

Figura 15. Angolo tra due vettori

Il programma restituirà quindi la lista di film apprezzati dagli utenti con gusti simili all'utente attivo ordinati per valore minimo di similarità del coseno, come mostrato in figura 16.

```
Insert favourite movie name:
Harry Potter
Your movie is: Harry Potter
Found results:
['Harry Potter and the Goblet of Fire (2005)']
Recommendations for : Harry Potter
1: Harry Potter and the Order of the Phoenix (2007), with distance of 0.2362881898880005
2: Harry Potter and the Prisoner of Azkaban (2004), with distance of 0.24916625022888184
3: Harry Potter and the Sorcerer's Stone (a.k.a. Harry Potter and the Philosopher's Stone) (2001), with distance of 0.27330267429351807
4: Harry Potter and the Chamber of Secrets (2002), with distance of 0.28429365158081055
5: Harry Potter and the Half-Blood Prince (2009), with distance of 0.3500920534133911
```

Figura 16. Output programma

RICERCA SU GRAFO

Il modulo graph_search ha lo scopo di ritrovare, nel grafo rappresentante la videoteca, i percorsi per arrivare agli scaffali contenenti i film consigliati dal modulo knn.

OBIETTIVI

- creazione di un grafo pesato relativo alla planimetria della videoteca:
  - la videoteca è composta da 55 scaffali, ognuno contenente 3520 film
  - a ogni scaffale corrisponde uno e un solo nodo (il più vicino secondo la planimetria). A ogni nodo però possono corrispondere più scaffali.
- una funzione "research", richiamata esternamente dal modulo "knn", che, ricevendo come parametro la lista dei film consigliati, e, per ogni film in essa presente, ritrovi e stampi a video i percorsi per arrivare dall'ingresso della videoteca fino allo scaffale contenente il primo film (nella prima iterazione), e poi di scaffale in scaffale per i restanti film. Ogni percorso deve avere un colore diverso per distinguerlo dagli altri.
    - Il metodo di ricerca utilizzato deve essere il Best-First-Search, in quanto utilizza un algoritmo ottimale e completo. Completo in quanto restituisce sicuramente una soluzione (se esiste), e ottimale in quanto la soluzione è la migliore.
        - Le complessità (spaziale e temporale) di questo algoritmo sono esponenziali rispetto al fattore di ramificazione.

IMPLEMENTAZIONE

Librerie utilizzate:

- igraph: https://igraph.org/python/, per la creazione e la stampa del grafo

Modulo “utils.py”:

Utilizzato per la traduzione dei nomi dei film della lista nei nodi del grafo.

- Lista di nomi dei film → Lista di id dei film → Lista di scaffali contenenti i film → Lista dei relativi nodi del grafo

Algoritmo di ricerca:

```
visited = []
queue = [[(start, 0)]]

while queue:
    queue.sort(key=path_cost)
    path = queue.pop(0)
    node = path[-1][0]
    visited.append(node)
    if node == end:
        return path

    else:
        adjacent_nodes = add_cost(graph, node, graph.neighbors(node))
        for (node2, cost) in adjacent_nodes:
            new_path = path.copy()
            new_path.append((node2, cost))
            queue.append(new_path)
```

Figura 17. Algoritmo di ricerca

VALIDAZIONE RISULTATI

Richiamando la funzione principale del modulo (research), e passandovi una lista di film consigliati generata dal modulo knn, visualizziamo la correttezza del risultato:

- Essendo gli scaffali 55, e il numero di film 193.600, ogni scaffale conterrà 3520 film, e il range di id di film contenuti in ognuno di essi è costituito da 3250 valori.
    - Es: lo scaffale 1 contiene i film con id 1-3249; lo scaffale 2 contiene i film con id 3250-6499, ecc.
    - La traduzione da id_film a id_scaffale avviene attraverso la funzione “get_rack()”.
- I nodi relativi agli scaffali sono invece verificabili secondo la tabella sottostante, che associa un insieme di scaffali a ogni nodo, secondo la struttura della planimetria.
    - La traduzione da id_scaffale a id_nodo avviene attraverso la funzione “get_node()”.

Nodo	Scaffale		Nodo	Scaffale		Nodo	Scaffale
1	1, 2, 3		13	/		25	32
2	15		14	17, 18, 19		26	28, 30
3	4, 5		15	10, 11, 12		27	50
4	6, 7, 9		16	13, 14		28	/
5	16, 55		17	21		29	49, 51
6	/		18	22		30	48
7	54		19	27		31	39, 42
8	/		20	23, 26		32	41
9	53		21	/		33	40, 46
10	25, 52		22	29, 21, 34		34	47
11	8		23	35, 37, 38		35	43, 44
12	20, 24		24	33, 36			

RISULTATO

Utilizzando la lista :

1: X-Men: Apocalypse (2016), ID: 122924

2: Captain America: Civil War (2016) ID: 122920

3: Wolverine, The (2013) ID: 103772

4: Rogue One: A Star Wars Story (2016) ID: 166528

5: Adventures of Tintin, The (2011) ID: 90746

Gli scaffali risultanti sono:

1: 122924 / 3520 = 35 → (35,tabella) → 23

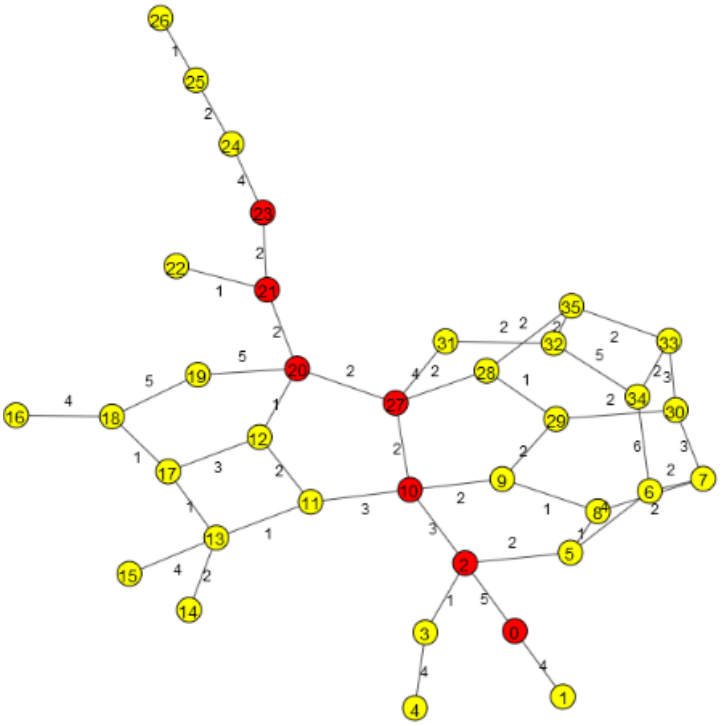

Figura 18. Path dall'entrata al primo film

2: 122920/ 3520 = 35 → (35,tabella) → 23

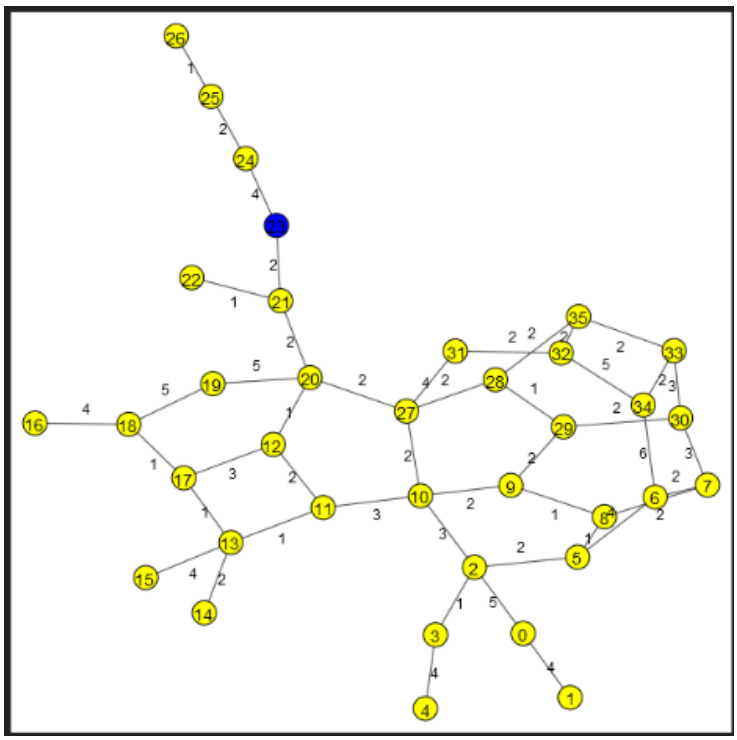

Figura 19. Path dal primo al secondo film

3: 103772 / 3520 = 30 → (30, tabella) → 26

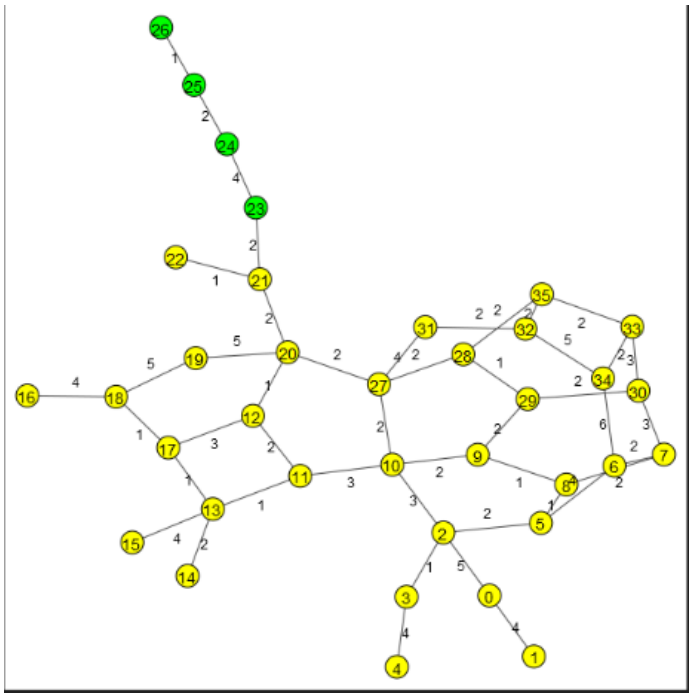

Figura 20. Path dal secondo al terzo

4: 166528 / 3520 = 48 → (48, tabella) → 30

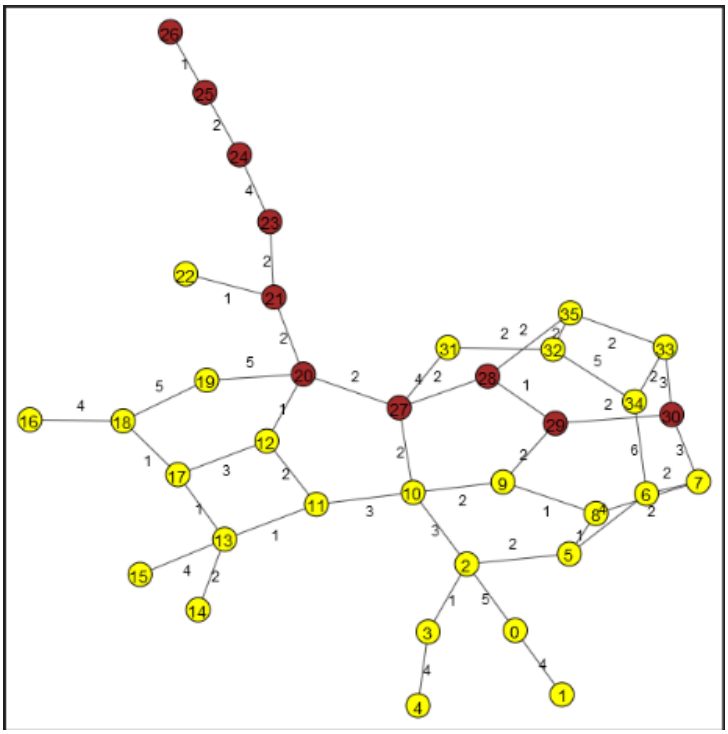

Figura 21. Path dal terzo al quarto film

5: 90746 / 3520 = 26 → (26, tabella) → 20

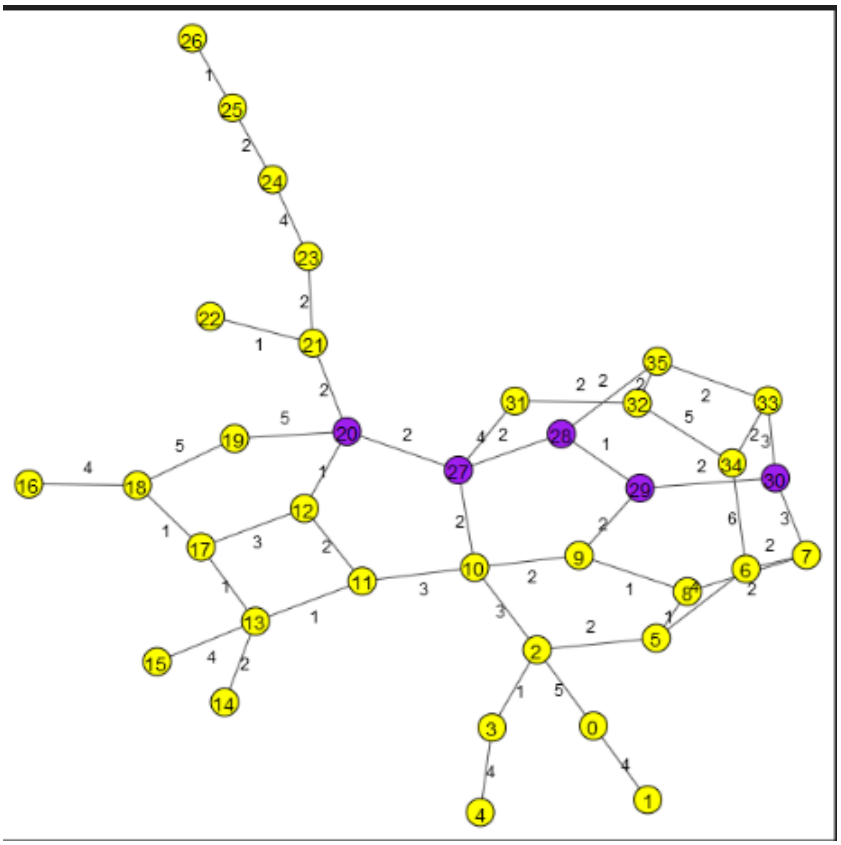

Figura 22. Path dal quarto al quinto film

CSP

Alcuni problemi nell'ambito dell'Intelligenza Artificiale possono essere risolti mediante features aventi domini e vincoli su di esse. Il problema di soddisfacimento dei vincoli (CSP) consiste nel trovare le assegnazioni totali che soddisfino tutti i vincoli, ed è composto da una tripla <X,D,C> dove X è un insieme di variabili, D è un insieme dei domini (uno per ogni variabile) e C è un insieme di vincoli che specificano combinazioni di valori ammesse. Un vincolo sulle variabili è una restrizione dei valori che quelle variabili possono assumere simultaneamente. Il grado della variabile è il numero di vincoli a cui essa è sottoposta. Un CSP presuppone un assegnamento iniziale, per poi proseguire assegnando via via valori alle variabili ancora libere. Una soluzione a un problema CSP è un'assegnazione completa e consistente, in altre parole una soluzione è un modello.
La libreria che è stata utilizzata è "Constraint" che consente di creare dei csp e di generare tutti i modelli. Nel nostro caso si è usata per gestire la videoteca in quanto abbiamo imposto dei vincoli sugli orari e sui film disponibili per ogni giornata lavorativa. Una volta generate tutte le soluzioni verrà richiesto all'utente di inserire

un giorno, un orario e un film tra quelli aggiunti al dominio.
Successivamente verrà ricercato un modello che ha come valori per le variabili quelli inseriti dall'utente.
La libreria dispone di 3 solver: Backtracking, Recursive Backtracking e Minimum Conflicts. Quello di default è Backtracking (Figura 17?) e abbiamo scelto di utilizzarlo per la intuitiva natura del nostro problema.

```
BT(Level)
    If all variables assigned
       PRINT Value of each Variable
       RETURN or EXIT (RETURN for more solutions)
                  (EXIT for only one solution)
    V := PickUnassignedVariable()
    Variable[Level] := V
    Assigned[V] := TRUE
    for d := each member of Domain(V)
       Value[V] := d
       OK := TRUE
       for each constraint C such that
                V is a variable of C
                and all other variables of C are assigned.
          if C is not satisfied by the current set of assignments
              OK := FALSE
       if(OK)
           BT(Level+1)
    return
```

Figura 23 Algoritmo backtracking

CONCLUSIONI

La lista delle dipendenze del progetto è presente nel file "requirements.txt" caricato su repository GitHub. Il lavoro è stato svolto in pair-programming coordinandoci su Discord per tutti i giorni della durata del progetto. Il nostro progetto cerca di essere originale e allo stesso tempo il più trasversale spaziando tra diversi argomenti del corso di ICON.