Laurea Triennale in Informatica - Università degli studi di Salerno- Corso di Fondamenti di Intelligenza Artificiale – Prof F. Palomba

TV Series' World

Toriello Matteo-Serretiello Simone

https://github.com/SimoneSerretiello/Tv-Series-World

SOMMARIO

--------1. Definizione del problema----------

--------2. Descrizione dell'agente----------

---------- 2.1 Obiettivi----------

----------2.2 Specifica P.E.A.S----------

----------3. Analisi del problema----------

--------4. Raccolta dei dati----------

---------- 4.1 Scelta del dataset----------

----------4.2 Grafici e Statistiche desrittive----------

--------- 5. Passi dell'algoritmo----------

--------6. Parametri----------

-------- --------6.1 Inizializzazione della popolazione----------

----------6.2 Codifica della popolazione----------

----------6.3 Calcolo della funzione di fitness----------

----------6.4 Selezione----------

----------6.5 Crossover----------

----------6.6 Mutazione----------

----------6.7 Stopping Condition----------

--------- 7. Conclusioni----------

Laurea Triennale in Informatica - Università degli studi di Salerno- Corso di Fondamenti di Intelligenza Artificiale – Prof F. Palomba

1. Definizione del problema

Il processo di sviluppo delle serie televisive è frutto di un percorso di adattamento e di evoluzione in un contesto tecnologico e competitivo sempre più dinamico. Le strutture formali e narrative delle serie TV sono progredite nel corso del tempo, come anche la loro popolarità, in qualsiasi fascia d'età, e la distribuzione sulle piattaforme di streaming online. Tuttavia, la maggior parte di queste piattaforme dispone di un sistema di suggerimento di visione dei contenuti limitate alle serie tv disponibili su di esse, limitando di molto la fruizione di possibili prodotti di interesse per l'utente. Individuato il problema, si è deciso di realizzare un sistema con l'ausilio di tecniche di intelligenza artificiale, che riesca a migliorare questa limitazione da parte delle piattaforme streaming estendendo il suggerimento a tutti i prodotti presenti sul mercato.

-Torna al sommario-

2. Descrizione dell'agente

2.1 Obiettivi

Lo scopo del progetto, quindi, è quello di realizzare un agente intelligente capace di suggerire un insieme di prodotti personalizzati, sulla base dei generi delle serie tv già viste.

2.2 Specifica PEAS

PERFORMANCE: La misura di performance dell'ambiente è la capacità di consigliare le serie tv che l'utente intende vedere.

ENVIRONMENT: L'ambiente è il frame con cui l'utente interagisce. Esso è:

- *Statico:* l'agente agisce soltanto sull'insieme di telefilm corrente;
- *Completamente osservabile:* in ogni momento l'agente conosce le caratteristiche delle serie tv;
- *Singolo:* un unico agente opera nell'ambiente preso in esame;
- *Episodico:* le decisioni future dell'agente non sono influenzate dalle scelte passate dell'utente;
- *Stocastico:* lo stato successivo non è possibile determinarlo in quanto sono presenti più componenti di probabilità, oltre ad elementi casuali;
- *Discreto:* l'ambiente fornisce un numero limitato di percezioni ed azioni;

ACTUATORS: Gli attuatori dell'agente consistono nella lista dei prodotti consigliati sulla base delle preferenze;

SENSORS: i sensori consistono nella lista in input delle serie tv già viste, oltre al bottone "Invia" presente nella versione grafica del progetto.

-Torna al Sommario-

3. Analisi del problema

Formulato il problema, era chiara la necessità di un algoritmo di **ottimizzazione.**
Data l'assenza di relazioni evidenti tra gli elementi del nostro dataset è stata scartata l'ipotesi di rappresentare il problema come un grafo.
Avendo un ambiente singolo le soluzioni mediante la teoria dei giochi non erano applicabili.
Inoltre, la mancanza di dataset con una specifica variabile target che ci permettesse di implementare un algoritmo di machine learning ci ha portato a scartare quest'ipotesi.
Quindi la scelta più promettente è stata quella di utilizzare un algoritmo genetico.
Il nostro obiettivo, infatti, è quello di suggerire un insieme di serie tv quanto più possibile simile ai prodotti inseriti in input dall'utente.

-Torna al Sommario-

4. Raccolta dei dati

4.1 Scelta del dataset

Per la scelta implementativa fatta, era necessario l'uso di un dataset e le possibili strade da seguire erano due:

1. **Creare** un dataset da zero, inserendo manualmente quanti più prodotti audiovisivi possibili con le relative caratteristiche;
2. **Cercare** sulla rete un dataset già formato e adeguarlo alle nostre esigenze;

La prima opzione era soggetta a limitazioni non trascurabili:

- Non affidabilità dei dati;
- Lentezza nel creare dati per avere un numero di prodotti considerevole per il testing dell'algoritmo;

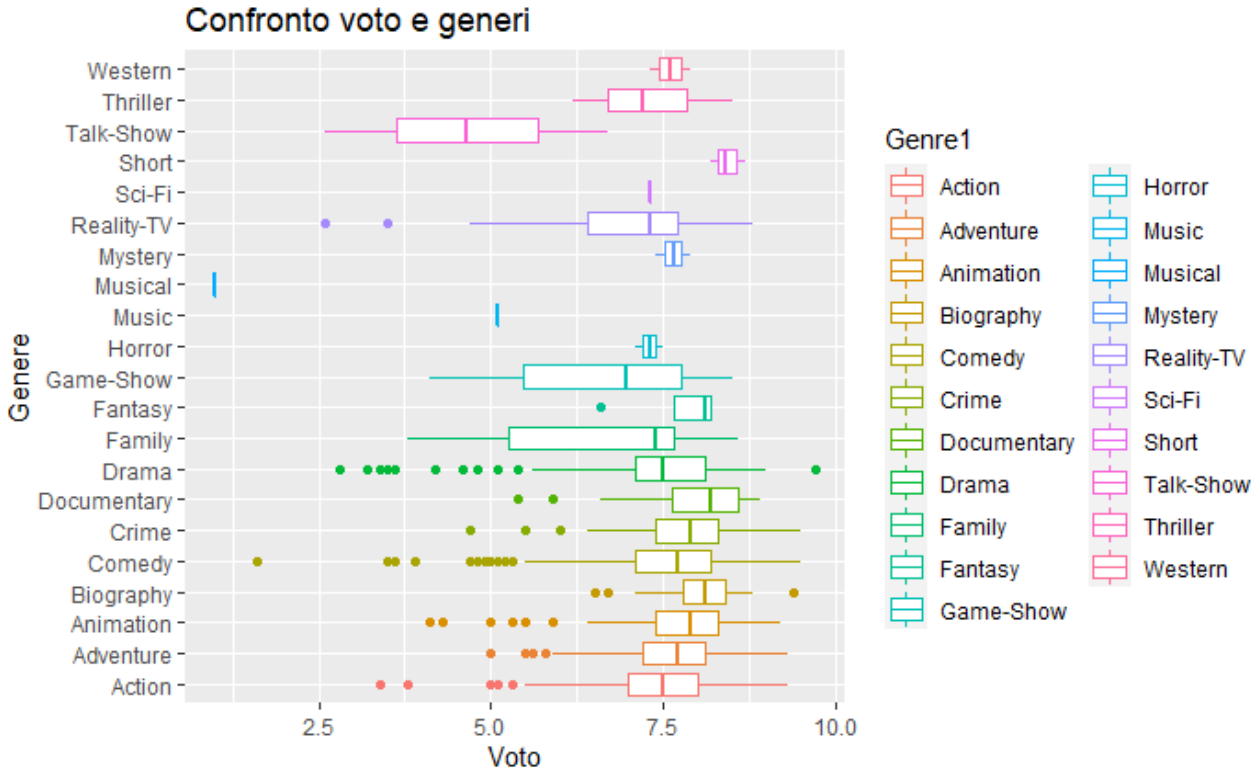
Si è deciso, pertanto, di procedere con la seconda soluzione, cercando in rete un dataset già pronto. Dopo svariati tentativi, la nostra attenzione si è posata su un dataset proveniente da https://www.kaggle.com/harshitshankhdhar/tv-series-dataset.

4.2 Grafici e Statistiche descrittive

Una volta ottenuto il dataset *IMDB TV Series* l'abbiamo caricato in una sessione R in modo da studiarne le variabili e gettare le basi, tramite statistiche descrittive, per la progettazione della funzione di valutazione del nostro algoritmo genetico.
Non appena caricato il file CSV (*Comma Separated Value*) ci siamo trovati di fronte a un dataset con 1283 osservazioni e 17 variabili. La nostra attenzione è stata subito catturata dalla variabile *Genre*, non utilizzabile in quanto ogni osservazione (o quasi) conteneva 3 generi (*es. Action, Drama, Thriller*). Si è deciso quindi di "*splittare*" la variabile in Genre1, Genre2 e Genre3 eliminando poi gli *NA* risultanti da questa operazione.

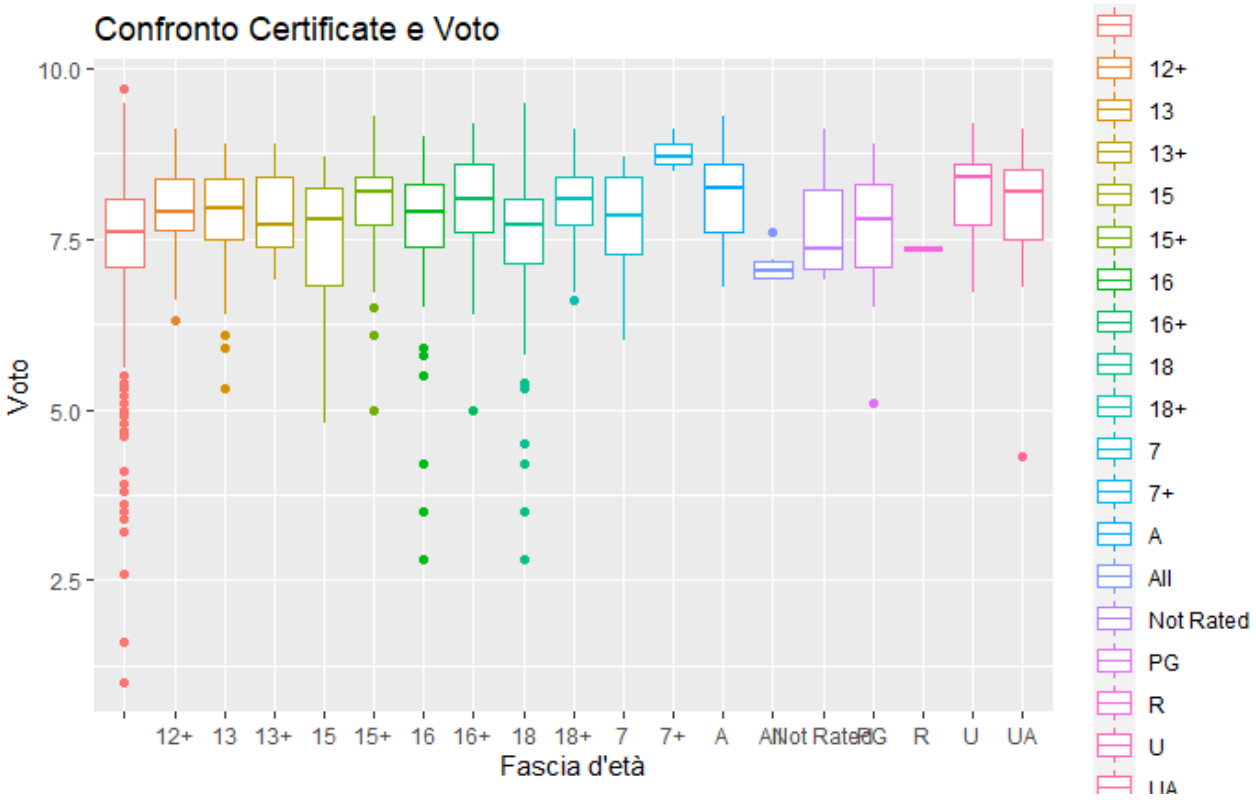
Assumendo che il Genre1 sia il più rilevante e basandoci sulla variabile IMDB_Rating, ovvero il voto della serie, abbiamo prodotto le prime statistiche descrittive tra le variabili:

- Abbiamo misurato l'indice di Cramèr per le variabili voto e genere che è risultato pari a 0.33, una correlazione tutto sommato alta visibile anche dal boxplot:



Dal grafico risultano molti punti distanti dalla mediana della rispettiva distribuzione ma questo era previsto in quanto, parlando di serie tv, i voti possono variare di molto in base al telefilm in questione.

- Abbiamo misurato l'indice di Cramèr per le variabili voto e certificate (variabile categoriale che indica la fascia d'età adatta per gli spettatori) pari a 0.19, correlazione bassa e ciò si nota anche nel grafico:



-Torna al Sommario-

5. Passi dell'algoritmo

1. Prendere in input serie tv inserite dall'utente;
2. Costruire una popolazione iniziale;
3. Codificare la popolazione iniziale in una la lista di 0 e 1, in cui 0 equivale a "*non consigliare"* e 1 equivale a *"consigliare";*
4. Calcolare il valore di fitness;
5. Selezionare un sottoinsieme di individui in base al loro punteggio di fitness;
6. Gli individui selezionati vengono fatti accoppiare con una determinata probabilità di crossover (pari a 0.8);
7. Gli individui ottenuti vengono fatti mutare con una probabilità di mutazione pari a 0.01 e si ritorna al passo 4;
8. Quando la dimensione della popolazione è minore di un certo valore oppure allo scadere del tempo di esecuzione l'algoritmo termina e restituisce la generazione attuale che rappresenta la migliore.

-Torna al Sommario-

6. Parametri

6.1 Inizializzazione della popolazione

La prima implementazione da noi proposta si basava su un'inizializzazione casuale della popolazione creando un numero di individui compreso tra 3 e 15 serie tv.
Problema: avendo un dataset ben popolato e non tenendo conto della lista di serie tv inserite dall'utente, i passi successivi dell'algoritmo erano eccessivamente influenzati dalla componente casuale ottenendo spesso risultati incerti.
Soluzione: si estrae un sottoinsieme di serie tv con i generi uguali a quelli delle serie inserite in input dall'utente. Sulla base di questo si scelgono tra le 3 e le 15 serie tv in modo casuale e si passano alla funzione di fitness.

6.2 Codifica della popolazione

La popolazione viene codificata casualmente in una lista di 0 e 1.

6.3 Calcolo funzione di fitness

La funzione di fitness che utilizza l'algoritmo si basa su:

- Generi simili: abbiamo predisposto un dizionario python per tenere traccia dei generi simili tra loro. La chiave è un genere univoco del dataset e i corrispondenti valori sono i generi simili ad essa.

Ad esempio:

Horror->Thriller, Mistery, Crime

Se i generi della serie presa in considerazione NON rientrano tra i generi simili a quelli delle serie tv dell'utente, il punteggio di fitness viene ridotto del 50%;

- IMBD Rating: la funzione di fitness aggiunge al punteggio di fitness 0.3 se il voto della serie presa in considerazione è maggiore di quello della serie inserita dall'utente;
- Overview: grazie ad una lista di parole chiave, la funzione di fitness confronta le sinossi delle 2 serie e per ogni keyword in comune aggiunge al punteggio di fitness 0.5.

6.4 Selezione

Inizialmente, la selezione avveniva tramite *Roulette Wheel Selection*.

Problema: come ben noto, questo metodo di selezione porta ad una convergenza prematura soprattutto in popolazioni piccole come la nostra e quindi questa implementazione è stata scartata.

Soluzione: abbiamo implementato il metodo di selezione *Two Tournament* che si è adattato particolarmente bene al nostro problema.

6.5 Crossover

Il problema in analisi ci ha portato ad implementare e valutare due tipi di Crossover: *Uniform Crossover* e *Single Point Crossover*. Per semplicità e per la qualità dell'output restituito dalla versione del progetto contenente il Single Point Crossover abbiamo deciso di utilizzare quest'ultima con probabilità pari all'80%.

6.6 Mutazione

Per semplicità abbiamo deciso di implementare il tipo di mutazione *Random Resetting*, con probabilità di mutazione pari a 1%.

6.7 Stopping condition

Basandoci sull'immediata risposta che l'utente vuole ricevere, l'algoritmo termina entro massimo 2 secondi oppure quando la selezione *Two Tournament* riduce i partecipanti al "torneo" ad un numero minore o uguale di 2.

-Torna al Sommario-

7. Conclusioni

Nonostante costrutti e librerie già predefinite per l'implementazione di algoritmi genetici, abbiamo preferito addentrarci nel mondo dell'intelligenza artificiale scrivendo da zero le varie componenti di questo progetto, per permetterci di apprendere e quindi apprezzare al meglio ogni argomento visto durante il corso.

Concludiamo dicendo che siamo soddisfatti del lavoro svolto e nonostante i risultati dell'algoritmo non siano particolarmente brillanti, la soluzione offerta raggiunge gli obiettivi che ci eravamo inizialmente prefissati.

-Torna al Sommario-