Report: Collaborative Filtering

Docente: Prof.essa Valentina Poggioni

Studente: Matteo Capparrucci (280278)

Introduzione

L'obiettivo di questa tesina è quello di fornire una panoramica generale nell'ambito dei Recommender System e, nello specifico, di quelli basati sulla tecnica del collaborative filtering basato sul calcolo delle misure di similarità User-User e Item-Item. Il dataset su cui sono stati svolti tutti i test è quello di MovieLens (sezione 1), mentre i Raccommender System utilizzati per i vari casi di studio sono dei semplici algoritmi di prova inventati da me.

1 Il dataset: MovieLens

MovieLens[1] è un sito nato con lo scopo di consigliare film agli utenti basandosi sullo storico delle preferenze degli stessi. Il gruppo di ricerca GroupLens ha ricavato dal sito una ricca serie di dataset[2] che, come anticipato, sono stati da me utilizzati per eseguire i test sulle performance dei vari reccomender system. I dati contenuti rappresentano la popolazione e le conseguenti votazioni del sito, dalla sua apertura fino a delle date specifiche. Per la precisione quelli utilizzati in questo lavoro sono:

- MoviLens100K: Stabile dataset di benchmark contenente $100,000$ ratings da 1000 utenti (ognuno con almeno 20 differenti recensioni) su circa 1700 film rilasciata nell'aprile 1998.
- MoviLens1M: Stabile dataset di benchmark contenente $1,000,000$ ratings da 6000 utenti (ognuno con almeno 20 differenti recensioni) su circa 4000 film rilasciata nel febbraio del 2003.

[1] http://movielens.org/

[2] il dataset è reperibile al link http://grouplens.org/datasets/movielens/

1.1 MoviLens100K

MoviLens100K è il più piccolo della famiglia di dataset MovieLens e conseguentemente il più maneggevole, per questo motivo è anche il dataset sul quale sono stati eseguiti il numero maggiore di test e il debug vero e proprio degli algoritmi.

Nella figura 3.1, osserviamo sull'asse delle y il numero di recensioni per ogni item e sull'asse delle x il numero di item aventi quel numero (y) di recensioni, una parte significativa degli oggetti recensiti nel dataset risultano avere un numero molto basso di recensioni. Nell'ottica di alleggerire il carico computazionale necessario per il calcolo delle misure di similarità e rendere queste più efficaci, eseguiremo dei test eliminando gli oggetti con numero di recensioni totali minore di 5.

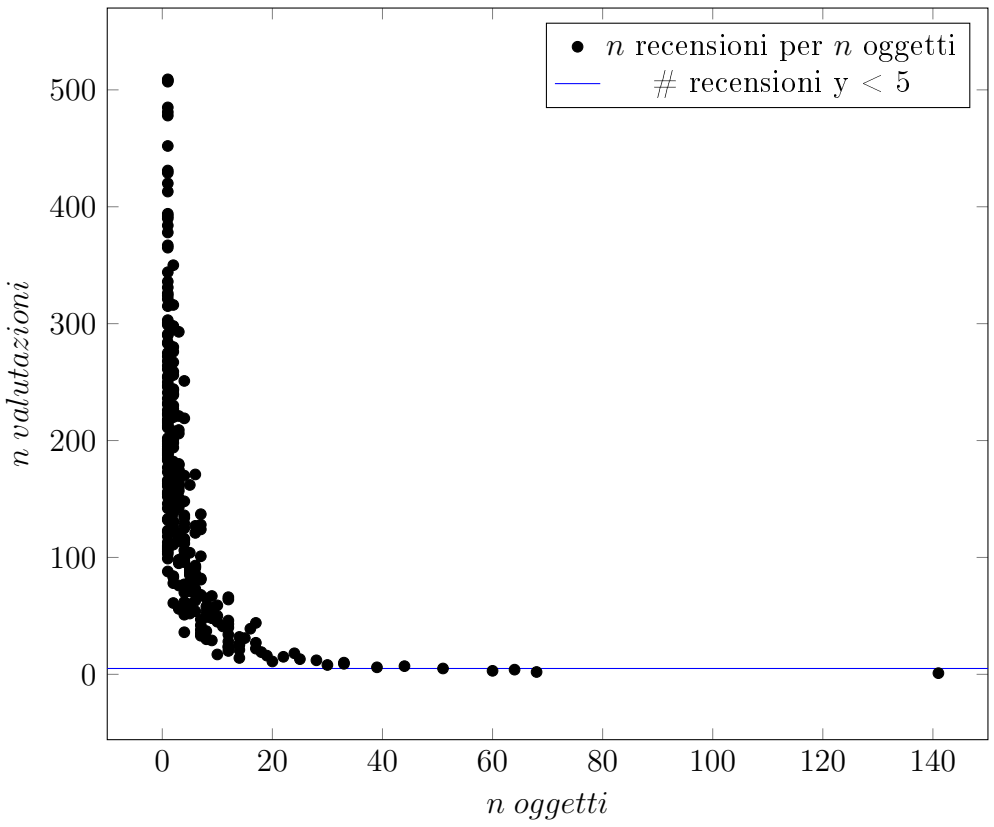

Figure 1: n oggetti x numero recensioni

1.2 MoviLens1M

MoviLens1M è un dataset di dimensione molto maggiore rispetto al precedente e dunque è stato utilizzato solo per pochi test dimostrativi vista la difficoltà (perlopiù temporale) data dal maneggiare un volume di dati di tali dimensioni.

Con il crescere del numero di review è possibile notare, come evidenziato in figura 3.2, un'inversione di tendenza rispetto al dataset di 'taglia' minore, in questo caso infatti il rapporto tra utenti e oggetti è sbilanciato dalla parte dei primi, che sono oltre 6000, in ogni caso, come con i precedenti è stato possibile eliminare una parte degli oggetti, eliminando quelli con numero di recensioni minori di 5, al fine di diminuire le dimensioni del dataset, e quindi consentire un calcolo più agevole della matrice di similarità e eliminare da essa i risultati meno significativi.

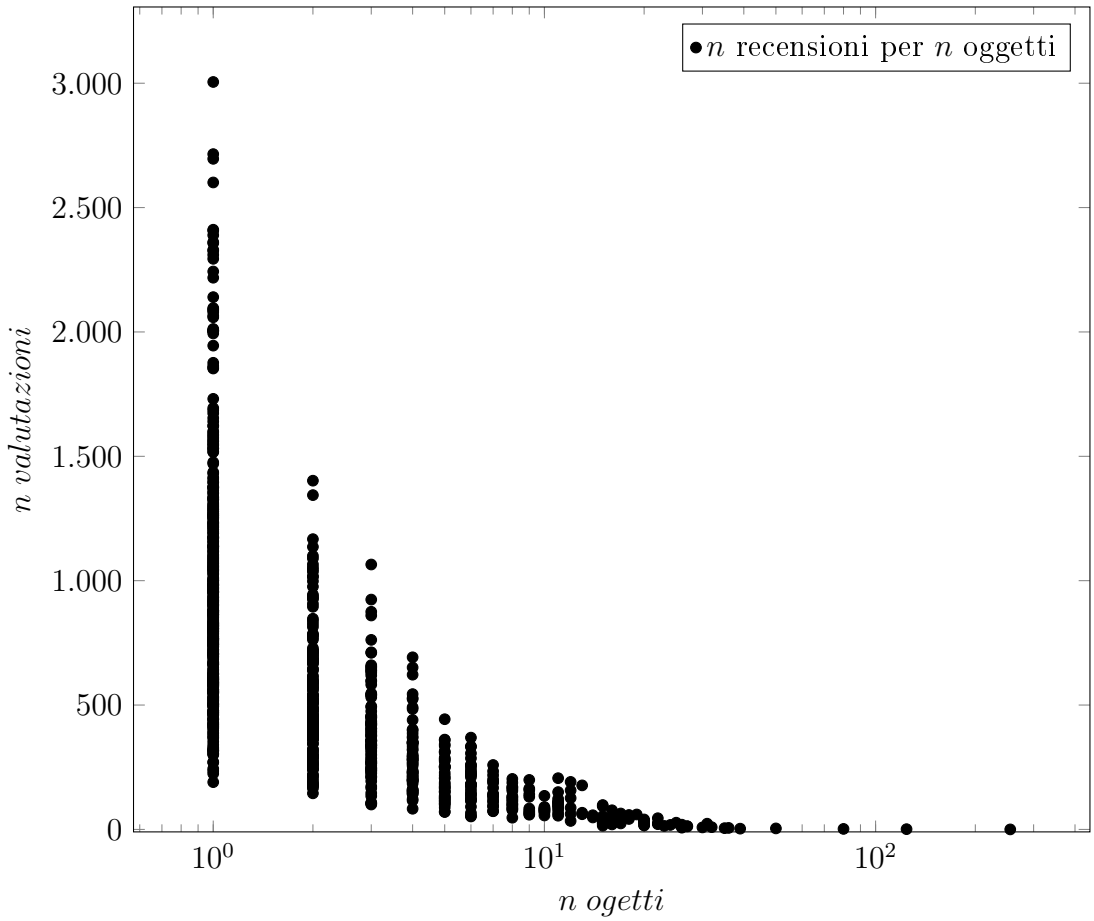

Figure 2: n oggetti x numero recensioni

2 Recommender System

Come anticipato, lo scopo di questi test era quello di valutare e testare alcuni semplici reccommender system, basati sulla tecnica del collaborative filtering, sui dataset MoviLens.

Un recommender system è un particolare tipo di algoritmo finalizzato alla formulazione di un rating o di una preferenza per un dato utente su un dato item. Nel caso dei reccomender system basati sul collaborative filtering questa 'predizione' viene formulata a partire da un modello costruito sulla

base di informazioni storiche degli utenti.

Entrambi gli algortimi proposti calcolano quindi per prima cosa una 'misura di similarità', sugli utenti o sugli item, utilizzando i dati sul comportamento passato degli utenti. Nel caso di algoritmi più evoluti vengono inoltre estratte delle ulteriori feature, anche non banalmente correlate al dato da predire, per raffinare la qualità dell'output.

In questo caso, comunque, la matrice di similarità sarà calcolata utilizzando la misura di similarità secondo Pearson, che è una delle più comuni e note.

2.1 Pearson Correlation Similarity

La similarità secondo Pearson è una delle misure più utilizzate per la realizzazione di recommender system basati sul vicinato (Nearest Neighbor algorithm), che rientrano nella famiglia dei collaborative filtering user-based. Nello specifico, questa consente di rappresentare il grado di 'similarità' tra due utenti x e y (nel caso della user-user) sulla base dei voti che questi hanno dato a degli oggetti comuni $r_{x,i}, r_{y,i} : i \in I$ e alla media aritmetica dei loro voti totali $\overline{r}_x$ e $\overline{r}_y$ (tramite l'equazione 1).

$$simil(x,y) = \frac{\sum_{i \in I_{x,y}} (r_{x,i} - \overline{r}_x)(r_{y,i} - \overline{r}_y)}{\sqrt{\sum_{i \in I_{x,y}} (r_{x,i} - \overline{r}_x)^2} \sqrt{\sum_{i \in I_{x,y}} (r_{y,i} - \overline{r}_y)^2}} \tag{1}$$

La stessa equazione, in ogni caso, può essere utilizzata anche per calcolare la similarità item-item selezionando opportunamente l'insieme dei vicini affinché rappresenti per ogni item x il gruppo di utenti I che hanno votato x.

Nel corso dei vari test, si è inoltre notato come l'equazione seguente fosse più efficiente rispetto alla precedente.

$$simil(x,y) = \frac{\sum_{i \in I_{x,y}} (r_{x,i} - \overline{r}_x)(r_{y,i} - \overline{r}_y)}{\sqrt{\sum_{i \in I_{x,y}} (r_{x,i} - \overline{r}_x)^2 (r_{y,i} - \overline{r}_y)^2}} \tag{2}$$

2.2 Recommender System User-based

Il primo algoritmo testato sui due dataset è un semplice recommender system basato sulla similarità user-user calcolata tramite Pearson (Eq 1) o sulla formula modificata (Eq 2).

Questo algoritmo prende in input la lista degli utenti, quella degli oggetti e il testset, scelto sulla base di un parametro numerico k che rappresenta il numero di predizioni da dover eseguire per ogni utente nel testset. Poi, per ogni utente u nel testset, cerca i k individui più simili a lui e da questi estrae i k *item* con le recensioni migliori, infine calcola la cardinalità $|item|$ di ogni *item* e 'raccomanda' i k *item* che compaiono più volte.

```
Procedure RecSys_UxU (TestSet, UserList, k1, k2, k3, k4)
begin
    foreach user u in TestSet
        Simil_u = select simil(u,i) > k4 : i ∈ UserList and i ≠ u
        foreach us in Simil_u
            RecList = select item : item ∈ u,us and rw(item) ≥ 4
        sort RecList by count(item)
        Out = RecList[0, ..., k3] : k3 > 0
        return Out
end
```

2.3 Dettagli Implementativi

Le procedure sono implementate nell'ottica di consentire l'impiego di grandi dataset, anche su macchine dotate di scarsa memoria. Per questo motivo la matrice ideale M composta da (Utenti x Oggetti) è rappresentata tramite liste di adiacenze. Ad ogni istante quindi sarà presente un'unica lista di oggetti, che rappresentano ognuno un utente nel dataset, accedibili direttamente tramite il loro id numerico univoco. Negli oggetti *usr* (o *itm* nel caso dell'item based) oltre a dati utili per la computazione della similarità, come ad esempio la medie dei voti delle review, viene allocata una lista contente i dati delle singole review. Le matrici di similarità, ovvero le matrici che contengono la misura di vicinanza tra tutti i vari utenti e oggetti, che all'atto pratico sono delle matrici diagonali superiori di float, sono invece calcolate riga per riga, ma per ognuna di queste viene mantenuto (e salvato) solo un sottoinsieme dei risultati formati dai valori maggiori di un certo k_4.

Infine, il coefficiente di similarità di pearson (2.1) ai fini di migliorare le performance generali del calcolo delle matrici di similarità, non è implementato sulla base dell'equazione 1, ma sulla seguente:

$$simil(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{r_{x,i}-\overline{r}_x}{s_x}\right)\left(\frac{r_{y,i}-\overline{r}_y}{s_y}\right) \tag{3}$$

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(r_{x,i} - \overline{r}_x)^2} \tag{4}$$

Per quanto riguarda il codice vero e proprio, è importante osservare le seguenti classi:

_initData : questa classe per prima cosa crea una sotto-cartella rispetto a quella del dataset (fornita come input nel main) e predispone la creazione del file LOG. Infine la classe initData restituisce la stringa $PATH$ contenente il percorso esatto da fornire in input ha tutte le altre classi in modo tale da avere i risultati delle varie esecuzioni nelle corrette sottocartelle.

_WriteMatrixCF : prende in input la variabile $PATH$, le dimensioni del dataset, e la percentuale di testset da estrarre (per ogni utente); legge da file tutte le recensioni in input salvando su due file "data_training" e "data_set" gli ogetti di tipo Usr e Itm opportunamente codificati. Se viene inoltre fornito il parametro "listaEsclusi" (di default settato a None) scarta sia per dataset che per datatraining tutte le recensioni contenenti gli oggetti contenuti in listaEsclusi.

_getMatrixCF : Prende in input la variabile $PATH$ e legge il file "PATH/data_training" per restituire la lista degli oggetti di tipo **Usr** contenente tutti i dati nel data training.

_getMatrixCF_TESTSET : Prende in input la variabile $PATH$ e legge il file "PATH/data _set" per restituire la lista degli oggetti di tipo **Itm** contenente tutti i dati nel data training.

_simil_UxU_ObjFull : calcola la matrice di similarità per tutte le righe del dataset utilizzando la similarità secondo pearson (equazione 4) implementata tramite la classe **_pearson**. In input richiede la lista degli utenti, le dimensioni del dataset, la variabile $PATH$, un valore di soglia k_4 e un booleano "Written" che se settato a $False$ calcolerà la matrice vera e propria mentre se settato a $True$ andrà semplicemente a leggerla da file. In output restituisce una lista di liste nella quale la prima rappresenta l'id numerico dell'utente a cui si riferisce la riga e la seconda contiene la lista degli utenti e il valore di

similarità di questi, esclusi tutti i valori con valore di similarità minore del parametro di soglia k_4.

_simil_UxU_ObjFull2 : Si comporta esattamente come la precedente ma questa volta restituisce una matrice calcolata sulla base dell'equazione 2.

_recSystemObjUxU : Prende in input una matrice di similarità, la lista degli utenti e quella del testet, e il parametro K_3 che definisce il numero di raccomandazioni da fare per ogni utente nel testset. Quindi esegue la procedura descritta nella sezione 2.2 restituendo una stringa conentente il risultato dell'intera esecuzione e un file (nella directory $PATH$) contenente l'output per ogni singolo utente.

3 Risultati Sperimentali

3.1 20% TEST SET

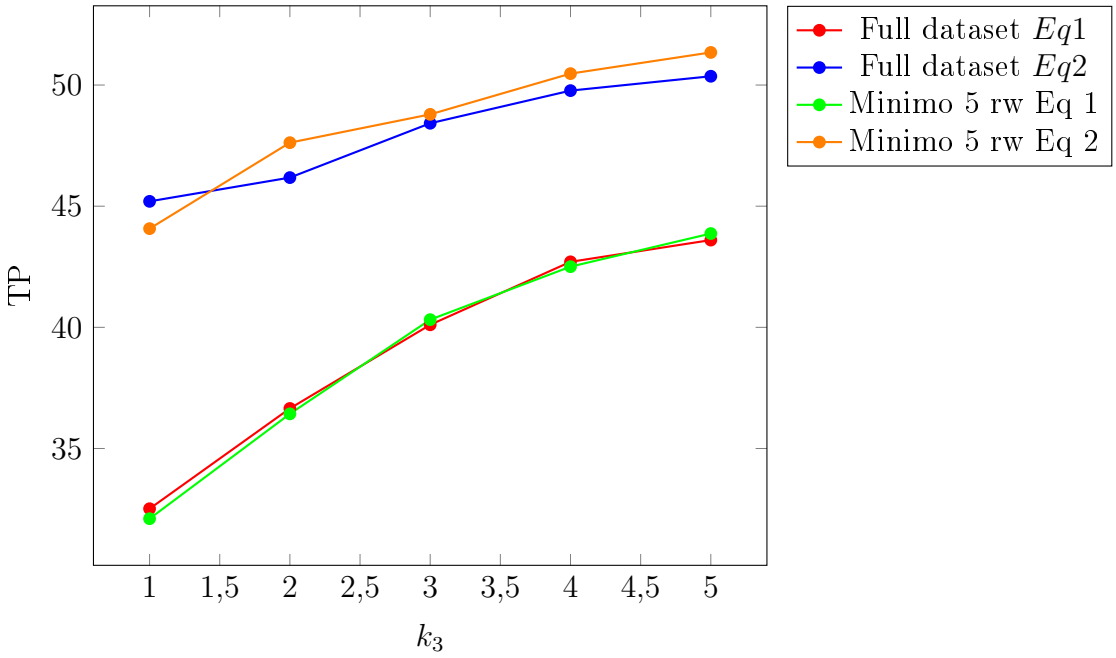

Figure 3: TP/Raccomandazioni effettive al variare del numero di raccomandazioni per utente.

3.2 30% TEST SET

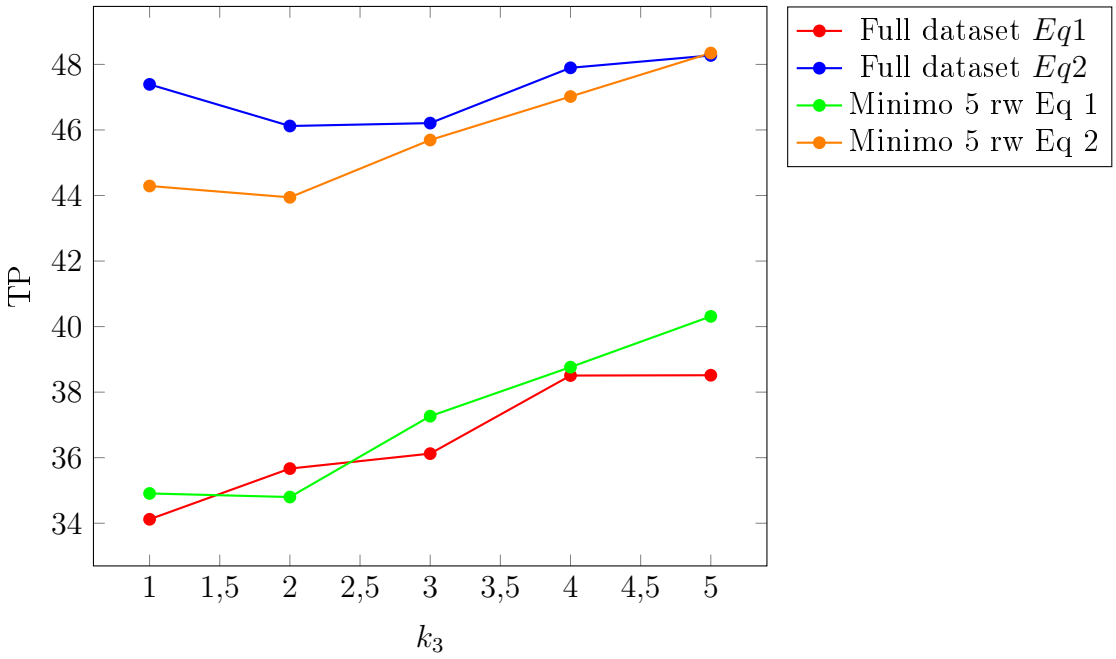

Figure 4: TP/Raccomandazioni effettive al variare del numero di raccomandazioni per utente.