Applicazione di Ricerca, Scoperta e Social per Ristoranti

Componenti del gruppo

- Marco Melucci [778426], m.melucci@studenti.uniba.it
- Gabriele Melucci [758392], g.melucci2@studenti.uniba.it

Link GitHub: progetto-ICON

A.A. 2024-2025

Indice generale

CAPITOLO INIZIALE – INTRODUZIONE

L’obiettivo di questo progetto è quello di creare la logica di un’applicazione basata sui ristoranti, con un contorno di aspetto sociale, tenendo conto delle caratteristiche (features) dei ristoranti, come tipo di cucina, o quelle degli utenti, come le loro valutazioni e i loro gusti. Il sistema allenato su data-set è in grado di categorizzare ristoranti in base alla loro descrizione, e può suggerire all’utente ristoranti che potrebbero essere di loro gradimento, offrendo inoltre funzioni di ricerca.

SOTTOCAPITOLO – REQUISITI FUNZIONALI

Il progetto è stato sviluppato utilizzando Python, un linguaggio di programmazione che offre un'ampia gamma di librerie per la gestione e l'analisi dei dati in modo semplice. La versione di Python utilizzata è la 3.11, mentre l'IDE scelto per lo sviluppo è VisualStudio Code.

Librerie utilizzate:

- ***pandas***, libreria per l'elaborazione e analisi di dati strutturati, come tabelle e serie temporali;
- ***numpy***, una libreria per il calcolo scientifico, specializzata in array multidimensionali e funzioni matematiche ad alte prestazioni;
- ***nltk***, una delle librerie principali per il Natural Language Processing (NLP) in Python;
- ***sklearn***, una libreria per il machine learning in Python, offre strumenti per clustering, classificazione e valutazione dei modelli
- ***tensorflow***, è una libreria open-source sviluppata da Google per il calcolo numerico e l'addestramento di modelli di machine learning (ML) e deep learning (DL);
- ***networkx***, è una libreria Python utilizzata per la creazione, manipolazione e studio di strutture di grafi e reti complesse.

SOTTOCAPITOLO – INSTALLAZIONE E AVVIO

Apri la cartella del progetto “Progetto-ICON” nel tuo IDE preferito e avvia il programma eseguendo il file Main.py.

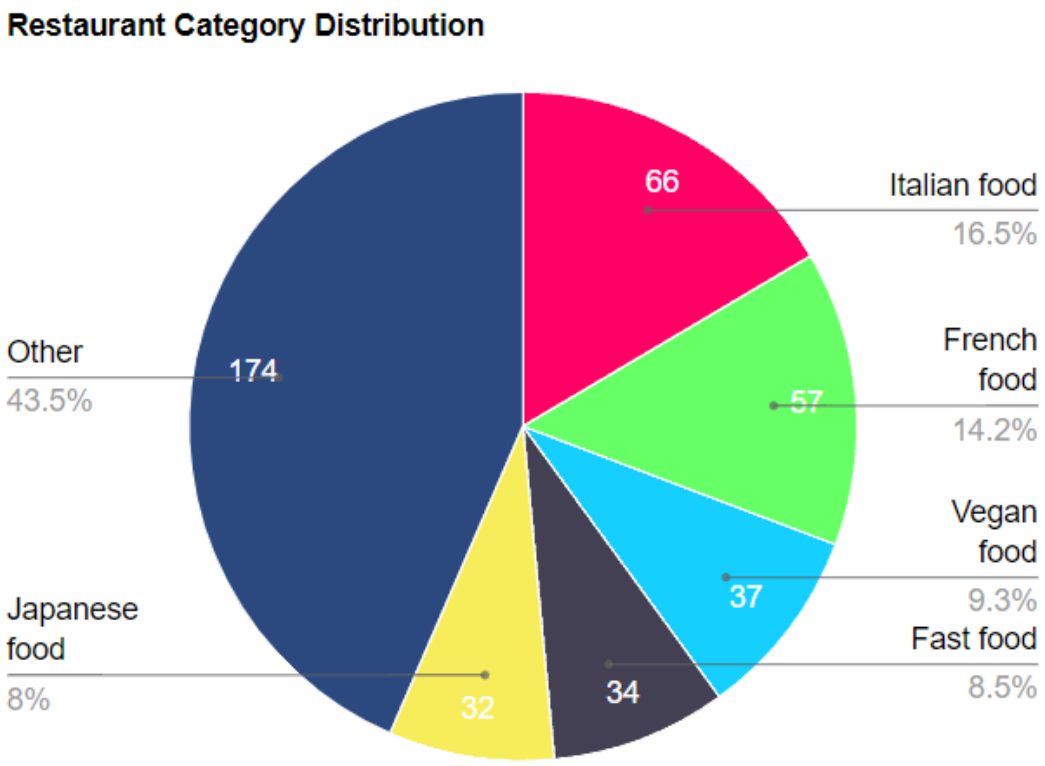
NB: Il programma controllerà la presenza di librerie e pacchetti necessari e procederà ad installare quelli che mancano. In base alla potenza della connessione internet ed altri fattori potrebbe volerci qualche minuto. Se il programma sembra non fare progressi dopo i controlli per i download, provare a fermarlo e rieseguirlo può correggere il problema.

1° CAPITOLO – CREAZIONE DEI DATA-SET (KB)

La creazione del data-set (KB) è avvenuta attraverso un processo strutturato e iterativo, suddiviso in tre fasi principali. In primo luogo, abbiamo individuato e selezionato un data-set open source online, utilizzato come base di partenza, garantendo che fosse pertinente al dominio di interesse. Questo ha fornito una solida struttura iniziale per il nostro lavoro.

Successivamente, abbiamo utilizzato un modello di intelligenza artificiale per estendere il data-set, generando un determinato numero di nuove istanze per ridurre i tempi di lavoro. Questa fase ha permesso di arricchire il data-set con informazioni aggiuntive, migliorando la rappresentatività e l'utilità per il modello di apprendimento supervisionato.

Infine, abbiamo intrapreso diverse iterazioni di controllo manuale sul data-set. Queste revisioni hanno avuto l'obiettivo di migliorare, chiarire e correggere eventuali errori presenti. Grazie a questo processo iterativo, abbiamo ottenuto un data-set più accurato e affidabile, ponendo le basi per migliori prestazioni dei modelli addestrati su di esso.



SOTTOCAPITOLO – FEATURES DEI DATA-SET

Features (regole e caratteristiche) effettivamente utilizzate nel file "*restaurantList.json"*:

- ***ID***, questo campo rappresenta un identificativo numerico univoco per il ristorante;
- ***Name***, il nome del ristorante;
- ***Description***, la descrizione del ristorante, ipoteticamente scritta da personale dei ristoranti stessi per l'applicazione;
- ***Categories***, identifica le tipologie di piatti e cucina che il ristorante offre.

Features utilizzate nel file *"userRatings.json"*:

- ***review_id***, questo campo identifica univocamente una recensione;
- ***user_id***, questo campo identifica l'utente che ha scritto la recensione;
- ***restaurant_id***, questo campo collega la recensione al ristorante specifico recensito;
- ***rating***, questo campo indica il punteggio assegnato dall'utente al ristorante.

Alcune features presenti nei file JSON (e quindi CSV, post conversione) non sono utilizzati in maniera effettiva nel programma. Tali informazioni sono orari di apertura e chiusura dei ristoranti, piatti principali dei ristoranti, ovvero le specialità e il time-stamp delle recensioni.
Tali dati sono stati aggiunti per nostra decisione nelle fasi iniziali, pensando ad un utilizzo futuro che non è effettivamente accaduto, ma rappresentano comunque informazioni utili che potrebbero essere utilizzate in espansioni delle funzioni.

SOTTOCAPITOLO – PREPROCESSING DEI DATA-SET

Il preprocessing del data-set per l'analisi e l'utilizzo dei dati, avviene tramite un semplice processo nel quale il testo raccolto viene sottoposto a vari passaggi:

- **Riduzione in minuscolo**;
- **Tokenizzazione**: divisione in unità piccole del testo, come singole parole, segni di punteggiatura etc...;
- **Rimozione delle stop word**: punteggiatura, articoli e altre parole non utili alla comprensione da parte della macchina);

- **Lemmatizzazione**: riduzione alle forme basi delle parole, come portare all'infinito i verbi, trasformare parole complesse come "migliore" in "buono" ed altre semplificazioni del testo.

Questo permette al programma di analizzare il testo (in particolare delle descrizioni dei ristoranti) in maniera semplice, ma rimanendo perfettamente capace di fornire risultati corretti ed accurati con ciò che impara da tali analisi.

```
    words = word_tokenize(text.lower())
    processed_words = [
        word for word in words
        if word not in stop_words and word.isalpha() and word not in string.punctuation
    ]
    processed_words = [lemmatizer.lemmatize(word) for word in processed_words]
    return processed_words
```

2° CAPITOLO – APPRENDIMENTO SUPERVISIONATO

L'obiettivo centrale dell'apprendimento supervisionato è che l'algoritmo apprenda una relazione tra gli input e le etichette (in questo caso le categorie dei ristoranti), in modo tale che, una volta addestrato, possa fare previsioni accurate anche su nuovi dati mai visti prima.

SOTTOCAPITOLO – SCELTA ARGOMENTO

Perché abbiamo usato l'apprendimento supervisionato?
Abbiamo utilizzato l'apprendimento supervisionato perché il nostro obiettivo era classificare i ristoranti in categorie specifiche basandoci sulle loro descrizioni, immaginando un applicazione in cui tale funzione semplificherebbe e renderebbe più godibile la ricerca di ristoranti da parte di un utente. Questo approccio richiede un data-set etichettato, dove ogni esempio è associato alla sua categoria corretta, permettendo al modello di apprendere le relazioni tra le caratteristiche (parole nelle descrizioni) e le etichette (categorie). Tale metodo garantisce un processo predittivo più preciso e mirato rispetto a tecniche non supervisionate.

Quali sono i pro dell'apprendimento supervisionato?

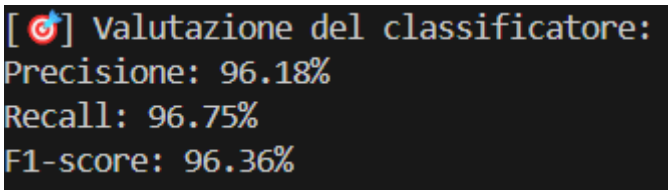
- **Accuratezza delle predizioni:** Grazie al data-set etichettato, il modello può essere addestrato a identificare con precisione le categorie basandosi su pattern chiari.
- **Facilità di valutazione:** Le prestazioni possono essere misurate con metriche come precision, recall e F1-score, fornendo una valutazione dettagliata dell'efficacia del modello.
- **Adattabilità:** È possibile iterare sul modello, migliorandolo tramite aggiornamenti dei dati o cambiando le tecniche di pre-processing applicate al testo in input.

Le principali fasi coinvolte nel processo di apprendimento supervisionato sono:

1. **Preparazione dei dati**: i dati da utilizzare per l’addestramento e il testing vengono organizzati e preparati per fornire i vari input, tramite l’uso di dizionari o liste;
2. ***Fase di addestramento***: durante questa fase, il modello viene alimentato con dati etichettati, nel nostro caso i ristoranti con le loro descrizioni e categorie e apprende la relazione tra gli input e gli output.
3. ***Fase di test***: una volta addestrato, il modello viene testato su un insieme di dati separato, non visto durante l'addestramento, per

verificarne l'efficacia. Nel codice del programma questa fase è eseguita su una porzione del data-set stesso riservata al testing, secondo la divisione 80%-20% per training e test;

4. ***Valutazione delle prestazioni***: in questa fase, si misurano le performance del modello, utilizzando metriche appropriate, per capire quanto accuratamente il modello riesce a fare previsioni sui nuovi dati.



Precisione si riferisce alla percentuale, o comunque l'accuratezza di una predizione del programma quando categorizza un ristorante.
Recall indica la capacità del ristorante di individuare tutti i ristoranti con una determinata categoria.
F1-score è una media di queste due metriche.
Il programma esegue questo calcolo per ogni categoria nel programma ed effettua una media per ottenere una valutazione complessiva su queste metriche.

SOTTOCAPITOLO – TESTING DELLE POSSIBILI SCELTE

Il processo di testing e scelta dell'algoritmo di apprendimento supervisionato ha seguito un approccio iterativo e data-driven, mirato a identificare la combinazione di tecniche e parametri che potessero massimizzare le prestazioni. Ecco una panoramica delle principali scelte effettuate:

Test iniziali con Naive Bayes:

- L'algoritmo **Naive Bayes** è stato scelto come punto di partenza per la sua semplicità e velocità computazionale;
- Tuttavia, i primi risultati hanno mostrato una **scarsa accuratezza**, segnalando che il modello non era sufficientemente accurato o adatto a catturare le relazioni e le complessità presenti nei dati, o la presenza di altri problemi relativi al data-set.

Introduzione dello smoothing di Laplace:

- Per migliorare le prestazioni del modello, è stato applicato lo **smoothing di Laplace**, una tecnica progettata per gestire problemi legati a frequenze nulle o estremamente basse nelle classi;

- Questa modifica ha prodotto un **miglioramento marginale**, evidenziando che ulteriori interventi erano necessari per ottenere risultati soddisfacenti.

Sperimentazione con TF-IDF:

- Per potenziare la rappresentazione delle caratteristiche testuali, è stata testata la tecnica **TF-IDF** (Term Frequency-Inverse Document Frequency), che riduce il peso dei termini comuni ed enfatizza quelli distintivi;
- Sebbene TF-IDF abbia apportato un **miglioramento limitato**, i risultati hanno indicato che la qualità e la struttura complessiva dei dati erano aspetti chiave da ottimizzare per superare un certo margine di accuratezza.

Combinazione di tecniche e cross-validation:

La scelta finale è ricaduta su un approccio che integrava:

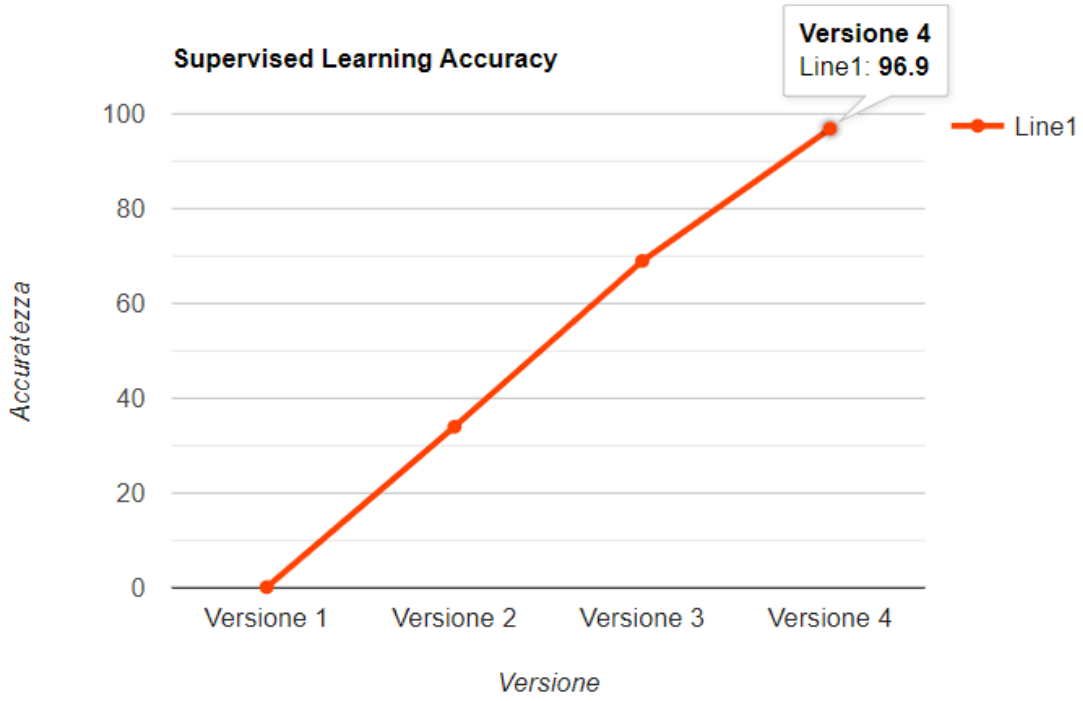
- **Naive Bayes** con **smoothing di Laplace**, per garantire un modello robusto;
- **K-fold Cross-validation**, per valutare le prestazioni in modo più affidabile e prevenire problemi di overfitting.

La combinazione di queste tecniche, in seguito a un finale miglioramento del data-set, senza includere direttamente TF-IDF, si è dimostrata la più efficace, fornendo risultati significativamente migliori rispetto a ogni tecnica utilizzata singolarmente.

Iterazioni con aggiornamenti del data-set:

- Ogni ciclo di testing è stato accompagnato da un processo di **aggiornamento e arricchimento del data-set**, con descrizioni più rappresentative e consone alle categorie assegnate;
  Questo aggiornamento continuo ha portato a miglioramenti consistenti, dimostrando che la qualità del data-set è un fattore determinante per il successo del modello.

Il processo di testing ha mostrato che, mentre tecniche come il **Naive Bayes** e **TF-IDF** possono fornire miglioramenti limitati individualmente, una combinazione di approcci e un attento affinamento del data-set offrono risultati ottimali. La decisione finale ha portato nella fase di testing il raggiungimento di percentuali molto alte, anche se esse possono fluttuare in base al seed utilizzato nel testing (come è possibile vedere nella differenza tra grafico ed output del programma). La validazione incrociata e lo smoothing di Laplace hanno contribuito significativamente, rendendo il modello più accurato e affidabile.



SOTTOCAPITOLO – ARGOMENTO USATO

L'apprendimento supervisionato all'interno del codice è implementato come un sistema di classificazione per prevedere la categoria di un ristorante in base alla sua descrizione, tramite uso di algoritmo **Naive Bayes**, miglioramento della performance con **smoothing di Laplace** e validazione e valutazione del testing attraverso **K-fold cross validation**:

Preparazione dei Dati:

Le descrizioni vengono pre-processate come menzionato prima (minuscole, tokenizzazione, stop words rimosse, lemmatizzazione);
Si calcolano due strutture:

- ***word_counts***: frequenza delle parole per ogni categoria;
- ***category_counts***: totale dei ristoranti per categoria.

Classificazione (Naive Bayes):

La funzione ***predict_category*** usa il teorema di Bayes per calcolare la probabilità logaritmica di una descrizione di appartenere a ogni categoria, applicando lo **smoothing di Laplace** [1] per parole rare.

Valutazione:

La funzione ***evaluate_classifier*** usa il **K-fold Cross-Validation** a 4 strati per confrontare categorie predette con quelle reali. Calcola poi precision, recall e F1-score come metriche per la valutazione delle performance dei metodi utilizzati.

```python
def evaluate_classifier(data, word_counts, category_counts, k=4):
    """
    Valuta il classificatore usando K-fold cross-validation e calcola precision, recall e F1-score.
    """
    # Divido in 4 pieghe mescolate in base a un seed i dati
    kf = KFold(n_splits=k, shuffle=True, random_state=42)
    true_labels = []
    predicted_labels = []

    # Testing del modello. passando per gli indici elencati nei fold, si raccolgono
    # le descrizioni e le categorie effettive, e si mettono in liste i label (categorie)
    # predette (tramite la funzione precedente) e reali al fine di misurare l'accuratezza.
    for train_index, test_index in kf.split(data):
        test_data = [data[i] for i in test_index]
        for row in test_data:
            if 'description' in row and 'categories' in row:
                description = row['description']
                true_category = row['categories']
                predicted_category = predict_category(description, word_counts, category_counts)
                true_labels.append(true_category)
                predicted_labels.append(predicted_category)
```

Interazione:

L'utente può visualizzare i ristoranti di una specifica categoria, usando il modello per classificare in base alle descrizioni; Il modello quindi apprende dai dati etichettati e usa le probabilità per classificare nuove descrizioni, garantendo un'accurata categorizzazione di nuovi ristoranti.

```
30. Thai food
31. Turkish food
32. Vegan food
33. Vietnamese food
[📑] Seleziona una categoria con un numero (0 per uscire): 31

[👨‍🍳] Ecco a te i ristoranti Turkish food:

- Turkish Delight
- Istanbul Kebab House
```

3° CAPITOLO – RETI NEURALI E MACHINE LEARNING

Le reti neurali sono modelli computazionali ispirati al cervello umano, utilizzati per risolvere problemi complessi. Sono composte da neuroni artificiali organizzati in strati: uno di input, uno o più strati nascosti e uno di output. Ogni neurone riceve un input, lo elabora con un peso e un bias, e produce un output attraverso una funzione di attivazione.

SOTTOCAPITOLO – SCELTA ARGOMENTO

Perché abbiamo usato le reti neurali?
Abbiamo utilizzato le reti neurali per modellare relazioni complesse tra utenti e ristoranti. In particolare, l'uso dell'autoencoder ci ha permesso di eseguire un'analisi di riduzione dimensionale e di identificare pattern nascosti nei dati, come preferenze implicite degli utenti o correlazioni tra ristoranti. Questo approccio è ideale per gestire grandi volumi di dati (come una matrice di valutazioni), dove le connessioni dirette non sono facilmente rilevabili.

Quali sono i pro delle reti neurali?

- **Capacità di apprendere rappresentazioni complesse:** Le reti neurali possono catturare pattern non lineari e relazioni intricate tra input e output, rendendole ideali per problemi complessi come le raccomandazioni.
- **Adattabilità a dati non strutturati:** Possono gestire dati scarsamente strutturati, come le valutazioni mancanti, trovando comunque informazioni utili.
- **Scalabilità:** Le reti neurali sono adatte per gestire grandi data-set, sfruttando parallelismo e ottimizzazioni hardware come GPU.
- **Miglioramento continuo:** Con l'aggiunta di nuovi dati, possono essere riaddestrate per migliorare la precisione delle raccomandazioni.

SOTTOCAPITOLO – ARGOMENTO USATO

Le reti neurali sono state utilizzate nel codice implementando un autoencoder, una rete neurale progettata per apprendere una rappresentazione compatta dei dati, usata qui per un sistema di raccomandazione in cui, in seguito a un' analisi, il programma è in grado di determinare una lista di ristoranti che potrebbero piacere all'utente, basandosi sui gusti che sono stati appresi.

La struttura è composta da:

- ***Encoder***: comprime le valutazioni degli utenti sui ristoranti in una rappresentazione a 32 dimensioni attraverso due layer densi (64 e 32 neuroni, con attivazione ReLU);
- ***Decoder***: ricostruisce le valutazioni originali usando due layer densi (64 neuroni e un layer di output con attivazione lineare).

Compilazione
- Ottimizzatore: ***adam*** (learning rate 0.001); [1]
- Funzione di perdita: ***mean_squared_error***, adatta per la ricostruzione continua. [2]

Addestramento
- La matrice delle valutazioni (utenti x ristoranti) viene normalizzata;
- La rete apprende per 50 epoche analizzando tutti i dati mescolati ogni volta, ottimizzando la ricostruzione delle valutazioni.

Per un utente specifico, l'autoencoder predice le valutazioni per tutti i ristoranti;
- I ristoranti già apprezzati (rating >= 4) vengono esclusi;
- I ristoranti con le valutazioni predette più alte che sono rimasti vengono poi suggeriti come raccomandazioni, stampando su schermo solo i primi 10.

L'autoencoder quindi sfrutta le preferenze degli utenti per individuare pattern nascosti e generare raccomandazioni personalizzate.

```
Dato che ti piacciono:
- Vietnamese Pho

Potrebbero piacerti anche:
1. Maxie's Fine Dining
2. Noodle House
3. Sweet Roost
4. Spanish Tapas
5. Soul Food Kitchen
6. Sushi Excellence
7. Sweeding Tree
8. Hungry Bar & Grill
9. Green White and Red Coffee
```

4° CAPITOLO – KNOWLEDGE GRAPH

Un Knowledge Graph (KG) è una struttura che organizza dati come entità e le relazioni tra di esse in un grafo. Le entità sono rappresentate come nodi, e le relazioni tra di esse come archi. I KG permettono di connettere e interrogare informazioni in modo efficiente, migliorando la comprensione e l'elaborazione di dati complessi.

SOTTOCAPITOLO – SCELTA ARGOMENTO

Perché abbiamo usato il knowledge graph?
Il knowledge graph è stato utilizzato per rappresentare e modellare le relazioni complesse tra utenti, ristoranti e categorie in modo strutturato e facilmente analizzabile. La rete di collegamenti che si va a creare è stata reputata appropriata per l'uso in una sezione del codice che fa riferimento a funzioni "social", come la scoperta di utenti con gusti simili. Questo tipo di modello consente di integrare diversi tipi di informazioni (recensioni, preferenze degli utenti, e categorie) in un'unica struttura, permettendo di eseguire query specifiche e ottenere insight utili come raccomandazioni personalizzate o analisi di utenti simili.

Quali sono i pro del knowledge graph?

- **Rappresentazione esplicita delle relazioni:** Permette di modellare chiaramente le connessioni tra entità diverse, come utenti, ristoranti e categorie.
- **Flessibilità:** È possibile aggiungere facilmente nuovi tipi di nodi o relazioni, adattandosi a modifiche nei dati o nei requisiti.
- **Query avanzate:** Consente di eseguire analisi complesse, come trovare utenti con gusti simili o scoprire ristoranti popolari in categorie specifiche.
- **Visualizzazione:** Offre una rappresentazione intuitiva e visivamente interpretabile delle relazioni, utile per comprendere pattern o connessioni.
- **Supporto per analisi di connettività:** Funzioni integrate come il calcolo della densità, dei cammini più brevi, e della centralità aiutano a comprendere meglio la struttura del sistema.

SOTTOCAPITOLO – ARGOMENTO USATO

Il codice implementa un **knowledge graph** (grafo di conoscenza) utilizzando la libreria networkx. Questo grafo rappresenta relazioni tra ristoranti, categorie e utenti, basandosi sui dati di due file CSV (ottenuti da data-set originariamente in formato JSON): uno per i dettagli dei ristoranti e l'altro per le recensioni degli utenti.

Struttura del Grafo

- Il grafo è **non orientato** e utilizza nodi ed archi per rappresentare le relazioni:
  - i nodi includono **ristoranti**, **categorie** e **utenti**;
  - gli archi collegano i ristoranti alle loro categorie e gli utenti ai ristoranti che hanno valutato.

Aggiunta dei Nodi

- Ogni ristorante viene rappresentato da un nodo con un'etichetta "restaurant" e collegato a uno o più nodi categoria con archi etichettati come "is_a";
- Ogni utente viene rappresentato da un nodo con etichetta "user" e collegato ai ristoranti che ha valutato con archi etichettati come "rated" e un peso pari al valore della valutazione.

Funzioni del Grafo

L'uso del grafo ci ha permesso di inserire funzioni specifiche che risultano efficaci grazie alla struttura dati usata.

Analisi del Grafo

- Calcola statistiche generali del grafo, come il numero di nodi, archi, densità, e il conteggio specifico di ristoranti, categorie e utenti.

Ristoranti più apprezzati

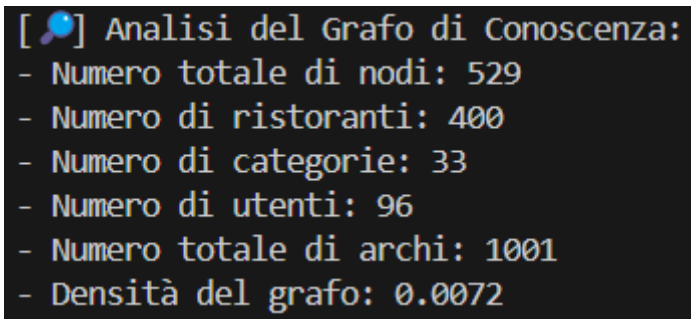
- Determina i ristoranti con una valutazione media superiore a una soglia specificata dall'utente in precedenza.

Ristoranti più popolari

- Identifica i 10 ristoranti con il maggior numero di recensioni, ordinati in base al numero di utenti collegati.

Utenti con gusti simili

- Trova utenti con preferenze simili basandosi sulle categorie dei ristoranti apprezzati dall'utente che effettua la ricerca. Confronta le valutazioni degli utenti per individuare interessi comuni, non sugli stessi ristoranti, ma sul tipo di cucina.



Il knowledge graph permette allora di analizzare e visualizzare le relazioni tra utenti, ristoranti e categorie

CONCLUSIONI E AGGIUNTE POSSIBILI

Abbiamo esplorato nella creazione del progetto diverse tecniche di intelligenza artificiale ed analisi dei dati per affrontare il problema della raccomandazione di entità e categorizzazione di dati, dimostrando come l'integrazione di approcci diversi possa portare a risultati significativi nella creazione della logica di un applicazione che mira ad essere utile quanto informativa, oltre che adattiva ai suoi utenti e ai loro gusti.

Tra le possibili aggiunte a cui si pensava di lavorare ci sono:

- Capacità di accettare data-set interamente nuovi per la categorizzazione di ristoranti da utilizzare nell'applicazione oltre i data-set di allenamento;
- Funzioni di ricerca con filtri per sfruttare dati come specialità, orari etc…
- Interazioni attive col database da parte dell'utente, ovvero la capacità effettiva di dare in input nuove recensioni per poter cambiare in tempo reale i suggerimenti del programma.

RIFERIMENTI BIBLIOGRAFICI

NumPy: Harris, C. R., Millman, K. J., van der Walt, S. J., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362.

NLTK: Bird, S., Loper, E., & Klein, E. (2009). *Natural language processing with Python*. O'Reilly Media.

Scikit-learn: Pedregosa, F., Varoquaux, G., Gramfort, A., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct), 2825-2830.

TensorFlow: Abadi, M., Agarwal, A., Barham, P., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv preprint arXiv:1603.04467*.

NetworkX: Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference* (Vol. 151, pp. 11-15). Pasadena, CA: SciPy.

Apprendimento Supervisionato
Reti Neurali
Knowledge Graph
[1] https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece

[2] https://it.wikipedia.org/wiki/Discesa_stocastica_del_gradiente#Adam
[3] https://it.wikipedia.org/wiki/Scarto_quadratico_medio
[4] https://it.wikipedia.org/wiki/Rettificatore_(reti_neurali)