DOCUMENTAZIONE PROGETTO

CORSO DI INGEGNERIA DELLA CONOSCENZA A.A 2021-2022

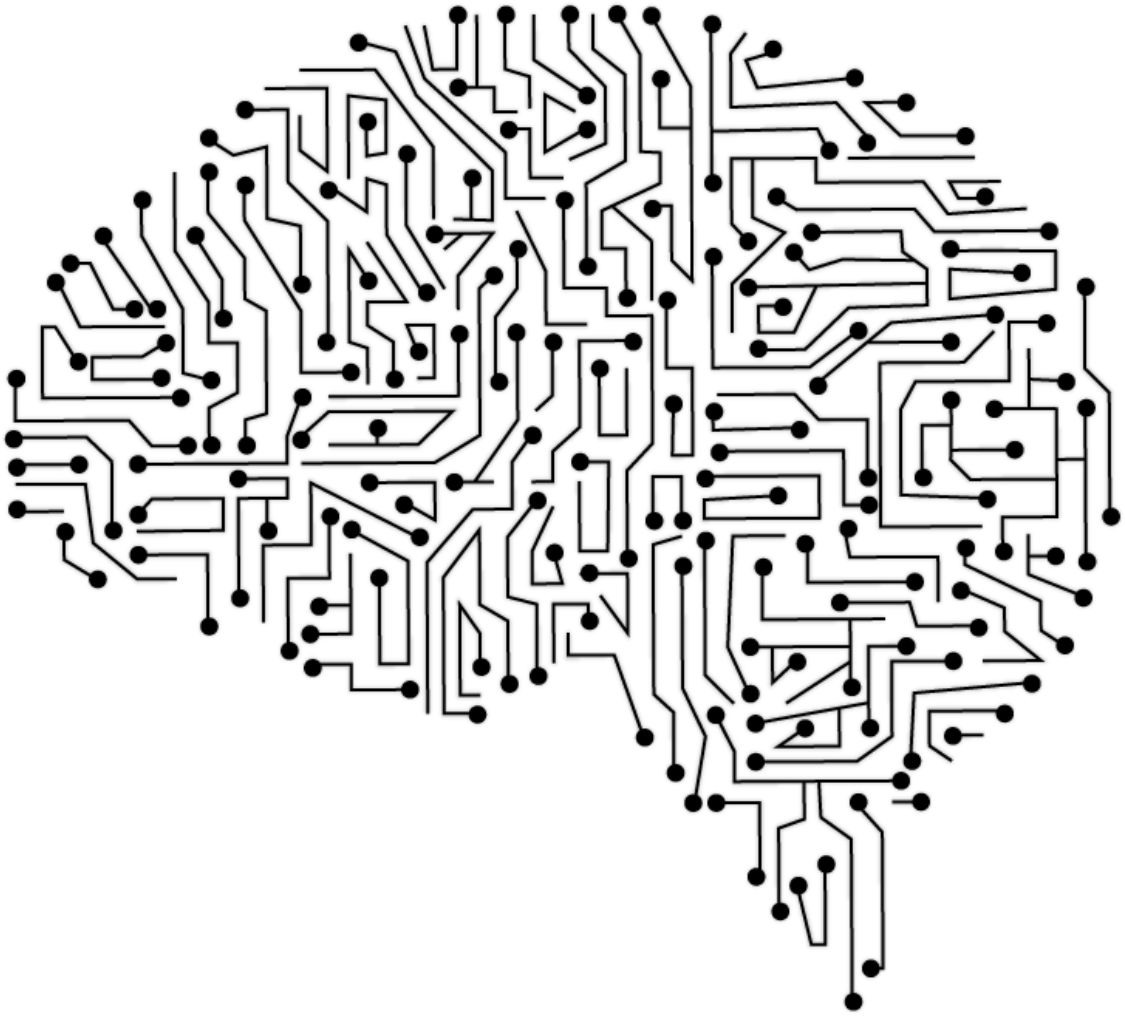
COSTRUZIONE DI UNA BASE DI CONOSCENZA, UTILIZZO DI TECNICHE DI APPRENDIMENTO SUPERVISIONATO E STRATEGIA DI RICERCA E OTTIMIZZAZIONE BASATA SU GRAFI PER:

1. *CONSIGLIARE UN FILM A PARTIRE DALLA PERSONALITÀ E DELL'UMORE DELL'UTENTE.*
2. *CALCOLARE IL PERCORSO MIGLIORE PER SPEDIRE IL FILM DAL PUNTO VENDITA AD UNO DI RITIRO.*

A CURA DI: LATINI DANIELE

d.latini1@studenti.uniba.it - MAT. 718150

LINK GITHUB: https://github.com/0xBush1do/Icon2122



1. Obiettivi di progetto

L'obiettivo di questo progetto è stato quello di realizzare un sistema intelligente che possa consigliare all'utente uno (o più) film sulla base della propria personalità e del proprio umore. Il secondo obiettivo consente invece ad un robot di consegne di valutare il percorso migliore che può seguire per consegnare il film "consigliato" dal punto vendita al punto di ritiro selezionato dall'utente stesso.

2. Strumenti utilizzati

Il progetto è stato realizzato principalmente nel linguaggio di programmazione Python, che offre diverse librerie utili per la gestione dei datasets e per l'utilizzo di tecniche di apprendimento supervisionato (K-NN). In particolare, sono state utilizzate le librerie che seguono:

- Pandas: Utile per la gestione dei datasets di grandi dimensioni grazie all'utilizzo dei Dataframe.
- Pyswip: Utile per poter creare, modificare e interrogare una base di conoscenza in prolog.
- Scikit-learn e Scipy: Utili per lavorare con il modello di classificazione K-NN.

Oltre al linguaggio di programmazione Python, è stato utilizzato il linguaggio Prolog per creare una base di conoscenza ed effettuare delle query al fine di comprendere i gusti dell'utente sulla base della propria personalità e del proprio umore.

3. Introduzione

Il progetto prevede l'utilizzo di due datasets: *movies.csv* e *ratings.csv*:

- Movies.csv presenta gli attributi: "movieId", "title", "genres" .
- Ratings.csv presenta gli attributi: "userId", "movieId", "ratings", "timestamp".

La struttura del progetto consta di tre fasi:

1. Creazione di una base di conoscenza in Prolog che ha lo scopo di scegliere la categoria di film più vicina alla personalità dell'utente e al suo umore tramite una query.
2. Nella seconda fase si utilizza la categoria di film in output per filtrare "movies.csv" per "genres" in modo tale che il sistema di raccomandazione possa inferire unicamente film della categoria scelta.
3. Infine, dopo aver consigliato i film da acquistare, ne viene simulata la spedizione per mezzo di un robot di consegne, il quale calcolerà il percorso ottimale per raggiungere il punto di ritiro selezionato dall'utente.

4. Prima fase: Ragionamento

Per la costruzione della base di conoscenza è stato utilizzato il linguaggio di programmazione Prolog, questo si basa sul calcolo dei predicati del primo ordine. La sintassi è limitata, però, alle cosiddette clausole di Horn che sono disgiunzioni di letterali del primo ordine, quantificate con al più un letterale positivo. Tramite la libreria PySwip è stato possibile gestire la creazione e l'interrogazione della base di conoscenza di modo che l'utente possa ottenere la categoria di film

consigliata sulla base della propria personalità (espressa sotto forma di acronimo: *Calmo, Estroverso, Coscienzioso, Serio* -> **cecs**) e sulla base del proprio umore (*happy* o *sad*) con il relativo livello (da 1 a 100).

5. Seconda fase: Recommender System – K-NN

Il K-Nearest Neighbors, noto anche come K-NN, è un algoritmo di apprendimento supervisionato che utilizza la prossimità per effettuare classificazioni o previsioni sul raggruppamento di un singolo punto di dati. Il K-NN fa parte della famiglia dei modelli di "apprendimento pigro" e consiste nell'individuare i K esempi più vicini ad un dato da classificare.

Preliminarmente è stato effettuato il preprocessing sui datasets *movies.cvs* e *ratings.csv*:

- Entrambi i datasets sono stati ridotti, filtrando in base alla categoria consigliata all'utente.
- Sono state eleminate le colonne ritenute superflue.
- Sono stati omessi, ai fini della raccomandazione, i film che hanno ricevuto meno recensioni di un certo valore di threshold. Questo ha permesso di ottenere un dataframe di soli film maggiormente recensiti.

```
  userId  movieId  rating          title  genres  RatingCount
       1      223     3.0  Clerks (1994)  Comedy          104
       9      223     4.0  Clerks (1994)  Comedy          104
      18      223     4.0  Clerks (1994)  Comedy          104
      19      223     3.0  Clerks (1994)  Comedy          104
      33      223     4.0  Clerks (1994)  Comedy          104
```

Superata la fase preliminare, l'algoritmo di raccomandazione acquisisce un film casuale estratto dal dataframe preprocessato e il parametro K di neighbors tramite la funzione di "sklearn.neighbors" *NearestNeighbors*:

```
knn.kneighbors(popular_rand_movie, n_neighbors=K)
```

A questo punto il modello identifica la distanza tra il film randomico e i k vicini più prossimi ad esso:

```
<+> FILM CONSIGLIATO PER TE <+>
1: Ferris Bueller's Day Off (1986), con distanza: 0.4804168438308247
2: Monty Python's Life of Brian (1979), con distanza: 0.5998609269889437
3: Meet the Parents (2000), con distanza: 0.6259918111422607
4: Clerks (1994), con distanza: 0.6547348977359602
5: Happy Gilmore (1996), con distanza: 0.6586309582935352
```

CONSIDERAZIONI SULLE METRICHE UTILIZZATE

La metrica utilizzata per calcolare la distanza tra un film selezionato e i suoi K neighbors è la similarità del coseno, che consiste nel misurare le differenze tra due vettori in un dato spazio.

APPLICATO AL CASO IN QUESTIONE, SUPPONENDO DI AVERE UNA TABELLA DEL TIPO:

title	Id1	Id2	Id3
Iron Man	4	3	5
Fast and Furios	5	5	1

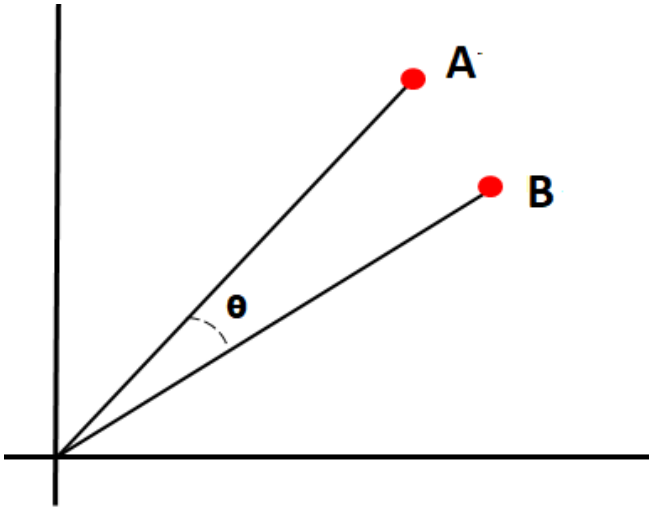
SI POSSONO RAPPRESENTARE I DUE FILM COME DUE VETTORI SEPARATI:

$$\vec{b} = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \qquad \vec{c} = \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix}$$

LA SIMILARITÀ DEL COSENO MISURERÀ LA SOMIGLIANZA TRA I DUE VETTORI SECONDO LA FORMULA:

$$similarity = \cos\theta = \frac{\vec{b}.\vec{c}}{\|\vec{b}\|\|\vec{c}\|}$$

GRAFICAMENTE:



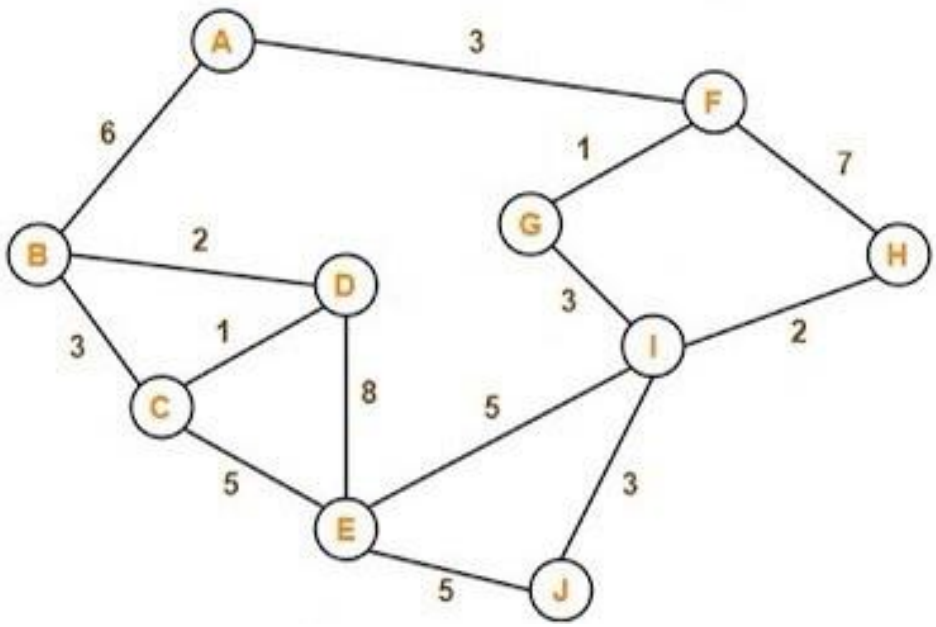
---

6. Terza fase: calcolo del percorso migliore con A*

L'algoritmo A* è un algoritmo di ricerca informata che combina la *Lowest-Cost-First search* con la *Best-First search* euristica. In particolare, nella selezione di un percorso "p" da espandere, A* utilizza una stima del costo di p sommata all'euristica h(p).

$$f(p) = cost(p) + h(p)$$

Di seguito viene riportata la piantina dei punti di consegna, sottoforma di grafo pesato:



Il punto vendita è il nodo etichettato con la lettera A, mentre l'utente può scegliere liberamente qualsiasi altro punto di ritiro. Una volta selezionato il suddetto punto, l'algoritmo troverà il percorso migliore per raggiungerlo, come mostrato di seguito:

```
> Il nostro servizio di delivery intelligente può spedire i film consigliati direttamente a casa tua <
> Se decidi di continuare, visualizzerai il percorso più breve che il nostro robot di consegne può - <
> - percorrere per raggiungere la tua abitazione <
> Premi 'c' per continuare, 'x' per non richiedere il ritiro. <
> c
<+> Importo la piantina del tuo quartiere <+>
<+> Il nostro punto vendita si trova in 'A', indica il punto di ritiro che preferisci (da 'B' a 'J') <+>
> j
Percorso migliore trovato: ['A', 'F', 'G', 'I', 'J']
<+> FINE SERVIZIO <+>
```

CONSIDERAZIONI SULL'EURISTICA UTILIZZATA E SUI COSTI

Il costo associato ad ogni arco è un valore numerico (intero, positivo) compreso tra 1 e 10, assegnato in base al livello di traffico presente; dunque, gli archi con valori più bassi rappresentano strade poco trafficate e viceversa.

La funzione euristica che è stata utilizzata è la *Distanza di Manhattan* (o *geometria del taxi*), che in maniera formale si definisce come:

"Dati P1, P2 punti nello spazio euclideo con coordinate rispettivamente (x1, y1) e (x2, y2), la distanza L1 si calcola come la somma delle lunghezze delle proiezioni sugli assi cartesiani dei segmenti che congiungono i due punti."

In formule:

$$L_1(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|.$$

Tale distanza è stata rapportata alla velocità media di percorrenza per ogni arco che collega il nodo di partenza ad un nodo *goal*, al fine di avere una stima più verosimile della distanza (in termini temporali) tra il nodo di partenza 'A' e il nodo di arrivo scelto dall'utente.

FINE.