A Framework for building Chat-based Recommender Systems

Fedelucio Narducci
Department of Computer Science
University of Bari Aldo Moro, Italy
fedelucio.narducci@uniba.it

Pierpaolo Basile
Department of Computer Science
University of Bari Aldo Moro, Italy
pierpaolo.basile@uniba.it

Andrea Iovine
Department of Computer Science
University of Bari Aldo Moro, Italy
andrea.iovine@uniba.it

Marco de Gemmis
Department of Computer Science
University of Bari Aldo Moro, Italy
marco.degemmis@uniba.it

Pasquale Lops
Department of Computer Science
University of Bari Aldo Moro, Italy
pasquale.lops@uniba.it

Giovanni Semeraro
Department of Computer Science
University of Bari Aldo Moro, Italy
giovanni.semeraro@uniba.it

ABSTRACT

Chat-based recommender systems are getting more and more attention in recent time given their natural interaction with the user. Indeed, chat-based recommender systems implement a paradigm where users define their preferences and discover items that best fit their needs through a dialog. A chat-based recommender system can be easily integrated in platforms such as social networks, e-commerce websites, bank websites. Therefore, the preferences can be directly provided by the users during the dialog or can be automatically extracted from their activities on the same platform that hosts the chatbot [3].

In this demo, we present a framework for building chat-based recommender systems. The framework, based on a content-based recommendation algorithm, is independent from the domain.

1 INTRODUCTION

Chat-based recommender systems have the capability of interacting with the user during the recommendation process [4]. Instead of asking users to provide all the requirements in one step, they guide the users through an interactive dialog [2]. This kind of interaction is particularly useful in domains such as music, TV [6] were the user interacts with the system while doing other activities.

Users can provide functional requirements or technical constraints used by the recommender for finding the items that best fit their needs. Accordingly, the acquisition of preferences is an incremental process that might not be necessarily finalized in a single step. A chat-based recommender system can provide several interaction modes and can offer explanation mechanisms [5]. Hence, the goal of these systems is not only to improve the accuracy of the recommendations, but also to provide an effective user-recommender interaction.

In this demo we will show a framework, not dependent on the domain, for generating conversational recommender systems. Our framework implements most of the capabilities that a recommender should offer, such as preference acquisition, profile exploration, critiquing strategies, and explanation.

By exploiting our framework, we successfully implemented instances of a conversational recommender system, as Telegram chatbot, in three different domains: movies, music, and books[1].

2 THE ARCHITECTURE

The main goal of this framework is to make easy the building of a new chat-based recommender system. Therefore, the components have been generalized making them independent from a specific domain. When the user desire to build a new chat-based recommender system for a new domain she should update the configuration file, and provide the list of entities and properties in the Wikidata[2] format. This last requirement depends on the implementation of the Entity Recognizer. In the following we provide a brief description of each component.

Dialog Manager. This is the core component of the framework whose responsibility is to supervise the whole recommendation process. The *Dialog Manager* (DM) is the component that keeps track of the dialog state. DM receives the user message, invokes the components needed for answering to the user request, and returns the message to be shown to the user.

Intent Recognizer. This component has the goal of defining the intent of the user formulated by natural language. The Intent Recognizer (IR) is based on DialogFlow[3] developed by Google. Our framework uses the DialogFlow APIs for sending the user message and to receive the intent recognized. DialogFlow requires a set of example sentences for each intent.

Sentiment Analyzer. The Sentiment Analyzer (SA) is based on the Sentiment Tagger of Stanford CoreNLP[4]. The Sentiment Tagger takes as input the user sentence and returns the sentiment tags identified. Afterwards, SA assigns

[1] On Telegram, search for: @MovieRecSysBot, @MusicRecSys, @BookRecSys.
[2] https://www.wikidata.org/
[3] https://dialogflow.com/
[4] https://stanfordnlp.github.io/CoreNLP/

the sentiment tags to the right entity identified into the sentence. For example, given the sentence *I like The Matrix, but I hate Keanu Reeves*, the Sentiment Tagger identifies a positive sentiment (i.e. like) and a negative one (i.e. hate). SA associates the positive sentiment to the entity *The Matrix* and the negative sentiment to the entity *Keanu Reeves*.

Entity Recognizer. The aim of the Entity Recognizer (ER) module is to find relevant entities mentioned in the user sentence and then to link them to the correct concept in the Knowledge Base (KB). The KB chosen for building our framework is Wikidata since it is a free and open-knowledge base and acts as a hub of several structured data coming from Wikimedia sister projects[5]. Moreover, Wikidata covers several domains and this is a key feature for developing a domain-independent framework. The ER module can be adapted to exploit a custom KB for particular domains not covered by Wikidata. The only requirements are: 1) the knowledge must be modeled through triples; 2) each concept must have one or more alias.

Recommendation Services This component collects the services strictly related to the recommendation process. The recommendation algorithm implemented is the PageRank with Priors [1], also known as Personalized PageRank. Another recommendation service offered by the framework is the *explanation* feature. The framework implements an explanation algorithm inspired by [5]. An example of natural-language explanation provided by the system is: "I suggest you the movie *Duplex* because you like movies where: the actor is *Ben Stiller* as in *Meet the Fockers*, the genre is *Comedy* as in *American Reunion*.". The last service implemented is the *critiquing*. This service allows to acquire a critique on a recommended item (e.g. *I like the movie Titanic, but I don't like the actor Bill Paxton*) and this feedback will be used in the next recommendation cycle by properly setting the weights of the nodes in the PageRank graph. As before stated, all these components are independent from the domain. The only requirement is that the entities have to be available in Wikidata.

3 DEMONSTRATION SUMMARY

During the demo we will show three different implementations of our chat-based recommender system in three different domains. More specifically, we will show a music recommender system, a movie recommender system, and a book recommender system. The peculiarity of these systems is that they have been implemented through the same framework with minimal code variations. Furthermore, we will show three different interaction modes: a button-based interaction, a natural-language based interaction, and a hybrid interaction (natural language and buttons combined). In Figure 1 two screenshots of the movie recommender system with the hybrid interaction, and the music recommender system with the NLP interaction are reported. The goal of the demonstration will be to show how to build a new chat-based recommender through our framework is an easy task.

[5]Wikipedia, Wikivoyage, Wikisource, and others

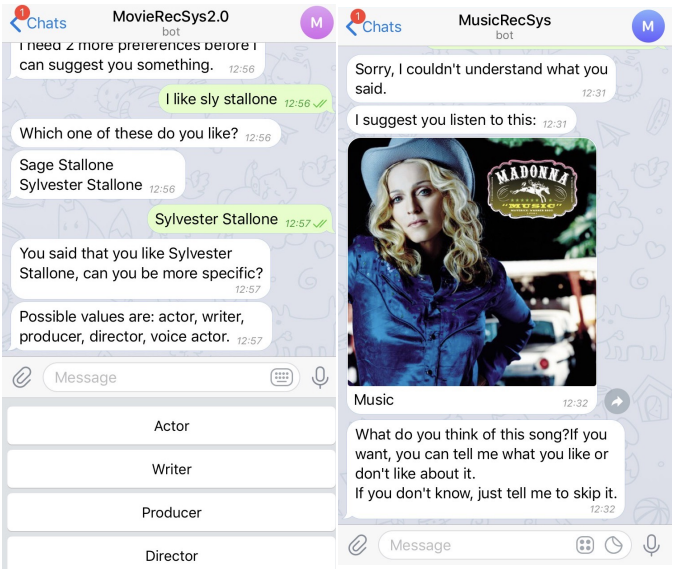

Figure 1: Two screenshots of the movie recommender system with the hybrid interaction (on the left side) and the music recommender with the NLP interaction (on the right side).

ACKNOWLEDGMENT

This work has been funded by the projects UNIFIED WEALTH MANAGEMENT PLATFORM - OBJECTWAY SpA - Via Giovanni Da Procida nr. 24, 20149 MILANO - c.f., P. IVA 07114250967, and PON01 00850 ASK-Health (Advanced system for the interpretations and sharing of knowledge in health care).

REFERENCES

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

[2] Michael Jugovac and Dietmar Jannach. 2017. Interacting with Recommenders: Overview and Research Directions. *ACM Trans. Interact. Intell. Syst.* 7, 3, Article 10 (Sept. 2017), 46 pages. DOI: https://doi.org/10.1145/3001837

[3] P. Lops, M. De Gemmis, G. Semeraro, F. Narducci, and C. Musto. 2011. Leveraging the LinkedIn social network data for extracting content-based user profiles. *RecSys'11 - Proc. of the 5th ACM Conf. on Recommender Systems* (2011), 293–296. DOI:https://doi.org/10.1145/2043932.2043986

[4] 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.

[5] 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.

[6] C. Musto, F. Narducci, P. Lops, G. Semeraro, M. De Gemmis, M. Barbieri, J. Korst, V. Pronk, and R. Clout. 2012. Enhanced semantic TV-show representation for personalized electronic program guides. In *Int. Conf. on User Modeling, Adaptation, and Personalization*, Vol. 7379 LNCS. 188–199. DOI: https://doi.org/10.1007/978-3-642-31454-4_16