Recommender Systems nell'era dell'Internet of Talking Things (IoTT)

Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro

Dipartimento di Informatica
Universitá degli Studi di Bari Aldo Moro
nome.cognome@uniba.it

Abstract

Nell'Internet of Talking Things, dispositivi intelligenti sono collegati tra loro per raccogliere e scambiare dati. Nella nostra visione dell'Internet of Talking Things, oggetti come frigoriferi, caldaie, tv intelligenti saranno in grado di comunicare con gli umani per impostare le preferenze e definire un profilo utente che consenta un utilizzo personalizzato dell'oggetto. In questo documento, presentiamo un recommender system implementato come un Bot di Telegram, che può adattarsi allo scenario precedente. Il sistema è un recommender system in grado di suggerire film che sfrutta le informazioni disponibili nella LOD (Linked Open Data) cloud per generare i suggerimenti e condurre la conversazione con l'utente. Questo recommender può essere facilmente visto come una componente di una smart TV in grado di colloquiare con l'utente.

1 Introduction

La caratteristica principale di un Conversational Recommender System (CoRS) è la sua capacità di interagire con l'utente durante il processo di suggerimento [3]. Infatti, l'utente può fornire feedback per influenzare il comportamento del recommender. Non è essenziale che sia stato creato un profilo utente completo prima di iniziare a fornire i suggerimenti e che tutte le preferenze siano state specificate in anticipo dall'utente. Esiste infatti un ciclo di interazioni tra il CoRS e l'utente ripetuto fino a quando l'utente ottiene il suggerimento di un oggetto di interesse. Di conseguenza, l'obiettivo di un CORS non è solo quello di migliorare l'accuratezza delle raccomandazioni, ma anche di fornire un'interazione utente-recommender system efficace e soddisfacente.

In questo articolo, proponiamo un recommender system conversazionale per il suggerimento di film implementato come Bot Telegram (@MovieRecSysBot). Un chatbot è un tipo di robot in grado di dialogare con l'umano. I principali vantaggi dell'utilizzo di un chatbot Telegram riguardano la facilità di interazione con un'interfaccia utente semplice e ben conosciuta (la stessa che viene quotidianamente usata per altri scopi sugli smartphone), l'assenza di credenziali d'accesso specifiche, dato che ogni account è identificato in Telegram dal numero di telefono e, infine, la possibilità di fornire le proprie risposte toccando semplicemente un pulsante. Il Bot si basa sulla LOD (Linked Open Data) cloud e più specificamente sulle proprietà codificate in DBpedia [1]. Queste proprietà sono sfruttate dal Bot per ricavare le preferenze dell'utente, fornire suggerimenti e generare spiegazioni personalizzate in linguaggio naturale. Il sistema è in grado di adattare il suo comportamento al feedback degli utenti implementando una strategia di valutazione proposta in [4]. I LOD sono già stati efficacemente utilizzati in altri scenari di recommendation [9] o in altri task come il cross-lingual information retrieval [7; 8].

Nella prossima sezione spiegheremo qual è il flusso seguito dal Bot e come è gestita l'interazione con l'utente.

2 Descrizione del Chatbot

Il flusso messo in atto dal Bot è riportato in Figura 1.

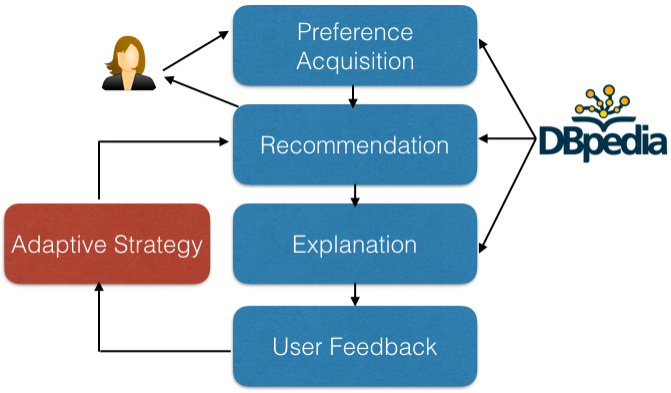

Figura 1: The Bot workflow

Nel primo step di *Preference Acquisition*, il Bot chiede all'utente di esprimere i propri interessi. Pone domande relative a entitità (ad es. film e persone) e alle loro proprietà presenti in DBpedia (ad es. genere per un film, ruolo per una persona). Naturalmente, quando l'utente inizia l'interazione, il suo profilo è vuoto, quindi il sistema di raccomandazione deve affrontare un classico problema di avviamento a freddo (cold-

[1] http://wiki.dbpedia.org/

start problem). Il sistema offre all'utente due diverse strategie per esprimere le sue preferenze: (i) valutare un insieme di *items* o *properties* proposto dal sistema stesso; (ii) digitare le entità o proprietà su cui vuole esprimere un giudizio. La prima opzione consente all'utente di esprimere le preferenze premendo dei pulsanti. La seconda opzione implementa un riconoscitore di entità basato sulla distanza di Levenshtein [**?**], attivato per mezzo di una funzione *Did you mean?*, in modo che, se l'utente commette errori di battitura, il sistema è comunque in grado di riconoscere la giusta entità o proprietà. Il secondo passo è la *Raccomandazione*. Il Bot implementa attualmente il PageRank con Priorità [2], noto anche come PageRank Personalizzato. Diversamente dal PageRank, che assegna a ciascun nodo del grafo su cui lavora una probabilità a priori distribuita uniformemente ($1/N$, dove N è il numero di nodi), il PageRank personalizzato assegna pesi diversi ai differenti nodi, al fine di ottenere un bias verso alcuni di essi (in questo caso, i nodi che rappresentano le preferenze di un utente specifico). L'algoritmo è stato efficacemente utilizzato in altri sistemi di raccomandazione [1]. Il modello adottato rappresenta le preferenze e le proprietà DBpedia in un singolo grafo. L'algoritmo viene eseguito per ciascun utente e l'assegnazione delle probabilità ai nodi è ispirata al modello proposto in [5]. L'algoritmo genera una classifica dei film potenzialmente interessanti per un determinato utente. Il Bot implementa anche un modulo di *Explanation*. Tintarev e Masthoff [10] sottolineano che la spiegazione di un recommender è generalmente volta a giustificare il suggerimento, ma potrebbe anche essere volta a fornire una descrizione dettagliata che permetta all'utente di comprendere le qualità dell'oggetto suggerito. Il Bot implementato è in grado di fornire entrambi i tipi di spiegazione. I dettagli su un oggetto possono essere ottenuti toccando il pulsante *Details* che mostra le informazioni estratte da IMDB su un determinato film. Il pulsante *Why?* implementa un algoritmo di spiegazione ispirato a [6]. L'dea è di utilizzare le connessioni tra le preferenze dell'utente e gli articoli consigliati per spiegare il motivo per cui un determinato articolo è stato raccomandato. Queste connessioni sono estratte dal grafo basato su LOD.

Un esempio di spiegazione in linguaggio naturale fornito dal sistema è: Ti suggerisco *Duplex* perchè ti piacciono i film in cui: l'attore è *Ben Stiller* come in *Meet the Fockers*, il genere è *Comedy* come in *American Reunion*.. In questo caso il sistema ha utilizzato le connessioni, estratte da DBpedia, tra il film consigliato *Duplex* e le preferenze dell'utente (che sono *Meet the Fockers*, *American Reunion* e *Ben Stiller*). Toccando il pulsante *Profile* l'utente può anche esplorare il suo profilo e aggiornare le sue preferenze.

Infine, il Bot consente all'utente di dare un feedback su un determinato suggerimento. Questo modulo implementa una *Adaptive Strategy* proposta in [4]. Toccando il pulsante *I like, but...* l'utente attiva la strategia *Refine*. Il *Refine* è una strategia di feedback che consente all'utente di esprimere una preferenza su di un film, ma anche di valutare separatamente le caratteristiche (ad esempio, *Mi piace questo film, ma dura troppo* o *Mi piace il film, ma non un attore*). Pertanto, l'utente può esprimere una preferenza anche su una singola caratteristica di un film. Il nodo associato alla caratteristica che l'utente non gradisce (ad esempio, Quentin Tarantino) verrà rimosso dal grafo utilizzato dal PageRank e il processo di raccomandazione ricomincerà dal nuovo grafo aggiornato. Infine, il Bot consente all'utente di esplorare e aggiornare il suo profilo. Attraverso queste funzioni l'utente può visualizzare le preferenze memorizzate nel suo profilo e modificarle. Alla fine, quando il profilo è stato aggiornato, il sistema eseguirà nuovamente il PageRank e genererà una nuova serie di raccomandazioni.

Riferimenti bibliografici

[1] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro. Content-based recommender systems+ dbpedia knowledge= semantics-aware recommender systems. In *Semantic Web Evaluation Challenge*, pages 163–169. Springer, 2014.

[2] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, 15(4):784–796, 2003.

[3] T. Mahmood and F. Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM, 2009.

[4] L. Mcginty and B. Smyth. Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems. *International Journal of Electronic Commerce*, 11(2):35–57, 2006.

[5] C. Musto, P. Lops, P. Basile, M. de Gemmis, and G. Semeraro. Semantics-aware graph-based recommender systems exploiting linked open data. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 229–237. ACM, 2016.

[6] C. Musto, F. Narducci, P. Lops, M. De Gemmis, and G. Semeraro. Explod: A framework for explaining recommendations based on the linked open data cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 151–154. ACM, 2016.

[7] F. Narducci, M. Palmonari, and G. Semeraro. Cross-language semantic retrieval and linking of e-gov services. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, pages 130–145, 2013.

[8] F. Narducci, M. Palmonari, and G. Semeraro. Cross-lingual link discovery with TR-ESA. *Inf. Sci.*, 394:68–87, 2017.

[9] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Trans. Intell. Syst. Technol.*, 8(1):9:1–9:34, Sept. 2016.

[10] N. Tintarev and J. Masthoff. Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):399–439, 2012.