Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Algoritmi di raccomandazione e Deep Neural Network

Francesco Sergio Pisani, Ettore Ritacco, Danilo Cistaro

RT-ICAR-CS-21-03 **Maggio 2021**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: *www.icar.cnr.it*

Sommario

Introduzione

Obiettivo del documento

Il presente documento ha come obiettivo quello di confrontare due approcci differenti per costruire un sistema di raccomandazione: uno basato sull'identificazione di similarità tra gruppi di utenti e uno più avanzato in cui si considera la qualità della predizione. L'obiettivo finale è quello di progettare sistemi di raccomandazione che possano essere di ausilio agli utenti delle piattaforme di visione di contenuti digitali con ampi cataloghi.
In questo documento verranno presentati due algoritmi di raccomandazione e i risultati ottenuti su un dataset benchmark.

Piano di dettaglio dell'attività

Nello specifico verranno mostrati i seguenti algoritmi:

1. BPR, descrizione e implementazione tramite rete neurale
2. SVAE, descrizione e architettura della rete neurale
3. Risultati sperimentali sul dataset movielens
    a. Descrizione del protocollo
    b. Risultati BPR
    c. Risultati SVAE

Definizioni e acronimi

NN	Neural Network
BPR	Bayesian Personalized Ranking
SVAE	Sequential Variational Autoencoder
MF	Matrix Factorization

Riferimenti

#	Titolo	Note	Link
[1]	BPR: Bayesian personalized ranking from implicit feedback		https://arxiv.org/abs/1205.2618
[2]	Modelling contextual information in session-aware recommender systems with neural networks		https://dl.acm.org/doi/abs/10.1145/2959100.2959162
[3]	Adversarial and Contrastive Variational Autoencoder for Sequential Recommendation		https://arxiv.org/abs/2103.10693
[4]	Sequential variational autoencoders for collaborative filtering		https://dl.acm.org/doi/abs/10.1145/3289600.3291007
[5]	Sequential recommendation with user memory networks		https://dl.acm.org/doi/abs/10.1145/3159652.3159668
[6]	Personalized top-n sequential recommendation via convolutional sequence embedding		https://dl.acm.org/doi/abs/10.1145/3159652.3159656
[7]	A theoretical analysis of NDCG type ranking measures		http://proceedings.mlr.press/v30/Wang13.html
[8]	Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents		https://www.sciencedirect.com/science/article/pii/S0952197605000825
[9]	Incorporating both qualitative and quantitative preferences for service recommendation		https://www.sciencedirect.com/science/article/pii/S0743731517303362

[10]	Matrix factorization techniques for recommender systems		https://ieeexplore.ieee.org/abstract/document/5197422/
[11]	A unified view of matrix factorization models		https://link.springer.com/chapter/10.1007/978-3-540-87481-2_24
[12]	Movielens dataset		https://grouplens.org/datasets/movielens

Algoritmi di raccomandazione - BPR

Il settore della ricerca nell'ambito della raccomandazione si è sempre concentrato su informazioni esplicite fornite dall'utente. Partendo dalle preferenze espresse da un utente si è cercato di costruire dei suggerimenti che potessero incontrare i suoi gusti. Con l'aumentare dei servizi online e degli utenti di tali servizi, costruire un profilo di riferimento è diventato un compito sempre più difficile. Allo stesso tempo, sistemi che non richiedono un'interazione attiva da parte dell'utente finale rappresentano un valore aggiunto perché agevolano l'utente nella ricerca di qualcosa da vedere, leggere, comprare, ecc. Di conseguenza, rappresentano un vantaggio per l'azienda. Il termine *implicit feedback* rappresenta il concetto fondamentale alla base di questo ambito di ricerca: si sfruttano delle informazioni implicite per costruire una lista di raccomandazione. Nell'ambito delle piattaforme di visione di contenuti video (film, serie tv, ecc.) si utilizza l'informazione su quali film l'utente ha già visto, quindi feedback implicito, e si suggerisce cosa potrebbe interessare all'utente basandosi su quanto hanno visto altri utenti con gusti simili.

L'algoritmo BPR (Bayesian Personalized Ranking) sfrutta il concetto di implicit feedback per modellare le preferenze dell'utente su uno spazio latente e usa questo modello per costruire una lista di raccomandazione. Non viene utilizzato il concetto di feedback negativo, cioè non si considerano eventuali preferenze negative ("non mi piace") fornite dall'utente.

L'obiettivo dell'algoritmo è quello di individuare una funzione tale da restituire uno score che privilegi i contenuti che l'utente trova preferibili e penalizzi ciò che l'utente non vorrebbe vedere. In pratica calcola uno score tale da costruire una lista di suggerimenti in cui i contenuti più vicini ai gusti dell'utente stiano in cima mentre il resto del catalogo si trovi nelle ultime posizioni. In maniera più precisa si può dire che si sta calcolando un sistema di ranking del catalogo in base alle preferenze dell'utente.

Descrizione dell'algoritmo

L'algoritmo considera le preferenze implicite. Se si considerano gli utenti e gli item come righe e colonne di una matrice NxM in cui N è il numero di utenti e M il numero di item (film, prodotti, ecc.) che l'utente "preferisce", le preferenze sono modellate come celle con un valore mentre per le altre non si hanno informazioni. Per ogni utente si indicano le preferenze esplicite come positive con il valore 1, mentre con il valore 0 si indicano gli item che non sono indicati nella lista delle preferenze dell'utente e vengono considerati come negativi.

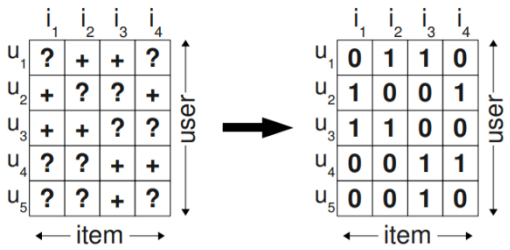


Figura 1 - Preferenze di un utente

L'algoritmo BPR si basa sulla valutazione di triple *(u, i, j)* in cui *u* è l'utente, *i* è un item positivo per l'utente *u*, cioè che l'utente preferisce e *j* è un item negativo, cioè che non preferisce. L'obiettivo è quello di calcolare uno score per *i* e *j* tale che lo score dell'item *i* sia maggiore di

quello dell'item *j*, cioè lo score deve valutare meglio gli item positivi rispetto ai negativi. In base a questo score verrà costruita la lista di raccomandazione.

Il modello BPR si basa sulla formulazione della seguente funzione di likelihood

$$p(\Theta|>_u) \propto p(>_u|\Theta)\, p(\Theta)$$

In cui Θ rappresenta il vettore dei parametri del modello e $>_u$ rappresenta le preferenze implicite dell'utente *u*. Quindi, per ogni utente, la relazione precedente lega i parametri del modello alle valutazioni dell'utente.

Partendo da questo concetto e facendo le seguenti due ipotesi semplificative:

1. Gli utenti agiscono in maniera indipendente
2. L'ordine di ogni coppia di item (i, j) non influisce sulle preferenze

Si può riscrivere la precedente esplicitando le preferenze relative agli item i e j come

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

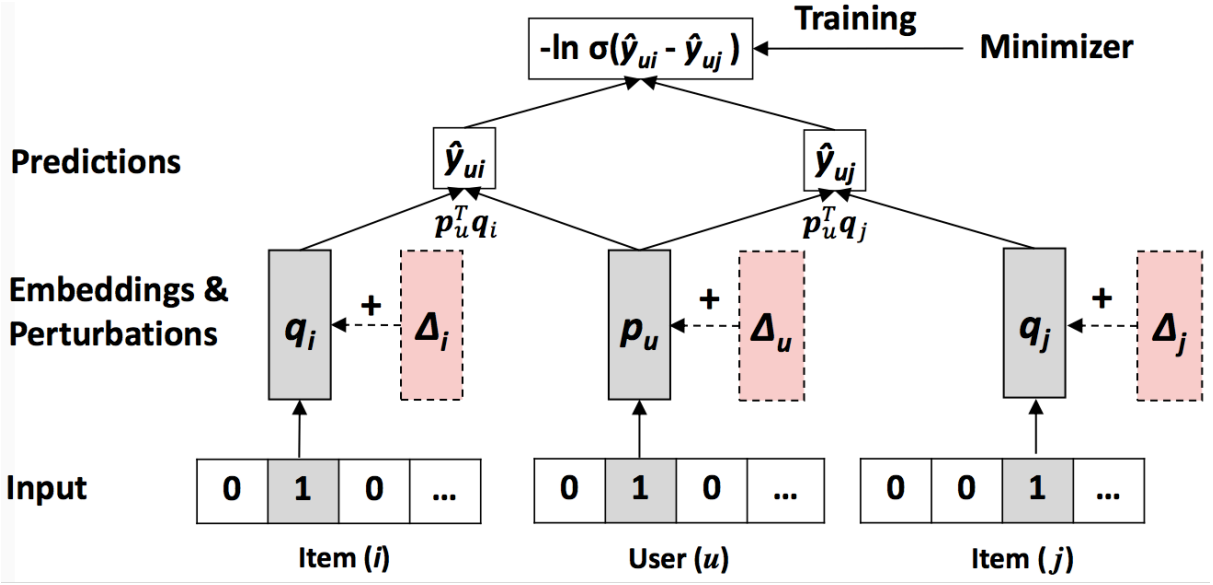
Dove il simbolo σ indica la funzione sigmoide. La relazione definisce che la probabilità che lo score di *i* sia maggiore dello score per l'item *j* per l'utente u è data dalla funzione x_{uij} che restituisce un valore che rappresenta la relazione tra l'utente *u* e le sue preferenze per *i* e *j*.
La funzione x_{uij} può essere stimata tramite una rete neurale oppure tramite dei metodi più tradizionali come la matrix factorization.
Partendo da questa formulazione e ipotizzando una funzione prior di riferimento come una distribuzione normale standard, è possibile definire il seguente stimatore da utilizzare per un processo di ottimizzazione

$$\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2$$

In cui il primo termine si riferisce alla probabilità calcolata dal modello e il secondo termine è il prodotto tra un fattore di regolarizzazione del modello per il vettore dei parametri che definiscono il modello.
Quello che si può notare è come il processo di ottimizzazione iteri su tutte le triple presenti in un training set di riferimento. Quindi il modello viene ottimizzato sulla base dei confronti positivi/negativi per ogni utente e quindi lo score restituito dipenderà dall'utente considerato.
Il protocollo di addestramento del modello BPR prevede la costruzione di un dataset costituito da una lista di triple (u, i, j) in cui *i* rappresenta gli item effettivamente visti dall'utente (positivi) e gli item *j* che non fanno parte delle preferenze dell'utente (negativi): il processo di ottimizzazione fa si che lo score associato ad *i* sia maggiore dello score associato a *j*.

Implementazione tramite rete neurale

La formulazione del BPR può essere vista come un framework generale poiché non impone dei vincoli su come realizzare il modello che calcola il contributo x_{uij}. Nell'ambito di questo progetto è stata utilizzata un'implementazione tramite rete neurale il cui modello è rappresentato nella figura seguente.



Il modello è basato su due layer di embedding: il primo stima il termine p_u mentre il secondo stima i termini q_i e q_j per gli item *i* e *j*. Lo score per gli item *i* e *j* è ottenuto dal prodotto *p*q*. In questo modo il modello apprende delle funzioni che combinate esprimono la relazione x_{uij}.
I blocchi indicati con il simbolo delta sono termini che facilitano la fase di apprendimento e possono essere considerati come una distorsione del dato in input.

Algoritmi di raccomandazione - SVAE

L'idea alla base del modello BPR assume che non ci sia relazione tra le coppie di item (i, j). Cioè suppone che per un utente due film qualunque abbiano lo stesso peso nelle sue preferenze. La ricerca più recente, invece, assume che la sequenza degli item abbia una sua semantica. In particolare l'ordine in cui vengono visualizzati dei film è importante, sia perché ci possono essere delle relazioni tra i film (i capitoli di una saga) sia perché i gusti di un utente possono variare nel tempo (dai film romantici sia passa alle commedie). Quindi i modelli più recenti introducono il concetto di sequenza nella modellazione.
L'algoritmo SVAE (Sequential Variational Autoencoder) basa il suo principio proprio sul concetto di sequenza. Gli item sono ordinati cronologicamente e l'obiettivo è quello di predire il prossimo item nella sequenza. A differenza del BPR, il concetto di score perde di significato a favore di una probabilità di comparire come prossimo item nella sequenza. La lista di raccomandazione viene costruita considerando gli item con probabilità decrescente.

Descrizione dell'algoritmo

L'algoritmo si basa sulla stima della probabilità che l'item *j* appaia come prossimo elemento nella sequenza degli item *1..i*. Si indica con *x* un generico utente e con il simbolo $x_{1:T}$ la sequenza di item associati all'utente *x* nell'intervallo temporale *1:T*. In questo caso l'intervallo temporale non considera il tempo intercorso tra due visualizzazioni successive ma è più semplicemente un indicatore della sequenzialità nella visualizzazione degli item, cioè l'utente visualizza prima l'item *i* e poi l'item *j*.
Quindi la probabilità di ottenere la sequenza $x_{1:T}$ è data dalla formula seguente

$$P\left(\boldsymbol{x}_{(1:T)}\right) = \prod_{t=0}^{T-1} P\left(\boldsymbol{x}_{(t+1)} | \boldsymbol{x}_{(1:t)}\right)$$

In cui si esplicita la dipendenza tra l'item successivo e tutti quelli che lo hanno preceduto. Si sta definendo il concetto che un utente visualizza un item in base a ciò che ha visto in precedenza.
Per calcolare questa probabilità si ricorre ad un modello generativo che "genera" le preferenze di un utente come una distribuzione multinomiale in cui le singole preferenze vengono generate secondo una distribuzione normale. Formalmente questo modello generativo viene definito ricorrendo alla notazione

$$\boldsymbol{z}_{u(t)} \sim \boldsymbol{N}\left(\boldsymbol{0}, \boldsymbol{I}_K\right), \qquad \pi\left(\boldsymbol{z}_{u(t)}\right) \sim \exp\left\{f_\phi\left(\boldsymbol{z}_{u(t)}\right)\right\},$$

$$\boldsymbol{x}_{u(t)} \sim \boldsymbol{Multi}\left(1, \pi\left(\boldsymbol{z}_{u(t)}\right)\right),$$

In cui la variabile *z* definisce lo spazio latente dei parametri della distribuzione delle preferenze che modella la storia dell'utente e $x_{u(t)}$ rappresenta le preferenze dell'utente al tempo *t*.
Utilizzando questo modello, si può ridefinire la probabilità precedente come una probabilità congiunta in cui si evidenzia la dipendenza tra gli item al tempo *t* e la variabile latente *z* al tempo *t*.

$$P(\boldsymbol{x}_{u(1:T)}, \boldsymbol{z}_{u(1:T)}) = \prod_t P(\boldsymbol{x}_{u(t)}|\boldsymbol{z}_{u(t)})P(\boldsymbol{z}_{u(t)}) .$$

A questo punto occorre definire come calcolare *z*. Quindi si ricorre ad una nuova formulazione che permette di realizzare un modello di NN in grado di approssimare questo calcolo. Si definisce la seguente probabilità condizionata che permette di calcolare *z* in funzione degli item $x_{1:t-1}$, cioè della storia dell'utente.

$$Q_\lambda(\boldsymbol{z}_{u(1:T)}|\boldsymbol{x}_{(1:T)}) = \prod_t q_\lambda(\boldsymbol{z}_{u(t)}|\boldsymbol{x}_{(1:t-1)})$$

In questo modo ho definito un modello che riesce a calcolare la probabilità che l'utente *x* visualizza l'item *j* al tempo *t+1* tramite una variabile latente *z* costruita considerando tutta la storia dell'utente fino al tempo *t*.
Per poter utilizzare un processo di ottimizzazione e costruire una rete neurale in grado di calcolare questa probabilità, è necessario definire una funzione obiettivo. In questo caso si definisce la seguente funzione di loss in cui per ogni utente *u* del dataset di training sono considerati due contributi. Il primo calcola l'errore di stima dei parametri del modello generativo, che in questo caso sono i parametri μ e σ di una funzione normale. Il secondo calcola l'errore di stima della probabilità del prossimo item da suggerire all'utente.

$$\mathcal{L}(\phi, \lambda, X) = \sum_u \sum_{t=1}^{N_u} \left\{ \frac{1}{2} \sum_k \left(\sigma_{\lambda,k}(t) - 1 - \log \sigma_{\lambda,k}(t) + \mu_{\lambda,k}(t)^2 \right) \right.$$
$$\left. - E_{\epsilon \sim \mathcal{N}(0,I)} \left[\log P_\phi \left(\boldsymbol{x}_{u(t)} | \boldsymbol{z}_\lambda(\epsilon, t) \right) \right] \right\} .$$

Tramite questo modello probabilistico si può costruire una rete neurale come la seguente in cui è possibile distinguere le tre componenti principali che definiscono l'architettura di rete e che approssima le funzioni definite dal modello probabilistico. I componenti GRU e ENC hanno il compito di stimare i parametri della distribuzione rappresentata dalla variabile z e che considerano la storia dell'utente, cioè tutti gli item fino al tempo *t*. Il componente DEC calcola la probabilità di un item al tempo *t+1*. Il layer di embedding codifica opportunamente l'input del modello.

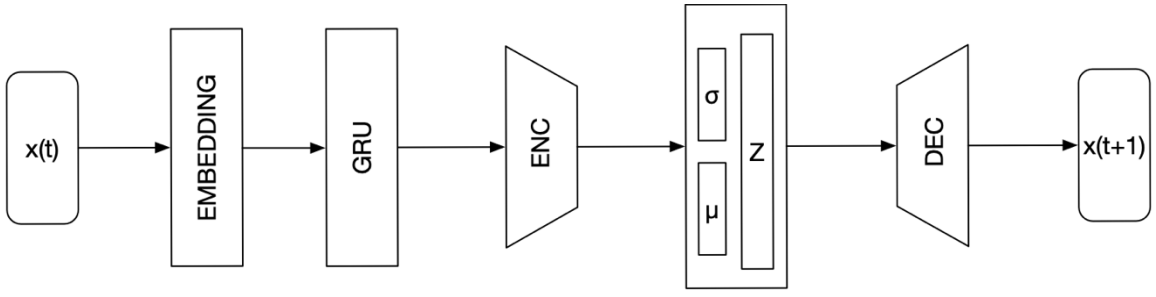


Figura 2 Modello di Neural Network

L'input del modello è una sequenza di item. L'output è la probabilità di essere il prossimo item associata ad ogni elemento della sequenza.

Risultati sperimentali

Per valutare le performance dei modelli descritti nei capitoli precedenti sono state effettuate delle prove su un dataset benchmark e in questo capitolo verranno presentati i risultati ottenuti. È stato scelto il dataset MovieLens perché rappresenta un indicatore comune in letteratura nell'ambito della raccomandazione.

Dataset

Il dataset MovieLens è stato creato all'interno del progetto di ricerca GroupLens dell'Università del Minnesota [12]. Lo scopo del progetto è di creare un sistema di raccomandazione per aiutare gli utenti nell'individuare nuovi film da vedere compatibili con i propri interessi. I dati raccolti all'interno del progetto sono stati resi pubblici, opportunamente anonimizzati, affinché studenti e ricercatori possano sviluppare nuovi algoritmi e tecnologie di personalizzazione e filtering.
Il dataset utilizzato durante la sperimentazione contiene i dati delle preferenze espresse da un gruppo di utenti che hanno utilizzato il portale MovieLens tra il 1997 e il 1998. La valutazione espressa da ogni utente è definita tramite una scala di 5 livelli, considerando il valore 5 come una valutazione massimamente positiva e il valore 1 come estremo opposto. Per ottenere un campione utile per analizzare le performance di un algoritmo, i dati sono stati selezionati seguendo tre criteri: tutte le preferenze devono avere una valutazione superiore a 4, gli utenti presenti devono aver espresso una preferenza per almeno 5 film e sono stati esclusi gli item con meno di 20 utenti che hanno espresso delle preferenze.
Il dataset contiene film suddivisi nei seguenti generi:

- Action
- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir
- Horror
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western

Inoltre nella tabella seguente vengono riportati i dati principali che descrivono il dataset dopo aver applicato i criteri di selezione definiti in precedenza.

Numero di utenti	6034
Numero di film	3533
Numero di preferenze	575271
Numero medio di film visti da un utente	95
Numero minimo di film visti da un utente	5
Numero massimo di film visti da un utente	1435
Numero mediano di film visti da un utente	58

Il grafico seguente mostra quanti film hanno visto gli utenti presenti nel dataset. La curva descrive un trend ben noto in cui pochi utenti guardano un numero elevato di film e molti utenti guardano un numero relativamente basso di film.

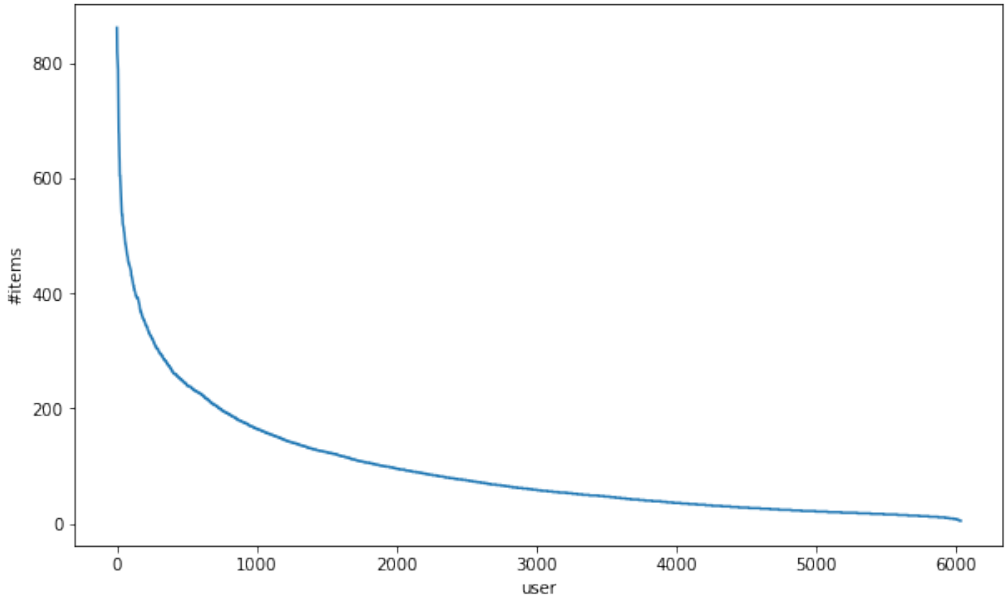


Figura 3 - Distribuzione del numero di film visti dagli utenti

Un trend simile è visibile nel grafico seguente dove è possibile vedere che sono pochi i film molto popolari mentre la maggior parte sono visualizzati da un numero basso di utenti. Questo comportamento replica quello che si può osservare nelle piattaforme di video on demand dove un ampio catalogo è dominato da pochi film mentre sono pochi i blockbuster, cioè quei film che catturano l'attenzione di un numero elevato di utenti.

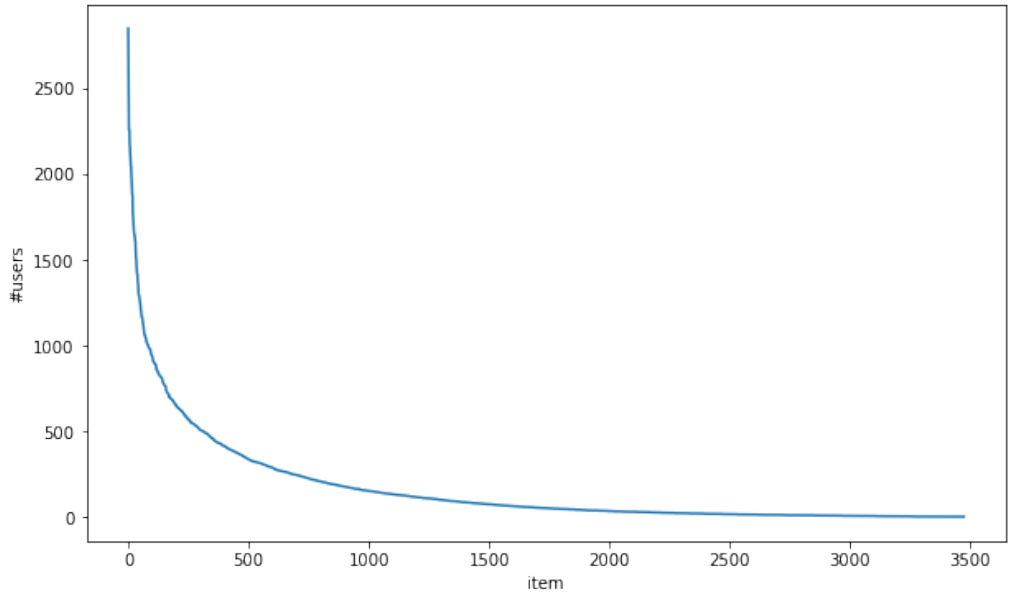


Figura 4 - Distribuzione dei film presenti nel dataset

Protocollo di valutazione

Per effettuare dei confronti è stato definito un protocollo comune per entrambi gli algoritmi.
Gli utenti sono stati suddivisi in tre gruppi distinti che costituiscono il dataset di training, validation e test. Il primo è utilizzato per addestrare i modelli, il secondo per scegliere il modello migliore e il terzo è utilizzato per valutare le performance. I risultati sono riportati nelle sezioni dedicate ai due algoritmi.
Per ogni utente la lista degli item di visione è ordinata cronologicamente e suddivisa in due parti. La prima è utilizzata come history, cioè cosa il sistema considera come storia dell'utente e in base alla quale predire il successivo item, mentre la seconda è utilizzata come sequenza da predire, cioè il primo elemento della sequenza è il prossimo item che il sistema dovrà suggerire.
Per valutare le performance degli algoritmi sono stati utilizzate tre metriche: precisione, recall e NDCG valutate su una lista di raccomandazione formata da *n* elementi.
La precisione è definita come

$$Prec@n = \frac{Hits@n}{n}$$

Dove *n* indica il numero di elementi nella lista di raccomandazione, *Hits* sono il numero di elementi correttamente predetti, cioè elementi che il modello suggerisce e che sono presenti nella seconda parte della lista degli elementi visti dall'utente.
La recall è definita come

$$Recall@n = \frac{Hits@n}{|R|}$$

Dove R è la dimensione della seconda parte della lista degli elementi visti dall'utente.
La metrica NDCG è definita come

$$NDCG@n = \frac{DCG@n}{IDCG@n}$$

$$DCG@n = \frac{r_i}{log_2(i+1)}$$

$$IDCG@n = \sum_{i}^{|R|} \frac{1}{log_2(i+1)}$$

Dove r_i indica la rilevanza dell'item, cioè se è presente o meno tra gli item correttamente predetti e indica la qualità della predizione nel suggerire gli elementi nella corretta sequenza temporale.

BPR

L'algoritmo BPR, considerando l'ordinamento cronologico degli item, non riesce ad ottenere dei risultati sufficienti per considerarlo come potenziale strumento di raccomandazione. Si nota come suggerendo 10 elementi, solo in 5.49 casi su 100 riesce a suggerire correttamente gli item. Anche considerando la qualità della predizione, si nota che si ha una precisione molto bassa e non riesce a "cogliere" la sequenzialità delle preferenze. Anche decuplicando la dimensione della lista delle raccomandazioni i risultati confermano questo trend. Aumentando di molto il numero di elementi suggeriti ci si aspetta un miglioramento delle metriche che si basano sul numero di elementi (recall) con conseguente peggioramento delle metriche di "qualità", cioè la precisione.
Questo tipo di risultati dimostra il problema del modello: non si tiene conto della sequenzialità delle preferenze e quindi la qualità della lista di raccomandazione ne risente.

Prec@10	7.84
Recall@10	5.49
NDCG@10	8.98
Prec@100	4.88
Recall@100	31.25
NDCG@100	17.39

SVAE

I risultati dell'algoritmo SVAE registrano dei miglioramenti significativi rispetto al modello BPR. Raccomandando 10 film, si riescono a ottenere dei risultati significativi sia in termini di precisione che in termini di qualità della predizione. La metrica NDCG suggerisce che il modello riesce a comprendere la corretta sequenzialità degli elementi in quasi 1 caso su 5. Questi numeri, seppur contenuti in valor assoluto, sono da valutare come un buon risultato considerando la difficoltà del problema. Inoltre rappresentano un miglioramento significativo rispetto al modello precedente.
Anche in questo caso aumentando la dimensione della lista di raccomandazione la precisione viene enormemente penalizzata. Mentre per le altre metriche si ha un miglioramento significiativo.

Prec@10	14.77
Recall@10	13.27
NDCG@10	18.27
Prec@100	6.91
Recall@100	49.42
NDCG@100	30.00

Conclusioni

In questo deliverable sono stati presentati due algoritmi per costruire sistemi di raccomandazione. Questi algoritmi sono rappresentativi di due concezioni contrapposte. La prima più tradizionale, in cui un sistema di raccomandazione dovrebbe suggerire gli item in base alle preferenze di utenti con interessi simili. Mentre la seconda è un'evoluzione della precedente in cui si considerano altri fattori per valutare la qualità dei suggerimenti che un sistema di raccomandazione dovrebbe proporre. In particolare si privilegia anche l'ordine in cui questi elementi vengono proposti, seguendo l'intuizione che la sequenza di visione per un utente è significativa per valutare la qualità della predizione. Cioè si cerca di considerare nella costruzione delle raccomandazioni anche l'evoluzione degli interessi di un utente nel tempo.
I risultati ottenuti tramite una sperimentazione hanno dimostrato come i nuovi modelli riescono a catturare queste caratteristiche e rappresentare un valore aggiunto rispetto ad algoritmi più tradizionali. I numeri confermano che la nuova generazione di algoritmi rappresenta una valida alternativa ad approcci più tradizionali e possono essere molto apprezzati dagli utenti delle sempre più numerose piattaforme di contenuti digitali. In questi servizi è facile trovare cataloghi molto ampi e nei quali spesso l'utente fatica nel trovare contenuti di proprio interesse. La complessità e la vastità dei cataloghi rappresenta uno dei problemi di queste piattaforme ed uno dei motivi di abbandono da parte dell'utente. Quindi migliorare la qualità delle raccomandazioni può essere uno strumento efficace per facilitare la scoperta di nuovi contenuti.
Inoltre, per entrambi gli algoritmi, sono state utilizzate implementazioni tramite reti neurali. Uno dei limiti delle prime implementazioni di algoritmi come il BPR è la numerosità degli item e degli utenti da utilizzare per la stima dei parametri del modello. Al crescere del numero di item e di utenti, le risorse computazionali da utilizzare e il tempo necessario alla computazione crescono vertiginosamente limitando la possibilità di realizzare sistemi complessi. Tramite l'impiego di reti neurali, invece, si riescono a raggiungere prestazioni ragguardevoli in tempi più che ragionevoli. Anche per algoritmi che non sono stati progettati specificatamente per questa tecnologia.