Recommender Systems in the Internet of Talking Things (IoTT)

Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro
Department of Computer Science
University of Bari Aldo Moro
Italy
name.surname@uniba.it

ABSTRACT

In the Internet of Things, smart devices are connected to collect and to exchange data. In our vision, in the Internet of Talking Things, objects such as intelligent fridges will be able to communicate with humans to set up preferences and profiling options which allow a personalized usage of the object. In this paper, we present a recommender system implemented as a Telegram Bot, that can fit with the previous scenario. The system is a movie recommender which exploits the information available in the Linked Open Data (LOD) cloud for generating the recommendations and leading the conversation with the user. It can be easily seen as an intelligent component of a connected TV.

ACM Reference format:
Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro. Copyrights held by the authors. Recommender Systems in the Internet of Talking Things (IoTT) . *RecSys 2017 Posters, Como, Italy, August 27-31 (RecSys 2017 Poster Proceedings),* 2 pages.

1 BACKGROUND AND MOTIVATIONS

The main distinctive feature of a conversational recommender system (CORS) compared to a classical one is its capability of interacting with the user during the recommendation process [3]. The user provides feedback and tries to get better recommendations. It is not essential that a complete user profile has been built before beginning the recommendation process and all preferences have been specified upfront by the user. There is a cycle of interactions between the CORS and the user repeated until the user reaches an item of interest. Accordingly, the goal of a CORS is not only to improve the accuracy of the recommendations, but also to provide an effective user-recommender interaction.

In this paper we propose a conversational movie recommender system implemented as Telegram Bot (@MovieRecSysBot). Chatbots are a kind of bots which emulate user conversations. The main advantages of using a Telegram bot are that it facilitates the interaction of the user by a clean and well-known user interface (the same that people daily use for other purposes on their smartphones), it does not require credentials since each account is identified in Telegram by the phone number, and lastly the user can answer by tapping a button. The Bot is based on the Linked Open Data (LOD) cloud, and more specifically on the properties encoded in DBpedia[1]. These properties are exploited by the Bot for eliciting user preferences, for providing recommendations as well as for generating personalized explanations in natural language. The system is capable of adapting its behavior to the user feedback by implementing a critiquing strategy proposed in [4]. LOD have already been effectively used in other recommendation scenarios [9]as well as for other tasks such as cross-lingual information retrieval [7, 8].

In the next Section how the Telegram Bot works and its interaction with the user are described.

2 DESCRIPTION OF THE CHATBOT

The workflow carried out by the Bot is depicted in Figure 1. In the

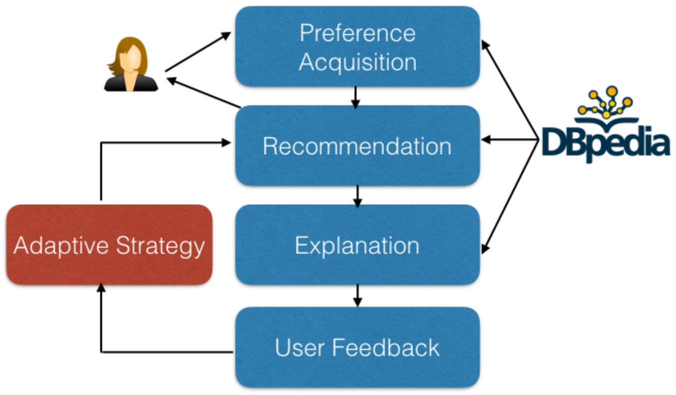


Figure 1: The Bot workflow

first step, *Preference Acquisition*, the Bot asks the user to express her interests. It asks questions related to entities (e.g, movies and persons) and their properties in DBpedia (e.g, genre, role). When the user starts the interaction, her profile is empty, so the recommender system needs to address a classical cold-start problem. The system offers the user two different strategies to express her preferences: (i) rating a set of *items* or *properties* proposed by the system; (ii) typing the entities or properties she is willing to rate. The first option allows the user to express the preferences by tapping buttons. The second option implements an entity recognizer based on the Levenshtein distance [11] by means of a *Did you mean* function (Figure 3 (a)), so that, if the user makes typos, the system is anyway able to recognize the right entity or property. The second step is the *Recommendation*. The Bot currently implements PageRank with Priors [2], also known as Personalized PageRank. Differently from PageRank, which assigns an evenly distributed prior probability to each node ($1/N$, where N is the number of nodes), the Personalized PageRank adopts a non-uniform personalization vector by assigning different weights to different nodes to get a bias towards some nodes (in this case, the preferences of a specific user). The algorithm has been effectively used in other recommendation environments

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

[1]. Figure 2 shows how the user preferences and the DBpedia properties are represented in a single graph. The algorithm is run for each user and the assignment of the probabilities to the nodes has been inspired by the model proposed in [5]. The algorithm generates a ranking of the items potentially interesting for a given user. The Bot also implements an *Explanation* module. Tintarev and Masthoff [10] point out that explaining a recommendation is generally intended as justifying the suggestion, but it might be also intended as providing a detailed description that allows the user to understand the quality of the recommended item. The Bot is able to provide these types of explanation. Details about an item can be obtained by tapping on a *Details* button (Figure 3 (b)) which shows information extracted from IMDB on a given movie. The *Why?* button implements an explanation algorithm inspired by [6]. The idea is to use the connections in the LOD-based graph between the user preferences and the recommended items for explaining why a given item has been recommended.

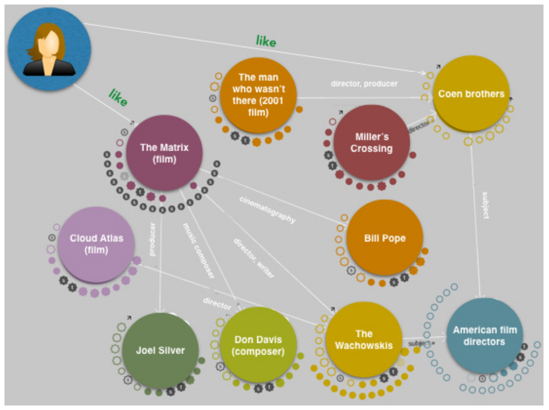
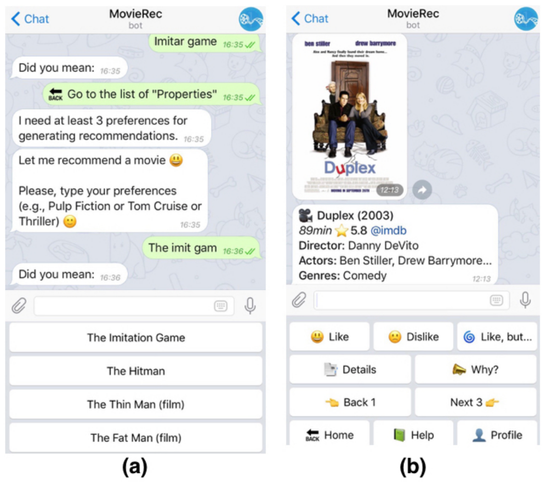


Figure 2: Example graph which connects users, items and entities in DBpedia

An example of natural-language explanation provided by the system is: "I suggest you *Duplex* because you like movies where: the actor is *Ben Stiller* as in *Meet the Fockers*, the genre is *Comedy* as in *American Reunion*. Moreover, I recommend *Duplex* because the actor is *Ben Stiller* and you like him". In this case the system used the connections, extracted from DBpedia, between the recommended movie *Duplex* and the user preferences (consisting of *Meet the Fockers*, *American Reunion*, and *Ben Stiller*). By tapping on the *Profile* button (Figure 3 (b)) the user can also explore her profile, and update her preferences.

Finally, the Bot allows the user to give feedback on a given recommendation. This module implements an *Adaptive Strategy* proposed in [4]. By tapping on the *Like, but...* button (Figure 3 (b)) the user activates the *Refine* strategy. The *Refine* is a critiquing strategy which allows the user to express a preference on a movie, but to separately evaluate its characteristics (e.g,. *I like this movie but it runs too* or *I like the movie, but not an actor*). Therefore, the user can express a preference on a single characteristic of a movie. The node associated to the characteristic the user does not like (e.g., Quentin Tarantino) will be removed from the graph used by the PageRank and the recommendation process starts again on the new updated graph. Finally, the Bot allows the user to explore and to update her profile. Through these functions the user can view the preferences stored in her profile and change them. At the end, when the profile has been updated, the system will run again the PageRank and generate a new set of recommendations.


Imitar game 16:35

Did you mean: 16:35

Go to the list of "Properties" 16:35

I need at least 3 preferences for generating recommendations. 16:35

Let me recommend a movie 😃

Please, type your preferences (e.g., Pulp Fiction or Tom Cruise or Thriller) 🙂 16:35

The imit gam 16:36

Did you mean: 16:36

Duplex (2003)
89min ⭐ 5.8 @imdb
Director: Danny DeVito
Actors: Ben Stiller, Drew Barrymore...
Genres: Comedy 12:13

Figure 3: A screenshot of the Bot during the training phase in typing mode (a), and the recommendation phase (b)

REFERENCES

[1] Pierpaolo Basile, Cataldo Musto, Marco de Gemmis, Pasquale Lops, Fedelucio Narducci, and Giovanni Semeraro. 2014. Content-based recommender systems+ DBpedia knowledge= semantics-aware recommender systems. In *Semantic Web Evaluation Challenge*. Springer, 163–169.
[2] Taher H Haveliwala. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering* 15, 4 (2003), 784–796.
[3] Tariq Mahmood and Francesco Ricci. 2009. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*. ACM, 73–82.
[4] Lorraine Mcginty and Barry Smyth. 2006. Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems. *International Journal of Electronic Commerce* 11, 2 (2006), 35–57.
[5] Cataldo Musto, Pasquale Lops, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2016. Semantics-aware graph-based recommender systems exploiting linked open data. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. ACM, 229–237.
[6] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2016. ExpLOD: A Framework for Explaining Recommendations based on the Linked Open Data Cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 151–154.
[7] Fedelucio Narducci, Matteo Palmonari, and Giovanni Semeraro. 2013. 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*. 130–145.
[8] Fedelucio Narducci, Matteo Palmonari, and Giovanni Semeraro. 2017. Cross-lingual link discovery with TR-ESA. *Inf. Sci.* 394 (2017), 68–87.
[9] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 2016. SPrank: Semantic Path-Based Ranking for Top-N Recommendations Using Linked Open Data. *ACM Trans. Intell. Syst. Technol.* 8, 1, Article 9 (Sept. 2016), 34 pages.
[10] Nava Tintarev and Judith Masthoff. 2012. Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction* 22, 4-5 (2012), 399–439.
[11] Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.