MyMediaLite

Introduzione

Evoluzione del framework MyMedia. A differenza di quest'ultimo, può essere usato per fini commerciali, può essere usato anche senza un database e contiene un set differente di algoritmi di recommendation, meno complessi rispetto a quelli del suo predecessore. È una libreria utilizzata per il Common Language Runtime (CLR). È scritto in C# ed è utilizzato per il collaborative filtering.

Per citarlo nel codice:

```
@inproceedings{Gantner2011MyMediaLite,
    author    = { Zeno Gantner and Steffen Rendle and
Christoph Freudenthaler and Lars Schmidt-Thieme },
    title     = { {MyMediaLite}: A Free Recommender System
Library },
    booktitle = { Proceedings of the 5th ACM Conference
                  on Recommender Systems (RecSys 2011) },
    year      = 2011
}
```

Permette di effettuare rating prediction e item prediction usando feedback impliciti (come click, likes, acquisti…).

La libreria è compatibile con diversi tipi di file, grazie ad un'interfaccia (**IDataSet**) che definisce delle funzioni base che poi vengono implementate in modo diverso da classi distinte. Un'altra interfaccia (**IdentityMapping**) definisce funzioni che permettono di "convertire" gli ID delle entità, in modo che si possano unire più dataset diversi senza avere problemi di ID ripetuti. Ci sono:

Rating files, contenenti tabelle relative ai voti che gli utenti hanno dato agli item. Le colonne delle tabelle sono:

- User ID
- Item ID
- Rating
- Data e ora (facoltativo)

Positive only feedback files, in cui nelle tabelle compaiono solo l'ID dell'item e quello dell'utente. Queste tabelle considerano le interazioni positive degli utenti con gli item Le tabelle contengono due colonne:

- ID dell'entità (utente o item che sia)
- ID dell'attributo

Attribute files, in cui sono rappresentate eventuali relazioni tra utenti o tra item. Ad esempio, “Il film A è il sequel del film B”. Tali relazioni non sono simmetriche. Le tabelle contengono due colonne:

- ID dell’entità
- ID dell’entità ad essa correlata

Entity files, contenenti gli ID degli item e degli utenti. Gli id degli utenti e quelli degli item sono poi contenuti in due liste separate di interi.

Item Recommendation files, contenenti gli ID degli utenti seguiti dagli ID degli item, con i rispettivi rating predetti, raccomandati per quell’utente. Gli item sono ordinati in maniera decrescente a seconda dei voti.

ESEMPIO:

0 [9:3.5, 7:3.4, 3:3.1]

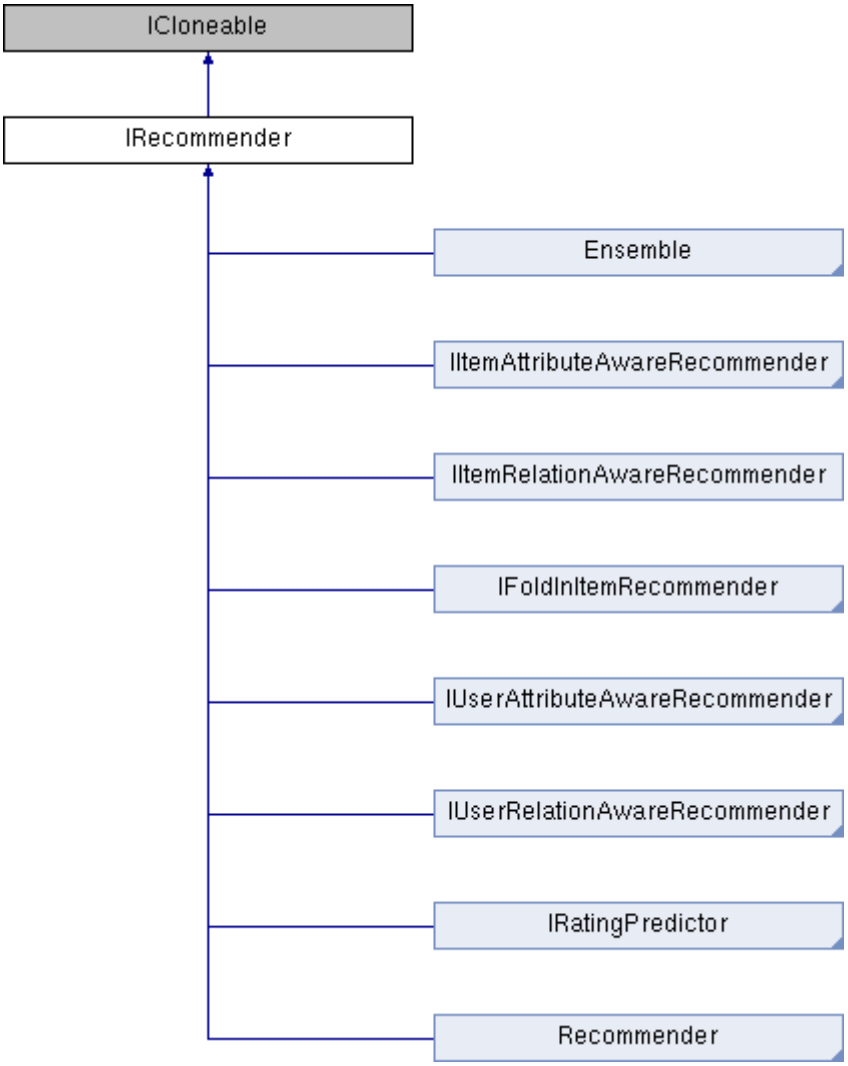
Vuol dire che all’utente 0 vengono consigliati gli item 9, 7, 3 rispettivamente con un rating predetto di 3.5, 3.4, 3.1.

Strutture dati

- **IBooleanMatrix**
- **IList**
- **TimedRatings:** Struttura dati per conservare ratings con informazioni sul timestamp
- **StaticRatings**: Struttura contenente tre vettor: uno per gli utenti, uno per gli item e uno per i rating
- **RatingsPerUserChronologicalSplit:** Struttura che consente di trattare i rating fino ad una certa data come dati di training, e quelli successivi come test

Architettura del framework

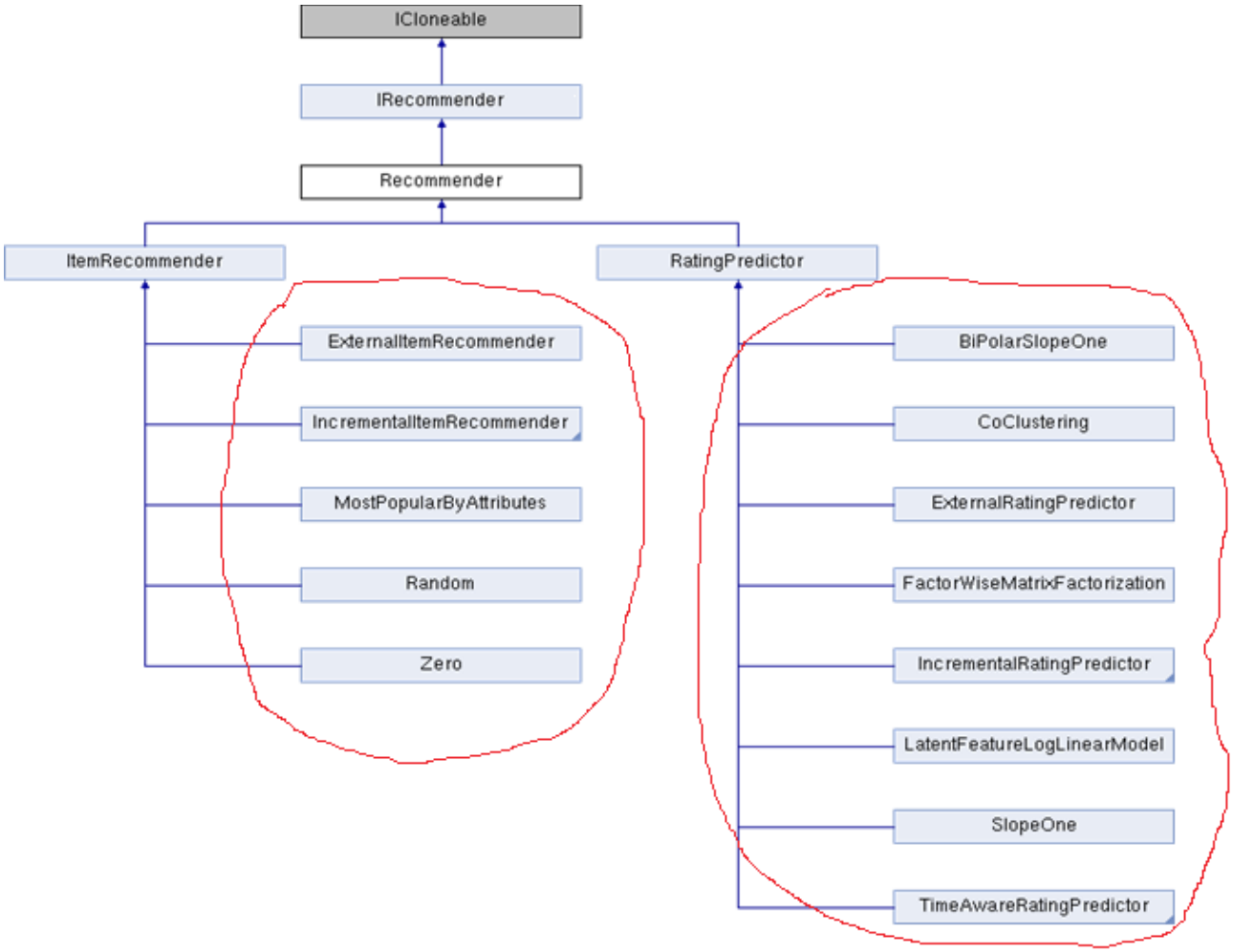
L’interfaccia principale prende il nome di **IRecommender**.



L'interfaccia mette a disposizione i metodi astratti principali che poi saranno implementati diversamente dai vari algoritmi. Questi metodi sono:

- Predict
- CanPredict
- Recommend
- Train
- LoadModel
- SaveModel
- ToString

Ci sono interfacce diverse a seconda delle relazioni che si vogliono prendere in considerazione. L'interfaccia IItemAttributeAwareRecommender ad esempio è per i recommender che prendono in considerazione relazioni binarie tra gli attributi degli item, mentre IItemRelationAwareRecommender riguarda i sistemi che considerano relazioni binarie tra gli item.

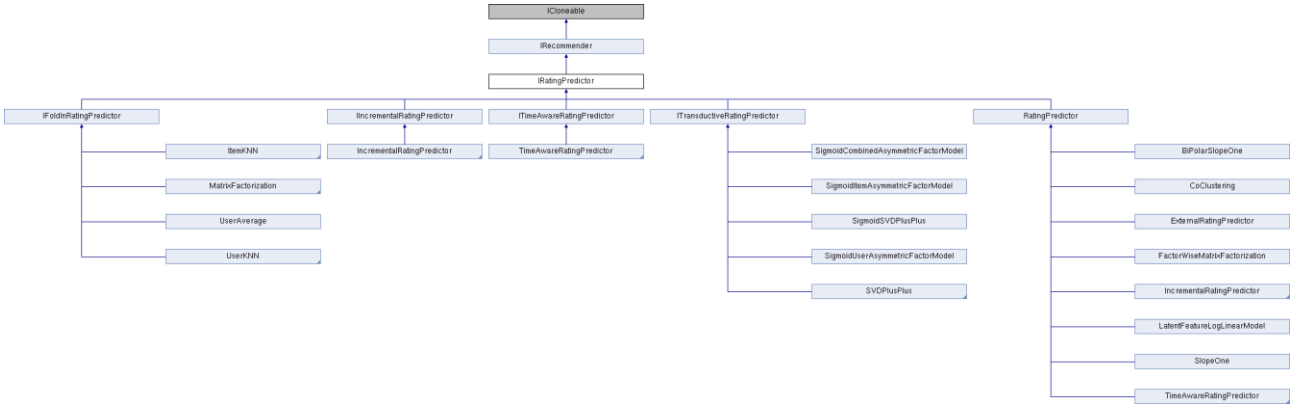


Le classi astratte ItemRecommender e RatingPredictor sono quelle che rispettivamente si occupano dei due task. Entrambe confluiscono nella classe astratta **Recommender**, che implementa alcuni dei metodi presenti enll'interfaccia IRecommend.

In ognuna delle due classi appena descritte confluiscono le classi relative ai diversi metodi memory-based che si possono utilizzare per la item recommendation e la rating prediction.

Le classi cerchiate in rosso sono quelle che definiscono le varie tecniche di recommendation e rating prediction. Ognuna di esse implementa diversamente i metodi Train e Predict. Queste ereditano la struttura delle classi genitori e, oltre ad implementare i metodi in esse definiti, ne possono aggiungere altri se necessario. Le classi cerchiate possono a loro volta essere delle classi astratte implementate a livelli inferiori.

L'interfaccia **IRatingPredictor** è quella dedicata alla rating prediction.



In essa confluiscono ulteriori altre interfacce che rappresentano diverse tecniche utilizzabili. Si può notare che l'ultima classe a destra nell'immagine sopra è la classe RatingPredictor. Essa dunque implementa sia l'interfaccia IRecommend che la classe astratta Recommender.

Cliccando sul link è possibile accedere al diagramma delle classi del framework:

http://mymedialite.net/documentation/doxygen/interface_my_media_lite_1_1_i_recommender.html

Metodi usati per l'item recommendation

https://lib.ugent.be/fulltxt/RUG01/002/304/661/RUG01-002304661_2016_0001_AC.pdf

Il tool per effettuare item recommendation è attivabile mediante il seguente comando:

item_recommendation --training-file=FILE --recommender=METHOD [OPTIONS]

Permette di specificare il file usato per la fase di training, eventualmente uno usato per il test, e l'algoritmo per ricavare le raccomandazioni, per il quale si possono settare le opzioni. Si può anche specificare il numero di item da consigliare e il file in cui salvare le predizioni.

- Raccomandazioni **random**
- Algoritmo **Zero**, che in pratica si comporta come un algoritmo che suggerisce item random, ma può essere utilizzato per la fase di debugging
- Raccomandazioni di item **più popolari**
- **BPRMF**: Approccio che effettua matrix factorization in cui la decomposizione della matrice principale viene svolta dall'algoritmo BPR. Si basa sui rating impliciti dagli utenti. Si serve dell'algoritmo BPR, che considera tre tipi di giudizi impliciti: **positivo** per gli item che un utente ha osservato, **negativo** per gli item che sono stati raccomandati ma che l'utente ha scelto di non osservare, **ignoto** per quegli item che l'utente non ha osservato perché non ha mai avuto l'occasione di farlo. Quest'ultimo aspetto lo differenzia dalle altre tecniche simili, che considerano come

negativi tutti gli item che l'utente non ha osservato. Funziona molto bene soprattutto se la tecnica di ranking utilizzata è la tecnica AUC (Area Under the Curve)

- **Multicore BPRMF** utilizza la tecnica BPRMF, ma il modello viene suddiviso in livelli su cui opera in maniera parallela. Funziona meglio rispetto al BPRMF
- **Weighted BPRMF** simile al BPRMF, ma in aggiunta considera dei pesi che vengono attribuiti agli utenti o agli item in base alla loro popolarità. Questo permette di limitare il problema del cold start. Questo può portare gli item più popolari ad essere preferiti a quelli meno popolari, per questo vengono inserite ulteriori variabili che limitano questo bias. Questo algoritmo ha un'accuratezza minore rispetto ai due precedenti. Ciò è dovuto al minor overfitting che garantisce.
- **WRMF**: (**Weighted Regularized Matrix Factorization**) (http://yifanhu.net/PUB/cf.pdf). Altro approccio che utilizza i feedback impliciti per fattorizzare una matrice. Questo algoritmo minimizza il tasso di errore. Il tasso di precisione è molto simile a quello del BPRMF, con cui condivide anche il fatto di non considerare l'overfitting.
- **Algoritmi di clustering basati sul k-nearest neighbor**, che generano una lista di item o user vicini, ordinati in base alla similarità con l'utente o l'item che si sta testando, e poi restituiscono i primi k elementi della lista. In particolare abbiamo:
    - **Item k-NN** e **user k-NN**, che usano la similarità del coseno per calcolare la distanza tra item o utenti. In particolare, l'item k-NN calcola la similarità tra item basandosi sui feedback impliciti espressi dagli utenti. Lo user k-NN invece individua gli utenti più vicini all'utente a cui si sta cercando di dare raccomandazioni, usando il principio della wisdom of crowd.
    - **Item-Attributes k-NN**: Clusterizza gli item tenendo conto degli attributi che hanno in comune, senza considerare i feedback degli utenti. Utilizzato nei content based recommender system. Solitamente questo algoritmo è meno preciso rispetto a quelli usati per il collaborative filtering, inoltre richiede che nel dataset vengano esplicitate le informazioni necessarie per effettuare i vari calcoli, per questo non è compatibile con alcuni dataset come MovieLens.
    - **User-Attributes k-NN:** Clusterizza gli utenti in base ai loro attributi in comune. È preferibile usarlo per trovare i vicini di un utente, e poi applicare l'algoritmo User k-NN per fornire raccomandazioni
- **Sparse Linear Methods (SLIM):** Algoritmi che hanno prodotto buoni risultati in termini di velocità ed accuratezza, in quanto usano a loro vantaggio la sparsità della matrice W. Tale matrice contiene i coefficienti di aggregazione tra gli utenti e gli item. Questi coefficienti sono calcolati basandosi sui giudizi impliciti o espliciti che gli utenti hanno dato agli item.
    - **LeastSquare SLIM:** Il calcolo della matrice W viene effettuato tramite un'ottimizzazione del metodo dei minimi quadrati. Dovrebbe fornire buoni

risultati a run time, ma per un test effettuato su MovieLens l'algoritmo ha impiegato 15 minuti per apprendere il modello. Dunque è possibile che questo algoritmo non sia stato implementato correttamente durante la realizzazione del framework.
    - **BPR-SLIM:** Utilizza il metodo BPR-Opt per calcolare la matrice W. È leggermente meno preciso e accurato rispetto agli altri algoritmi.
- Inclusione di recommender system esterni, che non sono stati implementati nella libreria e che si vuole aggiungere o testare.
  Comando per incorporare un recommender esterno: *item_recommendation --training-file=u1.base --test-file=u1.test -- recommender=ExternalItemRecommender --recommender-options="prediction_file=FileName"*

Metodi usati per la rating prediction

Il framework permette di utilizzare i seguenti metodi per predire il rating di un utente per un item:

- **UserAverage:** Media tra i voti dell'utente
- **ItemAverage:** Media tra i voti che gli altri utenti hanno dato all'item
- **GlobalAverage:** Media tra tutti i voti dati da tutti gli utenti a tutti gli item
- **Item-Attribute k-NN / User-Attribute k-NN**
- **Item k-NN / User k-NN**
- **Random**
- **Constant:** Predice sempre lo stesso valore
- **NaiveBayes**
- **SlopeOne:** Algoritmo in cui la deviazione tra due item è data dalla media delle differenze tra i voti dati da utenti che hanno votato entrambi gli item. Questa è la formula della predizione del voto per l'utente u sull'item i:

$$\hat{r}_{u,i} = \bar{r} + \frac{1}{|R_u - \{i\}|} \sum_{j \in |R_u - \{i\}|} d_{i,j}$$

    - **BipolarSlopeOne**: Variante dell'algoritmo slope-one, che divide l'insieme di item votati dagli utenti in due sottoinsiemi: item che sono piaciuti e item che non sono piaciuti. Ad entrambi gli insiemi viene applicato l'algoritmo slope-one e poi il calcolo finale avviene mediante una media pesata tra i valori ottenuti.
- **MatrixFactorization:** Fattorizzazione di matrici. L'apprendimento avviene tramite SGD (stochastic gradient descent).
    - **FactorWiseMatrixFactorization:** MatrixFactorization classica in cui l'apprendimento è di tipo factor-wise

    - **BiasedMatrixFactorization**
    - **SocialMF:** Tecnica usata nei social nework. Utilizza la tecnica della trust propagation: le feature di un utente sono dipendenti da quelle dei suoi vicini, che a loro volta sono dipendenti da quelle dei loro vicini. Si va dunque a creare una rete di fiducia che permette di fare predizioni.
- **TimeAwareBaseline:** Questo algoritmo enfatizza la popolarità di un item, posto che la popolarità non è un attributo globale ma varia da utente a utente. Utenti con gusti simili vengono raggruppati e vengono consigliati quegli item che recentemente hanno ricevuto un certo numero di rating da parte degli utenti. I voti più recenti sono più importanti di quelli più vecchi. Questo algoritmo confrontato con altri, su diversi dataset, ha restituito valori migliori in termini di precisione e nDCG.
    - **TimeAwareBaselineWithFrequencies**
- **UserItemBaseline:** Utilizza la media dei voti, più un bias regolarizzato per utenti e item
- **CoClustering:** Permette di clusterizzare contemporaneamente le righe e le colonne della matrice. Così si permette ad utenti e item di appartenere a più cluster.
- **Modelli asimmetrici**
    - **SigmoidCombinedAsymmetricFactorModel:** Modello asimmetrico in cui gli item sono rappresentati da gli utenti che li hanno votati. Gli utenti sono rappresentati dagli item che hanno votato
    - **SigmoidUserAsymmetricFactorModel:** Modello che rappresenta gli utenti nei termini degli item che hanno votato
    - **SigmoidItemAsymmetricFactorModel:** Modello che rappresenta gli item nei termini degli utenti che li hanno votati
- **SVDPlusPlus**: Metodo che è un'evoluzione del metodo SVD. Secondo quest'ultimo, ogni item può essere rappresentato da un vettore 'q_i', e ogni utente può essere rappresentato dal vettore 'p_u', tali che ***expected rating*** $= q_i^T * p_u$. Si può minimizzare l'errore quadratico tramite la seguente formula:

$$minimum(p,q) \sum_{(u,i)\epsilon K} \left(r_{ui} - q_i^T . p_u\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

    Il metodo SVDPlusPlus consente di considerare anche dei bias per utenti e item che permettono di migliorare la precisione del sistema.
    - GSVDPlusPlus: Metodo che prende in considerazione anche *quali* item sono stati votati dagli utenti
    - SigmoidSVDPlusPlus: Variante che utilizza una funzione sigmoidale.

- Anche per la rating prediction il framework permette di incorporare e testare **algoritmi esterni**, non presenti nella libreria.