Relazione progetto Metodi Informatici per la Gestione Aziendale

Recommendation System su dataset Amazon Reviews

Giorgio Casoni
Mat: 894495

A.A. 2024-2025

Indice

1 Executive Summary

Il presente progetto ha come obiettivo lo sviluppo, l'analisi e la valutazione di diversi sistemi di raccomandazione applicati a un sottoinsieme del dataset *Amazon Reviews 2023*[6][5], con particolare riferimento alla categoria merceologica *Books*. Il lavoro è stato strutturato secondo due dei tre livelli previsti dalla consegna (base e intermedio), il cui focus è tato sui sistemi di raccomandazione *collaborative filtering* e *content-based*.
La prima sfida è stata capire come gestire un dataset di dimensioni importanti come è quello utilizzato per il progetto (circa 15GB), e successivamente gestire, durante la fase di data preprocessing e data exploration, dati definibili "non-normativi", cioé discostati dalle specifiche teoriche del dataset.
Nel **progetto base** è stato sviluppato un sistema di raccomandazione collaborativo utilizzando dapprima l'algoritmo K-Nearest Neighbors (KNN), esplorandone diverse configurazioni in termini di misure di similarità, numero di vicini e approccio user- o item-based. Le prestazioni dei modelli sono state valutate tramite le metriche quantitative indicate in consegna (RMSE e MSE), permettendo di individuare una configurazione ottimale. In aggiunta, è stato implementato un approccio di *Matrix Factorization* basato su SVD, con l'obiettivo di confrontarne le prestazioni rispetto al KNN. Successivamente è stata effettuata una segmentazione degli utenti tramite clustering con K-Means, utilizzando come input le matrici di rating predette. Questa analisi ha consentito di individuare gruppi di utenti con comportamenti e preferenze simili, fornendo, in linea teorica, una lettura più strutturata della popolazione di utenti e permettendo di valutare la qualità del recommendation system.
Nel **progetto intermedio**, il sistema è stato esteso con un sistema di raccomandazione *content-based*, realizzato usando i metadati testuali dei prodotti. Sono stati costruiti e analizzati diversi tipi di embedding testuali, basati sia su tecniche di tipo statistico sia su modelli neurali (Transformer-based), al fine di valutare il loro impatto sulle prestazioni del sistema di raccomandazione.
I risultati finali mostrano come i sistemi di collaborative filtering forniscano prestazioni complessivamente migliori in termini di accuratezza predittiva, mentre gli approcci content-based risultano particolarmente utili in scenari di scarsità di dati o per migliorare l'interpretabilità delle raccomandazioni.

Method	RMSE	MSE
CF-KNN	0.9815	0.9635
CF-SVD	0.9641	0.9296
CB-TfIdf	0.9894	0.7784
CB-CBoW	0.9920	0.7850
CB-Transf	0.9871	0.7782

Tabella 1: La precisione finale delle varie tecniche usate.

Verranno poi analizzate, nelle sezione del report e in una sezione finale dedicata, anche le problematiche affrontate durante il progetto, causa probabile di alcuni risultati *"inaspettati"*.

2 Introduzione ed EDA

I sistemi di raccomandazione rappresentano oggi una componente fondamentale di numerose applicazioni informatiche, in particolare nei domini dell'e-commerce e dei contenuti di intrattenimento (streaming di film, musica, etc.), dove vengono utilizzati per supportare gli utenti nella scoperta di prodotti rilevanti all'interno di cataloghi estremamente ampi. In questo contesto, piattaforme come Amazon fanno largo uso di sistemi di raccomandazione per personalizzare l'esperienza utente, aumentare l'engagement e migliorare le prestazioni commerciali.
Il presente progetto si colloca in questo scenario e affronta il problema della progettazione e valutazione di sistemi di raccomandazione applicati a dati reali su larga scala, caratterizzati da elevata dimensionalità, forte sparsità e significativa eterogeneità delle informazioni disponibili.

2.1 Gestione del dataset

Come menzionato prima, una delle prime e principali difficoltà affrontate nel progetto riguarda la gestione di un dataset di dimensioni molto elevate. Il dataset *Amazon Reviews 2023*[5], nella categoria *Books* da me scelta, contiene centinaia di milioni di recensioni e metadati associati, rendendo impossibile un caricamento e un'analisi diretta in memoria senza opportune strategie di preprocessing. Per affrontare questa problematica, sono state adottate diverse strategie per ridurre l'impatto computazionale, tra cui:

1. conversione dei file originali in formato `JSONL` nel formato `Parquet`[1]. Questa strategia ha ridotto le dimensioni del dataset su disco da $\sim$ 25 GB a $\sim$ 9GB per il dataset *reviews* e da $\sim$ 11GB a $\sim$ 3GB per il dataset *metadata*, rendendo così più efficiente la fase di lettura e caricamento;

2. lettura e salvataggio dei dati in modalità *chunk-based*, per limitare l'uso della memoria e quindi velocizzare la conversione;

3. filtraggio degli utenti e dei prodotti con un numero minimo di interazioni, al fine di ridurre la sparsità della matrice di rating e migliorare la qualità delle analisi successive;

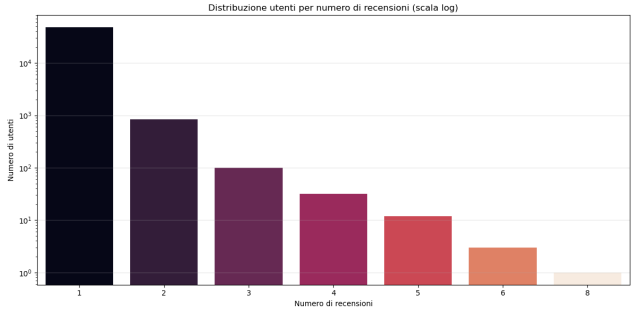
4. utilizzo di campioni rappresentativi del dataset per le fasi esplorative e di sviluppo del codice.

Queste scelte hanno permesso di mantenere un compromesso tra rappresentatività del dataset e sostenibilità computazionale, rendendo possibile l'applicazione di algoritmi di raccomandazione e clustering su sample sufficentemente grandi da essere statisticamente identici ai dati reali.
É stata inoltre applicata, in questa fase, una funzione atta a normalizzare il valore delle celle *'price'* a un float o a None.

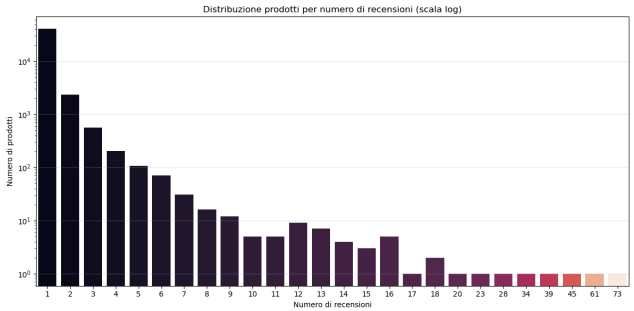
2.2 Analisi esplorativa dei dati - Reviews

Prima della costruzione dei modelli di raccomandazione, è stata condotta un'analisi esplorativa approfondita del dataset, con l'obiettivo di comprenderne le caratteristiche principali e guidare le scelte metodologiche successive.
Innanzitutto è stata fatta un'analisi di massima del dataset, per verificare i tipi di dato delle colonne e la presenza di valori nulli da rimuovere. Come si vede dalla figura 2a, questo ha rivelato che, ad eccezione della colonna rating e di poche altre, le colonne sono state up-castate in un tipo *object*, differente da quanto dichiarato dalle informazioni sul dataset. Questo ha ovviamente richiesto un successivo cast nei Types appropriati.
L'analisi ha evidenziato, come vediamo in figura 1, una distribuzione dei rating fortemente

sbilanciata, con il numero di recensioni per utente e per prodotto che risulta estremamente disomogeneo, e con una prevalenza di valutazioni alte, tipica dei sistemi di recensione online.
Un altro aspetto particolarmente rilevante emerso dall'analisi esplorativa, come vediamo in



(a) distribuzione numero di rating per utente



(b) distribuzione numero di rating per prodotto

Figura 1

```
### INFO DATASET REVIEWS (Books)

Sample size: 50000
Numero di righe: 50000
Numero di colonne: 10
Tipi di dato delle colonne:
rating                       int64
title                       object
text                        object
images                      object
asin                        object
parent_asin                 object
user_id                     object
timestamp            datetime64[ns]
helpful_vote                 int64
verified_purchase             bool
dtype: object
Numero di valori nulli:
rating               0
title                0
text                 0
images               0
asin                 0
parent_asin          0
user_id              0
timestamp            0
helpful_vote         0
verified_purchase    0
dtype: int64
```

(a) informazioni sul dataset

```
### STATISTICHE DESCRITTIVE RATING

count   50000.00000
mean        4.41124
std         1.06834
min         1.00000
25%         4.00000
50%         5.00000
75%         5.00000
max         5.00000
Name: rating, dtype: float64

Utenti unici: 48796
Prodotti unici (parent_asin): 43938
Sparsità (approx): 0.999977

Recensioni medie per utente: 1.024674153619149
Recensioni medie per prodotto: 1.1379671355091265
```

(b) statistiche di base sul dataset

Figura 2

figura 2b, è l'elevata sparsità della matrice utente-item, che rappresenta una sfida significativa per i sistemi di raccomandazione collaborativi. Per ovviare a questo problema, cioè una matrice user-item con una sparsità maggiore di 0.99, nella fase successiva, quella di realizzazione del RS collaborative filtering con KNN, il dataset è stato filtrato per selezionare soltanto gli utenti con più di n recensioni e gli item con più di m recensioni. Alla fine è stato scelto il valore $n = m = 20$, buon compromesso per ridurre notevolmente la sparsità della matrice senza rendere i risultati troppo distanti dalla realtà, cioè da un numero di recensioni medie per utente e per prodotto rispettivamente di 1.02 e 1.14.

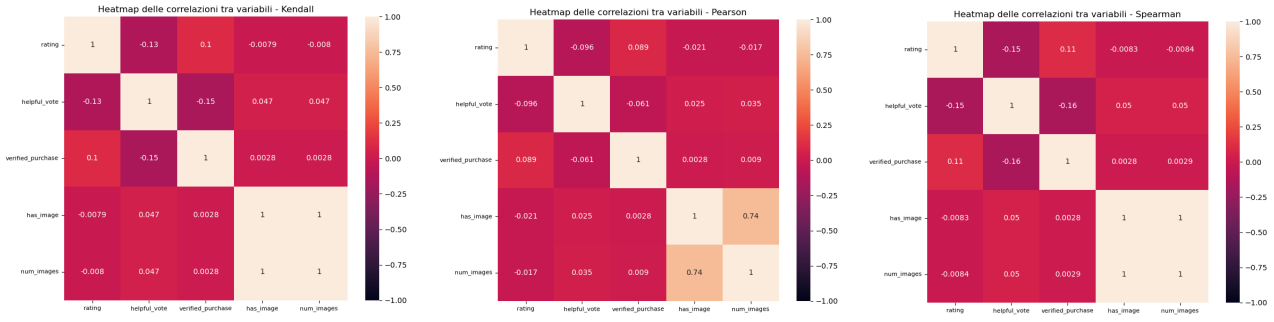
Lo step successivo è stato quello di cercare delle possibili correlazioni tra variabili disponibili. Le colonne utilizzate sono state quelle relative al voto dato al prodotto, al numero di voti helpful,

al flag di acquisto verificato nella review, alla presenza di immagini e al numero di immagini nella review. L'ultima colonna è stata aggiunta al dataset con l'apposito scopo di scoprire un'eventuale correlazione tra il numero di immagini presenti in una recensione e il numero di voti helpful. Il risultato finale dell'analisi della correlazione evidenzia un assenza totale di quest'ultima, sia positiva che negativa, per tutte le coppie di variabili, ad eccezione ovviamente di (has_images e num_images). La correlazione tra le variabili è stata valutata utilizzando tre diverse misure: correlazione di Pearson, Spearman e Kendall. L'uso congiunto di queste metriche consente di ottenere una visione più completa delle relazioni presenti nel dataset, tenendo conto delle diverse ipotesi statistiche sottostanti.

- La correlazione di Pearson[13] misura il grado di dipendenza lineare tra due variabili numeriche ed è particolarmente sensibile alla presenza di relazioni lineari e a valori anomali. Essa risulta appropriata quando le variabili sono distribuite in modo approssimativamente normale e quando si ipotizza una relazione lineare tra di esse; tuttavia, può fornire risultati fuorvianti in presenza di distribuzioni fortemente asimmetriche o di outlier, condizioni frequenti nei dataset reali di grandi dimensioni.

- La correlazione di Spearman[19] è una misura non parametrica basata sui ranghi delle osservazioni. Essa è in grado di catturare relazioni monotone, anche non lineari, ed è meno sensibile agli outlier rispetto alla correlazione di Pearson. Questo la rende particolarmente adatta all'analisi di dati con distribuzioni non gaussiane o caratterizzati da scale ordinali, come spesso accade nei sistemi di rating.

- La correlazione di Kendall[9], anch'essa non parametrica, misura la concordanza tra le coppie di osservazioni ed è generalmente più robusta in presenza di dataset rumorosi o di dimensioni ridotte. Sebbene sia computazionalmente più costosa e tenda a produrre valori di correlazione numericamente più bassi rispetto alle altre misure, Kendall fornisce una stima più stabile e interpretabile del grado di associazione tra variabili ordinali. L'impiego simultaneo di queste tre misure ha permesso di confrontare e validare i risultati ottenuti, riducendo il rischio di interpretazioni distorte e fornendo una valutazione più affidabile delle relazioni tra le variabili analizzate.



(a) correlazione con misura Kendall (b) correlazione con misura Pearson (c) correlazione con misura Spearman

Figura 3: heatmap della correlazione tra variabili per le tre misure scelte.

3 Collaborative Filtering RS con KNN

Questa sezione descrive lo sviluppo del sistema di raccomandazione basato su *collaborative filtering* tramite algoritmo K-Nearest Neighbors (K-NN) [2], come richiesto nel progetto base. L'implementazione è realizzata principalmente nello script `collaborative_filtering_KNN.py`, utilizzando la libreria `Surprise` per la modellazione e la validazione.

3.1 Motivazione dell'approccio KNN nel collaborative filtering

Il collaborative filtering sfrutta esclusivamente le interazioni utente-item (in questo caso i rating) per apprendere preferenze e similarità, senza richiedere informazioni descrittive sui prodotti. L'approccio KNN, in particolare, predice il rating atteso di un utente per un item aggregando i rating di utenti/item "vicini" secondo una misura di similarità.

Nel contesto di questo progetto, il KNN è particolarmente valido perché:

- è interpretabile (le raccomandazioni derivano da vicini simili);
- permette di testare chiaramente parametri significativi (metrica di similarità, k, user-based vs item-based);
- è una baseline solida per confronti successivi (es. Matrix Factorization).

3.2 Caricamento, filtraggio e campionamento del dataset

I dati utilizzati provengono dal file `Books.parquet` (categoria *Books*). La gestione della dimensione del dataset e della sparsità è affrontata tramite un preprocessing mirato di cui abbiamo già parlato.

Caricamento Lo script carica l'intero dataset in un dataframe Pandas:

```
reviews_full = pd.read_parquet(...)
```

Filtraggio per densità di interazioni Poiché i dataset di rating reali presentano tipicamente una distribuzione a lunga coda (molti utenti con poche recensioni e pochi utenti molto attivi), l'algoritmo KNN può risultare instabile se applicato su porzioni troppo sparse della matrice utente–item. Per mitigare questo problema, nel codice si calcola il numero di review per utente e per prodotto e si filtra mantenendo solo entità con almeno X interazioni:

- _X = soglia minima di review per `user_id`;
- _X = soglia minima di review per `parent_asin`.

Nel codice:

```
user_validi = [k for k,v in reviews_per_user.items() if v > _X]
item_validi = [k for k,v in reviews_per_item.items() if v > _X]
```

Questa scelta riduce la sparsità e aumenta la probabilità che, per una data coppia utente-item, esistano vicini utili per la predizione.

Campionamento per sostenibilità computazionale Dopo il filtraggio, si effettua un campionamento casuale di dimensione $\_N$:

```
reviews_final = reviews_filtered_B.sample(_N, random_state=42)
```

Il campionamento mantiene la struttura statistica del dataset filtrato, ma rende gestibili:

- la cross-validation ripetuta durante la Grid Search;
- la costruzione della matrice di predizione completa (computazionalmente costosa).

Risulta chiaro che la dimensione del campione è soltanto finalizzata a ridurre i tempi computazionali, in particolare del Grid Search, e che aumentando il campione, in combinazione con un filtraggio più restrittivo delle reviews, aumenta anche la precisione delle predizioni sia per KNN sia per SVD: aumentando il numero di possibili *neighbors* tra cui scegliere aumenta anche la probabilità di trovarne uno più simile.

3.3 Costruzione del dataset per Surprise

Per utilizzare `Surprise`, i dati vengono convertiti in un oggetto `Dataset` tramite `Reader`:

```
reader = Reader(rating_scale=(1,5))
dataset_knn = Dataset.load_from_df(reviews_final[[user, item, rating]],
reader)
```

La scelta della scala $(1, 5)$ è coerente con il dataset Amazon Reviews, dove i rating sono compresi tra 1 e 5.

3.4 Baseline iniziale con parametri di default

Prima di ottimizzare il modello, viene eseguito un test iniziale con una configurazione "ragionevole":

- similarità coseno;
- user-based;
- $k = 10$;
- `min_support=2`.

Nel codice:

```
sim_options_default = {name:  cosine, user_based:  True, min_support:
2}
knn_default = KNNBasic(k=10, sim_options=sim_options_default)
```

Motivazione dei parametri

- **Cosine similarity**: spesso efficace su dati sparsi, in quanto confronta l'orientamento dei vettori di rating più che la loro scala.
- **User-based**: adatto quando si ipotizza che utenti con pattern simili esprimano preferenze simili.

- **min_support**: impone un minimo numero di co-rating per considerare affidabile una similarità; valori troppo bassi aumentano rumore, valori troppo alti riducono eccessivamente i vicini disponibili.

La valutazione iniziale avviene tramite cross-validation a 3 fold:

```
cross_validate(knn_default, dataset_knn, measures=['RMSE','MSE'], cv=3)
```

I risultati del test baseline sono nell'immagine successiva. Possiamo osservare come i parametri

```
Evaluating RMSE, MSE of algorithm KNNBasic on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9862  0.9806  0.9781  0.9816  0.0034
MSE (testset)     0.9726  0.9617  0.9566  0.9636  0.0067
Fit time          8.51    9.44    9.44    9.13    0.43
Test time         0.12    0.08    0.07    0.09    0.02

### TEST KNN CON PARAMETRI DEFAULT:

k=10, name=cosine, user_based=True, min_support=2
RMSE medio: 0.9816 +- 0.0034
MSE  medio: 0.9636 +- 0.0067
Tempo: 28.17 secondi
```

Figura 4: I risultati su 3 fold del KNN baseline.

standard siano già buoni, con un RMSE minore di 1 e una deviazione standard comunque bassa.

3.5 Ottimizzazione tramite Grid Search

Fondamentale nella realizzazione di un RS con KNN e SVD è l'identificazione della configurazione ottimale del KNN: nel codice viene implementata una Grid Search tramite `GridSearchCV`[7] di `scikit-learn`.

Spazio degli iperparametri La griglia esplora:

- $k \in \{3, 5, 10, 15, 20, 25\}$;
- similarità `name` $\in \{\text{cosine}, \text{pearson}, \text{msd}\}$;
- `user_based` $\in \{\text{True}, \text{False}\}$;
- `min_support` $\in \{1, 2, 5\}$.

Nel codice:

```
param_grid_knn = { k:  [...], sim_options:  {name:  [...], user_based:
[...], min_support:  [...]}}
```

Motivazione delle scelte nella griglia

- **Cosine** è robusta su dati sparsi e non richiede centratura.
- **Pearson** cattura relazioni lineari "centrate" (normalizza rispetto alla media), ma può diventare instabile se ci sono pochi co-rating.

- **MSD** (mean squared difference) misura la distanza media tra rating su item in comune risulta utile quando si vogliono penalizzare forti differenze assolute.
- **User-based vs item-based**: permette di verificare quale struttura (somiglianza tra utenti o tra item) è più informativa nel dataset filtrato.

Metriche e cross-validation La Grid Search valuta RMSE e MSE:

```
GridSearchCV(KNNBasic, param_grid_knn, measures=['rmse','mse'], cv=3)
```

Per valutare i risultati di un KNN usiamo due metriche: RMSE e MSE. RMSE è la metrica principale perché penalizza maggiormente errori grandi, mentre MSE fornisce una lettura più "robusta" e interpretabile in media.
Nella figura sotto possiamo vedere i 30 migliori parametri risultati dal GridSearch. Salta subito all'occhio un fatto: MSE, RMSE e pure le deviazioni standard sono identiche per ognuna delle 30 combinazioni. Questo è dovuto a una combinazione di due fattori:

1. **errori di arrotondamento:** sebbene 4 decimali siano sufficientemente precisi, è possibile, e probabile, che stampando più decimali i risultati sarebbero differenti, anche dato che per l'ordinamento crescente per RMSE dei risultati, i decimali sono stati considerati tutti;
2. **il dataset è estremamente "piatto":** ovvero anche dopo il filtraggio la matrice è molto densa localmente, *'min_support'* è soddisfatto per quasi tutte le coppie e aumentare *'k'* oltre una certa soglia non cambia i vicini effettivi, perché i candidati disponibili di base sono molto pochi.

Nel nostro caso questo non crea un effettivo problema, dato che con una scala di rating composta da $\{1, 2, 3, 4, 5\}$, sarebbe problematico soltanto un errore maggiore di 1, mentre qualunque valore tra 0.01 e 0.99 produce lo stesso effetto. Questo significa anche che possiamo scegliere i parametri con ragioni più "euristiche", ad esempio se vogliamo maggiore velocità computazionale sceglieremo un $'k'$ basso.

```
### TUTTE LE COMBINAZIONI ORDINATE PER RMSE:
Rank  1 | k=10 | sim=msd     | Item-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  2 | k=20 | sim=cosine  | Item-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  3 | k=20 | sim=cosine  | Item-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  4 | k=20 | sim=cosine  | User-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  5 | k=20 | sim=cosine  | User-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  6 | k=20 | sim=cosine  | User-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  7 | k=15 | sim=msd     | Item-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  8 | k=15 | sim=msd     | Item-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank  9 | k=15 | sim=msd     | Item-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 10 | k=15 | sim=msd     | User-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 11 | k=15 | sim=msd     | User-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 12 | k=15 | sim=msd     | User-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 13 | k=15 | sim=pearson | Item-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 14 | k=15 | sim=pearson | Item-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 15 | k=15 | sim=pearson | Item-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 16 | k=15 | sim=pearson | Item-based | min_sup=1 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 17 | k=15 | sim=pearson | User-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 18 | k=15 | sim=pearson | User-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 19 | k=15 | sim=pearson | User-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 20 | k=15 | sim=pearson | User-based | min_sup=1 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 21 | k=15 | sim=cosine  | Item-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 22 | k=15 | sim=cosine  | Item-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 23 | k=15 | sim=cosine  | Item-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 24 | k=15 | sim=cosine  | User-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 25 | k=15 | sim=cosine  | User-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 26 | k=15 | sim=cosine  | User-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 27 | k=25 | sim=msd     | Item-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 28 | k=10 | sim=msd     | Item-based | min_sup=3 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 29 | k=10 | sim=msd     | Item-based | min_sup=2 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
Rank 30 | k=10 | sim=msd     | User-based | min_sup=5 | RMSE=0.9816+-0.0089 | MSE=0.9636+-0.0174
```

Figura 5: Risultati del GridSearch

3.6 Valutazione finale del modello migliore

Dopo aver individuato la configurazione ottimale, lo script esegue un test finale con split train/test di diverse dimensioni (0.1, 0.2, 0.3) per verificare stabilità e robustezza rispetto alla quantità di dati di training:

```
train_test_split(...  test_size=0.1/0.2/0.3 ...)
accuracy.rmse(predictions)
accuracy.mse(predictions)
```

Questa scelta implementativa consente di osservare se:

- le prestazioni degradano significativamente riducendo il training set;
- il modello è sensibile al campionamento casuale;
- RMSE/MSE rimangono coerenti con quanto osservato in cross-validation.

3.7 Costruzione della matrice di rating predetti e generazione raccomandazioni

Una volta selezionato il modello migliore, lo script effettua il training sul *full trainset* e costruisce una matrice di predizione completa (utenti $\times$ item) necessaria per:

- generare raccomandazioni Top-N per ciascun utente;
- svolgere la segmentazione utenti usando profili utente densi.

Nel codice, se il rating reale è già presente (`res.r_ui is not None`), il valore viene impostato a 0 per distinguere le celle osservate da quelle predette; altrimenti si utilizza la stima `res.est`. Questa scelta è utile per:

- evitare che i rating noti alterino la logica di ordinamento delle raccomandazioni;
- rendere esplicito quali suggerimenti derivano da predizioni.

Riempimento della matrice completa La matrice viene poi costruita iterando su tutti gli utenti e tutti gli item nel campione:

```
for uid in users_id:  for iid in items_id:  algo.predict(uid,iid)
```

Persistenza su disco Dato il costo computazionale di questo passaggio, la matrice viene salvata in formato `.npy`:

```
np.save(...  'KNN_rec_matrix', filled_rating_matrix)
```

Questa scelta è fondamentale per rendere riproducibile l'intera pipeline senza dover ricalcolare la matrice a ogni esecuzione (in particolare per le fasi di clustering e analisi successive).

Top-N recommendation Infine, le raccomandazioni vengono costruite ordinando per ogni utente gli item in base al rating predetto, ottenendo una lista ordinata di item consigliati:

```
sorted(row.items(), key=lambda x:  x[1], reverse=True)
```

3.8 Commenti

Il KNN collaborative filtering tende a performare bene quando esistono sufficienti sovrapposizioni tra utenti e item (co-rating), ma può soffrire in scenari di:

- cold-start (nuovi utenti/item con poche interazioni);
- matrice estremamente sparsa;
- distribuzioni fortemente sbilanciate dei rating.

Nel progetto, le scelte di filtraggio e la selezione della metrica di similarità sono state determinanti per garantire un numero sufficiente di vicini significativi e ottenere stime stabili. Il modello K-NN rappresenta quindi un baseline fondamentale sia per il confronto con SVD sia come base per la segmentazione utenti, dove la matrice di rating predetta è utilizzata per costruire profili utente teoricamente densi e confrontabili.

3.9 Clustering con K-Means

Una volta costruita la matrice completa di rating predetti tramite il modello KNN ottimizzato, è stata effettuata un'ulteriore analisi esplorativa basata sul clustering degli utenti.
L'obiettivo di questa fase non è quello di migliorare direttamente l'accuratezza del sistema di raccomandazione, bensì di ottenere una segmentazione degli utenti utile a interpretare e analizzare i pattern di preferenza appresi dal modello.

3.9.1 Motivazione dell'approccio

Il clustering viene applicato non sui rating originali, ma sulla matrice di rating *predetti* dal modello KNN. I rating predetti riflettono una versione "smussata" e regolarizzata delle preferenze degli utenti, riducendo il rumore dovuto a interazioni isolate. In questo modo, ogni utente viene rappresentato da un vettore numerico di dimensione pari al numero di item considerati, che descrive il suo profilo di preferenza stimato dal sistema di raccomandazione.

3.9.2 Algoritmo di clustering: K-Means

Per la segmentazione degli utenti è stato utilizzato l'algoritmo K-Means, una tecnica di clustering non supervisionato che mira a partizionare i dati in K cluster minimizzando la distanza intra-cluster[8][10].
I vantaggi di K-Means sono che:

- è computazionalmente efficiente anche su dataset di dimensioni medio-grandi;
- produce cluster facilmente interpretabili tramite i centroidi;
- è ampiamente utilizzato in letteratura per la segmentazione degli utenti nei sistemi di raccomandazione.

3.9.3 Scelta della tecnica di normalizzazione

Prima dell'applicazione dell'algoritmo K-Means, è stata effettuata una fase di normalizzazione dei dati, necessaria poiché il clustering si basa sulla distanza euclidea ed è quindi fortemente influenzato dalla scala delle variabili.
Sono state considerate diverse tecniche di scaling comunemente utilizzate, tra cui StandardScaler, MinMaxScaler e RobustScaler. Lo **StandardScaler** [20] normalizza i dati sottraendo la

media e dividendo per la deviazione standard, risultando appropriato in presenza di distribuzioni approssimativamente gaussiane e in assenza di outlier significativi. Tuttavia, nel contesto del presente progetto, i dati risultano fortemente asimmetrici, rendendo questa tecnica poco adatta. Il **MinMaxScaler**[11] riporta invece i dati in un intervallo predefinito, tipicamente $[0, 1]$, ma è particolarmente sensibile ai valori estremi: pochi outlier possono comprimere la maggior parte delle osservazioni in una regione ristretta dello spazio dei dati, riducendo la capacità discriminante del clustering. Per questi motivi è stato adottato il **RobustScaler**[16], che utilizza la mediana e l'intervallo interquartile per la normalizzazione. Questa tecnica risulta più robusta alla presenza di outlier e meglio adatta a distribuzioni non gaussiane, consentendo di preservare le differenze relative tra i profili degli utenti senza essere eccessivamente influenzata da comportamenti anomali.

3.9.4 Scelta del numero di cluster

La scelta del numero ottimale di cluster K è stata effettuata analizzando l'andamento di:

- *inertia*, la somma delle distanze intra-cluster;
- *silhouette score*[17], che misura la separazione tra cluster.

Sono stati testati diversi valori di K, e la scelta finale è stata effettuata individuando un compromesso tra:

- riduzione significativa dell'inertia (più basso è meglio);
- migliore valore del silhouette score (più vicino a $+1$ meglio è);

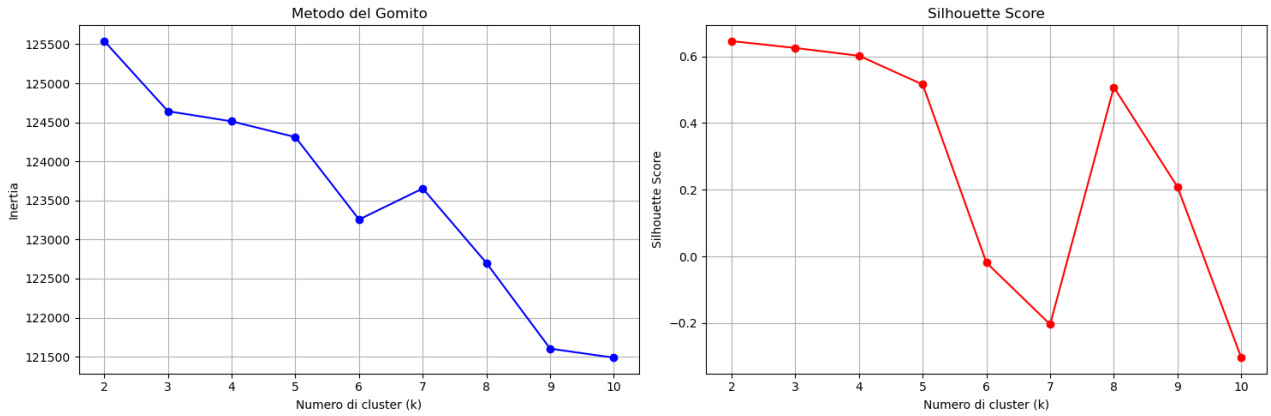


Figura 6: Inertia e Silhouette Score

Osservando i due grafici, è stato scelto $k = 4$ che è il valore con l'inertia migliore mantenendo una silhouette score maggiore di 0.6. Alternativamente, si sarebbe potuto usare $k = 8$, che ha una silhouette score un leggermente più bassa ma un inertia molto più bassa, ma solo in valore assoluto: in percentuale la perdita nella silhouette score è molto maggiore rispetto al guadagno in Inertia.

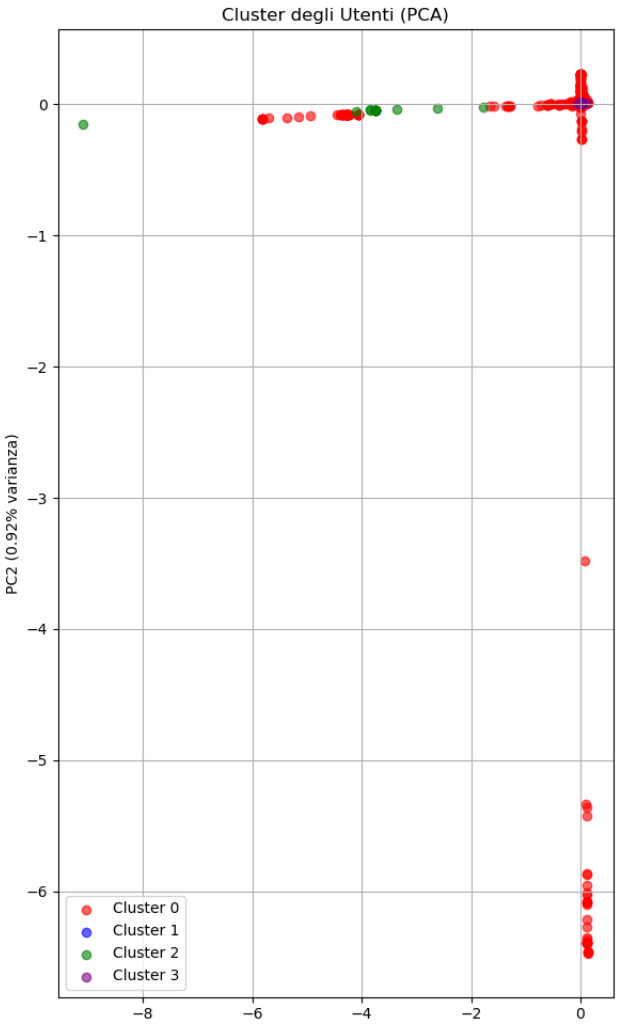
3.9.5 Cluster finali e analisi

Trovato il numero ideale di cluster si è proseguito con la loro analisi. Come possiamo osservare nelle immagini sottostanti, l'applicazione di K-Means alla matrice di rating predetti dal modello KNN ha prodotto una segmentazione fortemente sbilanciata, con un cluster dominante e alcuni cluster di dimensione molto ridotta. Questo risultato è coerente con le caratteristiche del

modello KNN e con la struttura del dataset analizzato. Infatti, il cluster principale raccoglie la grande maggioranza degli utenti e rappresenta il comportamento medio appreso dal sistema di raccomandazione: utenti con poche interazioni e una marcata tendenza a valutazioni positive. Tale comportamento riflette una proprietà nota dei dataset di recensioni online, in cui la distribuzione dei rating è fortemente sbilanciata verso valori alti e molti utenti contribuiscono con un numero limitato di recensioni.
I cluster di dimensione ridotta identificano invece gruppi di utenti con profili predetti significativamente differenti, caratterizzati da un numero maggiore di interazioni, da una maggiore variabilità nei rating o da una propensione a valutazioni più critiche. La quasi assenza di similarità coseno tra i centroidi dei cluster (figura 8) indica che questi gruppi occupano regioni nettamente distinte nello spazio delle preferenze latenti apprese dal modello K-NN.
Infine, la proiezione PCA[12][15] (figura 7) evidenzia invece che la maggior parte della varianza è spiegata dalla prima componente principale, associabile a una tendenza globale dei rating. Tuttavia, la separazione dei cluster avviene lungo componenti a varianza inferiore, coerentemente con il fatto che le differenze tra utenti nei sistemi di raccomandazione sono spesso sottili ma strutturate.
È tuttavia importante sottolineare che il clustering non viene applicato sui rating originali, ma su quelli predetti dal modello, i quali rappresentano una versione regolarizzata e smussata delle preferenze degli utenti. In questo contesto, il clustering fornisce uno strumento interpretativo utile per comprendere come il modello di raccomandazione organizzi implicitamente gli utenti, piuttosto che una segmentazione definitiva del comportamento reale degli utenti stessi.

```
### Applicazione K-means con k=4

### Analisi cluster:

Cluster 0:
        - Utenti: 40615
        - Rating medio: 4.346
        - Deviazione std rating: 0.982
        - Recensioni medie per utente: 1.5

Cluster 1:
        - Utenti: 13
        - Rating medio: 3.902
        - Deviazione std rating: 1.026
        - Recensioni medie per utente: 6.3

Cluster 2:
        - Utenti: 18
        - Rating medio: 3.956
        - Deviazione std rating: 1.242
        - Recensioni medie per utente: 2.5

Cluster 3:
        - Utenti: 11
        - Rating medio: 4.138
        - Deviazione std rating: 1.026
        - Recensioni medie per utente: 2.6
```

(a) Analisi dei cluster



(b) Analisi con PCA

Figura 7

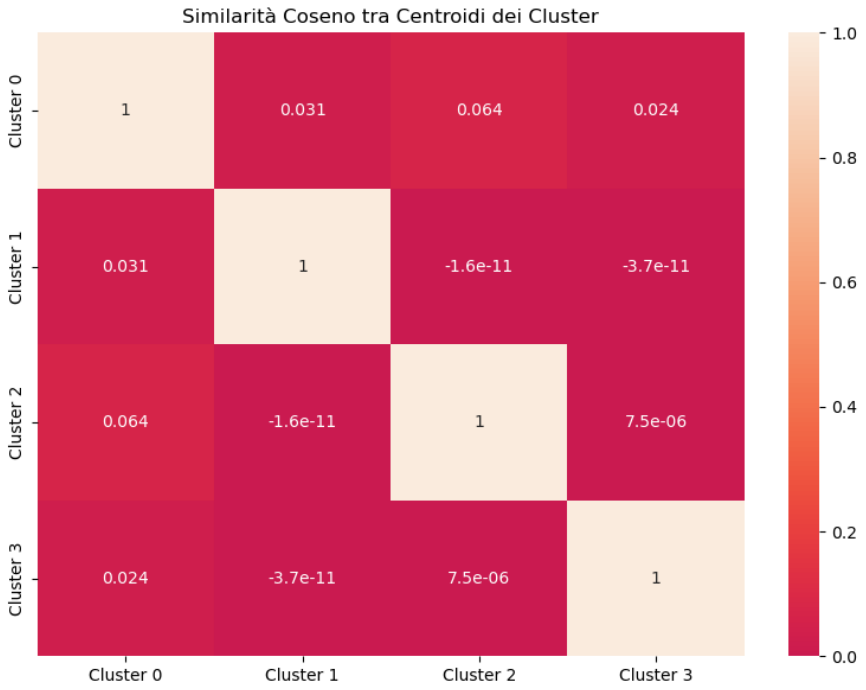


Figura 8: Similarità tra i centroidi dei cluster

4 Collaborative Filtering RS con SVD

In questa sezione viene descritto il sistema di raccomandazione collaborativo basato su *Matrix Factorization*, implementato tramite l'algoritmo *Singular Value Decomposition*[3] (SVD).
L'obiettivo principale è confrontare questo approccio con il collaborative filtering basato su KNN, analizzandone le differenze in termini di accuratezza predittiva, robustezza alla sparsità e capacità di modellare preferenze latenti.
L'implementazione è contenuta nello script `collaborative_filtering_SVD.py` e utilizza la libreria `Surprise`, che fornisce una versione ottimizzata dell'algoritmo SVD ampiamente utilizzata nei sistemi di raccomandazione.

4.1 Motivazione dell'approccio SVD

A differenza dei metodi basati sulla similarità diretta, come il KNN, la Matrix Factorization (nel nostro caso SVD) mira a rappresentare utenti e item in uno spazio latente di dimensione ridotta. Matematicamente, SVD fattorizza la nostra matrice user-item (di dimensione $m \times n$)nel prodotto di tre matrici, cioè $A = U\Sigma V*$, dove:

- U è una matrice unitaria di dimensione $m \times m$;
- Σ è una matrice diagonale rettangolare di dimensione $m \times n$;
- V^* è la trasposta coniugata di una matrice unitaria V di dimensioni $n \times n$.

Declinato ai recommender system, abbiamo che, approssimativamente parlando, le tre matrici rappresentano rispettivamente:

- U quello che piace agli utenti, le loro preferenze "generali";
- Σ quanto importante è ogni fattore, cioè ogni rating;
- V^* quanto gli item sono simili tra loro.

Il rating predetto è ottenuto come prodotto scalare tra tali vettori[18].
L'SVD ha quindi il vantaggio di:

- essere meno sensibile alla sparsità della matrice utente-item;
- consentire di catturare fattori latenti non osservabili direttamente;
- tendere a generalizzare meglio rispetto ai metodi basati su similarità;
- rappresentare un termine di confronto naturale rispetto al KNN.

4.2 Dataset e preprocessing

Per garantire un confronto equo con il sistema KNN, l'SVD è stato applicato allo stesso dataset, preprocessato con le stesse identiche funzioni e gli stessi parametri:

- filtraggio di utenti e item con almeno $\_X$ interazioni;
- campionamento casuale di $\_N$ recensioni;
- utilizzo della stessa scala dei rating $(1, 5)$.

In questo modo, eventuali differenze nelle prestazioni possono essere attribuite all'algoritmo e non a variazioni nei dati di input, sebbene SVD sia di base più veloce nella computazione rispetto a KNN, e avrebbe quindi permesso di usare campionamenti più grandi e migliorare ulteriormente la precisione dei risultati.

4.3 Costruzione del dataset per Surprise

I dati vengono convertiti in un oggetto `Dataset` tramite un `Reader`:

```
reader = Reader(rating_scale=(1,5))
dataset_svd = Dataset.load_from_df(reviews_final[[user, item, rating]],
reader)
```

Questo passaggio consente di utilizzare le funzionalità integrate di cross-validation e Grid Search offerte dalla libreria.

4.4 Baseline iniziale

Come primo riferimento quantitativo, viene valutata una configurazione base dell'SVD (texttt-svd_baseline = SVD()) con parametri di default:

- `n_factors`: 100;
- `n_epochs`: 20;
- `lr_all`: 0.005;
- `reg_all`: 0.02.

Il modello viene valutato tramite cross-validation a 3 fold, utilizzando RMSE e MSE:

```
cross_validate(svd_baseline, dataset_svd, measures=['RMSE','MSE'], cv=3)
```

Questa fase fornisce un primo confronto diretto con il baseline KNN.

```
### TEST SVD CON PARAMETRI DEFAULT:

RMSE medio: 0.9691 +- 0.0100
MSE  medio: 0.9393 +- 0.0195
Tempo: 1.98 secondi
```

Figura 9: test SVD baseline

4.5 Ottimizzazione degli iperparametri

Per migliorare le prestazioni del modello, è stata effettuata una Grid Search molto dettagliata sugli iperparametri principali dell'SVD. I risultati sono presentati in figura 10.

Spazio degli iperparametri

- `n_factors`: dimensione dello spazio latente;
- `n_epochs`: numero di epoche di training;
- `lr_all`: learning rate;
- `reg_all`: coefficiente di regolarizzazione.

Nel codice:

```
GridSearchCV(SVD, param_grid_svd, measures=['rmse','mse'], cv=4)
```

Motivazione delle scelte

- un numero ridotto di fattori latenti può portare a underfitting;
- un numero eccessivo di fattori aumenta il rischio di overfitting;
- la regolarizzazione è fondamentale per controllare il rumore nei dati sparsi;
- il learning rate influenza la stabilità della convergenza.

```
### GRID SEARCH SVD

Miglior RMSE: 0.964150
Miglior MSE : 0.929648
Migliori parametri (RMSE): {'n_factors': 50, 'n_epochs': 30, 'lr_all': 0.0075, 'reg_all': 0.01}
Tempo Grid Search: 313.75 secondi

### TOP 10 CONFIGURAZIONI (RMSE):
    n_factors  n_epochs  lr_all  reg_all  mean_rmse  std_rmse  mean_mse   std_mse  rank_rmse
40         50        30  0.0075     0.01   0.964150  0.007898  0.929648  0.015186          1
35         50        25  0.0100     0.04   0.964157  0.008238  0.929666  0.015839          2
42         50        30  0.0075     0.03   0.964261  0.008151  0.929865  0.015674          3
41         50        30  0.0075     0.02   0.964309  0.008150  0.929958  0.015672          4
23         50        20  0.0100     0.04   0.964347  0.008169  0.930032  0.015710          5
33         50        25  0.0100     0.02   0.964357  0.008256  0.930053  0.015876          6
83         75        25  0.0100     0.04   0.964369  0.007917  0.930070  0.015226          7
20         50        20  0.0100     0.01   0.964399  0.008084  0.930131  0.015548          8
47         50        30  0.0100     0.04   0.964425  0.008212  0.930183  0.015794          9
21         50        20  0.0100     0.02   0.964462  0.008154  0.930254  0.015683         10
```

Figura 10: I risultati del GridSearch

4.6 Valutazione finale del modello

Il modello ottimizzato viene valutato su split train/test di dimensione 0.1, 0.2 e 0.3:

```
train_test_split(...)
accuracy.rmse(predictions)
accuracy.mae(predictions)
```

Questo consente di analizzare la stabilità delle prestazioni rispetto alla quantità di dati disponibili per il training. Il risultato finale è presentato nell'immagine sottostante.

```
RMSE test_size=0.2, random_state=19: 0.9174169645224899
MSE test_size=0.2, random_state=19: 0.8416538867936594
MAE test_size=0.2, random_state=19: 0.7085433715012494
RMSE: 0.9152
MSE: 0.8376
MAE:  0.7125
```

Figura 11: L'RMSE finale su un test split di 0.2

4.7 Costruzione della matrice di rating predetti

Una volta selezionato il modello migliore, l'SVD viene addestrato sull'intero training set e utilizzato per costruire una matrice completa di rating predetti:

```
best_svd.fit(trainset)
best_svd.predict(uid, iid)
```

Analogamente al KNN, i rating già osservati vengono impostati a zero per distinguere le celle note da quelle predette e la matrice risultante viene salvata su disco per velocizzarne gli usi successivi durante la fase di clustering.

4.8 Clustering con K-Means

Una volta ottenuta la matrice completa di rating predetti tramite il modello SVD ottimizzato, è stata condotta un'analisi di clustering degli utenti con l'obiettivo di segmentare la popolazione in gruppi omogenei sulla base dei profili di preferenza appresi dal modello. Come nel caso del KNN, questa fase non è finalizzata ad aumentare direttamente l'accuratezza predittiva, bensì a fornire una lettura interpretativa dei comportamenti degli utenti, sfruttando una rappresentazione densa e confrontabile.
I vantaggi del clustering su una matrice di predizione creata da SVD sono principalmente che i rating predetti dall'SVD sono intrinsecamente regolarizzati, grazie alla fattorizzazione e ai termini di regolarizzazione, risultando meno rumorosi e che la Matrix Factorization cattura fattori latenti: utenti con preferenze simili tendono ad avere profili predetti simili anche in presenza di poche interazioni.
In altre parole, il profilo vettoriale di ciascun utente, ottenuto dai rating stimati, rappresenta una "firma" delle sue preferenze secondo il modello SVD, particolarmente adatta a operazioni di segmentazione.

4.8.1 Preparazione dei dati e normalizzazione

Poiché K-Means utilizza la distanza euclidea, è necessario rendere confrontabili le diverse dimensioni del vettore utente (ossia gli item). In assenza di scaling, pochi valori estremi potrebbero dominare la distanza e distorcere il clustering.
AcuteE stato adottato il `RobustScaler`[16] (scikit-learn), che normalizza ogni variabile sottraendo la mediana e dividendo per l'intervallo interquartile (IQR). Questa scelta è coerente con le caratteristiche del dataset:

- distribuzioni asimmetriche dei rating (spesso sbilanciate verso valori alti);
- presenza di outlier (utenti molto attivi o molto critici) che influenzerebbero media e deviazione standard;
- necessità di evitare che pochi utenti "estremi" comprimano la variabilità del resto della popolazione.

Si veda la sezione 3.9.3 per una descrizione più completa degli Scaler e i dettagli dell'algoritmo K-Means.

4.8.2 Scelta del numero di cluster

La scelta di K è stata effettuata combinando i due criteri complementari standard:

- *Elbow method* sull'inertia: si osserva l'andamento decrescente dell'inertia al crescere di K e si cerca un "gomito" oltre il quale i miglioramenti marginali diventano ridotti;

- *Silhouette score*[17]: misura la separazione tra cluster confrontando coesione interna e distanza dai cluster vicini.

Il valore finale di K è stato selezionato come compromesso tra:

- qualità quantitativa (silhouette non troppo bassa e inertia in diminuzione significativa);
- interpretabilità (numero di gruppi non eccessivo, cluster non troppo piccoli);
- stabilità (risultati simili a parità di random seed / inizializzazione).

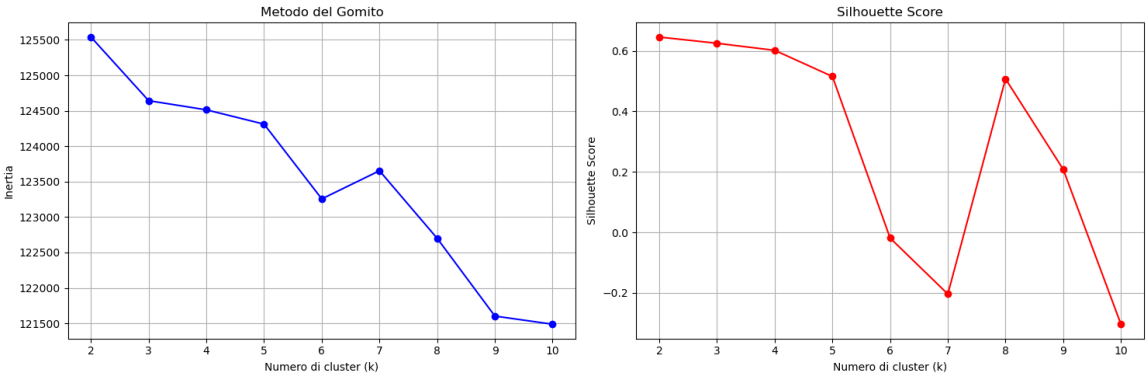


Figura 12: Inertia e Silhouette Score

In particolare, come possiamo osservare confrontando la figura 12 con la figura 6, i due grafici dell'Inertia e della Silhouette Score per le recommendation matrix prodotte da KNN e da SVD sono identici. Questo comportamento è attribuibile al fatto che, dopo la fase di normalizzazione, le due matrici inducono strutture di distanza tra utenti molto simili. In particolare, entrambi i modelli producono una rappresentazione dominata da una componente principale associata al rating medio globale, come evidenziato dall'analisi PCA. Di conseguenza, le differenze tra KNN e SVD, pur presenti a livello di predizione individuale, risultano poco influenti sulle metriche globali di clustering basate su distanza euclidea, quali inertia e silhouette score.
Ulteriore dimostrazione del fatto che sia dovuto alle matrici e non sia un errore nel codice è che, come si può osservare in figura 13, le *pairwise distances* sono uguali.
Per finire, alla luce delle considerazioni fatte sopra, ho scelto un k diverso da quello usato per

```
computato le pairwise distances knn
computato le pairwise distances svd
differenza abs: 0.0
```

Figura 13: Valore assoluto delle differenze pairwise tra tutti i punti delle due matrici di raccomandazione

la pipeline KNN, cioè $k = 8$ invece che $k = 4$. Questo perchè una sperimentazione con $k = 8$ nella sezione del KNN aveva evidenziato una struttura dei cluster meno significativa, con un cluster enorme con > 40.000 utenti, uno piccolo con > 20 utenti e tutti gli altri unitari, con un solo utente. Come vediamo nella sezione successiva invece, con SVD la situazione è diversa.

4.8.3 Analisi e interpretazione dei cluster

L'applicazione dell'algoritmo K-Means alla matrice di rating predetti dal modello SVD ha prodotto una segmentazione caratterizzata da un cluster dominante e da diversi cluster di dimensione ridotta, ma con profili comportamentali distinti. Questo risultato è coerente sia con

la struttura del dataset Amazon Books sia con le proprietà della Matrix Factorization.
Il cluster principale raccoglie la grande maggioranza degli utenti e rappresenta il comportamento medio appreso dal modello: utenti con poche interazioni e una marcata propensione a valutazioni positive. Tale cluster riflette una caratteristica intrinseca dei sistemi di recensione online, nei quali molti utenti contribuiscono con un numero limitato di rating prevalentemente alti.
Rispetto al clustering basato su KNN, l'SVD mostra una maggiore capacità di differenziare sottogruppi di utenti meno numerosi ma più informativi. I cluster di dimensione ridotta evidenziano profili distinti, tra cui utenti più critici (rating medio più basso), utenti con maggiore variabilità nei giudizi e utenti con comportamenti più coerenti e stabili. Questi gruppi emergono grazie alla rappresentazione latente appresa dall'SVD, che riesce a catturare differenze sottili anche in presenza di pochi dati osservati. L'analisi della similarità coseno tra i centroidi dei cluster (figura 15) mostra valori prossimi allo zero (e in alcuni casi lievemente negativi), indicando che i cluster occupano regioni ben separate dello spazio latente. Ciò suggerisce che i gruppi individuati non rappresentano semplici variazioni del comportamento medio, ma segmenti realmente distinti secondo il modello.
Infine, la proiezione PCA[12][15] (figura 16) evidenzia invece che la maggior parte della varianza è spiegata dalla prima componente principale, associabile a una tendenza globale dei rating. Tuttavia, la separazione dei cluster avviene lungo componenti a varianza inferiore, coerentemente con il fatto che le differenze tra utenti nei sistemi di raccomandazione sono spesso sottili ma strutturate.
Nel complesso, il clustering basato su SVD fornisce una segmentazione più informativa e articolata rispetto a quella ottenuta dal KNN, rafforzando il ruolo della Matrix Factorization non solo come strumento predittivo, ma anche come mezzo per l'analisi interpretativa delle preferenze degli utenti.

4.8.4 Considerazioni

A livello concettuale, il clustering SVD tende spesso a produrre gruppi più informativi rispetto al clustering basato su KNN, poiché l'SVD costruisce una rappresentazione latente che cattura differenze più sottili tra utenti anche quando il numero di interazioni osservate è ridotto. Inoltre, la regolarizzazione dell'SVD riduce l'effetto del rumore, consentendo ai cluster di riflettere maggiormente pattern sistematici piuttosto che variazioni casuali. In sintesi, il clustering non supervisionato costituisce una componente di analisi avanzata che arricchisce la valutazione del sistema di raccomandazione: non misura solo l'accuratezza predittiva (RMSE/MSE), ma fornisce anche insight sulla struttura degli utenti e sulla rappresentazione implicita delle preferenze appresa dai modelli. Nel confronto con il KNN, l'SVD mostra una maggiore capacità di

generalizzazione e una minore sensibilità alla sparsità, a fronte di una minore interpretabilità diretta. Il clustering basato sull'SVD tende inoltre a produrre una segmentazione più informativa e bilanciata degli utenti, rendendolo particolarmente adatto a fasi di analisi esplorativa avanzata.

```
### Analisi cluster:

Cluster 0:
        - Utenti: 40508
        - Rating medio: 4.348
        - Deviazione std rating: 0.981
        - Recensioni medie per utente: 1.5

Cluster 1:
        - Utenti: 10
        - Rating medio: 3.739
        - Deviazione std rating: 1.307
        - Recensioni medie per utente: 4.6

Cluster 2:
        - Utenti: 24
        - Rating medio: 4.294
        - Deviazione std rating: 0.965
        - Recensioni medie per utente: 2.1

Cluster 3:
        - Utenti: 15
        - Rating medio: 4.395
        - Deviazione std rating: 0.679
        - Recensioni medie per utente: 2.5

Cluster 4:
        - Utenti: 35
        - Rating medio: 4.247
        - Deviazione std rating: 0.830
        - Recensioni medie per utente: 2.7

Cluster 5:
        - Utenti: 11
        - Rating medio: 3.700
        - Deviazione std rating: 0.966
        - Recensioni medie per utente: 3.6

Cluster 6:
        - Utenti: 31
        - Rating medio: 3.686
        - Deviazione std rating: 1.273
        - Recensioni medie per utente: 1.6

Cluster 7:
        - Utenti: 23
        - Rating medio: 3.975
        - Deviazione std rating: 0.931
        - Recensioni medie per utente: 5.3
```

Figura 14: Analisi dei cluster finali

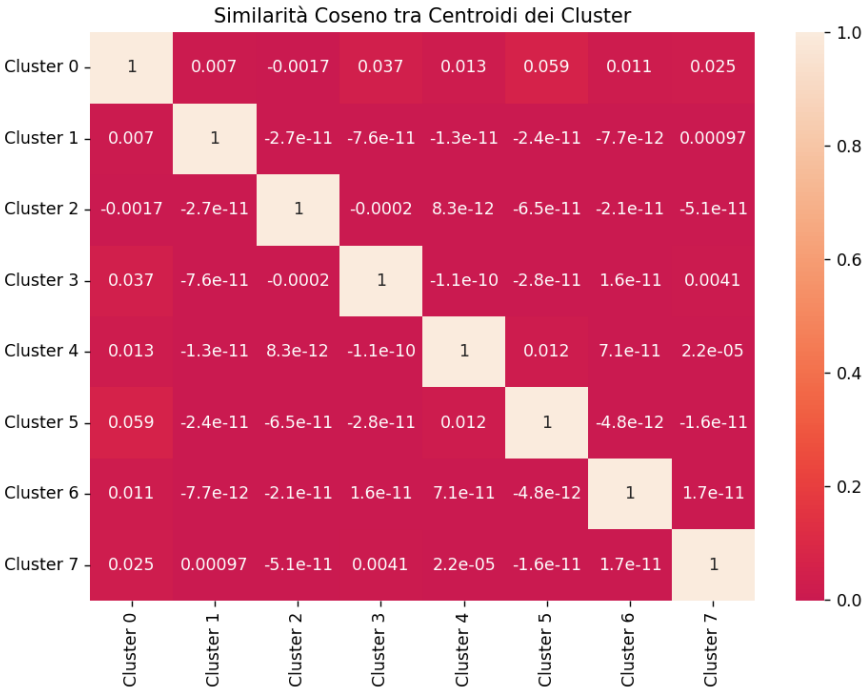


Figura 15: similarità del coseno tra i centroidi dei cluster.

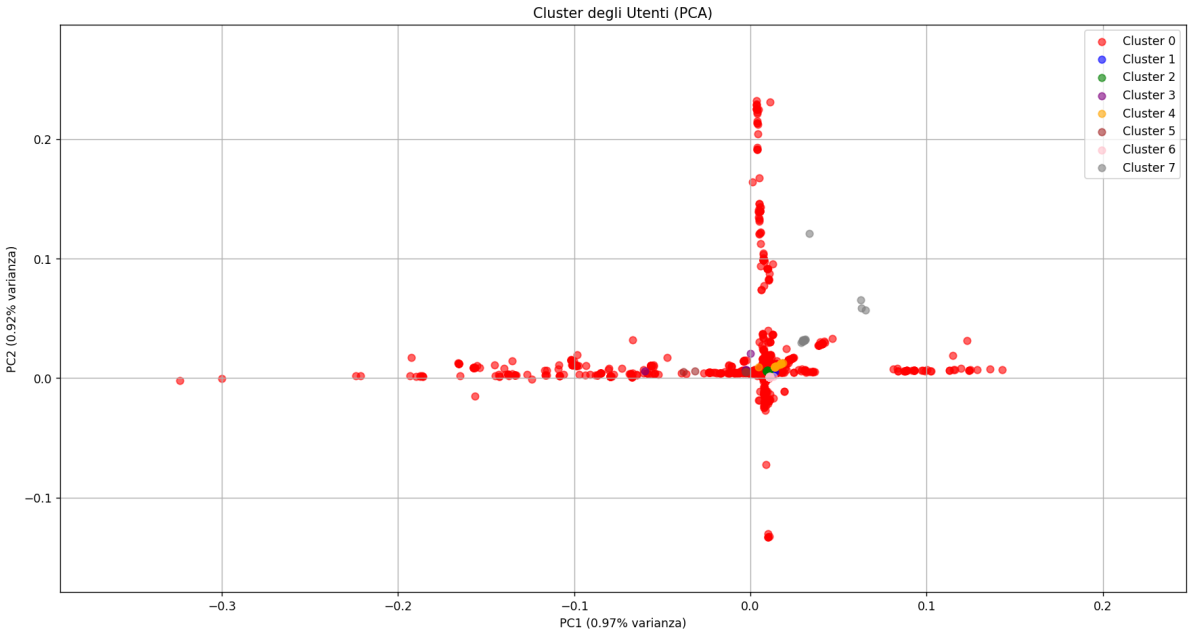


Figura 16: PCA dei cluster

5 Data Embedding

Nel progetto intermedio, il sistema di raccomandazione è stato esteso introducendo un approccio *content-based*, basato sull'analisi delle informazioni testuali associate ai prodotti. In particolare, sono stati utilizzati i metadati descrittivi dei libri (ad esempio titolo, descrizione e categorie) per costruire rappresentazioni numeriche dense (*embeddings*) in grado di catturare la similarità semantica tra gli item.

L'obiettivo principale di questa fase è duplice:

- ridurre la dipendenza esclusiva dalle interazioni utente-item, tipica del collaborative filtering;
- migliorare la capacità del sistema di gestire scenari di *cold-start*, in particolare per nuovi prodotti e utenti con poche o nessuna recensione.

5.1 Preprocessing dei testi

Prima della costruzione degli embedding, i testi dei metadati sono stati sottoposti a una fase di preprocessing, necessaria per ridurre il rumore e uniformare il formato dei dati. Le operazioni principali includono:

- concatenazione dei campi testuali rilevanti (titolo, descrizione, autore, categorie);
- conversione in minuscolo;
- rimozione di caratteri speciali e simboli non informativi;
- gestione di valori mancanti o descrizioni incomplete.

Questa fase garantisce una rappresentazione testuale coerente, adatta all'applicazione delle diverse tecniche di embedding.

5.2 Tecniche di embedding considerate

Nel progetto intermedio sono state sperimentate tre diverse tecniche di embedding testuale, caratterizzate da livelli crescenti di complessità e capacità di modellare la semantica dei testi: TF-IDF, CBoW (Continuous Bag-of-Words) e embedding basati su Transformer. L'obiettivo di questo confronto è valutare l'impatto della rappresentazione testuale sulla qualità delle raccomandazioni content-based.

5.2.1 TF-IDF

La tecnica TF-IDF (*Term Frequency - Inverse Document Frequency*) [21][22] rappresenta ciascun documento come un vettore sparso, in cui ogni dimensione corrisponde a un termine del vocabolario e il valore associato riflette l'importanza del termine all'interno del documento rispetto all'intera collezione.

Vantaggi

- semplicità concettuale e computazionale;
- elevata interpretabilità delle feature;
- buona efficacia nel catturare termini distintivi.

Svantaggi

- rappresentazione ad alta dimensionalità e fortemente sparsa;
- incapacità di catturare relazioni semantiche tra parole;
- sensibilità a sinonimi e variazioni lessicali.

TF-IDF è stato utilizzato come baseline per valutare i benefici introdotti da tecniche di embedding più avanzate.

5.2.2 CBoW (Word2Vec)

La seconda tecnica si basa su Word2Vec in modalità *Continuous Bag-of-Words* (CBoW)[23][24], in cui le parole vengono rappresentate tramite vettori densi appresi da un modello neurale addestrato sul corpus testuale. L'embedding di ciascun item viene ottenuto aggregando (tramite media "pesata") i vettori delle parole che compongono il documento.

Vantaggi

- rappresentazione densa e a dimensionalità ridotta;
- capacità di catturare relazioni semantiche di base tra parole;
- maggiore robustezza rispetto a TF-IDF sui sinonimi.

Svantaggi

- perdita dell'informazione sull'ordine e sul contesto delle parole;
- aggregazione semplice che può appiattire il significato globale;
- qualità dell'embedding dipendente dal corpus di addestramento.

Questo approccio rappresenta un compromesso tra semplicità computazionale e capacità semantica.

5.2.3 Embedding basati su Transformer

La terza tecnica utilizza modelli Transformer pre-addestrati per generare embedding contestuali dell'intero documento. Nel progetto è stato utilizzato il modello:
`sentence-transformers/all-MiniLM-L6-v2`[4], progettato specificamente per la produzione di sentence e document embeddings ottimizzati per il confronto semantico. Il testo completo associato a ciascun item viene tokenizzato e processato dal modello; l'embedding finale è ottenuto tramite *mean pooling*[4][14] sui token validi, seguito da una normalizzazione L2 per rendere stabile l'uso della similarità coseno.

Vantaggi

- embedding contestuali che catturano il significato globale del testo;
- elevata capacità di modellare relazioni semantiche complesse;
- prestazioni superiori nei task di similarità testuale e retrieval.

Svantaggi

- maggiore costo computazionale rispetto a TF-IDF e CBoW;
- minore interpretabilità delle singole dimensioni;
- necessità di una gestione attenta delle risorse di calcolo.

L'utilizzo di un modello Transformer consente di ottenere rappresentazioni semantiche più ricche e robuste, in linea teorica particolarmente adatte alle descrizioni dei libri che contengono informazioni complesse e articolate.

5.3 Analisi geometrico-strutturale degli embeddings

Per comprendere in modo più profondo le proprietà delle rappresentazioni testuali utilizzate nel progetto intermedio, è stata innanzitutto condotta un'analisi geometrico-strutturale degli embedding. Questa analisi ha l'obiettivo di caratterizzare lo spazio vettoriale generato da ciascuna tecnica, valutandone densità, struttura latente, distribuzione delle distanze e capacità discriminante, indipendentemente da un utilizzo diretto in un sistema di raccomandazione.

5.3.1 Misure considerate

Le principali misure utilizzate nell'analisi sono le seguenti:

- **Sparsità**: Misura la percentuale di valori nulli negli embedding. Elevata sparsità indica rappresentazioni ad alta dimensionalità con molte feature inattive, tipiche di approcci bag-of-words, mentre embedding densi tendono a codificare l'informazione in modo distribuito.
- **Norma L2**:La norma L2 dei vettori fornisce informazioni sulla scala degli embedding. Vettori normalizzati (norma costante) rendono la similarità coseno più stabile e confrontabile, mentre norme variabili possono influenzare le distanze euclidee.
- **Similarità coseno tra coppie casuali**: La distribuzione della similarità coseno tra coppie di item estratte casualmente descrive quanto lo spazio embedding sia concentrato o disperso. Valori medi molto bassi indicano uno spazio quasi ortogonale, mentre valori più elevati suggeriscono una maggiore coerenza semantica globale.
- **PCA e varianza spiegata (EVR)**: L'analisi delle componenti principali (PCA)[15] consente di stimare quanta informazione sia concentrata nelle prime direzioni dello spazio embedding. Una varianza spiegata elevata nelle prime componenti indica una struttura latente semplice; una distribuzione più uniforme suggerisce una rappresentazione più ricca e articolata.
- **Dimensione effettiva**: La dimensione effettiva approssima il numero di direzioni realmente informative nello spazio embedding. Valori bassi indicano una forte compressione dell'informazione, mentre valori più alti riflettono una rappresentazione semantica più distribuita.

5.3.2 Analisi degli embedding TF-IDF

Gli embedding TF-IDF sono caratterizzati da un'elevatissima dimensionalità (50 000) e da una sparsità pari a circa il 99.2%. Questa struttura riflette la natura puramente lessicale della rappresentazione, in cui ogni dimensione corrisponde a un termine specifico del vocabolario.
Dal punto di vista geometrico, i vettori TF-IDF risultano normalizzati in norma L2, producendo

una distribuzione delle similarità coseno tra coppie casuali fortemente concentrata attorno allo zero. Ciò indica che la maggior parte dei documenti è quasi ortogonale nello spazio vettoriale, condividendo pochi termini rilevanti.
L'analisi PCA mostra una varianza spiegata estremamente bassa: la prima componente principale cattura poco più dell'1% della varianza totale, e anche considerando decine di componenti la quantità di informazione spiegata rimane limitata. La dimensione effettiva relativamente elevata indica una rappresentazione distribuita ma priva di una struttura latente compatta.
Nel complesso, TF-IDF produce uno spazio adatto a confronti basati su sovrapposizione lessicale, ma poco strutturato dal punto di vista semantico.

5.3.3 Analisi degli embedding CBoW

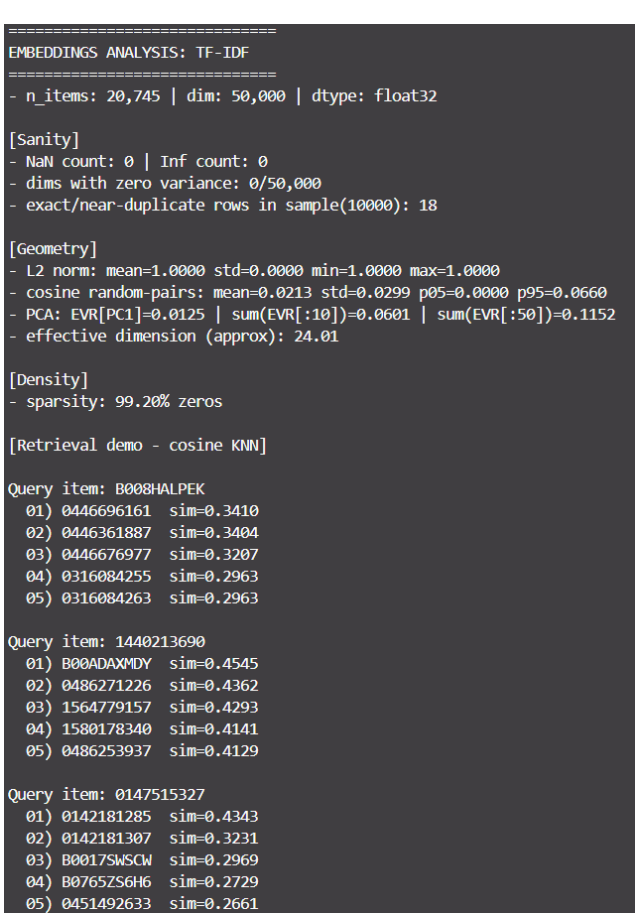
Gli embedding CBoW generano vettori densi di dimensionalità pari a 200, senza sparsità. I valori degli embedding presentano una distribuzione ampia e una norma L2 variabile, riflettendo l'assenza di una normalizzazione esplicita.
La distribuzione delle similarità coseno tra coppie casuali mostra valori medi significativamente più elevati rispetto a TF-IDF, indicando che lo spazio embedding è più compatto e che gli item condividono una maggiore affinità semantica di base.
L'analisi PCA evidenzia una forte concentrazione della varianza: la prima componente principale spiega circa il 46% della varianza, e le prime dieci superano il 75%. La dimensione effettiva stimata è molto bassa, suggerendo che lo spazio embedding è dominato da poche direzioni latenti principali.
Questa struttura indica che CBoW cattura efficacemente macro-temi semantici, ma tende a comprimere eccessivamente l'informazione, limitando la capacità di distinguere sfumature semantiche più fini.

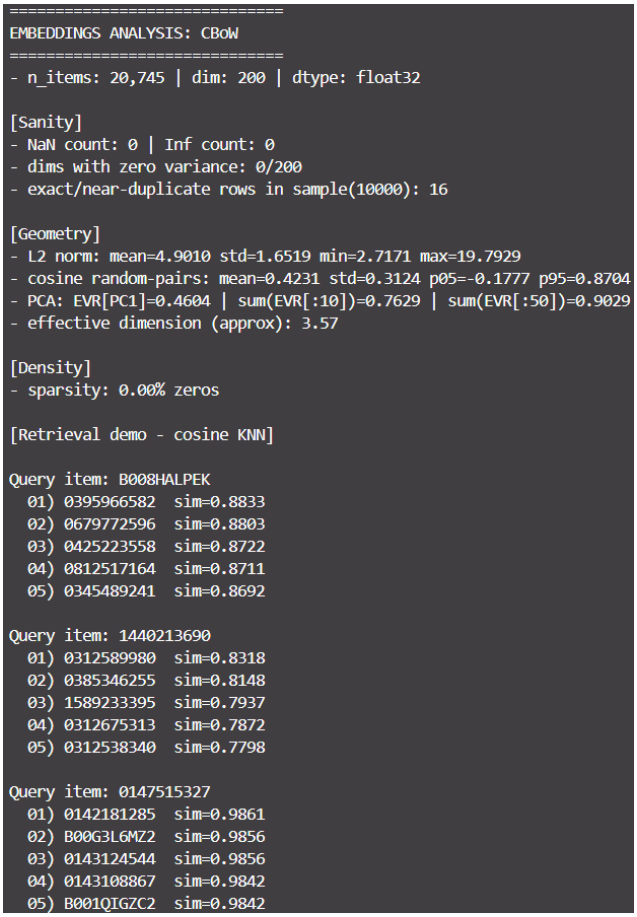
5.3.4 Analisi degli embedding basati su Transformer

Gli embedding basati su Transformer (`all-MiniLM-L6-v2`) producono vettori densi di dimensionalità 384, tutti normalizzati in norma L2. Questa caratteristica rende la similarità coseno una misura particolarmente stabile e significativa.
La distribuzione delle similarità coseno tra coppie casuali presenta valori medi intermedi, con una dispersione controllata. Ciò indica che lo spazio embedding consente di rappresentare gradi differenti di similarità semantica, evitando sia l'eccessiva ortogonalità di TF-IDF sia la forte compressione osservata in CBoW.
L'analisi PCA mostra una distribuzione più equilibrata della varianza: la prima componente principale spiega una frazione limitata della varianza, mentre un numero maggiore di componenti contribuisce in modo significativo. La dimensione effettiva elevata (circa 32) suggerisce una rappresentazione semantica ricca e distribuita su molte direzioni latenti.
Queste proprietà indicano che gli embedding Transformer riescono a catturare relazioni semantiche complesse e multilivello, risultando particolarmente adatti sia per il retrieval semantico sia per la costruzione di sistemi di raccomandazione content-based.

5.3.5 Considerazioni conclusive

L'analisi geometrico-strutturale evidenzia una progressione chiara tra le tecniche considerate: da rappresentazioni sparse e lessicali (TF-IDF), a embedding densi ma compressi (CBoW), fino a rappresentazioni dense, normalizzate e semanticamente ricche (Transformer). Questi risultati forniscono una base quantitativa e interpretativa solida per le scelte progettuali adottate nelle successive fasi del progetto.





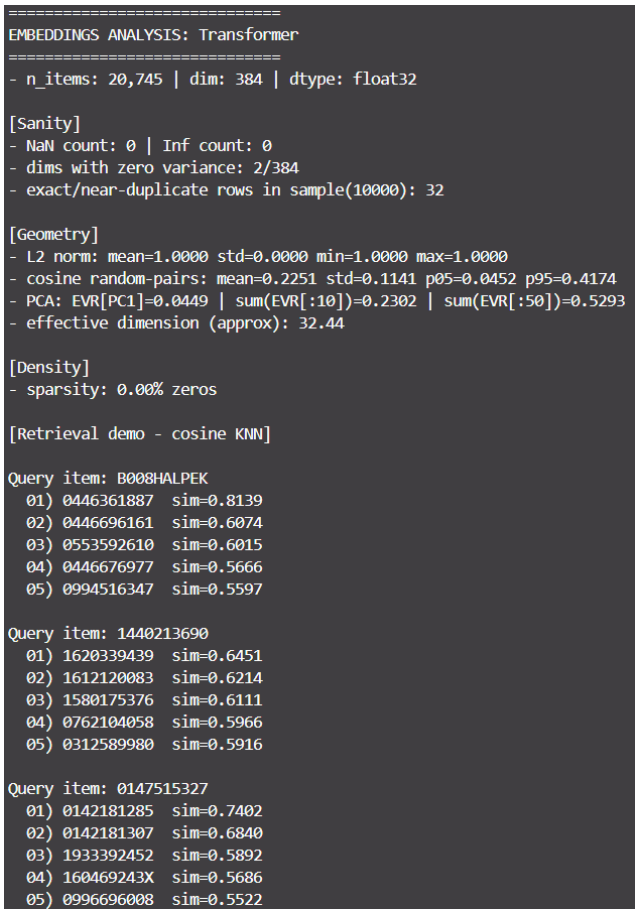


Figura 18: Analisi geometrica-strutturale dei tre embeddings

5.4 Analisi degli embeddings tramite clustering

Oltre all'analisi strutturale e geometrica degli embedding, è stata condotta un'ulteriore analisi basata sul clustering non supervisionato degli item tramite K-Means[10][8]. L'obiettivo di questa fase non è la costruzione diretta di un sistema di raccomandazione, bensì la valutazione della qualità delle rappresentazioni testuali dal punto di vista della loro capacità di organizzare semanticamente lo spazio degli item. Il clustering sugli embedding è sensato in questo contesto perché:

- gli embedding rappresentano direttamente il contenuto semantico degli item;
- cluster ben separati indicano che lo spazio embedding cattura strutture latenti significative;
- metriche come inertia e silhouette permettono di confrontare in modo quantitativo rappresentazioni diverse.

5.4.1 Clustering degli embedding TF-IDF

Nel caso degli embedding TF-IDF, il clustering K-Means mostra valori di inertia relativamente bassi ma una riduzione molto graduale al crescere del numero di cluster (fig 18). Questo comportamento è tipico di rappresentazioni ad alta dimensionalità e fortemente sparse, in cui le distanze tra i punti risultano poco informative.
La silhouette score assume valori molto bassi (dell'ordine di 0.01-0.02) per tutti i valori di k considerati, indicando una scarsa separazione tra i cluster. Ciò suggerisce che TF-IDF tende a distribuire gli item in modo quasi uniforme nello spazio, senza formare gruppi semanticamente ben definiti.
Questo risultato è coerente con le analisi precedenti: la similarità tra item dipende principalmente dalla sovrapposizione lessicale, e la mancanza di una struttura semantica globale limita la capacità di TF-IDF di produrre cluster interpretabili.

5.4.2 Clustering degli embedding CBoW

Gli embedding CBoW mostrano un comportamento sensibilmente diverso. L'inertia decresce in modo più marcato all'aumentare di k (fig 19), suggerendo una maggiore capacità di K-Means di identificare strutture compatte nello spazio embedding.
La silhouette score assume valori più elevati rispetto a TF-IDF, soprattutto per un numero ridotto di cluster. Questo indica che CBoW è in grado di produrre gruppi di item più coesi, sebbene la qualità del clustering diminuisca progressivamente all'aumentare del numero di cluster. Tuttavia, il rapido decadimento del silhouette score riflette il fatto che lo spazio CBoW è dominato da poche direzioni principali, come già evidenziato dall'analisi PCA. Di conseguenza, il clustering tende a separare macro-gruppi tematici, ma fatica a distinguere sottostrutture più fini.

5.4.3 Clustering degli embedding basati su Transformer

Il clustering degli embedding basati su Transformer presenta un comportamento intermedio ma più bilanciato. L'inertia decresce in modo regolare al crescere di k (fig 20), indicando una struttura dello spazio embedding più articolata rispetto a TF-IDF e meno compressa rispetto a CBoW.
La silhouette score rimane relativamente stabile su valori moderati per diversi numeri di cluster, suggerendo che lo spazio embedding consente una segmentazione più robusta e meno dipendente

dalla scelta specifica di k. Questo comportamento è coerente con una rappresentazione semantica ricca e distribuita, in cui le differenze tra item sono codificate lungo più dimensioni latenti. Dal punto di vista interpretativo, questo risultato indica che gli embedding Transformer riescono a catturare relazioni semantiche complesse, permettendo la formazione di cluster che non si basano esclusivamente su parole chiave comuni, ma sul significato globale delle descrizioni.

5.4.4 Confronto complessivo e interpretazione

Il confronto tra le tre tecniche evidenzia differenze chiare:

- TF-IDF produce uno spazio poco clusterizzabile, riflettendo una rappresentazione prevalentemente lessicale e non semantica;
- CBoW consente una segmentazione più netta, ma limitata a pochi macro-gruppi dominanti;
- gli embedding basati su Transformer offrono il miglior compromesso tra struttura, stabilità e capacità discriminante.

Nel complesso, l'analisi di clustering conferma e rafforza quanto emerso dalle analisi geometriche e di retrieval: gli embedding Transformer forniscono una rappresentazione più adatta non solo alla raccomandazione content-based, ma anche all'analisi esplorativa e interpretativa dello spazio degli item.

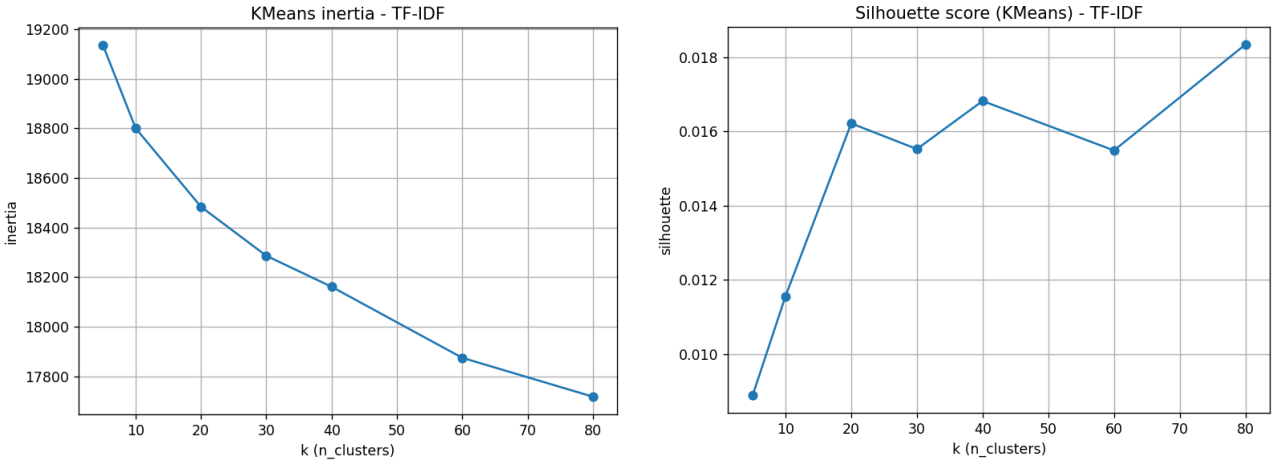


Figura 19: Inertia e Silhouette Score degli embeddings TF-IDF

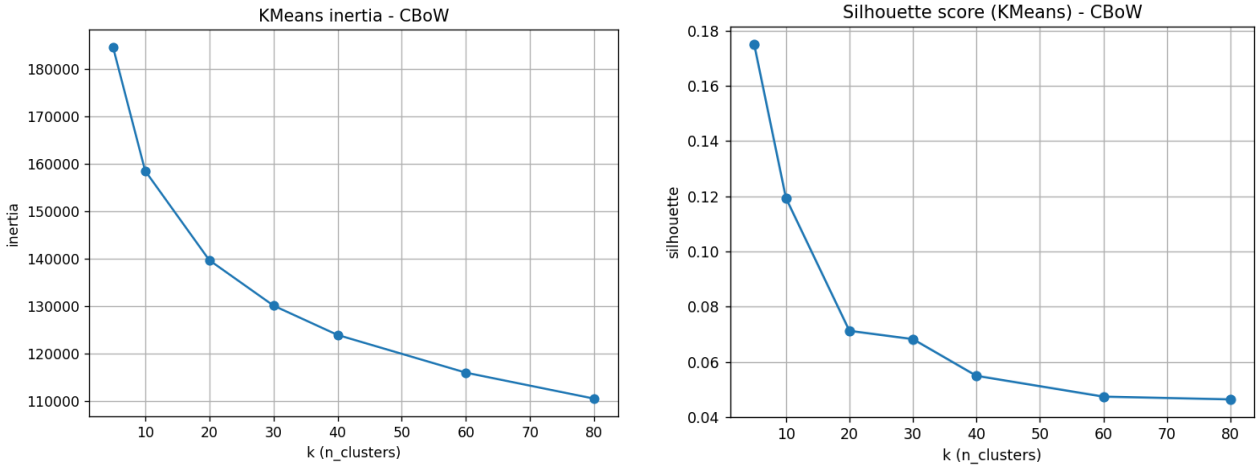


Figura 20: Inertia e Silhouette Score degli embeddings CBoW

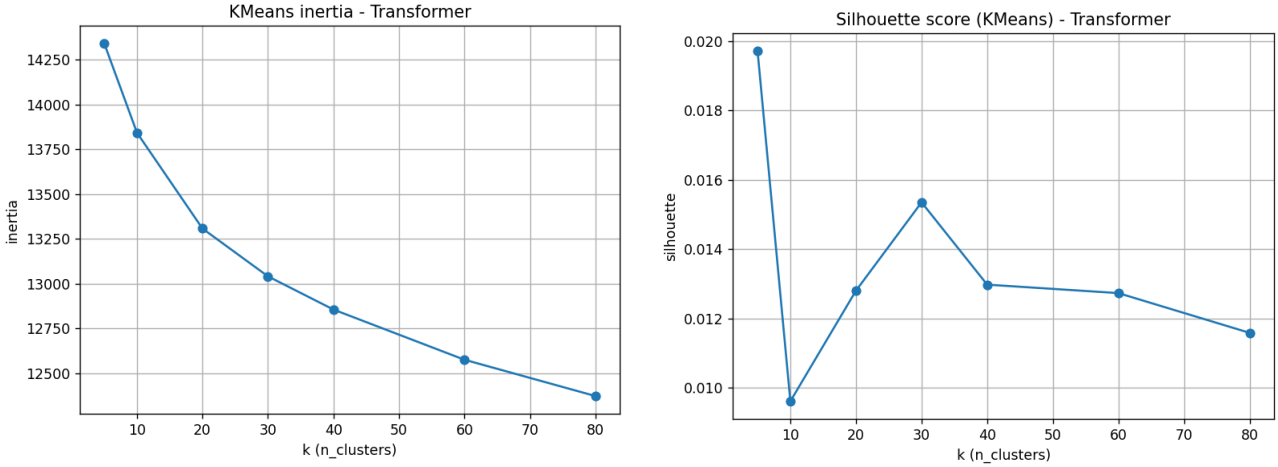


Figura 21: Inertia e Silhouette Score degli embeddings Transformer

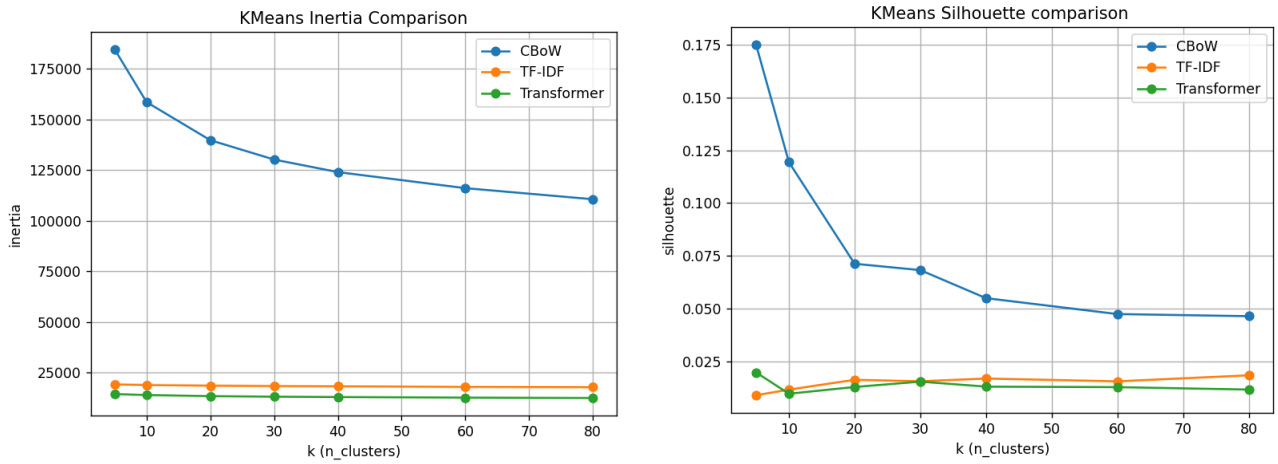


Figura 22: Inertia e Silhouette Score comparativa tra i tre embeddings

6 Content Based RS con KNN

In questa sezione viene presentato il sistema di raccomandazione content-based sviluppato nel progetto intermedio.
La sezione segue la stessa struttura adottata per i sistemi di collaborative filtering, al fine di rendere il confronto metodologico e sperimentale il più chiaro e coerente possibile.

6.1 Motivazioni dell'approccio content-based

L'introduzione di un sistema di raccomandazione content-based è motivata dai limiti strutturali del collaborative filtering, già emersi nel progetto base. In particolare:

- la dipendenza dalle interazioni storiche rende difficile gestire nuovi item;
- la sparsità del dataset limita la capacità di generalizzazione;
- i modelli collaborativi tendono a enfatizzare item popolari.

Nel dominio delle reviews di libri, i metadati testuali associati ai prodotti (titolo, descrizione, categorie) contengono informazioni semantiche ricche che possono essere sfruttate per individuare similarità tra item indipendentemente dalle interazioni degli utenti. L'approccio content-based consente quindi di generare raccomandazioni anche in scenari di *cold-start* sugli item, rappresentando un complemento naturale ai modelli collaborativi.

6.2 Data preprocessing

La fase di preprocessing per il sistema content-based differisce in modo significativo da quella adottata nel collaborative filtering, poiché l'unità di analisi principale non è l'interazione utente-item, ma l'item stesso e i suoi metadati testuali.
Come implementato nello script `content_based_RS.py`, il preprocessing prevede:

- selezione degli item per i quali sono disponibili metadati testuali sufficientemente informativi;
- concatenazione dei campi testuali rilevanti in un'unica stringa descrittiva;
- pulizia del testo ("minuscolizzazione" e rimozione di caratteri speciali);
- gestione esplicita dei valori mancanti o incompleti.

Rimane implementato un filtro sul numero minimo di recensioni per user e per item, e un sampling dei prodotti da analizzare, per evitare un'esplosione del tempo di calcolo. Il preprocessing è quindi orientato a massimizzare la qualità e la coerenza della rappresentazione semantica degli item.

6.3 Grid Search e analisi dei risultati

Per valutare in modo sistematico l'impatto dei diversi embedding testuali sulla qualità delle raccomandazioni content-based, è stata condotta una fase di ottimizzazione dei parametri basata su Grid Search[7]. L'obiettivo di questa analisi è duplice: da un lato confrontare le diverse rappresentazioni testuali in termini di accuratezza, dall'altro valutare il costo computazionale associato a ciascun approccio.
La valutazione quantitativa del sistema content-based è stata condotta tramite una Grid Search sul parametro k, che rappresenta il numero di item semanticamente più simili considerati per la

generazione delle raccomandazioni. La procedura è implementata nello script `content_based_RS.py`, con il supporto delle funzioni di valutazione definite in `functions.py`. Per ciascuna tecnica di embedding (TF-IDF, CBoW, Transformer), il sistema è stato valutato per:

$$k \in \{5, 10, 15, 20, 30, 40\}$$

calcolando RMSE, MAE e tempo di esecuzione.

TF-IDF Gli embedding TF-IDF mostrano valori di RMSE compresi tra circa 0.99 e 1.00, con variazioni limitate al variare di k. Il miglior risultato si ottiene per $k = 10$. Tuttavia, i tempi di esecuzione risultano significativamente più elevati rispetto agli altri approcci, a causa dell'elevata dimensionalità e della sparsità della rappresentazione.

CBoW Gli embedding CBoW presentano valori di RMSE leggermente inferiori rispetto a TF-IDF, con il minimo osservato per $k = 15$. I tempi di esecuzione sono notevolmente più contenuti, rendendo questo approccio un buon compromesso tra efficienza computazionale e qualità delle raccomandazioni.

Transformer Gli embedding basati su Transformer ottengono i risultati migliori in termini di accuratezza, con il valore minimo di RMSE osservato per $k = 5$. Questo indica che un numero ridotto di item semanticamente molto simili è sufficiente a produrre raccomandazioni efficaci, confermando la maggiore qualità informativa di questa rappresentazione.

```
### GRID SEARCH RESULTS
    k    rmse     mae  time_sec
1  10 0.98949 0.77841 143.22190
0   5 0.98988 0.78146 125.91992
2  15 0.99306 0.77868 166.12782
3  20 0.99454 0.77860 158.87114
4  30 0.99844 0.77903 156.04493
5  40 1.00147 0.77904 146.44873
```

```
### GRID SEARCH RESULTS
    k    rmse     mae  time_sec
0   5 0.99209 0.78508   6.39972
1  10 0.99250 0.78456   6.55444
2  15 0.99273 0.78326   6.42884
3  20 0.99671 0.78444   6.63577
5  40 0.99834 0.78090   6.36541
4  30 0.99897 0.78288   6.46587
```

```
### GRID SEARCH RESULTS
    k    rmse     mae  time_sec
2  15 0.98715 0.77823   6.54365
0   5 0.98819 0.78202   6.64495
1  10 0.98883 0.78049   6.57332
3  20 0.98889 0.77801   6.70666
4  30 0.99529 0.77825   6.68583
5  40 0.99796 0.77804   6.74974
```

Figura 23: Risultati dei tre GridSearch. Da sinistra a destra, TF-IDF, CBoW e Transformers

6.4 Confronto tra i risultati dei tre embedding

Il confronto tra i tre approcci evidenzia differenze sistematiche, sebbene molto contenute, nei valori di RMSE e MAE per le varie tecniche di embedding. Questa osservazione è coerente con la natura del problema: il sistema content-based non modella direttamente le preferenze individuali degli utenti, ma fornisce una stima basata sulla similarità tra item.
TF-IDF rappresenta una baseline semplice ma penalizzata sia dal costo computazionale sia dalla limitata capacità semantica. CBoW migliora l'efficienza e l'accuratezza, ma mostra una saturazione rapida dei benefici all'aumentare di k.
Gli embedding basati su Transformer emergono come la soluzione più efficace: non solo producono i valori di RMSE più bassi, ma lo fanno con un numero ridotto di vicini, riducendo la complessità della fase di aggregazione. Questo comportamento è coerente con le analisi geometriche e di clustering presentate nelle sezioni precedenti, che mostrano una rappresentazione semantica più ricca e meglio strutturata.
Questi risultati sono pienamente coerenti con quanto emerso dall'analisi geometrico-strutturale e dal clustering degli embedding, rafforzando la scelta di utilizzare gli embedding basati su Transformer come base principale del sistema content-based.

7 Conclusioni

In questo progetto sono stati sviluppati e analizzati diversi sistemi di raccomandazione applicati al dominio delle recensioni di libri, seguendo un percorso progressivo che va dal collaborative filtering basato su interazioni utente-item all'utilizzo di informazioni semantiche di contenuto tramite embedding testuali. L'obiettivo principale è stato valutare punti di forza e limiti di ciascun approccio, sia dal punto di vista dell'accuratezza predittiva sia da quello interpretativo.
Nel progetto base, i sistemi di collaborative filtering basati su KNN e SVD hanno mostrato prestazioni comparabili in termini di RMSE e MAE. Il modello KNN si è dimostrato semplice e interpretabile, ma fortemente dipendente dalla struttura della matrice utente-item e dalla sparsità dei dati. Il modello SVD, pur producendo miglioramenti limitati in termini di errore, ha mostrato una maggiore capacità di generalizzazione e una rappresentazione latente più informativa, come evidenziato dall'analisi di clustering degli utenti.
L'analisi dei cluster per entrambi i modelli collaborativi ha rivelato una struttura fortemente sbilanciata, con un cluster dominante che rappresenta il comportamento medio degli utenti e diversi cluster di dimensione ridotta. Questo risultato riflette una caratteristica intrinseca del dataset e suggerisce che, in presenza di utenti poco attivi, le metriche di clustering basate su distanza colgono principalmente differenze macroscopiche nei profili di rating.
Nel progetto intermedio, l'introduzione di un sistema di raccomandazione content-based ha permesso di superare alcune delle limitazioni del collaborative filtering, in particolare il problema del cold-start sugli item. L'analisi comparativa degli embedding testuali (TF-IDF, CBoW e Transformer) ha evidenziato una chiara progressione nella qualità delle rappresentazioni: da approcci lessicali e sparsi a rappresentazioni dense e semanticamente ricche.
Le analisi geometrico-strutturali e di clustering degli embedding hanno mostrato che gli embedding basati su Transformer producono uno spazio più articolato e meglio strutturato, in grado di catturare relazioni semantiche complesse tra i prodotti. Questo risultato è stato confermato anche dalla valutazione del sistema content-based tramite Grid Search, in cui gli embedding Transformer hanno ottenuto i migliori risultati in termini di RMSE, richiedendo al contempo un numero ridotto di vicini per la generazione delle raccomandazioni.
Nel complesso, i risultati indicano che nessun approccio è universalmente dominante: il collaborative filtering rimane efficace per utenti con una storia di interazioni sufficientemente ricca, mentre il content-based si dimostra particolarmente utile in scenari di sparsità e cold-start. L'integrazione di informazioni collaborative e semantiche emerge quindi come una direzione naturale per lo sviluppo di sistemi di raccomandazione più robusti e flessibili, ponendo le basi per ulteriori estensioni, come l'utilizzo del sentiment delle recensioni o di modelli ibridi più avanzati.

8 References

Riferimenti bibliografici

[1] Apache Software Foundation. *The .parquet data format*. 2013. URL: https://parquet.apache.org/docs/file-format/.

[2] *k-NN inspired algorithms*. 2015. URL: https://surprise.readthedocs.io/en/stable/knn_inspired.html.

[3] *Matrix factorization based algorithms*. 2015. URL: https://surprise.readthedocs.io/en/stable/matrix_factorization.html.

[4] Huggingface 'Community week'work. *all-MiniLM-L6-v2*. 2021. URL: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

[5] McAuley Lab. *Amazon Reviews '23*. 2024. URL: https://amazon-reviews-2023.github.io/.

[6] J. McAuley et Al. Y. Hou. «Bridging Language and Items for Retrieval and Recommendation». In: *arXiv preprint arXiv:2403.03952* (2024).

[7] *GridSearchCV*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[8] *K-Means clustering*. 2025. URL: https://en.wikipedia.org/wiki/K-means_clustering.

[9] *Kendall rank correlation coefficient*. 2025. URL: https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient.

[10] *KMeans*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[11] *MinMaxScaler*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[12] *PCA*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[13] *Pearson correlation coefficient*. 2025. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[14] *Pooling layer*. 2025. URL: https://en.wikipedia.org/wiki/Pooling_layer.

[15] *Principal Component Analysis*. 2025. URL: https://en.wikipedia.org/wiki/Principal_component_analysis.

[16] *RobustScaler*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[17] *Silhouette (clustering)*. 2025. URL: https://en.wikipedia.org/wiki/Silhouette_(clustering).

[18] *Singular Value Decomposition*. 2025. URL: https://en.wikipedia.org/wiki/Singular_value_decomposition.

[19] *Spearman's rank correlation coefficient*. 2025. URL: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient.

[20] *StandardScaler*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[21] *Tf-idf*. 2025. URL: https://en.wikipedia.org/wiki/Tf%E2%80%93idf.

[22] *TfidfVectorizer*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[23] *Word2vec*. 2025. URL: https://radimrehurek.com/gensim/models/word2vec.html.

[24] *Word2vec*. 2025. URL: https://en.wikipedia.org/wiki/Word2vec.