Università degli Studi di Padova

Dipartimento di Matematica

"Tullio Levi-Civita"

Corso di Laurea Magistrale in Informatica

Sviluppo e confronto di metodi per la RecSys Challenge 2016

Tesi di laurea magistrale

Relatore:

Prof. Fabio Aiolli

Correlatore:

Dott. Mirko Polato

Laureando:

Andrea Domenico Giuliano

Anno Accademico 2016-2017

Indice

Introduzione

La crescita esplosiva della quantità di informazioni digitali disponibili e il numero di visitatori su Internet hanno creato una situazione di sovraccarico dell'informazione che ostacola l'accesso tempestivo a elementi di interesse su rete da parte degli utenti. Inizialmente sistemi di Information Retrieval come Google, DevilFinder e Altavista hanno parzialmente risolto questo problema, ma continuava a risultare assente un qualsiasi tipo di personalizzazione di tale ricerca riguardo alle caratteristiche, interessi e preferenze dell'utente. Tale situazione ha portato ad una crescente domanda di sistemi di raccomandazione.

Essi sono sistemi di filtraggio dell'informazione che affrontano il problema dell'eccesso di dati disponibili filtrando il frammento di informazioni vitali fuori dalla totalità dei dati. Tale filtraggio viene generato dinamicamente in base alle preferenze dell'utente, ad i suoi interessi o in base ai suoi comportamenti analizzati. I sistemi di raccomandazione hanno come scopo ultimo quelli di predire con quali elementi un utente interagirà nel futuro, basandosi sulle sue informazioni e comportamenti.

In questa tesi viene analizzata una tipologia particolare dei sistemi di raccomandazione, ossia i sistemi di job recommendation, sistemi specializzati sulla raccomandazione riguardante i business social network, come Xing o Linkedin. In particolare sono stati analizzati i modelli proposti durante il RecSys Challenge 2016, dai quali sono stati selezionati due modelli per essere implementati.

Successivamente sono state effettuate delle modifiche a tali modelli allo scopo di migliorarne le performance.

Capitolo 1

Recommender systems

> Quello che l'informazione consuma è piuttosto ovvio: consuma l'attenzione dei suoi destinatari. Dunque un'abbondanza di informazione crea povertà d'attenzione, ed il bisogno di scegliere come distribuire in maniera efficiente questa attenzione tra la sovrabbondanza di informazioni che potrebbero consumarla.
> *H.A. Simon*

1.1 Sistemi di Raccomandazione

La continua diffusione delle tecnologie informatiche e la sempre crescente propagazione dei contenuti digitali offrono agli utenti la possibilità di fare del media digitale uno strumento universale, da utilizzare in ogni occasione e situazione, rendendo di fatto tale dispositivo centrale nella vita dell'utilizzatore stesso.
Questa situazione rende il problema della gestione e dell'organizzazione di tale mole informativa una questione di primaria importanza, che cerca soluzioni nuove e che facilitino l'esperienza utente.
In tale contesto rientrano i sistemi di raccomandazione (RSs), ossia tecniche e strumenti software atti a raccomandare l'insieme di item da "proporre" per ogni utente [2,3,1].

Con il termine "item" si intendono le entità con cui interagiscono gli utenti. Un RS usualmente è specializzato in una particolare tipologia di item, riguardante un ambito specifico, ed anche il sistema stesso sarà specializzato nel suddetto campo. Per permettere tutto ciò, tali RSs collezionano informazioni riguardo alle preferenze ed ai comportamenti di ogni singolo user. Codeste informazioni vengono collezionate attraverso il feedback rilasciato dall'utente, esso può essere due due tipologie:

- **Explicit Feedback**

  Esso avviene collezionando i rating rilasciati dai vari user. Come è facile intuire dal nome, esso viene richiesto in maniera esplicita dal sistema. Codesto metodo permette di avere dei dati più facili da interpretare, ma può risultare intrusivo per quanto riguarda l'utente.

- **Implicit Feedback**

  Esso avviene analizzando il comportamento dell'utente in maniera implicita, ossia osservando le sue varie azioni e comportamenti [4,5,6].

Lo studio dei sistemi di raccomandazione è relativamente recente, infatti lo sviluppo di tali sistemi è emerso come settore indipendente verso la metà degli anni 90 [8,2,9,7].

1.2 Principali Approcci

I principali approcci utilizzati nei sistemi di raccomandazione sono:

- **Content based (CB)**

  Il sistema raccomanda gli item più simili a quelli su cui l'utente ha già mostrato interesse, quindi l'analisi delle caratteristiche degli Item ha un importanza cruciale [10].Tale approccio risulta molto utile nelle situazione nelle quali non si hanno molte informazioni sull'utente (cold-start problem).

- **Collaborative Filtering (CF)**

  Il sistema raccomanda ad ogni utente item similari a quelli che piacciono ai suoi simili o, viceversa, gli item più simili a quelli che piacciono a lui, quindi in tali approcci risulta necessario avere una nozione di similarità Item-Item ed User-User. Si noti che in tali approcci non si usa conoscenza specifica su oggetti e utenti ma solo caratteristiche del comportamento sociale determinato dall'interazione users-items.

- **Metodi Ibridi**

  Nulla vieta di creare metodi che utilizzino entrambi gli approcci. Usualmente le metodologie CB viengono usate con una maggiore rilevanza rispetto a quelle CF quando ci si trova in una situazione di cold start, mentre i metodi CF vengono usati in maniera più rilevante rispetto a quelli CB quando l'informazione implicita contenuta sulla interazione, ossia la rete sociale users-items, diventa prevalente rispetto a quella esplicita del contenuto.

1.3 Problemi in RS

I sistemi di raccomandazione sono usati per la risoluzione dei seguenti problemi:

- **Rate prediction**

  Situazione usualmente legata al explicit feedback. L'obbiettivo è predire il punteggio dei rates futuri relativi agli users. Nel Collaborative Filtering il rate prediction è un problema di regressione, per risolverlo una delle metodologie più usate risulta essere la Matrix Factorization [11], ovvero viene appresa una rappresentazione per gli utenti e per gli items in modo che il loro prodotto scalare approssimi i rates presenti.

- **TOP-N recommendation**

  L'obbiettivo è quello di predire gli N items di maggior gradimento per un dato user, ossia quelli con cui è più facile che si verifichino delle interazioni. Nel Collaborative Filtering il TOP-N recommendation è un problema di ranking, per risolverlo una delle metodologie più comnuni è quella di effettuare

una Matrix Factorization su preferenze [11], ossia un ranking tra coppie di items/users.

1.4 RecSys 2016

RecSys è una delle più grandi conferenze a livello mondiale riguardanti i sistemi di raccomandazione ed è attualmente l'unica completamente indirizzata su di essi [13].L'edizione del 2016 è stata co-organizzata da XING, CrowdRec and MTA ZTAKI [12]. Xing è un social network riguardante il mondo del lavoro molto popolare a livello europeo, in particolare in germania [14].

1.4.1 La Challenge

La challenge proposta nel 2016 riguarda lo sviluppo di un sistema di raccomandazione specifico per Xing, ossia un sistema di job recommendation, con lo scopo finale di predire in maniera efficiente i lavori con il quale un determinato utente interagirà in maniera positiva in futuro [12].

1.4.2 Job Recommendation

Creare un sistema di job recommendation richiedere la creazione di un RS estremamente specializzato. L'utenza base ha un periodo di attività estremamente limitato, durante il quale tende ad interagire ripetutamente con gli stessi items visti in precedenza, per tale motivo ha senso raccomandare quest'ultima tipologia di oggetti. Codesti items inoltre hanno una decadenza temporale estremamente accentuata, il che sta a significare che con l'avanzare del tempo essi diventono poco interessanti per gli users con grande rapidità, inoltre essi tendono a comparire e scomparire in fretta, in base alla loro disponibilità [30]. Inoltre una delle caratteristiche della job recommendation è quello di raccomandare per ogni users non solo items che corrispondano ai suoi gusti personali, ma anche in base alle caratteristiche dell'utente stesso, come, ad esempio, il suo livello di preparazione o la sua esperienza [31].

Come si può evincere, la job reccomandation è un tipo di raccomandazione molto differente da quelle riguardanti altri tipi di items, come per esempio film o prodotti di un e-commerce, poiché mentre se in quest'ultimi è improbabile che un utente guardi nuovamente lo stesso film oppure acquisti nuovamente lo stesso prodotto, nel caso dei lavori risulta frequente che un utente guardi ripetutamente lo stesso annuncio, allo scopo di paragonarlo con altri o per richiedere un colloquio.

1.4.3 Il Dataset

Il dataset fornito per la challenge risulta costituito da quattro gruppi principali, suddivisi in altrettanti documenti:

- **Users**

  Elenco degli utenti che utilizzano il sistema Xing, assieme alle loro relative informazioni. Essi si suddividono tra normal-users e target-users, questi ultimi sono gli utenti per cui bisogna effettuare la raccomandazione. I rispettivi dati sono contenuti all'interno del file users.csv.

- **Items**

  Elenco degli item, ossia dei job, da raccomandare ai target-user, assieme alle loro relative informazioni. I rispettivi dati sono contenuti all'interno del file items.csv.

- **Interactions**

  Elenco delle interazioni che i vari user hanno effettuato con i vari item, assieme alle loro relative informazioni. Essi si suddividono in quattro categorie:

  1. Click
  2. Bookmark
  3. Reply
  4. Delete

  Tutti le tipologie di interazioni, eccetto per quanto riguarda la categoria "Delete", sono considerate positive. Tali dati sono contenuti all'interno del file interactions.csv.

- **Impressions**

  Elenco degli items che sono stati proposti dal precedente sistema di job recommendation presente su Xing, assieme alle loro relative informazioni. Tali dati sono contenuti all'interno del file impressions.csv.

La tabella 1.1 mostra il numero degli elementi di ciascuna categoria:

Gruppo	# elementi
Users	1,367,057
Items	1,358,098
Interactions	8,826,678
Impressions	10,130,410

Tabella 1.1: Numero di elementi per ciascun gruppo

1.4.4 Metrica di Valutazione

Nella RecSys Challenge 2016 viene usato il tasso di successo del top-N recommendation come metrica di valutazione di base. In particolare, se l'utente interagisce con almeno uno dei top-N item raccomandati, la raccomandazione è considerata un successo.

Usualmente la metodologia di valutazione più utilizzata per valutare un sistema di raccomandazione è l'online AB test [26]. In tale tipo di valutazione, per ogni sistema di raccomandazione viene selezionato in maniera casuale un insieme di users ed il risultato è basato sulla misurazione delle interazioni reali di tali users [27]. Purtroppo, una tale strategia di valutazione on-line richiederebbe significativi sforzi di tipo ingegneristico per essere effettuata [15].

In questo concorso, gli organizzatori hanno fornito una metodologia alternativa per eseguire la valutazione offline. Durante il periodo di test, il sistema ha registrato il comportamento dei vari users ed ha generato un insieme ground truth T di tuple user-item, dove $t_i = T(u_i)$ da la lista degli item con il quale l'utente u_i ha interagito durante il periodo di test. Quindi, dato un periodo di raccomandazione S , il sistema di valutazione online produrrà un puteggio $score(R, T)$ che misurerà la

rilevanza del sistema di raccomandazione sull'insieme di ground truth.
La funzione che produce tale punteggio è definita come mostrato nella formula 1.1 :

$$\begin{aligned} score(R,T) &= \sum_{i=1}^{N} s(u_i) \\ s(u_i) &= 20(P_2 + P_4 + R + UserSuccess)) + 10(P_6 + P_{20}). \end{aligned} \tag{1.1}$$

dove P_2, P_4, P_6, P_{20} rappresentano la Precisione alla top-N posizione, dove $N = 2, 4, 6, 20$ ed R è di Recall.
La funzione P_N riguardante il calcolo dello score della Precisione alla top-N posizione ritorna un valore numerico compreso [0...1] ed indica il rapporto tra quanti degli item proposti siano effettivamente presenti all'interno del test-set, definiti come true positive (tp), rispetto al numero di raccomandazioni errate, definite come false positive (fp). Essa è definita come mostrata nella formula 1.2 :

$$P = \frac{tp}{tp + fp} \tag{1.2}$$

La funzione R riguardante il calcolo dello score della Recall ritorna un valore numerico compreso [0...1] ed indica il rapporto tra quanti degli item presenti all'interno del test-set siano stati effettivamente raccomandati all'user con successo, definiti come true positive (tp), ripetto al numero degli oggetti sempre presenti all'interno del test-set che non sono stati oggetto di raccomandazione, definiti come false negative (fn). Essa è definita come mostrata nella formula 1.3 :

$$R = \frac{tp}{tp + fn} \tag{1.3}$$

La funzione $UserSuccess$ ritorna un valore binario $\{0, 1\}$,esso sarà 1 nel caso almeno un item raccomandato dal sistema per l'user sia stato realmente soggetto di un interazione, 0 altrimenti. Essa è definita come mostrata nella formula 1.4 :

$$UserSuccess(A(u), T_u) = I(|A(u)_{:30} \cap T_u| > 0) \tag{1.4}$$

dove $A(u)_{:30}$ è l'insieme dei primi 30 item raccomandati per l'user di riferimento, T_u è il test-set del medesimo utente ed I è la funzione identità.

1.4.5 Soluzioni proposte

Durante lo svolgimento della RecSys Challenge 2016 numerosi team da tutto il mondo hanno proposto le loro metodologie di job recommendations. Ogni gruppo partecipante, oltre a presentare i risultati del proprio metodo, ossia per ogni users i vari items da raccomandare, doveva presentare un paper di massimo 4 pagine che illustrava il metodo usato in tale challenge. Tali paper sono stati esaminati e valutati dalla seguente commissione:

- **Alejandro Bellogín**, Universidad Autónoma de Madrid, Spain
- **Paolo Cremonesi**, Politecnico di Milano, Italy
- **Simon Dooms**, Trackuity, Belgium
- **Balasz Hidasi**, Gravity R&D, Hungary
- **Levente Kocsis**, Hungarian Academy of Sciences, Hungary
- **Andreas Lommatzsch**, TU Berlin, Germany
- **Katja Niemann**, XING AG, Germany
- **Alan Said**, University of Skövde, Sweden
- **Yue Shi**, Yahoo Labs, USA
- **Marko Tkalcic**, Free University of Bozen-Bolzano, Italy

Tra tutti i paper presentati la commissione ha selezionato quelli più interessanti e rappresentativi [12], essi sono:

- **Job Recommendation with Hawkes Process** [15]
  Metodo sviluppato e proposto dal team composto da Wenming Xiao, Xiao Xu, Kang Liang, Junkang Mao and Jun Wang.
  In tale approccio per prima cosa viene cotruito un modello gerarchico a coppie con l'apprendimento degli insiemi come quadro generale di previsione. Successivamente vengono integrate le informazioni relative al contenuto e al comportamento relativo agli user/item nel processo di ingegnerizzazione delle funzionalità sviluppato dal team. In tale sezione vengono costruiti da tre modelli di classificazione, ossia Regressione Logistica (LR), gradient boosting regression tree (GBDT) [28] ed una nuova tipologia di albero chiamato eXtreme Gradient Boosting (XGBOOST) [29].

In particolare, allo scopo di generare una raccomandazione rilevante anche dal punto di vista temporale, viene utilizzato il *Processo Hawkes*. Per ultimo viene affrontato il problema di cold start utilizzando una strategia semantica basata sulle informazioni degli users.
Lo schema concettuale del modello proposto è illustrato nella figura 1.1:

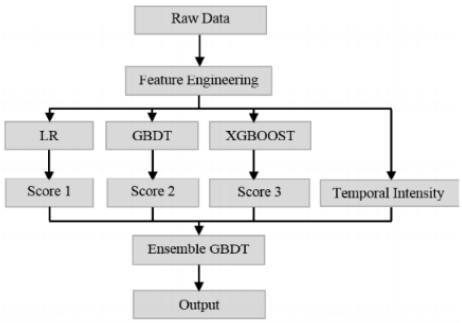

Figura 1.1: Diagramma concettuale del modello

Tale approccio si è classificato in prima posizione nella challenge, producendo le raccomandazioni che hanno ricevuto la valutazione maggiore [15].

- **RecSys Challenge 2016: job recommendations based on preselection of offers and gradient boosting** [20]
  Metodo sviluppato e proposto dal team composto da Andrzej Pacuk, Piotr Sankowski, Adam Witkowski, Karol Wegrzycki e Piotr Wygocki.
  In questo modello per ogni user viene effettuata una selezione dei candidati, ossia degli item che sono considerati più promettenti per una possibile interazione futura. Tale selezione viene effettuata per evitare di dover confrontare ogni user con ogni item in di ridurre i calcoli e si basa sulle interazioni degli users e degli items. Successivamente viene effettuato per ogni item candidato una stima delle probabilità di interazione futura. Per stimare le probabilità viene usato XGBOOST [29] , una libreria riguardante l'apprendimento automatico implementata in GBDT [28]. Infine viene vengono selezionati i primi 30 items considerati più promettenti.

Uno schema concettuale di tale modello può essere osservato nella figura 1.2:

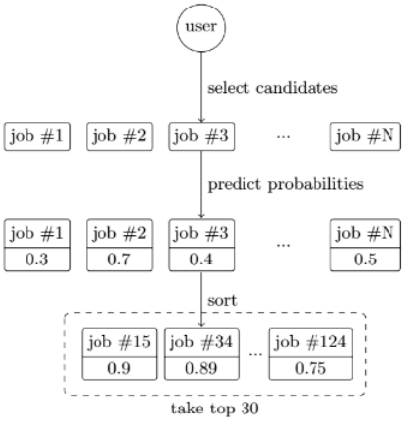

Figura 1.2: Schema concettuale del modello

Tale approccio si è classificato in seconda posizione nella challenge [20].

- **A Combination of Simple Models by Forward Predictor Selection for Job Recommendation** [24]

  Metodo sviluppato e proposto da Dávid Zibriczky.

  Il principio della tecnica proposta è quello di definire diversi sistemi che acquisiscano la specificità del dataset e poi per trovare le combinazioni ottimali di tali metodologie, considerando diverse categorie di utenti. Tale tecnica viene effettuata attraverso la combinazione di vari predittori che tengono conto sia delle caratteristiche degli user e degli item, sia degli eventi avvenuti tra questi ultimi.

  Tale approccio si è classificato in terza posizione nella challenge [24].

- **Multi-Stack Ensemble for Job Recommendation** [16]

  Metodo sviluppato e proposto dal team PumpkinPie composto da Tommaso Carpi, Marco Edemanti, Ervin Kamberoski, Elena Sacchi, Paolo Cremonesi, Roberto Pagano e Massimo Quadrana.

  L'approccio proposto dal team consiste nella generazione di un insieme di algoritmi di raccomandazione che utilizzano tecniche diverse, per poi combinarle successivamente con un approccio multi-stack.

Tale metodo si basa sull'idea che la combinazione di diversi approcci possono apprendere ed interpretare diverse relazioni riguardanti gli users e gli items [32,33]. Come mostrato nella figura 1.3 l'approccio combina sia algoritmi basati sull'approccio collaborative filtering (CF), sia content based (CB), basati sulle passate interactions ed impressions relative sia agli users che agli items ed infine l'algoritmo di Baseline. Quest'ultimo metodo è un algoritmo fornito direttamente da XING e leggermente modificato dal team PumpkinPie dove per ogni item viene calcolato uno score relativo ad un utente basato direttamente sulle caratteristiche comuni, come la regione geografica, industy e discipline. Tale metodo risulta comodo nel caso ci si trovi in una situazione di cold start.

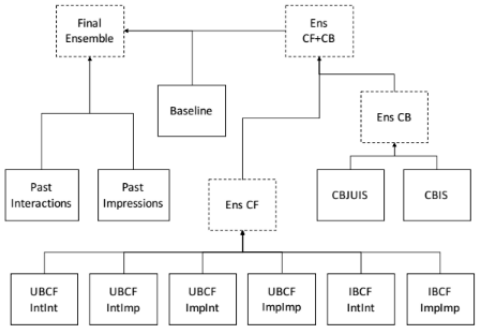

Figura 1.3: Schema concettuale del modello

Tale approccio si è classificato in quarta posizione nella challenge [16].

- **Temporal Learning and Sequence Modeling for a Job Recommender System** [18]

  Metodo sviluppato e proposto dal team composto da Kuan Liu, Xing Shi, Anoop Kumar, Linhong Zhu e Prem Natarajan.

  L'idea base del metodo proposto è quella di combinare l'elemento temporale nell'apprendimento con una sequenza di approcci differenti allo scopo di apprendere il modello di interazione relativo agli users ed agli items. Per prima cosa viene effettuato un modello di classificazione basato sullo storico degli users e degli items e viene creata una matrice di fattorizzazione ibrida basata sulle interazioni tra users ed items.

Successivamente vengono sfruttate le proprietà di interazione user-item per creare un modello di raccomandzione basato sull'RNN.

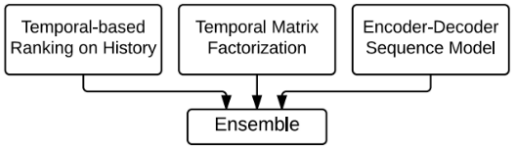

Figura 1.4: Schema concettuale del modello

Tale approccio si è classificato in quinta posizione nella challenge [18].

- **Job Recommendation Based on Factorization Machine and Topic Modelling** [19]

  Metodo sviluppato e proposto dal team composto da Vasily Leksin and Andrey Ostapets.

  Il modello proposto include tre tipi di approcci, Fattorizzazione, collaborative filtering basata sugli items e content based basato sui tags. Inoltre vengono combinati approcci di tipo collaborative filtering e content based all'interno del modello.

  Tale approccio si è classificato in settima posizione nella challenge [19].

- **A Scalable, High-performance Algorithm for Hybrid Job Recommendations** [17]

  Metodo sviluppato e proposto dal team composto da Toon De Pessemier, Kris Vanhecke e Luc Martens.

  Il team ha proposto un modello ibrido costituito da un approccio KNN combinato con uno content based, quest'ultimo è basato su un pre-selezione di items candidabili per ogni users e sulle informazioni temporali delle interazione degli user stessi. L'approccio KNN si basa sul cercare items similari a quelli con il cui l'utente ha interagito in passato [17].

- **A Bottom-Up Approach to Job Recommendation System** [22]

  Metodo sviluppato e proposto dal team composto da Sonu K. Mishra e Manoj Reddy.

  L'approccio proposto ha una struttura bottom up, ossia comincia con un analisi delle proprità principali dei dati, come le interactions user-item ed il loro valore temporale, per poi creare il modello vero e proprio, basato su vari approcci tradizionali per i sistemi di raccomandazione, come il collaborative filtering. Il modello finale proposto dal team è basato sull'algoritmo Gradient Boosting [34] ed è quello che offre le prestazioni migliori.

  Tale approccio si è classificato in ventesima posizione nella challenge [22].

- **Jobandtalent at RecSys Challenge 2016** [23]

  Metodo sviluppato e proposto dal Jobandtalent Recommendation team composto da Jose Ignacio Honrado, Oscar Huarte, Cesar Jimenez, Sebastian Ortega, Jose R. Perez-Aguera, Joaquin Perez-Iglesias, Alvaro Polo e Gabriel Rodriguez.

  Tale metodo si basa sull'approccio "Learning to Rank", ossia usare delle tecniche di apprendimento automatico allo scopo di creare modelli classificati. Usualmente tali tipologie di approcci tendono ad analizzare e combinare varie caratteristiche di una query allo scopo di classificare dei documenti per essa [35]. Il team applica l'approccio definito come "listwise" che produce una classificazione dei documenti. Inoltre per utilizzare tale tipologia di strategia di ricerca il team ha creato delle query specifiche da associare ai documenti considerati rilevanti. Per adattare tale approccio alla challenge le query sono considerati gli users, i documenti gli items e gli elementi analizzati sono le interazioni tra users ed items.

  Tale approccio si è classificato in undicesima posizione nella challenge [23].

- **An Ensemble Method for Job Recommender Systems** [21]

  Metodo sviluppato e proposto dal team composto da Chenrui Zhang e Xueqi Cheng.

  L'approccio proposto consiste nella combinazione di diversi algoritmi collaborative filtering che analizzano le interazioni tra users ed items, tra cui il modello Latent Semantic(LSI), che ha lo scopo di individuare user similari in base alle interactions effettuate, ed Item2Vec allo scopo di identificare item similari.

  Tale approccio si è classificato in decima posizione nella challenge [21].

- **A preliminary study on a recommender system for the Job Recommendation Challenge** [25]

  Metodo sviluppato e proposto dal team composto da Mirko Polato e Fabio Aiolli.

  Il metodo proposto dal team consiste in una combinazione lineare di diversi predittori, in maniera tale che diversi aspetti relativi agli users ed agli items siano analizzati in maniera tale da effettuare una corretta raccomandazione.

  Tale approccio si è classificato in ventisettesima posizione nella challenge [25].

In questo progetto di tesi sono stati analizzati nel dettaglio due approcci in particolare, ossia il modello proposto da Dávid Zibriczky e quello proposto dal team composto da Fabio Aiolli e Mirko Polato. Tale scelta è stata effettuata poiché entrambi i metodi proposti risultano similari ed inoltre poiché entrambi presentano una completa descrizione della metodologia implementata, il che permette una loro completa e corretta implementazione.

Capitolo 2

Metodi Analizzati

La RecSys Challenge 2016 ha ricevuto diverse tipologie di modelli creati e sviluppati da diversi team internazionali. La possibilità di accedere ad una tale varietà di metodologie ed approcci ci ha permesso di studiarli allo scopo di poter identificare delle caratteristiche comuni che ci permettano di creare un sistema ibrido dalle alte prestazioni. Tra tutti i modelli proposti in tale challenge due sono stati selezionati a scopo di studio ed implementazione, ossia quelli di Dávid Zibriczky e Polato et al. Tale scelta è stata effettuata poiché i modelli utilizzano un approccio similare, inoltre entrambi presentano nel paper infrmazioni sufficienti ad una completa implementazione.

2.1 Modello di Dávid Zibriczky

La soluzione proposta da Dávid Zibriczky per la RecSys Challenge 2016 consiste in una combinazione di vari predittori. Attraverso una tale varietà di predittori il sistema sarà in grado di sfruttare le diverse caratteristiche del dataset allo scopo di effettuare un adeguata raccomandazione [24].

2.1.1 Funzioni Comuni

Data la seguente definizione delle dipologie di eventi tra users ed items:

1. Click
2. Bookmark
3. Reply
4. Delete
5. Impression

e data la funzione $w_y(e)$ che ritorna la tipologia dell'evento e, viene assegnato un peso ad ogni evento usando la formula 2.1 :

$$w(e) = w_y(e)(1 - \delta^{t_{max} - t(e)}) \tag{2.1}$$

dove:

- $w_y(e)$
  valore costante assegnato in base alla tipologia dell'evento e, tale valore può essere:
  $$w(e) = \begin{cases} 0.05 & \text{se } y(e) = 5 \\ 1 & \text{altrimenti} \end{cases}$$
- $t(e)$
  time stamp indicante il tempo in cui è avvenuto l'evento e.
- t_{max}
  time stamp più recente presente nel dataset.
- $\delta^{t_{max} - t(e)}$
  Funzione di decadimento temporale che può assumere un valore compreso tra 0.05 e 0.3 in base alla differenza tra t_{max} e $t(e)$.

Sia Y un arbitrario insieme di tipi di eventi, per un dato user u ed item i, la funzione $w_Y(u, i)$ ritorna il peso dell'evento più recente avvenuto tra tali user ed item, assumendo che $y(e) \in Y$. Nel caso non sia presente alcun evento (u, i) della tipologia appartenente ad Y allora $w_Y(u, i) = 0$.

Usando quest'ultima notazione è possibile definire la funzione di supporto dell'item i come mostrato nella formula 2.2 :

$$s_Y(i) = \sum_u w_Y(u, i) \tag{2.2}$$

Infine la funzione di co-occorrenza tra l'item i e l'item j, dati i loro rispettivi insiemi di tipologie di eventi Y e Z, viene definita come mostrato nella formula 2.3 :

$$s_{Y,Z}(i, j) = \sum_u w_Y(u, i) w_Z(u, j) \tag{2.3}$$

2.1.2 IKNN: Item-based K-nearest neighbors

Data la nozione di similarità tra due items definita come mostrato nella formula 2.4 :

$$sim_{I,O}(i, j) = \frac{s_{I,O}(i, j)}{(s_I(i) + \lambda)^{\alpha}(s_I(j) + \lambda)^{1-\alpha}} \tag{2.4}$$

dove λ è il coefficiente di regolarizzazione ed α è un esponente di normalizzazione, il predittore dell'item i per l'utente u viene definito come mostrato nella formula 2.5 :

$$p(u, i) = \left(\sum_{e \in E(u)} w(e) \right)^{-1} \sum_{e \in E(u)} w(e) sim_{I,O}(i(e), i) \tag{2.5}$$

dove $E(u)$ è l'insieme degli eventi dell'user u ed $i(e)$ è l'item protagonista dell'evento e [24].

2.1.3 RCTR: Recalling recommendations

Tale predittore si basa sull'idea che nel contesto della job reccomandation un user tenda a visualizzare più volte gli items già visualizzati nelle settimane precedenti. Per tale proprietà è stato sviluppato un modello basato sul tasso di click che l'oggetto ha ricevuto nella sua storia.

Il predittore dell'item i per l'utente u viene definito come mostrato nella formula 2.6 :

$$p(u,i) = \sum_{e \in E(u)} \frac{s_C(i(e))}{s_R(i(e))} \mathbb{1}(y(e) = 5) \tag{2.6}$$

dove $E(u)$ è l'insieme degli eventi dell'user u, C ed R rappresentano rispettivamente gli insiemi dei tipi di eventi relativi ai clicks ed alle impressions, mentre $\mathbb{1}$ è la funzione identità [24].

2.1.4 AS: Already seen items

Tale predittore prende spunto dall'idea che nel contesto della job reccomandation un user tenda a visualizzare più volte gli items già visti in precedenza. Allo scopo di modellare questo fenomeno, tali items sono raccomandati in base alla tipologia di evento avvenuta ed all'invecchiamento di tale evento.
Il predittore dell'item i per l'utente u viene definito come mostrato nella formula 2.7 :

$$p(u,i) = \sum_{e \in E(u)} w(e) \mathbb{1}(y(e) \in I) \tag{2.7}$$

dove $E(u)$ è l'insieme degli eventi dell'user u.
Tale metodo tende ad avere basse prestazioni se usato da solo, ma in combinazione con gli altri approcci proposti tende ad avere un positivo impatto sulla raccomandazione effettuata. Da notare che il metodo mostrato nella formula 2.7 viene indicato con $AS(I)$, dove I è l'insieme dei tipi di eventi relativi agli eventi di tipologia input [24].

2.1.5 UPOP: User metadata-based popularity

Nell'ambito della job recommendation una grande quantità di user è "nuova", ossia senza alcun tipo di storico relativo alle interactions ed impressions, rendendo necessario l'utilizzo di altri dati riguardanti gli utenti, come i gruppi di appartenenza.

Per tale motivo è stato costruito un modello sulla base degli item più popolari ai gruppi di cui un tale user fa parte.
Per definire i gruppi utenti sono stati i seguenti parametri:

- jobroles
- edu field of studies

La popolarità di un item i in un gruppo di utenti g viene definita come mostrato nella formula 2.8 :

$$pop_g(i) = \frac{1}{|U(g)|} \sum_{u \in U(g)} \sum_{e \in E(u)} w(e) \mathbb{1}(i = i(e)) \tag{2.8}$$

dove $E(u)$ è l'insieme degli eventi dell'user u, $U(g)$ è l'insieme degli user appartenenti al gruppo g ed $i(e)$ è l'item protagonista dell'evento e.
Il predittore dell'item i per l'utente u viene definito come mostrato nella formula 2.9 :

$$p(u, i) = \frac{1}{|G(u)|} \sum_{g \in G(u)} \frac{pop_g(i)}{pop_g(i) + \lambda} \tag{2.9}$$

dove $G(u)$ è l'insieme dei gruppi a cui appartiene l'utente u e λ è il coefficiente di regolarizzazione [24].

2.1.6 MS: Meta cosine similarity

Allo scopo di avere un predittore con approccio content based, è stato modellata una similarità coseno tra items basato sui metadata [10]. Tali items sono modellati in vettori basati sui loro metadata e che utilizzano il sistema dei pesi tf-idf [36], per cui la similarità coseno $sim(i, j)$ tra gli items i e j viene calcolata come la similarità coseno dei loro vettori.

I vettori degli item sono costruiti usando i seguenti parametri:

- tags
- title
- industry id
- geo country
- geo region
- discipline id

Il predittore dell'item i per l'utente u viene definito come mostrato nella formula 2.10 :

$$p(u,i) = \left(\sum_{e \in E(u)} w(e) \right)^{-1} \sum_{e \in E(u)} w(e) sim(i(e), i) \tag{2.10}$$

dove $E(u)$ è l'insieme degli eventi dell'user u ed $i(e)$ è l'item protagonista dell'evento e [24].

2.1.7 AP: Age-based popularity change

Una delle caratteristiche della raccomandazione nell'ambito della job recommendation è quella di avere degli item che tendono a perdere popolarità molto velocemente col passare del tempo. Per tale motivo è stato costruito un modello basato su questo abbassamento di popolarità.
Il predittore dell'item i per l'utente u viene definito come mostrato nella formula 2.11 :

$$p(u,i,t) = \frac{\sum_{k=1}^{7} pop_a(age(i,t)+k)}{28 * pop_a(age(i,t))} \tag{2.11}$$

dove $pop_a(d)$ è la popolarità attesa per un item con età d e $age(i,t)$ è una funzione che ritorna l'età dell'oggetto i al tempo t in giorni. Per la challenge è stato usato $p(u,i,t_{max})$ [24].

2.1.8 Ottimizzazione

Per quanto riguarda l'ottimizzazione è stato generato un test-set usando l'ultima settimana del training set. Ogni predittore è stato ottimizzato usando il metodo della discesa del gradiente. I valori risultati più accurati sono stati nominati come candidati per la combinazione lineare dei valori dei predittori. Allo scopo di mantenere il numero di tali valori basso, è stato utilizzato il seguente algoritmo goloso di selezione dei predittori:

1. Sia $P = \{p_1, p_2, ..., p_k\}$ un insieme di predittori e siano assegnati l'insieme di pesi $W = (w_1, w_2, ..., w_k)$ ai rispettivi indici di P. Sia indicata come precisione della combinazione dei predittori P usando i rispettivi pesi W nel test set T come $\tau(P, W, T)$.

2. Si effettui la creazione di n test fold $F = T_1, T_2, ..., T_n$, dove $T_j \subset T$.

3. Sia $P_C = P$, $P_S = \emptyset$ e $W = (0, 0, ..., 0)$.

4. Per ogni $p_i \in P_C$, sia $P_S^i = P_S \cup p_i$ e viene calcolato il guadagno in precisione attraverso $g_i = \tau(P_S^i, W_i, T) - \tau(P_S, W, T)$ dove $W_i = \frac{1}{|F|} \sum_{T_j \in F} (\arg_W \max \tau(P_S^i, W, T_j))$ viene trovato attraverso l'ottimizzazione della discesa del gradiente.

5. Viene selezionato un i dove g_i risulti massimale, viene spostato p_i da P_C a P_S e viene settato $W = W_i$.

6. Vengono ripetuti i passaggi 4 e 5 fino a quando $g_i > 0$ e $P_C \neq \emptyset$

Tale metodo applica una cross-validation che permette di evitare una situazione di overfitting durante il processo [24].

2.1.9 OM: The omit method

A volte durante il processo di raccomandazione si possono verificare delle situazione dove un item viene raccomandato troppo o troppo poco. Allo scopo di affrontare tale situazione viene proposto il seguente metodo:

1. Il training set originale viene diviso in un insieme sub-train e sub-test.
2. I metodi di raccomandazione sono addestrati nel insieme di sub-train.
3. Per ogni insieme di sub-test, una raccomandazione Top-N viene calcolata.
4. Per ogni elemento dell'insieme Top-N, vengono calcolate le prestazioni nel caso tale elemento sia escluso.
5. Tale valore viene sommato per tutti gli items.
6. Gli items con un significativo valore negativo vengono omessi dalla raccomandazione.

Omettendo tali items che tendono a procurare un calo di performance viene generato un aumento delle prestazioni del sistema [24].

2.2 Modello di Polato et al.

L'approccio proposto dal team composto da Fabio Aiolli e Mirko Polato segue le linee guida delle metodologie memory-based e consiste nella creazione di una struttura basata sull'approccio collaborative filtering dove diversi predittori possono essere combinati in maniera lineare attraverso un algoritmo di apprendimento ispirato da precedenti lavori del docente Aiolli [37] e del team composto da Y. Hu, Y. Koren, e C. Volinsky [38].
La combinazione di diversi predittori permette di sfruttare in maniera efficace le diverse informazioni riguardanti gli users, gli items e le loro interazioni.

2.2.1 Idea Base

Viene definita la matrice $\mathbf{W} \in \mathbb{R}^{n \times k}$ come la proiezione degli user in uno spazio vettoriale di dimensione k con la rappresentazione degli users $\mathbf{w}_u$ come righe.
In maniera similare viene definita la matrice $\mathbf{X} \in \mathbb{R}^{k \times m}$ come la proiezione degli items nel medesimo spazio vettoriale con la rappresentazione degli items $\mathbf{x}_i$ come colonne. La raccomandazione viene effettuata sulla base della classificazione indotta dalla fattorizzazione 2.12 :

$$\hat{\mathbf{R}} \sim \mathbf{WX} \tag{2.12}$$

In tale approccio la coppia **WX** viene formata da una rappresentazione evidence-based e similarity-based. Grazie a tale rappresentazione risulta possibile dividere i predittori in due principali categorie, ossia Item-based, utilizzato ogni qual volta che una rappresentazione similarity-based viene e effettuata tra due items distinti, ed User-based, utilizzato ogni qual volta che una rappresentazione similarity-based viene e effettuata tra due users distinti.

2.2.2 Funzioni Comuni

Come detto in precedenza, risulta possibile dividere i vari predittori in due principali categorie, Item-based ed User-based. In entrambe le situazioni, il punteggio normalizzato del predittore k per la coppia user-item (u, i) viene calcolata utilizzando la formula 2.13 :

$$\hat{r}_{ui}^{(k)} = \frac{\mathbf{w}_u \mathbf{x}_i}{\| \mathbf{w}_u \|_2 \ \| \mathbf{x}_i \|_2} \tag{2.13}$$

All'interno dei predittori viene utilizzata come funzione di similarità quella coseno. Dati due vettori $\mathbf{a}, \mathbf{b} \in \{0, 1\}^q$ la loro similarità coseno è cacolata come mostrato nella formula 2.14 :

$$cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\| \mathbf{a} \| \ \| \mathbf{b} \|} \tag{2.14}$$

Infine viene indicato con ϕ_x la rappresentazione dell'oggetto x nel feature space.

2.2.3 Predittori Item-based

In questa sezione vengono presentati i predittori item-based. La caratteristica di tali predittori è quella di avere una rappresentazione evidence-based per gli users ed una similarity-based per quanto riguarda gli items. Per ogni predittore verrà mostrato come esso viene rappresentato all'interno della struttura creata da Fabio Aiolli e Mirko Polato.

- **Interaction-based (IB)**

  Tale predittore suggerisce ad un user u items similari a quelli con cui lui/lei ha avuto un evento di tipo interaction in passato. Esso è rappresentato come mostrato nella formula 2.15 :

$$\mathbf{W} = \mathbf{R}, \quad \phi_i = \mathbf{r}_i, \quad \mathbf{X} : x_{ij} = cos(\phi_i, \phi_j). \tag{2.15}$$

- **Evidence-Impression-based (EIB)**

  Tale predittore differisce dal precedente per la tipologia di rapporto tra user ed item, di fatti tale esso suggerisce ad un user u items similari a quelli con cui lui/lei ha avuto un impressions in passato. Data una matrice $\mathbf{E} \in \{0,1\}^{n \times m}$ dove la cella (u,i) risulterà 1 se tra l'user u e l'item i è avvenuto un avvenimento di tipo impression in passato. Il predittore è rappresentato come mostrato nella formula 2.16 :

$$\mathbf{W} = \mathbf{E}, \quad \phi_i = \mathbf{r}_i, \quad \mathbf{X} : x_{ij} = cos(\phi_i, \phi_j). \tag{2.16}$$

- **Impression-based (IEB)**

  Tale predittore è una variante di quello precedente, di fatti viene utilizzata come evidenza le user interaction e la rappresentazione degli item relativa alle impressions. Esso è rappresentato come mostrato nella formula 2.17 :

$$\mathbf{W} = \mathbf{R}, \quad \phi_i = \mathbf{e}_i, \quad \mathbf{X} : x_{ij} = cos(\phi_i, \phi_j). \tag{2.17}$$

- **Tag-based (TGB)**

  In tale predittore la rappresentazione degli items è caratterizzata dai loro relativi tags. Sia $\mathbf{G}$ l'insieme di tutti i possibili tags e sia $\mathbf{g}_i \in \{0,1\}^{|G|}$ il vettore rappresentate l'item i, dove tutte le celle riferite ai tags dell'item sono impostate ad 1, allora il predittore è rappresentato come mostrato nella formula 2.18 :

$$\mathbf{W} = \mathbf{R}, \quad \phi_i = \mathbf{g}_i, \quad \mathbf{X} : x_{ij} = cos(\phi_i, \phi_j). \tag{2.18}$$

- **Title-based (TTB)**

  Tale predittore è una variante di quello visto in precedenza, di fatti la rappresentazione degli items è caratterizzata dai loro relativi termini che lo compongono il suo titolo. Sia $\mathbf{L}$ l'insieme di tutti i possibili termini riguardanti i titoli degli items e sia $\mathbf{l}_i \in \{0,1\}^{|L|}$ il vettore rappresentate l'item i, dove tutte le celle riferite ai termini che compongono il titolo dell'item sono impostate ad 1, allora il predittore è rappresentato come mostrato nella formula 2.19 :

$$\mathbf{W} = \mathbf{R}, \quad \phi_i = \mathbf{l}_i, \quad \mathbf{X} : x_{ij} = cos(\phi_i, \phi_j). \tag{2.19}$$

2.2.4 Predittori User-based

In questa sezione vengono presentati i predittori user-based. La caratteristica di tali predittori è quella di avere una rappresentazione evidence-based per gli items ed una similarity-based per quanto riguarda gli users. Per ogni predittore verrà mostrato come esso viene rappresentato all'interno della struttura creata da Fabio Aiolli e Mirko Polato.

- **Popularity-based (PB)**

  Tale predittore raccomanda all'user gli item in base alla loro popolarità, ossia al numero di interazioni positive che hanno avuto in passato. Esso è rappresentato come mostrato nella formula 2.20 :

  $$\mathbf{W} = \mathbf{1}, \quad \mathbf{X} = \mathbf{R}. \tag{2.20}$$

- **Interaction-based (UB)**

  Tale predittore suggerisce ad un user u items che sono stati valutati in maniera positiva da users simili a lui, usando come le iteractions come evidenza per quanto riguarda gli items. Esso è rappresentato come mostrato nella formula 2.21 :

  $$\phi_u = \mathbf{r}_u, \quad \mathbf{W} : w_{uv} = cos(\phi_u, \phi_v), \quad \mathbf{X} = \mathbf{R}. \tag{2.21}$$

- **Evidence-Impression-based (EUB)**

  Tale predittore è una variante di quello visto in precedenza dal quale differisce per quanto riguarda la tipologia di evidenza riguardante gli items, di fatti in tale predittore vengono usate le impressions. Data una matrice $\mathbf{E} \in \{0,1\}^{n \times m}$ dove la cella (u, i) risulterà 1 se tra l'user u e l'item i è avvenuto un avvenimento di tipo impression in passato. Il presittore è rappresentato come mostrato nella formula 2.22 :

  $$\phi_u = \mathbf{r}_u, \quad \mathbf{W} : w_{uv} = cos(\phi_u, \phi_v), \quad \mathbf{X} = \mathbf{E}. \tag{2.22}$$

- **Impression-based (UEB)**

  Tale predittore è una variante di quello visto in precedenza, esso viene ottenuto invertendo le evidence riguardante gli items con la rappresentazione degli users. Esso è rappresentato come mostrato nella formula 2.23 :

$$\phi_u = \mathbf{e}_u, \quad \mathbf{W} : w_{uv} = cos(\phi_u, \phi_v), \quad \mathbf{X} = \mathbf{R}. \tag{2.23}$$

- **FOS-based (FB)**

  In tale predittore la rappresentazione degli users è caratterizzata dai loro relativi "Field of Studies". Sia $\mathbf{F}$ l'insieme di tutti i possibili Field of Studies e sia $\mathbf{f}_u \in \{0,1\}^{|F|}$ il vettore rappresentate l'user u, dove tutte le celle riferite ai Field of Studies sono impostate ad 1, allora il predittore è rappresentato come mostrato nella formula 2.24 :

$$\phi_u = \mathbf{f}_u, \quad \mathbf{W} : w_{uv} = cos(\phi_u, \phi_v), \quad \mathbf{X} = \mathbf{R}. \tag{2.24}$$

- **Job role-based (JB)**

  In tale predittore la rappresentazione degli users è caratterizzata dai loro relativi Jobroles. Sia $\mathbf{J}$ l'insieme di tutti i possibili Field of Studies e sia $\mathbf{j}_u \in \{0,1\}^{|J|}$ il vettore rappresentate l'user u, dove tutte le celle riferite ai Jobroles sono impostate ad 1, allora il predittore è rappresentato come mostrato nella formula 2.25 :

$$\phi_u = \mathbf{j}_u, \quad \mathbf{W} : w_{uv} = cos(\phi_u, \phi_v), \quad \mathbf{X} = \mathbf{R}. \tag{2.25}$$

2.2.5 Apprendimento

Allo scopo di ottenere il punteggio finale vengono combinati in maniera lineare i valori ottenuti dai predittori visti in precedenza, dove i pesi utilizzati sono appresi attraverso un metodo ispirato da precedenti lavori del docente Aiolli [37] e del team composto da Y. Hu, Y. Koren, e C. Volinsky [38].
Dato un insieme di predittori $\{\Psi_1, \Psi_2, ..., \Psi_K\}$ tali che $\forall i,\ \Psi_i : U \,\mathrm{x}\, I \rightarrow S$, dove

$S \equiv [0, 1]$ è l'insieme dei possibili punteggi normalizzati, viene definito il punteggio finale della coppia user-item come mostrato nella formula 2.26 :

$$\hat{r}_{ui} = \sum_{k=1}^{K} \eta_k \cdot \Psi_k(u, i) = \sum_{k=1}^{K} \eta_k \cdot \hat{r}_{ui}^{(k)} \tag{2.26}$$

dove $\eta_k \geq 0$ e $\sum_{k=1}^{K} \eta_k = 1$.

Dato un user u ed un item i viene definito il vettore dei punteggi relativo alla coppia (u, i) come mostrato nella formula 2.27 :

$$s_{ui} = \left[\hat{r}_{ui}^{(1)}, \hat{r}_{ui}^{(2)}, ..., \hat{r}_{ui}^{(K)}\right] \tag{2.27}$$

Allo scopo di apprendere η viene definito il seguente problema di regressione dove l'obbiettivo è quello di minimizzare la differenza quadrata tra il punteggio stimato s_{ui} attraverso la combinazione dei predittori ed il punteggio reale r_{ui}.
Esso viene definito come mostrato nella formula 2.28 :

$$\min_{\eta} \sum_{u,i} c_{ui}(\eta^\top s_{ui} - r_{ui})^2 \tag{2.28}$$

Il problema mostrato nella formula 2.28 può essere facilmente ridotto al problema di ottimizzazione mostrato nella formula 2.29 :

$$\min_{\eta} \frac{1}{2}\eta^\top \left(\sum_{u,i} c_{ui}(s_{ui}s_{ui}^\top)\right)\eta - \eta^\top \left(\sum_{\substack{u,i \\ r_{ui}=1}} c_{ui}s_{ui}\right) \tag{2.29}$$

dove $\| \eta \|_1 = 1$, $\eta_k \geq 0$ e c_{ui} è una costante non negativa che rappresenta la confidenza della coppia (u, i). In generale, $c_{ui} > c_{uj}$ per ogni $i \in I_u$ e $j \notin I_u$, mentre nel caso sia i che j siano entrambi positivi, ossia entrambi $\in I_u$, o negativi, ossia $\notin I_u$, entrambi avranno lo stesso peso. L'idea base è quella di dare maggiore importanza alle interazioni che sappiamo essere positive, mentre le negative, che sono intrinsecamente ambigue, avranno un valore di confidenza minore.
Il problema di ottimizzazione mostrato nella formula 2.29 è conosciuto come *problema di ottimizzazione quadratica* e può essere efficentemente risolto con librerie software già esistenti, come per esempio CVXOPT [39].

2.3 Differenze tra i modelli

Entrambi gli approcci analizzati si basano sull'idea di base di creare un ambiente nel quale combinare diverse tipologie di predittori, in maniera tale da poter apprendere i diversi aspetti del dataset, inoltre entrambi i modelli sono ispirati alla strategia KNN [40]. Nonostante tale similitudine, entrambi i metodi mostrano delle sostanziali differenze tra loro:

- **Varietà dei Predittori**

  Il modello proposto da Dávid Zibriczky [24] offre una buona varietà di predittori, infatti alcuni di essi sono ispirati all'approccio KNN [40] , come il predittore IKNN ed il MS, altri predittori hanno una struttura completamente differente, come il RCTR e l'AP. Il modello proposto dal team composto da Fabio Aiolli e Mirko Polato [25] invece propone una combinazione lineare di diversi predittori, ma quest'ultimi hanno tutti una struttura similare ed essi variano unicamente per il metodo in cui sono rappresentati i dati.

- **Gestione delle Interactions**

  Nel modello proposto da Dávid Zibriczky [24] viene contemplata la possibilità che un user possa avere più interactions con lo stesso item, o viceversa, durante il suo storico e che tali interactions possano essere di tipologia differente. Il modello proposto dal team composto da Fabio Aiolli e Mirko Polato [25] non considera il fatto che un user possa avere più di un interactions con lo stesso item, o viceversa, durante il suo storico, ma considera unicamente se è avvenuta almeno un interazione di tipologia positiva. Un ulteriore differenza sta nel fatto che mentre il modello di Dávid Zibriczky [24] considera tutte le tipologie di interactions in maniera paritaria, attribuendo ad esse lo stesso peso, che varia unicamente in base a quado tale evento sia avvenuto, il modello proposto dal team composto da Fabio Aiolli e Mirko Polato [25] considera unicamente le interazioni considerate positive durante il processo di apprendimento e non considera il fattore temporale nell'attribuire un peso a tale evento.

- **Gestione delle Impressions**

  Nel modello proposto da Dávid Zibriczky [24] viene contemplata la possibilità che un user possa avere più impression con lo stesso item, o viceversa, durante il suo storico e che tali impressions possano essere di tipologia differente. Il modello proposto dal team composto da Fabio Aiolli e Mirko Polato [25] non considera il fatto che un user possa avere più di un impressions con lo stesso item, o viceversa, durante il suo storico, ma considera unicamente se ne sia avvenuta almeno una. Un ulteriore differenza sta nel fatto che mentre il modello di Dávid Zibriczky [24] attribuisce alle varie impressions un peso che varia unicamente in base a quado tale evento sia avvenuto, mentre il modello proposto dal team composto da Fabio Aiolli e Mirko Polato [25] non considera effettua tale considerazione ed assegna un peso identico a tutte gli eventi di tipologia impressions.

- **Decadenza Temporale degli Eventi**

  Nel modello proposto da Dávid Zibriczky [24] viene tenuto conto del fattore temporale sia per quanto riguarda l'attribuzione dei pesi riguardanti i vari eventi, sia di tipologia interaction che impressions, inoltre il predittore AP effettua una discriminazione di tipo temporale verso gli item più vecchi. Il modello proposto dal team composto da Fabio Aiolli e Mirko Polato [25] non tiene conto di quando tali eventi siano avvenuti, ma attribuisce un valore unitario a tutte le tipologie di eventi.

Capitolo 3

Sviluppo e Sperimentazione

Durante questo lavoro di tesi è stato effettuata un'implementazione dei metodi analizzati, allo scopo di analizzarli nel dettaglio e di poter creare un metodo ibrido che combini gli aspetti migliori riguardanti i modelli analizzati. Inoltre è stato effettuato anche uno studio preliminare del dataset relativo alla challenge allo scopo di analizzarne le caratteristiche e poter creare un sistema di job recommendation specifico per essi.

3.1 Studio e Trattamento dei Dati

Il dataset fornito per la challenge è un campione dei dati riguardanti il social network Xing. Tali informazioni sono state trattate in maniera da rendere anonime sia le identità dei vari users, sia quelle dei vari items, ossia i jobs.

3.1.1 File del Dataset

Come detto in precedenza tali dati sono contenuti in quattro file, che erano liberamente scaricabili per chiunque intendesse partecipare alla challenge:

- **users.csv**
  File contenente i dati relativi a gli users
- **items.csv**
  File contenente i dati relativi a gli items
- **interactions.csv**
  File contenente i dati relativi a le interactions tra users ed items
- **impressions.csv**
  File contenente i dati relativi a le impressions tra user ed items

3.1.2 Classi del Dataset

A partire dai file descritti precendentemente è stato creato un file database generico allo scopo di poter studiare e trattare i dati in maniera più agevole. Tale database è strutturato secondo le seguenti classi:

- **Users**
  Gruppo di dati contenente le informazioni riguardanti gli utenti. Come detto in precedenza tali dati sono stati modificati allo scopo di anonimizzarli. Queste informazioni sono organizzate secondo i campi della tabella 3.1 :

Nome Campo	Tipo	Descrizione
id	Int	Valore numerico identificativo dell'user
career level	Text	Valore numerico rappresentativo del livello di carriera dell'user
discipline id	Text	Valore numerico rappresentativo della tipologia di disciplina riguardante l'user
industry id	Text	Valore numerico rappresentativo della tipologia di ambito lavorativo dell'user
country	Text	Stringa riguardante lo stato di appartenenza dell'user
region	Text	Valore numerico rappresentativo della regione geografica dell'user
experience n entries class	Text	Valore numerico rappresentante il livello di esperienza dell'user
experience years experience	Text	Valore numerico rappresentante il livello di esperienza dell'user
experience years in current	Text	Valore numerico rappresentante il livello di esperienza dell'user
edu degree	Text	Valore numerico rappresentate il livello di istruzione dell'user

Tabella 3.1: Informazioni riguardanti gli users

Inoltre per quanto riguarda gli users sono presenti due gruppi di dati riguardanti i jobroles ed i field of studies. Codesti gruppi di informazioni sono utili per definire i gruppi per i quali un user fa parte ed essi sono organizzati rispettivamente secondo la tabella 3.2 e 3.3:

Nome Campo	Tipo	Descrizione
user id	Int	Valore numerico identificativo dell'user
jobrole	Text	Valore numerico identificativo del jobrole

Tabella 3.2: Informazioni riguardanti i jobroles degli users

Nome Campo	Tipo	Descrizione
user id	Int	Valore numerico identificativo dell'user
fos	Text	Valore numerico identificativo del field of studies

Tabella 3.3: Informazioni riguardanti i field of studies degli users

- **Items**

  Gruppo di dati contenente le informazioni riguardanti i lavori che poi andranno raccomandati ai vari utenti. Come detto in precedenza tali dati sono stati modificati allo scopo di anonimizzarli. Queste informazioni sono organizzate secondo i campi della tabella 3.4 :

Nome Campo	Tipo	Descrizione
id	Int	Valore numerico identificativo dell'item
career level	Text	Valore numerico rappresentativo del livello di carriera dell'item
discipline id	Text	Valore numerico rappresentativo della tipologia di disciplina riguardante l'item
industry id	Text	Valore numerico rappresentativo della tipologia di ambito lavorativo dell'item
country	Text	Stringa riguardante lo stato di appartenenza dell'item
region	Text	Valore numerico rappresentativo della regione geografica dell'item
latitude	Text	Valore numerico indicante la latitudine dell'item
longitude	Text	Valore numerico indicante la longitudine dell'item
employment	Text	Valore numerico indicante la tipologia di occupazione proposta dall'item
created at	Text	Valore numerico indicante quando l'item è stato creato
active during test	Text	Valore numerico indicante se l'item è risultato attivo durante i test

Tabella 3.4: Informazioni riguardanti gli items

Inoltre per quanto riguarda gli items sono presenti due gruppi di dati riguardanti i titles ed i tags. Codesti gruppi di informazioni sono utili per definire i gruppi per i quali un item fa parte ed essi sono organizzati rispettivamente secondo la tabella 3.5 e 3.6:

Nome Campo	Tipo	Descrizione
item id	Int	Valore numerico identificativo dell'item
title	Text	Valore numerico identificativo del title riguardante l'item

Tabella 3.5: Informazioni riguardanti i titles degli items

Nome Campo	Tipo	Descrizione
item id	Int	Valore numerico identificativo dell'item
tag	Text	Valore numerico identificativo del tag riguardante l'item

Tabella 3.6: Informazioni riguardanti i tags degli items

- **Interactions**

  Gruppo di dati riguardanti le interazioni effettuati dagli utenti con gli item. Esistono 4 tipologie di interazione:

  1. Click
  2. Bookmark
  3. Reply
  4. Delete

  Le interazioni di tipologia 1,2 e 3 sono considerate interazioni positive, mentre le interazioni di tipologia 4 sono considerate negative. Le informazioni riguardanti tali interazioni sono organizzate secondo la tabella 3.7:

Nome Campo	Tipo	Descrizione
user id	Int	Valore numerico identificativo dell'user
item id	Int	Valore numerico identificativo dell'item
interaction type	Text	Valore numerico identificativo della tipologia di interazione effettuata
created at	Text	Valore numerico indicante quando è stata effettuata l'interazione

Tabella 3.7: Informazioni riguardanti le interactions

- **Impressions**

  Gruppo di dati riguardanti gli items che sono stati proposti agli users dal precedente sistema di job raccomendation di Xing. Essi sono sono organizzati sono organizzate secondo la tabella 3.8:

Nome Campo	Tipo	Descrizione
id	Int	Valore numerico identificativo dell'impression
user id	Int	Valore numerico identificativo dell'user
year	Text	Valore numerico indicante l'anno nel quale l'impression è avvenuta
week	Text	Valore numerico indicante la settimana nella quale l'impression è avvenuta

Tabella 3.8: Informazioni riguardanti le impressions

A differenza delle interactions, una singola impression può interessare più item allo stesso tempo, inoltre un singolo item può comparire più volte all'interno della stessa impression, per tale motivo le informazioni riguardanti gli item interessati dalla impression sono organizzati secondo la tabella 3.9:

Nome Campo	Tipo	Descrizione
imp id	Int	Valore numerico identificativo dell'impression
item id	Int	Valore numerico identificativo dell'item
ord	Int	Valore numerico indicante l'ordine dell'item all'interno dell'impression

Tabella 3.9: Informazioni riguardanti gli items relativi alle impressions

Uno schema completo di tale dataset e delle sue ralazioni può essere osservato nell'immagine 3.1:

Figura 3.1: Diagramma riguardante le relazioni delle classi

3.1.3 Studio dei dati

Sono stati effettuati diversi studi preliminari sui dati allo scopo di poterne comprendere appieno le caratteristiche legate ad esso. La tabella 3.10 mostra le statistiche principali relativi al dataset:

Dataset	# eventi	# users	# items
E(1): Click	7,183,038	769,396	998,424
E(2): Bookmark	206,191	59,063	142,908
E(3): Reply	422,026	107,463	190,099
E(4): Delete	1,015,423	44,595	215,844
E(5): Impression	201,872,093	2,755,167	846,814
E(1-5): All events	210,698,771	2,792,405	1,257,422
Catalog	-	1,367,057	1,358,098
Catalog and E(5)	-	1,292,662	843,589
Catalog and E(1-4)	-	784,687	1,025,582

Tabella 3.10: Statistiche relative al dataset

I risultati mostrano che il 95% degli eventi sono di tipologia impressions ed essi sono circa 4 volte il numero degli eventi di tipo interactions. Un altro risultato interessante è che dalle osservazioni effettuate sui dati, circa il 28% degli users risulta inattivo e che per ogni item in media è protagonista di 8 eventi, il che è numero basso.

In base a tali statistiche 3 categorie di utenti possono essere distinte:

1. **Nuovi Utenti**

   Utenti per i quali sono presenti solamente le informazioni relative ai metadata. Tale categoria comprende circa il 12% degli users.

2. **Utenti Inattivi**

   Utenti per i quasi sono presenti le informazioni relative ai metadata ed alle impressions. Tale categoria comprende circa il 16% degli users.

3. **Utenti Attivi**

   Utenti per i quali sono presenti le informazioni relative ai metadata, alle impressions ed alle interactions. Tale categoria comprende circa il 73% degli users.

In maniera analoga è possibile suddividere gli items in 2 differenti categorie:

1. **Nuovo Item**

   Items che non sono ancora stati protagonisti di alcuna tipologia di evento nel loro storico. Tale categoria comprende circa il 25% degli items.

2. **Vecchio Item**

   Items che sono stati protagonisti di almeno un qualche tipo di evento nel loro storico. Tale categoria comprende circa il 75% degli items.

La figura 3.2 mostra la probabilità che si effettui un evento di tipologia interaction con un nuovo item. Com'è possibile osservare c'è una probabilità del 25% che un item nuovo diventi protagonisti di almeno un interazione entro 24 ore dal tempo dell'ultimo training e tale probabilità aumento a circa il 50% nel tempo di una settimana. Complessivamente, la probabilità di interazione con un nuovo item nei successivi 7 giorni è del 30,2%.

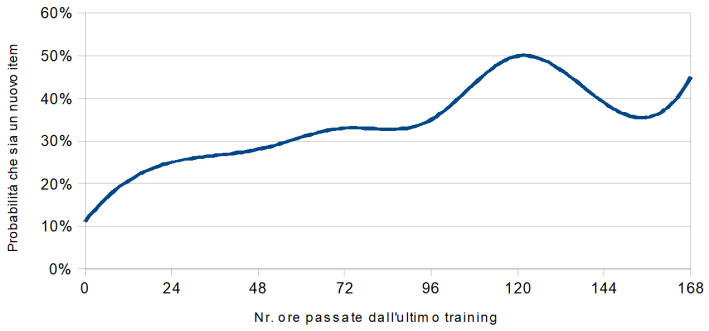

Figura 3.2: Probabilità di interazione con un nuovo item

Una caratteristica osservata riguarda agli users è che essi tendono ad interagire con item già visionati in precedenza. Come si può evincere dalla figura 3.3 mostra la probabilità che visiti nuovamente un item dopo che siano state effettuate delle interazioni su altri items. I risultati mostrano che la prima interazione di un user

sarà con un item che ha già visionato in precedenza con una probabilità di poco superiore al 15%.

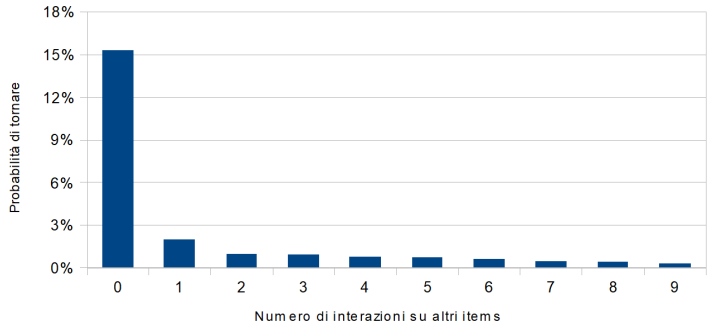

Figura 3.3: Probabilità di interagire nuovamente con un item già visionato

Un comportamento osservato relativo alle impressions riguarda il fatto che esse tendono a raccomandare ripetutamente gli stessi items. La figura 3.4 riassume la distribuzione degli eventi di tipologia impressions in base a quante settimane siano passate dalla prima raccomandazione di un dato item per un dato user. I risultati mostrano che approssivamente il 38% degli eventi di tale tipologia sono effettuati su item che erano stati già raccomandati in passato, quindi evidenziano questo comportamento del precedente sistema di raccomandazione di Xing.

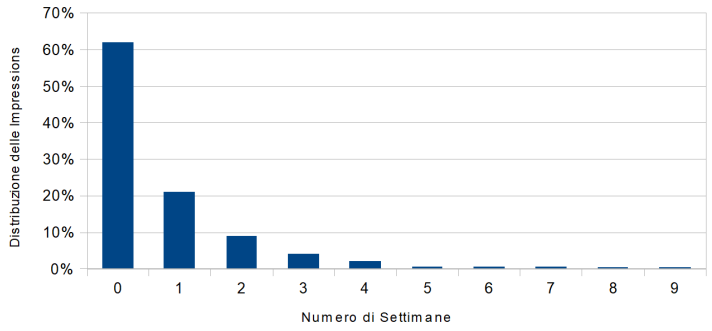

Figura 3.4: Distribuzione delle impressions

Infine è stato analizzato il comportamento riguardante il cambiamento di popolarità di un item con l'avanzare del tempo. La figura 3.5 raffigura il cambio di popolarità stimato per un nuovo user ogni 7 giorni. I risultati mostrano che

la popolarità di un item con un età di 25-30 giorni tende a diminuire in maniera considerevole, il che indica che tale popolarità tende a diminuire rapidamente con l'avanzare del tempo e che in media si hanno 25-30 giorni utili per raccomandare un dato item.

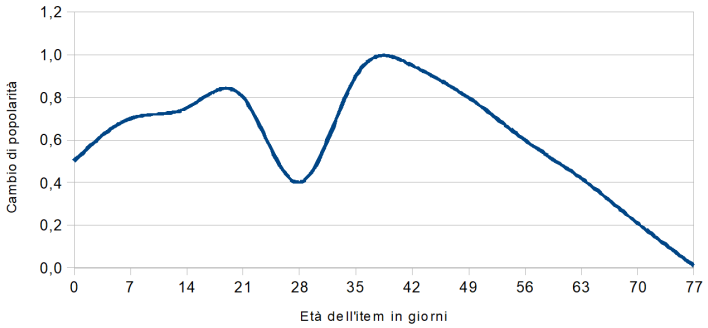

Figura 3.5: Cambio di popolarità di un item

3.1.4 Problematiche Riscontrate e Scelte di Sviluppo

I dati rilasciati per la challenge sono dati presi direttamente dal portale Xing, i quali sono stati anonimizzati allo per motivi di privacy. Tale caratteristi permette di avere una simulazione realistica del sistema di job recommendation, ma porta anche alla presenza di alcune situazioni anomale:

- **Ripetizione di Dati**

  Durante la creazione del database è stato riscontrato che alcuni dati venivano dichiarati più volte. Come scelta di sviluppo in questi casi è sempre stata sempre considerata l'ultima dichiarazione dei dati, questo poiché è realistico pensare che essa sia la versione più recente di quet'ultimi.

- **Presenza di Dati Parziali**

  Durante l'analisi dei dati è stata evidenziata la presenza nelle classi user ed item del database di situazioni nelle quali in alcuni campi comparisse il valore NULL. Tale situazione affligge sia la classe USERS che ITEMS del database e risulta abbastanza frequente all'interno di dati reali, poichè capita spesso che non vengano compilati tutti i campi non obbligatori riguardanti le

informazioni. Come scelta di sviluppo per ogni campo è stato selezionato un valore non presente originariamente in tale campo ed è stato usato nel caso si fosse in presenza di dati mancanti.

- **Presenza di Dati Anomali**

  Durante l'analisi dei dati è stata evidenziata la presenza di alcui dati riguardanti le classi INTERACTIONS ed IMPRESSIONS di eventi riguardanti users ed items che non avevano alcun tipo di dichiarazione all'interno delle loro rispettive classi. Come scelta di sviluppo tali dati sono stati eliminati all'interno del database, tale scelta è stata effettuata poiché non era possibile dedurre alcun tipo di informazione riguardo agli user/item a cui si riferivano tali dati, rendendo impossibile effettuare una corretta raccomandazione per essi.

- **Assenza del Test Set**

  Questa problematica è stata evidenziata fin dall'inizio di questo progetto di tesi, poiché non è stato possibile avere accesso al test set utilizzato nella challenge per calcolare il punteggio da attribuire ad un metodo. Tale situazione è potenzialmente critica, poiché non permette alcun tipo di verifica riguardante la correttezza o meno del sistema di raccomandazione implementato. Come scelta di sviluppo, per risolvere tale situazione è stato è stata effettuata una suddivisione temporale del training set in 6 settori, come mostrato nella figura 3.6.

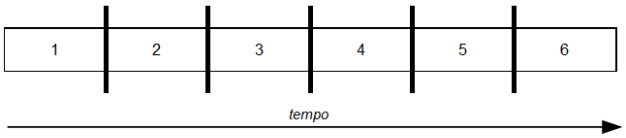

Figura 3.6: Divisione temporale del test-set

Per ogni users è stato controllato in quali settori venissero effettuate delle interazioni di tipologia "clicks" ed è stato creato un test set all'interno dei relativi settori inserendo gli items protagonisti di tali eventi. Da tale controllo

è stato escluso la prima sezione temporale. Successivamente per ogni user è stato controllato quale fosse il settore cronologicamente più recente dove ci fosse la presenza di un test set valido, quindi per tale utente è stato usato il test set presente in tale settore, mentre i settori precedenti sono stati usati come training set.

Inoltre è stato creato un database ridotto, contenente 5000 user ed i loro rispettivi eventi. Per quanto riguarda gli items, sono stati selezionati coloro che fossero interessati dalle interazioni dei primi 5000 utenti.
Infine per ogni utente è stato controllata la presenza di un test set valido, in tale modo sono stati individuati 2593 target users all'interno di tale database ridotto.
Tale operazione è stata effettuata allo scopo di poter effettuare un maggior numero di test sui modelli.

3.2 Implementazione Metodo di Dávid Zibriczky

L'implementazione di tale metodo è avvenuta utilizzando il linguaggio di programmazione Python 2.7.12. Il programma creato allo scopo ha la seguente struttura:

- **Sqlite.py**

  File contenente la classe Sqlite, la quale si occupa della creazione del database e della correzzione dei dati anomali.

- **Transfer_to_RAM.py**

  File contenente la classe T_to_RAM, la quale di occupa della creazione della struttura dati rappresentante i dati e del pre-calcolo dei predittori fissi, ossia dei metodi di cui il valore di ritorno rimane fisso a prescindere dall'evento dell'user analizzato.

- **Predictors.py**

  File contenente la classe Predictor, la quale si occupa del calcolo dei predittori non fissi, ossia dei metodi di cui il valore di ritorno varia in base all'evento dell'user analizzato. Inoltre si occupa anche della parte di apprendimento,

ossia della combinazione dei vari valori dei predittori e del calcolo del punteggio da attribuire alla predizione.

- **Main.py**

  File contenente la classe Main del programma.

3.2.1 Rappresentazione dei dati

Allo scopo di rendere agevole il calcolo dei predittori è stata definita una rappresentazione dei dati basata su 5 strutture di tipo dizionario:

1. **Users Dict**

   Tale struttura dati è stata strutturata in maniera tale da rilasciare tutte le informazioni inerenti l'utente di cui viene inserito il codice identificativo ed è strutturata nella seguente maniera:

$Users\_d[User\_id] =$

	0	1	2	
	Career_level	Discipline_id	Industry_id	...
	(int)	(int)	(int)	

	3	4	5	6	
...	Country	Region	Exp_in_entries_class	Exp_year_experienced	...
	(int)	(int)	(float)	(float)	

	7	8	9	10	
...	Exp_year_current	Edu_degree	U_fos_l	U_jbrl_l	...
	(float)	(int)	[...]	[...]	

	11	12	
...	U_nr_gr	U_per_d	
	(int)	{...}	

   dove:

   - **U_fos_l**

     Lista dei codici identificativi riguardanti i Field of Studies a cui appartiene l'utente.

   - **U_jbrl_l**

     Lista dei codici identificativi riguardanti i Jobroles a cui appartiene l'utente.

- **U_nr_gr**

  Numero dei gruppi a cui appartiene l'utente.

- **U_per_d**

  Dizionario conente le informazioni utente relative ai vari periodi temporali nel quale è stato suddiviso il training set. Esso è strutturato nella seguente maniera:

  $U\_per\_d[Period] =$

	0	1	2	3	
	U_int_l	U_imp_l	w_e	RCTR_s	...
	[...]	[...]	(float)	(float)	

	4	5	
...	AS_score_v	Test_set	
...	[...]	[...]	

  dove:

  - **U_int_l**

    Lista riguardante le interazioni dell'utente avvenute nell'attuale settore temporale ed in quelli precedenti. Essa contiene il codice identificativo dell'item con il quale tale evento è avvenuto, la tipologia dell'interazione, il time stamp indicante quando tale evento sia avvenuto ed il peso da attribuire ad esso. Tale lista è strutturata nella seguente maniera:

    $U\_int\_l[i] =$

0	1	2	3
Item_id	Int_type	Created_at	Score
(int)	(int)	(int)	(float)

  - **U_imp_l**

    Lista Contenente i codici identificativi riguardanti gli eventi di tipologia impressions dell'utenti avvenute nell'attuale settore temporale ed in quelli precedenti.

  - **w_e**

    Valore numerico risultato della sommatoria di tutti i pesi di tutti gli eventi dell'utente avvenuti nell'attuale settore temporale ed in quelli precedenti.

  - **RCTR_s**

Punteggio del predittore RCTR relativo all'utente.

- **AS_score_v**

  Vettore riguardante i punteggi del predittore AS relativo all'utente. Esso contiene i punteggi relativi alle varie tipologie di interazione ed è strutturato nella seguente maniera:

	0	1	2	
$AS\_score\_v =$	AS_score_1	AS_score_2	AS_score_3	...
	(float)	(float)	(float)	

3	
AS_score_4	...
(float)	

- **Test_set**

  Lista contenente i codici identificativi degli items che sono stati soggetti ad eventi di tipologia "Click". Tale lista viene usata come test set per verificare la bontà del sistema.

2. **Items Dict**

   Tale struttura dati è stata strutturata in maniera tale da rilasciare tutte le informazioni inerenti l'item, ossia il job di cui viene inserito il codice identificativo, ed è strutturata nella seguente maniera:

	0	1	2	3	
$Items\_d[Item\_id] =$	Career_level	Discipline_id	Industry_id	Country	...
	(int)	(int)	(int)	(int)	

4	5	6	7	8	9	
...	Region	Latitude	Longtude	Employment	Created_at	Diff_t	...
	(int)					

10	11	12	13	14	
...	Active_dur_test	It_titles_d	It_tags_d	R_cos_v	Ap_score	...
	(float)	{...}	{...}	[...]	(float)	

15	16	17	18
...	Sim_cos_titl_den	Sim_cos_tags_den	Sim_cos_den_v_rest	I_per_d
	(float)	(float)	(float)	{...}

dove:

- **Diff_t**

  Valore numerico indicante la differenza temporale tra la creazione dell'item di riferimento e la creazione dell'item più recente presente nel training set.

- **It_titles_d**

  Dizionario contenente i punteggi idf riferiti ai titles di cui appartiene l'item di riferimento. Esso è utilizzato per il calcolo della similarità coseno tra items relativa ai loro titles.

  $$It\_titles\_d[Title\_id] = Title\_idf\_score$$

- **It_tags_d**

  Dizionario contenente i punteggi idf riferiti ai tags di cui appartiene l'item di riferimento. Esso è utilizzato per il calcolo della similarità coseno tra items relativa ai loro tags.

  $$It\_tags\_d[Tags\_id] = Tags\_idf\_score$$

- **R_cos_v**

  Vettore conentente i valori rimanenti necessari al calcolo della similarità coseno tra items. Esso è strutturato nella seguente maniera:

  $R\_cos\_v =$

0	1	2	3
Industry_id	Country	Region	Discipline_id
(int)	(int)	(int)	(int)

- **Ap_score**

  Punteggio del predittore AP relativo all'item di riferimento.

- **Sim_cos_titl_den**

  Valore numerico utilizzato come denominatore per il calcolo della similarità coseno tra items relativa ai loro titles.

- **Sim_cos_tags_den**

  Valore numerico utilizzato come denominatore per il calcolo della similarità coseno tra items relativa ai loro tags.

- **Sim_cos_den_v_rest**

  Valore numerico utilizzato come denominatore per il calcolo della simila-

rità coseno tra items relativa ai valori contenuti all'interno del vettore R_cos_v.

- **I_per_d**

  Dizionario conente le informazioni dell'item relative ai vari periodi temporali nel quale è stato suddiviso il training set. Esso è strutturato nella seguente maniera:

$I\_per\_d[Period] =$

0	1	2	
Users_si_so_d	RCTR_single_sc	Iknn_si_sc	...
{...}	(float)	(float)	

3	
Iknn_so_sc	...
(float)	

  dove:

  - **Users_si_so_d**

    Dizionario contenente i valori utilizzati per il calcolo del predittore IKNN ed RCTR. Ognuno dei punteggi viene preceduto dal time stamp relativo a quando è avvunuto il relativo evento. Esso è strutturato nella seguente maniera:

$Users\_si\_so\_d[User\_id] =$

0	1	
Iknn_si_last_t	Iknn_si_last_sc	...
(int)	(float)	

2	3	4	
Iknn_so_last_t	Iknn_so_last_sc	RCTR_cl_last_t	...
(int)	(float)	(int)	

5	
RCTR_cl_last_sc	...
(float)	

  - **RCTR_single_sc**

    Punteggio del predittore RCTR relativo all'item di riferimento.

  - **Iknn_si_sc**

    Valore numerico utilizzato come denominatore per il calcolo della similarità tra items relativo al predittore IKNN.

– **Iknn_so_sc**

Valore numerico utilizzato come denominatore per il calcolo della similarità tra items relativo al predittore IKNN.

3. **Impressions Dict**

Tale struttura dati è stata strutturata in maniera tale da rilasciare tutte le informazioni inerenti le impressions, ed esso è strutturato nella seguente maniera:

$Imp\_d[Imp\_id] =$

	0	1	2	3	4		5	6
	User_id	Year	Week	Diff_t	Nr_Items	...	Items_d	Score_s_imp
	(int)	(int)	(int)	(int)	(int)		{...}	(float)

dove:

- **User_id**

  Valore identificativo dell'utente protagonista dell'impression.

- **Year**

  Valore numerico indicante l'anno nel quale è avvenuto l'evento.

- **Week**

  Valore numerico indicante la settimana nel quale è avvenuto l'evento.

- **Diff_t**

  Valore numerico indicante la differenza temporale tra la creazione dell'impression di riferimento e la creazione dell'evento di tale tipologia più recente presente nel training set.

- **Nr_Items**

  Valore numerico indicante il numero di items protagonisti dell'impression.

- **Items_d**

  Dizionario che contiene come chiavi gli items protagonisti dell'impressions e che ritorna quante volte questo oggetto compaia all'interno dell'evento di riferimento.

  $$Items\_d[Item\_id] = Nr\_Times$$

- **Score_s_imp**

  Valore numerico indicante il peso da attribuire all'impression.

4. **Field of Studies Dict**

   Tale struttura dati è stata strutturata in maniera tale da rilasciare le informazioni interenti ai vari gruppi Field of Studies ed è strutturata nella seguente maniera:

$$Fos\_d[Fos\_id] = \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \text{Nr\_Users} & \text{Users\_l} & \text{Per\_d} \\ \hline \text{(int)} & [...] & [...] \\ \hline \end{array}$$

   dove:

   - **Nr_Users**

     Valore numerico indicante il numero di utenti cha fanno parte del gruppo.
   - **Users_l**

     Lista degli utenti facente parte del gruppo.
   - **Per_d**

     Dizionario che riceve in ingresso il periodo di riferimento ed il codice identificativo relativo all'item di interesse e ritorna la somma dei pesi degli eventi avvenuti su tale job dagli utenti del gruppo.

$$Per\_d[Period][Item_id] = Item\_pop_g\_score$$

5. **Jobroles Dict**

   Tale struttura dati è stata strutturata in maniera tale da rilasciare le informazioni interenti ai vari gruppi Jobroles ed è strutturata nella seguente maniera:

$$Jbrl\_d[Jbrl\_id] = \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \text{Nr\_Users} & \text{Users\_l} & \text{Per\_d} \\ \hline \text{(int)} & [...] & [...] \\ \hline \end{array}$$

   dove:

   - **Nr_Users**

     Valore numerico indicante il numero di utenti cha fanno parte del gruppo.
   - **Users_l**

     Lista degli utenti facente parte del gruppo.

- **Per_d**

  Dizionario che riceve in ingresso il periodo di riferimento ed il codice identificativo relativo all'item di interesse e ritorna la somma dei pesi degli eventi avvenuti su tale job dagli utenti del gruppo.

  $$Per\_d[Period][Item_id] = Item\_pop_g\_score$$

3.2.2 Predittori ed Apprendimento

Una volta ultimala la procedura di creazione delle strutture dati relative ad i dati ed al rispettivo pre-calcolo dei predittori fissi, ossia dei metodi il cui risultato non varia in base all'evento analizzato, avviene il calcolo dei rimanenti predittori e la loro relativa combinazione.
Per ogni user viene calcolato il punteggio di ogni item presente all'interno del dataset ed è calcolato nella seguente maniera:

1. **Valori Fissi**

   Viene definito un punteggio inziale per l'utente di riferimento come mostrato nella formula 3.1 :

   $$User\_score = RCTR\_score + AS\_score \tag{3.1}$$

   dove $RCTR\_score$ è il punteggio del predittore RCTR ed AS_score è la sommatoria degli valori del predittore AS riguardanti gli eventi di tipologia 1, 3 e 4, ossia Click, Reply e Delete.
   Partendo da tale punteggio, per ogni item viene definito il punteggio iniziale della coppia user/item come mostrato nella formula 3.2 :

   $$Base\_score = User\_score + AP\_score \tag{3.2}$$

   dove AP_score è il punteggio del preditore AP dell'item.

2. **Predittore UPOP**

   Viene definito il punteggio dei gruppi dell'utente riferito all'item sotto esame come mostrato nella formula 3.3 :

   $$Upop\_score = \frac{1}{|U(g)|} \sum_{g \in U(g)} \frac{Item\_pop_g\_score}{Item\_pop_g\_score + \lambda} \tag{3.3}$$

Quindi viene effettuata una sommatoria dei valori UPOP riferiti all'item, normalizzando tale valore sia con il coefficente di regolazione λ e sia che con il numero dei gruppi utente.

3. **Predittore IKNN ed MS**

   I punteggi relativi ai predittori IKNN ed MS vengono definiti effettuando una sommatoria delle rispettive similarità dell'item sotto esame assieme agli items protagonisti degli eventi sia interactions che impressions memorizzati nello storico dell'utente. Tali punteggi vengono normalizzati utilizzando il peso dei vari event, il coefficente di regolazione λ e l'esponente di normalizzazione α. Per il predittore IKNN vengono usate le configurazioni IKNN(C,C), IKNN(R,R) ed IKNN(R,C), dove R e C si riferiscono rispettivamente agli eventi di tipologia impressions ed interactions.

4. **Apprendimento**

   Una volta effettuato il calcolo dei vari predittori non fissi riferiti all'item sotto esame, i valori ottenuti vengono sommati tra di loro allo scopo di ottenere il punteggio finale. Nell'implementazione non è stato usato il processo di ottimizzazione suggerito dal modello di Dávid Zibriczky per la combinazione dei valori dei predittori per ridurre la complessità di calcolo del programma e poiché dai test effettuati da Dávid Zibriczky stesso esso non è risultato estremamente efficace.

5. **Ordinamento e calcolo punteggio**

   Una volta definito il punteggio finale da attribuire a tutti gli items per un user, essi vengono ordinati in maniera decrescente e viene creata una lista oridnata contenente i 30 con il punteggio più alto. Tale lista viene confrontata con il test set attribuito all'user e ne viene calcolato il punteggio di tale raccomandazione utilizzando il sistema della challenge.

Il punteggio finale da attribuire al modello è definito come la somma dei punteggi di tutte le raccomandazioni effettute agli users.

3.2.3 Test

I test sul modello di Dávid Zibriczky sono stati effettuati presso il nodo *dellsrv0* del centro di calcolo del dipartimento di Matematica dell'università di Padova [41] sulla seguente configurazione:

- CPU: 2 x Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
- RAM: 160GB
- #Cores: 20
- Connectivity: Ethernet 10GB
- Local Storage: 200GB

Per i test sono stati usati i valori $\lambda = 0.01$ ed $\alpha = 0.4$, inoltre allo scopo di velocizzare il calcolo sono stati creati 25 jobs paralleli che computavano solamente le raccomandazioni degli users a loro assegnate.
Ogni job ha impiegato un tempo medio di 7 ore per ultimare tutte le raccomandazioni dei 2593 target users ed il punteggio finale ottenuto è di 12781.34. Essendo tale risultato in linea con i punteggi dichiarati nel paper [24] , esso conferma la comparabilità del dataset ridotto da noi creato con quello completo e la corretta implementazione del metodo.

3.3 Implementazione Metodo di Polato et al.

L'implementazione di tale metodo è avvenuta utilizzando il linguaggio di programmazione Python 2.7.12. Il programma creato allo scopo ha la seguente struttura:

- **Sqlite.py**

  File contenente la classe Sqlite, la quale si occupa della creazione del database e della correzzione dei dati anomali.
- **Representation.py**

  File contenente la classe Representation, relativa alla rappresentazione dei dati.
- **Predictors.py**

  File contenente la classe Pred e delle classi figli PredU e PredI, che si occupano

rispettivamente del calcolo dei predittori User-based ed Item-based.

- **Reco.py**

  File contenente la classe Reco, relativa alle operazioni di raccomandazione degli items agli users. Tale classe si interfaccia con il file Regression.py per l'operazione di apprendimento e si occupa infine anche della creazione della lista ordinata relativa agli items da raccomandare ai vari users.

- **Regression.py**

  File contenente le classi Regression e Regression2, relative alle operazioni di apprendimento.

- **Main.py**

  File contenente la classe Main del programma.

3.3.1 Rappresentazione dei dati

Allo scopo di rendere agevole il calcolo dei predittori è stata definita una rappresentazione dei dati basata su 6 strutture di tipo dizionario:

1. **u2i_train**

   Dizionario che dato il codice identificativo di un user ritorna l'insieme degli items con il quale tale utente ha avuto almeno un interazione di tipologia positiva nel suo storico, ossia di tipologia Click, Bookmark o Reply:

   $$u2i\_train[user\_id] = Int\_items\_set_u \tag{3.4}$$

2. **u2i_impressions**

   Dizionario che dato il codice identificativo di un user ritorna la lista degli items con il quale tale utente ha avuto almeno un interazione di tipologia positiva nel suo storico:

   $$u2i\_impressions[user\_id] = Imp\_items\_list_u \tag{3.5}$$

3. **i2u_train**

   Dizionario che dato il codice identificativo di un item ritorna l'insieme degli users con il quale tale job ha avuto almeno un interazione di tipologia positiva nel suo storico, ossia di tipologia Click, Bookmark o Reply:

$$i2u\_train[item\_id] = Int\_users\_set_i \tag{3.6}$$

4. **u_norm**

   Dizionario che dato il codice identificativo di un user ritorna il valore della norma da utilizzare per il calcolo della similarità coseno con altri utenti:

$$u\_norm[user\_id] = Norm\_value_u \tag{3.7}$$

5. **i_norm**

   Dizionario che dato il codice identificativo di un item ritorna il valore della norma da utilizzare per il calcolo della similarità coseno con altri jobs:

$$i\_norm[item\_id] = Norm\_value_i \tag{3.8}$$

6. **filtered**

   Dizionario che dato il codice identificativo di un user ritorna la lista degli items che saranno presi in esame per effettuare la raccomandazione relativa all'utente:

$$filtered[user\_id] = Item\_list_u \tag{3.9}$$

3.3.2 Predittori ed Apprendimento

Una volta ultimata la procedura di creazione delle strutture dati relative ai dati avviene la definizione delle strutture dati rappresentanti i vari predittori.
Essi sono definiti in 3 strutture dati principali:

$preds = [pred\_u, pred\_i, pred\_po]$

1. **pred_u**

   Struttura dati relativa ai predittori user-based.

2. **pred_i**

   Struttura dati relativa ai predittori item-based.

3. **pred_po**

   Struttura dati relativa al predittore popularity-based relativo agli users.

Dopo tale procedimento viene effettuata la fase di apprendimento ed ottimizzazione, dove vengono inoltre definiti i valori dei η e dei pesi da attribuire ai singoli valori dei predittori in base alle caratteristiche degli users su cui effettuare l'attività di job reccomandation.
Successivamente per ogni target-user viene effettuata l'effettiva computazione delle raccomandazioni, calcolando il punteggio per ogni items pre-selezionato per la raccomandazione all'interno della struttura dati $filtered$.
Infine per ogni user viene comparata la raccomandazione proposta con il rispettivo test-set e ne viene calcolato il punteggio secondo la metodologia definita dalla challenge.

3.3.3 Test

I test sul modello di Fabio Aiolli e Mirko Polato sono stati effettuati presso il nodo *dellsrv0* del centro di calcolo del dipartimento di Matematica dell'università di Padova [41] sulla seguente configurazione:

- CPU: 2 x Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
- RAM: 160GB
- #Cores: 20
- Connectivity: Ethernet 10GB
- Local Storage: 200GB

Per ogni user sono stati calcolati gli score relativi a tutti gli item presenti all'interno del dataset. Tale scelta è stata effettuata con il proposito di testare in maniera completa il modello, inoltre allo scopo di velocizzare il calcolo sono stati creati 25 jobs paralleli che computavano solamente le raccomandazioni degli users a loro assegnate.

Ogni job ha impiegato un tempo medio di 4 ore per ultimare tutte le raccomandazioni dei 2593 target users ed il punteggio finale ottenuto è di 8131,41. Essendo tale risultato in linea con i punteggi dichiarati nel paper [25] , esso conferma la comparabilità del dataset ridotto da noi creato con quello completo e la corretta implementazione del metodo.

3.4 Sperimentazioni

Successivamente all'implementazione dei metodi proposti nei paper analizzati sono state effettuate delle sperimentazioni su di essi allo scopo di provare a migliorarne le performance. Tali test sono stati effettuati senza modificare la struttura principale dei rispettivi programmi e delle rappresentazioni dei dati. Come per i modelli originali, tutti i test sono stati effettuati sul dataset ridotto, allo scopo di poter effettuare un maggior numero di test.

3.4.1 Modifiche al modello di Dávid Zibriczky

Una prima idea è stata quella di confrontare direttamente i campi comuni relativi dei metadati relativi agli users e degli items, in maniera tale da evitare di raccomandare ad un utente dei jobs per il quale esso non fosse qualificato. Tali valori sono:

- Car_level
- Discipline_id
- Industry_id
- Country
- Region

Un primo approccio è stato quello di togliere dalla raccomandazione per i vari users qualsiasi item i quali metadati comuni non combaciassero, ma sebbene tale approccio si dimostrasse vantaggioso dal punto di vista della complessità computazionale, effettuava troppe esclusioni, lasciando circa il 14% degli users senza alcun item disponibile per la raccomandazione.

Un altro approccio è stato quello di promuovere gli items che contenessero valori comuni con quelli dell'user per il quale si stesse facendo la raccomandazione.

Da tale idea per ognuno dei valori comuni di comparazione è stato definito un predittore dell'item i per l'utente u come mostrato nella formula 3.10 :

$$p(u,i) = \frac{0.05}{1 + |val_u - val_i|} \tag{3.10}$$

dove:

- val_u: valore di confronto dell'user.
- val_i: valore di confronto dell'item.

Codesto predittore si basa sul fatto che tutti i dati comuni siano definiti come valori numeri e sulla supposizione che valori numerici contigui indichino categorie similari.
Tali predittori aggiuntivi, chiamati rispettivamente SIM_CAR, SIM_DIS, SIM_IND, SIM_COU e SIM_REG, sono stati inizialmente aggiunti e testati singolarmente al modello di Dávid Zibriczky allo scopo di testarne l'accuratezza degli stessi. Purtroppo tali test non hanno dato risultati soddisfacenti, mostrando una diminuzione di prestazioni generali del sistema dal 2% al 7% in meno.
Successivamente è stato effettuato un test con l'aggiunta simultanea dei vari predittori aggiuntivi, ma anche questo tentativo ha dato risultati insoddisfacenti, registrando un punteggio di 12017,83, ossia un peggioramento delle prestazioni di circa il 4% generale.
Esso è facilmente spiegabile dal fatto che i vari users non interagiscono unicamente con gli items con per i quali mostrino un interesse diretto, ma anche per motivi di curiosità o per effettuare comparazioni con altri jobs.

3.4.2 Modifiche al modello di Polato et al.

Allo scopo di migliorare le performance del modello proposto da Fabio Aiolli e Mirko Polato è stato deciso di adottare l'approccio di pre-selezionare gli items che saranno analizzati per ogni users, allo scopo di rendere il modello più snello da un punto di vista computazionale ed ovviamente di migliorare la qualità della raccomandazione stessa.
Durante l'analisi dei dati è stato individuato il comportamento comune degli utenti di interagire ripetutamente con gli stessi items, allo scopo di analizzarli nuovamente od effettuare dei confronti tra di essi.

Da tale osservazione è stato deciso di implementare tale caratteristica comportamentale degli users all'interno del modello, indirizzando il sistema a raccomandare per ogni utente prevalentemente gli items con i quali esso ha avuto un interazione di tipologia positiva nel suo storico.
Tale implementazione è stata effettuata modificando la struttura dati $filtered$, ossia il dizionario che dato un user ritorna la lista degli items che saranno soggetti di raccomandazione da parte del sistema.
Come scelta d'implemenzione è stato deciso di proporre alla raccomandazione per ogni user una lista di 300 items. Tali jobs sono stati selezionati secondo il seguente approccio:

1. Ad ogni user vengono proposti per la raccomandazione tutti i jobs che sono stati protagonisti di un evento di tipologia positiva nel suo storico. Tali items sono contenuti all'interno di una lista seguendo l'ordine cronologico di interazione, partendo dal più recente per arrivare al più distante cronologicamente. Nel caso un item sia protagonista di più di un evento, esso sarà considerato unicamente per l'interazione più recente. Sono considerati, nel caso siano presenti, unicamente i primi 210 items di tale lista.
2. Per ogni periodo temporale del training set viene creata una lista degli item creati in tale settore temporale ordinata in maniera decrescente in base alla loro data di creazione. Tale lista viene usata per riempire gli elementi rimanenti delle rispettive liste utenti contenenti gli items da proporre alla raccomandazione, controllando che il job da inserire non sia già presente all'interno della lista.

Figura 3.7: pre-selezione dei dati

Tale approccio consente per ogni utente di promuovere gli items con i quali esso ha avuto interazioni in passato, e quindi con cui è più probabile che avvengano

eventi in futuro. Inoltre assicura che almeno un 30% dei jobs soggetti alla raccomandazione sia nuovo, e che quindi ci sia sempre la possibilità che ad un user venga proposto un item con cui non avesse avuto alcun tipo di interazione positiva in passato. Il problema di cold start presente in diversi users viene risolto proponendo alla raccomandazione items nuovi, e che quindi è probabile che diventino popolari nell'immediato futuro.

I test sul modello conentente l'approccio di pre-selezione dei dati è stato effettuato presso il nodo *dellsrv0* del centro di calcolo del dipartimento di Matematica dell'università di Padova [41]. Per ogni user sono stati calcolati gli score relativi a tutti gli item presenti all'interno della rispettiva lista del dizionario $filtered$. Il programma è stato eseguito su un singolo job ed è stato impiegato in un tempo totale di 22 secondi per ultimare tutte le raccomandazioni dei 2593 target users, ottenendo un punteggio finale di $15945,11$, mostrando un aumento di quest'ultimo di quasi il 97% rispetto al modello originale e migliorando in maniera evidente anche il tempo di calcolo richiesto per la roccomandazione.

A fronte di tali risultati è stato deciso di sperimentare quest'approccio anche sul modello proposto da Dávid Zibriczky [24], pre-selezionando gli items da analizzare per la raccomandazione in maniera analoga.

I test su quest'ultimo modello conentente l'approccio di pre-selezione dei dati è stato effettuato presso il nodo *dellsrv0* del centro di calcolo del dipartimento di Matematica dell'università di Padova [41]. Il programma è stato eseguito su un singolo job ed è stato impiegato in un tempo totale di 128 secondi per ultimare tutte le raccomandazioni dei 2593 target users, ottenendo un punteggio finale di $18472,42$, mostrando un aumento di quest'ultimo di oltre il 44% rispetto al modello originale.

Capitolo 4

Conclusioni

In questo progetto di tesi è stato presentato uno studio riguardante i metodi proposti da Dávid Zibriczky e da Polato et al. per la RecSys Challenge 2016, riguardanti la job recommendation e sono state testate delle modiche a quest'ultimi, allo scopo di migliorarne le performance e ridurne il carico di calcono necessario alla computazione.
É stato mostrato come un approccio di pre-selezione degli items su cui effettuare le operazioni di raccomandazione basato sulle carattestiche del dataset risulti efficace in tali tipologie di raccomandazione, mostrando nei modelli analizzati un aumento prestazionale fino al 97% e riducendo in maniera evidente la complessità di calcolo richiesta.
Una applicazione di tale approccio anche a diverse tipologie di sistemi di raccomandazione può essere scopo di lavori futuri.

Ringraziamenti

È arrivato il momento di ringraziare tutte le persone che mi hanno aiutato in questa avventura.

Voglio inanzitutto ringraziare il professore Fabio Aiolli ed il dottore Mirko Polato per avermi seguito ed aiutato durante il mio periodo di tesi.
Grazie per i loro preziosi suggerimenti, per le correzioni ed in particolare per la loro disponibilità a pazienza.

Ringrazio infinitamente mia madre, mio padre e tutta la mia famiglia per avermi costantemente supportato e sostenuto, credendo in men anche nei momenti più difficili.

Vorrei ringraziare Umberto Martinati e tutti gli amici che ho trovato durante il mio percorso di laurea magistrale per avermi aiutato ed aver reso questi anni incredibili.

Grazie a tutti i miei amici di Reggio Emilia, probabilmente le uniche persone assieme alla mia famiglia in grado di sopportarmi per più di 10 anni ormai, per incoraggiarmi costantemente e non rendere mai alcun momento della mia vita noiosa.

Bibliografia

[1] Burke, R.: Hybrid web recommender systems. In: The Adaptive Web, pp. 377–408. SpringerBerlin / Heidelberg (2007)

[2] Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversationalstrategies. In: C. Cattuto, G. Ruffo, F. Menczer (eds.) Hypertext, pp. 73–82. ACM (2009)

[3] Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM40(3), 56–58 (1997)

[4] K. Choi, D. Yoo, G. Kim, Y. Suh, A hybrid online-product recommendation-system: combining implicit rating-based collaborative filtering andsequential pattern analysis. Electronic Commerce Research andApplications, in press, doi: 10.1016/j.elerap.2012.02.004.

[5] S.K. Lee, Y.H. Cho, S.H. Kim, Collaborative filtering with ordinal scale-basedimplicit ratings for mobile music recommendations, Information Sciences180 (11) (2010) 2142–2155.

[6] E.R. Núñez-Valdéz, J.M. Cueva-Lovelle, O. Sanjuán-Martınez, V. Garcıa-Dıaz, P.Ordoñez, C.E. Montenegro-Marın, , Implicit feedback techniques onrecommender systems applied to electronic books, Computers in HumanBehavior 28 (4) (2012) 1186–1193.

[7] Anand, S.S., Mobasher, B.: Intelligent techniques for web personalization. In: Intelligent Techniques for Web Personalization, pp. 1–36. Springer (2005)

[8] Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM 35(12), 61–70 (1992)

[9] McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the net. In: KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 627–636. ACM, New York, NY, USA (2009)

[10] Michael J. Pazzani, Daniel Billsus: Content-Based Recommendation Systems, (2007)

[11] Yehuda Koren, Robert Bell, Chris Volinsky,: MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS, IEEE Computer Society (2009)

[12] RecSys Challenge 2016 `http://2016.recsyschallenge.com/`

[13] ACM Recys `https://recsys.acm.org/`

[14] Xing Informations `https://corporate.xing.com/en/about-xing/`

[15] W. Xiao and J. Wang: Job Recommendation with Hawkes Process, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[16] Tommaso Carpi, Marco Edemanti, Ervin Kamberoski, Elena Sacchi, Paolo Cremonesi, Roberto Pagano and Massimo Quadrana: Multi-Stack Ensemble for Job Recommendation, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[17] Toon De Pessemier, Kris Vanhecke and Luc Martens: A Scalable, High-performance Algorithm for Hybrid Job Recommendations, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[18] Kuan Liu, Xing Shi, Anoop Kumar, Linhong Zhu and Prem Natarajan: Temporal Learning and Sequence Modeling for a Job Recommender System, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[19] Vasily Leksin and Andrey Ostapets: Job Recommendation Based on Factorization Machine and Topic Modelling, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[20] Andrzej Pacuk, Piotr Sankowski, Adam Witkowski, Karol Wegrzycki and Piotr Wygocki: RecSys Challenge 2016: job recommendations based on preselection of offers and gradient boosting, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[21] Chenrui Zhang and Xueqi Cheng: An Ensemble Method for Job Recommender Systems, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[22] Sonu K. Mishra and Manoj Reddy: A Bottom-Up Approach to Job Recommendation System, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[23] Jose Ignacio Honrado, Oscar Huarte, Cesar Jimenez, Sebastian Ortega, Jose R. Perez-Aguera, Joaquin Perez-Iglesias, Alvaro Polo and Gabriel Rodriguez: Jobandtalent at RecSys Challenge 2016, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[24] Dávid Zibriczky: A Combination of Simple Models by Forward Predictor Selection for Job Recommendation, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[25] Mirko Polato and Fabio Aiolli: A preliminary study on a recommender system for the Job Recommendation Challenge, RecSysChallenge '16, September 15, 2016, Boston, MA, USA

[26] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. Springer, 2011.

[27] John O'Donovan & Barry Smyth: Trust in Recommender Systems, IUI'05, January 9–12, 2005, San Diego, California, USA.

[28] J. H. Friedman: Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4):367–378, 2002.

[29] C. G. ianqi Chen. Xgboost: A scalable tree boosting system. In Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.

[30] D. Agarwal, B.-C. Chen, R. Gupta, J. Hartman, Q. He, A. Iyer, S. Kolar, Y. Ma, P. Shivaswamy, A. Singh, et al. Activity ranking in linkedin feed. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1603–1612. ACM, 2014.

[31] M. Bastian, M. Hayes, W. Vaughan, S. Shah, P. Skomoroch, H. Kim, S. Uryasev, and C. Lloyd. Linkedin skills: large-scale topic extraction and inference. In Proceedings of the 8th ACM Conference on Recommender systems, pages 1–8. ACM, 2014.

[32] L. Rokach. Ensemble-based classifiers. Artificial Intelligence Review, 33(1-2):1–39, 2010.

[33] R. Polikar. Ensemble based systems in decision making. IEEE Circuits and systems magazine, 6(3):21–45, 2006.

[34] Jerome H. Friedman: GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE, The Annals of Statistics 2001, Vol. 29, No. 5, 1189–1232.

[35] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview, IEEE Circuits and systems magazine.

[36] Massimo Melucci: Information Retrieval, Collana Informatica FrancoAngeli, pages 140-158, 2013.

[37] F. Aiolli. Convex AUC optimization for top-N recommendation with implicit feedback. In ACM Recommender Systems Conference, pages 293–296, New York, USA, 2014.

[38] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In ICDM, pages 263–272, 2008.

[39] CVXOPT home page http://cvxopt.org/

[40] Gongde Guo, Hui Wang, David Bell, Yaxin Bi and Kieran Greer: KNN Model-Based Approach in Classification, R. Meersman et al. (Eds.), CoopIS/-DOA/ODBASE 2003, LNCS 2888, pp. 986–996, 2003,© Springer-Verlag Berlin Heidelberg 2003

[41] Cluster del dipartimento di Matematica http://computing.math.unipd.it/highpc/description