A domain-independent Framework for building Conversational 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

Conversational Recommender Systems (CoRSs) implement a paradigm where users can interact with the system for defining their preferences and discovering items that best fit their needs. A CoRS can be straightforwardly implemented as a chatbot. Chatbots are becoming more and more popular for several applications like customer care, health care, medical diagnoses. In the most complex form, the implementation of a chatbot is a challenging task since it requires knowledge about natural language processing, human-computer interaction, and so on. In this paper, we propose a general framework for making easy the generation of conversational recommender systems. The framework, based on a content-based recommendation algorithm, is independent from the domain. Indeed, it allows to build a conversational recommender system with different interaction modes (natural language, buttons, hybrid) for any domain. The framework has been evaluated on two state-of-the-art datasets with the aim of identifying the components that mainly influence the final recommendation accuracy.

ACM Reference Format:
Fedelucio Narducci, Pierpaolo Basile, Andrea Iovine, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2019. A domain-independent Framework for building Conversational Recommender Systems. In *Proceedings of Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop 2018 (co-located with RecSys 2018)*. ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

Conversational Recommender Systems (CoRSs) are characterized by the capability of interacting with the user during the recommendation process [11]. Instead of asking users to provide all requirements in one step, CoRSs guide the users through an interactive dialog [8].

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. CoRSs can provide several interaction modes and can offer explanation mechanisms. 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 paper we propose 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 capability. Furthermore, it offers three different interaction modes: natural language, buttons, and a combination of the two previous ones.

The most complex interaction mode is certainly the one based on natural language. Indeed, a conversational recommender based on natural language needs at least four components: an intent recognizer, an entity recognizer, a sentiment analyzer, and a recommendation algorithm. The next sections explain the tasks that each component carries out. Usually, the first three components are also used for purposes different from the recommendation task. For example, an entity recognizer is useful in several applications where the identification of named entities in a given text is needed, such as news classification, search algorithms, customer support. In this work we first generalized, combined, and integrated the aforementioned components in order to make easy the development of a new conversational recommender system, then we investigated the impact of each component on the recommendation process.

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

The rest of the paper is organized as follows: the relevant literature is analyzed in Section 2; Section 3 describes the architecture of our framework, and finally, the experimental evaluation and the discussion of results are reported in Section 4. Section 5 draws the conclusion and the future work.

Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop 2018 (co-located with RecSys 2018), October 7, 2018, Vancouver, Canada.

[1] On Telegram, search for: @MovieRecSysBot, @MusicRecSys, @BookRecSys.

2 RELATED WORK

Conversational Recommender Systems fall in the area of Goal-Oriented Dialog Systems. A Goal-Oriented Dialog System, also known as Chatbot, is designed for helping users to achieve a given goal (e.g. to book a restaurant). These systems are generally closed-domain, thus can be exploited in scenarios like recommendation [13, 14], retrieval [15] and can be integrated in larger systems, such as Amazon Alexa[2], to give the impression of a general coverage [7]. In the literature there is a distinction between modular and end-to-end dialog systems. The former are composed of at least two components: a dialog-state tracking component and a response generator; the latter do not rely on explicit internal states and learn a dialog model based on past conversations [2]. In this work we define a modular framework for generating goal-oriented dialog systems for the recommendation task in any domain. These systems are very used on social networks since they can acquire information on the user by analyzing their activities on the platform [10].

There are several work in the literature that tried to improve various aspects of the conversational recommendation process [8]. In [4] the authors demonstrated that a speech-based interaction model produces higher user satisfaction and needs less interaction cycles. In [16] the authors propose a chat-based group recommender system that iteratively allows users to express and revise their preferences during the decision making process. In [6] the authors present an interactive visualization framework that combines recommendation techniques with visualization ones to support human-recommender interaction. Several researchers developed integrated frameworks for conversational recommender systems [3, 19] by combining conversational functionalities with adaptive and recovery functions. To the best of our knowledge, the framework proposed in this paper is the first solution that allows to generate a CoRS for any domain by offering a complete suite of functions for a multi-modal interaction.

A commercial solution is proposed by Microsoft with the Bot Framework[3] that provides tools for building, connecting, testing, and deploying intelligent bots. Even though the framework is not designed for generating conversational recommender systems, it allows to integrate services from Microsoft Azure like the recommendation engine[4]. However, the effort for integrating and connecting the different components is borne by the user of the framework. Furthermore, the framework does not offer features like critiquing strategies or explanation functions. Also, the different interaction modes (e.g., buttons) have to be implemented by the user. Last but not least, in this work we studied the accuracy of each component integrated in our framework. Conversely, other solutions can be used only as black box models.

[2]https://developer.amazon.com/alexa
[3]https://dev.botframework.com/
[4]https://azure.microsoft.com/en-us/resources/videos/building-a-recommender-system-in-azure-ml-studio/

3 THE FRAMEWORK ARCHITECTURE

The architecture of our framework is depicted in Figure 1. The main goal of this framework is to make easy the building of a new CoRS. Therefore, the components have been generalized making them independent from a specific domain. When the user wants to build e new CoRS for a new domain she should update the configuration file, and provide the list of entities and properties in the Wikidata[5] format. This requirement will be better explained in the next sections and depends on the Entity Recognizer. The interaction follows a *slot-filling* model, where a set of features need to be filled in order to accomplish the user goal. As an example, the recommendation step requires that the user preferences have been filled. In the following we analyze each component in detail.

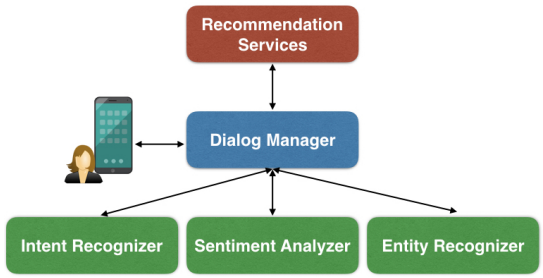

Figure 1: The Framework Architecture

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. When all the information for fulfilling the user request are available, the *Dialog Manager* returns the message to the client. DM is completely independent from the client, indeed it receives a text message and returns a json message as a response. In this way the client can be any messaging platform like Facebook Messenger, Telegram, Web apps, and so on.

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[6] 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. There are four main intents to be recognized:

- *preference*: the user is providing a preference. The preference can be expressed on a new item, or on a recommended item. In the latter case, the preference is also considered as a critique (e.g., I like this movie, but I don't its director).

[5]https://www.wikidata.org/
[6]https://dialogflow.com/

- *recommendation*: the user asks to receive a recommendation. This intent is the condition for other sub-intents such as *explanation* where the user asks the motivation for a given recommendation, *critiquing* where the user expresses a critique on one or more features of the recommended item, *more info* where the user asks more details on the recommended item (e.g., the plot, the trailer).

- *show profile*: the user asks to visualize (and modify) her list of preferences.

- *help*: the user asks for help from the system to complete a given task.

Each intent can be composed of a set of sub-intents that activate specific functions. For example the intent *profile* has *delete preference, update preference, reset profile* as sub intents. The motivation behind this hierarchical organization is that, generally, the sub-intent can be activated only when the parent intent is activated too. The activation of a parent intent is managed by the Dialog Manager.

Sentiment Analyzer The Sentiment Analyzer (SA) is based on the Sentiment Tagger of Stanford CoreNLP[7]. The Sentiment Tagger takes as input the user sentence and returns the sentiment tags identified. Afterwards, SA assigns 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*. The association sentiment-entity is performed by computing the distance between the sentiment tag and the entity into the sentence. The distance is in terms of number of tokens that separate the sentiment tag from the entity. Given the aforementioned example, the distance between LIKE and *The Matrix* is zero, as well as the distance between HATE and *Keanu Reeves*. The sentiment tag identified by CoreNLP is thus associated to the closest entity in the sentence. Furthermore, SA implements a *Property Type Recognizer* that is able to identify property-mentions in the sentence. Given the sentence *I like The Matrix, but I hate the director*, SA identifies the property DIRECTOR and assigns the negative sentiment to it. Afterwards, SA will retrieve the entity associated to that property, *Larry e Andy Wachowski* in the given example. The list of properties the framework is able to recognize is provided in the configuration file.

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[8]. Moreover, Wikidata covers several domains and this is a key feature for developing a domain-independent framework. We choose to develop a custom entity recognizer for two reasons: 1) existing entity recognizer/entity linking algorithms are hard to customize to a specific domain; 2) entity recognizers included in existing dialog manager toolkits generally require annotated data to build a new model for a specific domain, while we implemented a knowledge based approach that does not need any annotated data. The task is challenging since more than one surface form *Spielberg*, *Steven Spielberg* can refer to *Steven_Spielberg:director*, and the same surface form *Spielberg* can refer to more than one concept *Steven_Spielberg:director* and *Sasha_Spielberg:actor* in case of ambiguous entities. Moreover, we need to limit the entities' type according to the domain of the CoRS and the list of concepts and properties provided during the configuration of the framework.

In order to recognize the entities in the user request, we build a search engine based on a classical Vector Space Model in which for each entity we store all the possible alias provided by Wikidata. For example, for the concept *Q8877* (*Steven Spielberg*), we store the alias *Steven Allan Spielberg*, *Spielberg* and *Steven Spielberg*. The index is exploited for retrieving a list of candidate concepts according to the input text. In particular, given a text T as input, the ER module performs a chunking operation in order to identify nominal chunks by using the Apache OpenNLP library. Each nominal chunk is sent as a query to the search engine in order to retrieve a list of candidate concepts. The output of this first recognition step is a list of candidate concepts assigned to each nominal chunk. The list is sorted according to the score assigned by the search engine. We use Apache Lucene as library for implementing our search engine.

The last step consists in selecting the correct concept for each chunk. The idea is to choose the concept that is more similar to the other concepts occurring in the text following the hypothesis of one topic for discourse. The motivation behind this approach is that the user tends to cite in the same text entities that are in some way related. The score $s(c_j)$ assigned to each candidate concept c_i for the chunk e_j is computing according to the Equation 1, where E is the set of the other nominal chunks in the text. The score $s(c_j)$ is the sum for each chunk e_k in E of the maximum similarity score between all the candidate concepts c_i of e_k and c_j.

$$s(c_j) = \sum_{e_k \in E} argmax_{c_i \in C_{e_k}} sim(c_j, c_i) \quad (1)$$

In order to compute the score $s(c_j)$, we need to define a similarity function between concepts. In our approach we rely on graph embeddings that have recently gained considerable attention [17]. These approaches allow to represent entities and relations through an embedding, which is a continuous vector representation able to capture the semantics of an entity or a relation. We investigate holographic embeddings (HolE) [18], which exploit the circular correlation of entity embeddings to create compositional representations of binary relational data coming from Wikidata. By exploiting HolE, each entity is represented by an embedding and we

[7] https://stanfordnlp.github.io/CoreNLP/
[8] Wikipedia, Wikivoyage, Wikisource, and others

can compute the similarity between two entities by the cosine similarity between the corresponding embeddings. The exploited graph is built by querying Wikidata. Finally, the score $s(c_j)$ is averaged with the score returned by the search engine and the list of candidate concepts is re-sorted in descending order. The first element of each candidate list is the concept assigned to each nominal chunk in the text. 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 [5], also known as Personalized PageRank. It works on a graph where the nodes are the entities the recommender deals with (e.g., for the movie domain, actors, movies, directors, genre, etc.). These entities are extracted from Wikidata, and their connections (edges in the graph) are extracted from DBpedia[9]. Hence, for example the movie *The Matrix* is connected to the director node *Larry e Andy Wachowski*, to the genre node *science fiction*. The algorithm has been effectively used in other recommendation environments [1]. Another recommendation service offered by the framework is the explanation feature. The framework implements an explanation algorithm inspired by [12]. The idea is to use the connections between the user preferences and the recommended items for explaining why a given item has been recommended. 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*.". In this case the system used the connections between the recommended movie *Duplex* and the user preferences (*Meet the Fockers*, *American Reunion*, and *Ben Stiller*). 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.

4 EXPERIMENTAL EVALUATION

The goal of the experimental evaluation was to define the accuracy of each component involved in our framework. For this experiment we used the bAbI dataset developed by Facebook Research[10]. The dataset collects a list of utterances, and each utterance contains a list of preferences, followed by the recommendation request, and the recommended item. An example of utterance is:

"Beauty and the Beast, Aladdin, Schindler's List, The Shawshank Redemption, and The Silence of the Lambs are movies I loved. Would you recommend something I might like? Ghost".

We used this dataset for testing the impact of the Entity Recognizer, the Sentiment Analyzer, and the Intent Recognizer on the recommendation process. Since the dataset has the goal of evaluating end-to-end dialog systems, the dataset is split in three sub sets: training, test, and dev set. Given the goal of our experiment, we excluded the training set due to its huge dimension, and we used the test, and dev sets composed respectively of 6,667 and 6,733 examples (each example contains the preference elicitation, the recommendation request, and the recommendation). We defined four different configurations of the framework:

- Upper bound (UB): this configuration tests the accuracy of our recommendation algorithm. The preferences and the recommendation requests are filled programmatically, so, except for the recommendation algorithm, the other components of the frameworks do not work.

- Intent Recognizer Test (IR): in this configuration the only component that works is the Intent Recognizer. The component detects the intention of the user of expressing a preference and receiving the recommendation. If both intents are correctly recognized, the recommendation is performed by setting the entities and their sentiments programmatically.

- Entity Recognizer Test (ER): in this configuration the only component that works is the Entity Recognizer. It detects the entities on which the user expressed a preference. The sentiment on the entities correctly identified are set programmatically.

- Sentiment Recognizer Test (SR): in this configuration the only component that works is the Sentiment Recognizer. The component detects the sentiments in the sentence. The entities on which the sentiments is expressed are set programmatically.

These configurations allow to test one component at time by excluding the influence of the other ones in the process. For each configuration we computed the *HitRate@n* as the ratio of the hits in the recommendation list with *n= 5, 10, 20*. In Table 1, the first row reports the upper bound in terms of *HitRate@n*: this is the best result that our recommendation algorithm can achieve on this dataset in the ideal situation where the other components work with a 100% of accuracy. The other rows report the loss in terms of *HitRate@n* of each configuration compared to the upper bound. Due to the space limit, we report only the results on the test set of bAbI, since the dev set follows the same trend.

It is worth to note that it is not surprising that the upper bound is very low. Indeed, in the bAbI dataset even though the top predictions contain items that might be good for the user, if the actual single true label is not recommended, the recommendation fails.

The analysis of the loss shows that the entity recognizer (ER) is the component with the highest negative impact on the recommendation process. Even though the entity recognizer was able to recognize $\sim$ 85% of the entities on the bAbI dataset, the error ($\sim$ 15% entities not recognized) determined a strong loss in terms of accuracy. The second component in

[9] http://wiki.dbpedia.org/
[10] https://research.fb.com/downloads/babi/

terms of negative impact on the recommendation accuracy is the Intent Recognizer (IR). In this case, the component was able to correctly recognize $\sim 77\%$ of the intents (both preference elicitation, and recommendation request). The component with the lowest impact on the recommendation accuracy is the Sentiment Recognizer (SR), in this case the component was able to correctly recognize $\sim 83\%$ of the sentiments. By analyzing these results, it emerges that the Entity Recognizer plays a crucial role in the recommendation process of a conversational recommender. This aspect is particularly crucial when a rich user profile is not available, and the recommender has to work on a small number of preferences as in the bAbI dataset. The second component that negatively influences the recommendation is the IR. Also in this case, if the CoRS is not able to correctly identify the user intention, it will not be able to activate the correct process to satisfy the request. Finally, the SR is the component with the lowest negative impact. However, in the bAbI dataset all the sentences have a positive sentiment, and this facilitates the work of the component. We also tested our framework on a dataset recently released by Grouplens [9]. This dataset has been collected with the aim of analyzing the recommendation requests of real users to a conversational recommender. In this dataset we could only analyze the accuracy of the entity recognizer and the intent recognizer for the request-recommendation intent. The dataset is composed of 694 sentences. The framework correctly recognized the 7.64% of request-recommendation intents, and the 64.39% of the entities. The very low performance of the IR is due to the fact that in this dataset the user requests the recommendation in a very varied and synthetic form (e.g., *"action movies", "exploitations films", "film with sharks", "i'm looking for a hard sci-fi movie"*), so this requires a specific training of the IR component. However, this limit could be overcome in a real-world scenario since the system can ask for reformulating the sentence when it is not able to understand. Moreover, the ER accuracy is lower than the one measured on the bAbI dataset, probably because the entities are written directly by the users and might contain errors, or might not correspond exactly to the entities in our database (e.g., *"call work orange"*). This requires a disambiguation step that can not be performed in an in-vitro experiment.

	HR@5	**HR@10**	**HR@20**
UB	0.75	1.21	1.93
	Loss@5	**Loss@10**	**Loss@20**
IR	-34.00%	-30.86%	-24.03%
ER	-46.00%	-35.80%	-27.13%
SR	-20.00%	-16.05%	-14.73%

Table 1: Loss for each configuration in terms of HitRate

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a framework for building conversational content-based recommender systems in any domain. The only requirement to be satisfied is to provide a list of entities and properties in the Wikidata format. We will soon release the source code of the framework and make it available for the community. The preliminary experimental evaluation on two state-of-the-art datasets shows the impact of each component in a classical movie recommendation scenario. In this way, the user is aware of the limitations of the single modules implemented. In the next future, we plan to run an experimental evaluation on another synthetic dataset and to perform an in-vivo evaluation with real users on three different domains (movie, book, music) with the aim of investigating the impact of other capabilities (e.g. critiquing, explanation) on the recommendation process.

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] 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] Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931* (2015).

[3] M Goker and Cynthia Thompson. 2000. The adaptive place advisor: A conversational recommendation system. In *Proceedings of the 8th German Workshop on Case Based Reasoning*. Citeseer, 187–198.

[4] Peter Grasch, Alexander Felfernig, and Florian Reinfrank. 2013. ReComment: Towards Critiquing-based Recommendation with Speech Interaction. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 157–164. DOI:https://doi.org/10.1145/2507157.2507161

[5] 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.

[6] Chen He, Denis Parra, and Katrien Verbert. 2016. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications* 56 (2016), 9 – 27. DOI:https://doi.org/10.1016/j.eswa.2016.02.013

[7] Vladimir Ilievski, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2018. Goal-Oriented Chatbot Dialog Management Bootstrapping with Transfer Learning. *CoRR* abs/1802.00500 (2018). arXiv:1802.00500 http://arxiv.org/abs/1802.00500

[8] 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

[9] Jie Kang, Kyle Condiff, Shuo Chang, Joseph A. Konstan, Loren G. Terveen, and F. Maxwell Harper. 2017. Understanding How People Use Natural Language to Ask for Recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, Paolo Cremonesi, Francesco Ricci, Shlomo Berkovsky, and Alexander Tuzhilin (Eds.). ACM, 229–237. DOI:https://doi.org/10.1145/3109859.3109873

[10] 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

[11] 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.

[12] 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.

[13] 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

[14] F. Narducci, P. Basile, C. Musto, P. Lops, A. Caputo, M. de Gemmis, L. Iaquinta, and G. Semeraro. 2016. Concept-based item representations for a cross-lingual content-based recommendation process. *Information Sciences* 374 (2016), 1339–1351. DOI:https://doi.org/10.1016/j.ins.2016.09.022

[15] F. Narducci, M. Palmonari, and G. Semeraro. 2013. Cross-language semantic retrieval and linking of e-gov services. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8219 LNCS, PART 2 (2013), 130–145. DOI:https://doi.org/10.1007/978-3-642-41338-4_9

[16] Thuy Ngoc Nguyen and Francesco Ricci. 2017. Dynamic Elicitation of User Preferences in a Chat-based Group Recommender System. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, New York, NY, USA, 1685–1692. DOI: https://doi.org/10.1145/3019612.3019764

[17] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[18] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, and others. 2016. Holographic Embeddings of Knowledge Graphs.. In *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. 1955–1961.

[19] Francesco Ricci and Fabio Del Missier. 2004. Supporting travel decision making through personalized recommendation. In *Designing personalized user experiences in eCommerce*. Springer, 231–251.