**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](https://github.com/GabrieleMelucci/Progetto-ICON)

A.A. 2024-2025

INDICE

**0° CAPITOLO – 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 dataset 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 VisualStudioCode.

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 il progetto nel tuo IDE preferito e avvia il programma eseguendo il file Main.py.

**1° CAPITOLO – CREAZIONE DEL DATASET**

La creazione del dataset è avvenuta attraverso un processo strutturato e iterativo, suddiviso in tre fasi principali. In primo luogo, abbiamo individuato e selezionato un dataset open source 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 dataset, generando nuove istanze e ampliando la varietà e la copertura dei dati. Questa fase ha permesso di arricchire il dataset con informazioni aggiuntive, migliorando la rappresentatività e l'utilità per il modello di apprendimento supervisionato.

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

Features utilizzate nel file *resturantList.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;
* ***resturant\_id***, questo campo collega la recensione al ristorante specifico recensito;
* ***rating***, questo campo indica il punteggio assegnato dall'utente al ristorante.

**SOTTOCAPITOLO** **– PREPROCESSING DEL DATASET**

Il preprocessing del dataset per l’analisi e l’utilizzo dei dati, avviene tramite un semplice processo nel quale il testo raccolto e viene poi sottoposto a vari passaggi:

**Riduzione in minuscolo**;

**Tokenizzazione**: divisione in unità piccole del testo, come singole parole, segni di punteggiatura etc...;

**Rimozione delle stop words**: 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.

**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.

Le principali fasi coinvolte nel processo di apprendimento supervisionato sono:

1. ***Selezione degli iper-parametri***: si tratta di scegliere i parametri che guideranno il processo di addestramento del modello, come la velocità di apprendimento o la profondità di un albero decisionale;
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, apprende la relazione tra gli input e gli output, e ottimizza i suoi parametri.
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 dataset 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.

**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:

1. **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 potente da catturare le relazioni e le complessità presenti nei dati, o altri problemi relativi al dataset.
2. **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.
3. **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.
4. **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;
    - **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 dataset, senza includere direttamente TF-IDF, si è dimostrata la più efficace, fornendo risultati significativamente migliori rispetto a ogni tecnica utilizzata singolarmente.

1. **Iterazioni con aggiornamenti del dataset**:
   * Ogni ciclo di testing è stato accompagnato da un processo di **aggiornamento e arricchimento del dataset**, con descrizioni più rappresentative e dettagliate;
   * Questo aggiornamento continuo ha portato a miglioramenti consistenti, dimostrando che la qualità del dataset 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 dataset offrono risultati ottimali. La validazione incrociata e lo smoothing di Laplace hanno contribuito significativamente al successo finale, rendendo il modello più accurato e affidabile.

**SOTTOCAPITOLO** **–** **ALGORITMO 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:

1. **Preparazione dei Dati**:
   * Le descrizioni vengono preprocessate 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.
2. **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 per parole rare.
3. **Valutazione**:
   * La funzione ***evaluate\_classifier*** usa il **K-Fold Cross-Validation** per confrontare categorie predette con quelle reali. Calcola poi precision, recall e F1-score come metriche per la valutazione delle performance dei metodi utiizzati.
4. **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.

**3° CAPITOLO – RETI NEURALI**

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.

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.

1. **Struttura**
   * ***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).
2. **Compilazione**
   * Ottimizzatore: ***adam*** (learning rate 0.001);
   * Funzione di perdita: ***mean\_squared\_error***, adatta per la ricostruzione continua.
3. **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.

**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.

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 dataset originariamente in formato JSON): uno per i dettagli dei ristoranti e l'altro per le recensioni degli utenti.

1. **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.
2. **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.
3. **Gestione dei Dati**
   * I dati sui ristoranti e sulle recensioni vengono letti da file CSV;
   * Le categorie dei ristoranti sono separate da virgole e ogni categoria viene aggiunta come nodo unico.

* **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.

**Da aggiungere** → 1) abbiamo trovato un dataset open source; 2) abbiamo utilizzato un modello IA per estendere le istanze; 3) abbiamo fatto diverse iterazioni di controllo a mano per migliorare, chiarire e correggere errori contenuti nel dataset.