User Modeling and User-Adapted Interaction
https://doi.org/10.1007/s11257-019-09250-7

An investigation on the user interaction modes of conversational recommender systems for the music domain

Fedelucio Narducci[1] · Pierpaolo Basile[1] · Marco de Gemmis[1] · Pasquale Lops[1] · Giovanni Semeraro[1]

Received: 14 August 2018 / Accepted in revised form: 11 November 2019

Abstract
Conversational Recommender Systems (CoRSs) implement a paradigm that allows users to interact in natural language with the system for defining their preferences and discovering items that best fit their needs. CoRSs can be straightforwardly implemented as chatbots that, nowadays, are becoming more and more popular for several applications, such as customer care, health care, and medical diagnoses. Chatbots implement an interaction based on natural language, buttons, or both. The implementation of a chatbot is a challenging task since it requires knowledge about natural language processing and human–computer interaction. A CoRS might be particularly useful in the music domain since music is generally enjoyed in contexts when a standard interface cannot be exploited (driving, doing homeworks, running). However, there is no work in the literature that analytically compares different interaction modes for a conversational music recommender system. In this paper, we focus on the design and implementation of a CoRS for the music domain. Our CoRS consists of different components. The system implements content-based recommendation, critiquing and adaptive strategies, as well as explanation facilities. The main innovative contribution is that the user can interact through different interaction modes: natural language, buttons, and mixed. Due to the lack of available datasets for testing CoRSs, we carried out an in vivo experimental evaluation with the goal of investigating the impact of the different interaction modes on the recommendation accuracy and on the cost of interaction for the final user. The experiment involved 110 people, and 54 completed the whole process. The analysis of the results shows that the best interaction mode is based on a mixed strategy that combines buttons and natural language. In addition, the results allow to clearly understand which are the steps in the dialog that are particularly strenuous for the user.

✉ Fedelucio Narducci
fedelucio.narducci@uniba.it

[1] Department of Computer Science, University of Bari Aldo Moro, Bari, Italy

Published online: 28 November 2019

Keywords Conversational recommender systems · Music recommender systems · Natural language processing

1 Introduction

Music plays a key role in our life, and everybody remembers a song associated with an important moment. Nowadays, we are able to listen to music in every moment and every context, thanks to music streaming services such as Apple Music, Spotify, Amazon Music, Google Play, and Deezer. Listening to music is one of the principal activities on the Internet, and using a smartphone to listen to music is increasing rapidly worldwide. Some statistics better explain this phenomenon. For example, in one minute, Spotify delivers 40,000 h of music,[1] 90% of paid audio streamers uses a smartphone for music listening, and[2] 45% of Internet users engages in licensed audio streaming. Accordingly, the way in which we listen to music is changing rapidly and we need new tools that help us to easily access to our preferred songs.

Among these tools, conversational recommender systems (CoRSs) are characterized by the capability of interacting with the user during the recommendation process through a dialog (Mahmood and Ricci 2009). Instead of asking users to provide all the requirements in one step, CoRSs guide the users through an interactive dialog (Jugovac and Jannach 2017). CoRSs are fundamental in those contexts in which we cannot interact with a classical user interface, but we can dialogue with the system using, for example, the voice. Indeed, the music domain, compared to other domains like books, electronic devices, and goods in general, can extremely benefit from an interaction based on the dialog. Let us consider some situations where we listen to music: driving, running, doing the housework, cooking, relaxing on sofa. Those are situations where, actually, we are not able to use a standard interface in a comfortable way. A voice user interface would be ideal. Pandora, for example, recently introduced voice commands in its app, and researchers are working on conversational interaction modes.[3]

Through the conversational interaction with a CoRS, users can express functional requirements or technical constraints exploited by the recommender for finding the songs that best fit their needs. Indeed, the main difference reported in the literature between a CoRS and a standard recommender system is just the capability offered to the user to interact and consequently modify the recommender system behavior (Mahmood and Ricci 2009). Let us think about Amazon's recommender system that does not give any possibility to express a judgment on the suggested items.

CoRSs usually 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 thus

[1] Visual Capitalist, What Happens in an Internet Minute in 2017?, http://www.visualcapitalist.com/happens-internet-minute-2017/.

[2] IFPI, Connecting with music consumer—Insight report, http://ifpi.org/downloads/Music-Consumer-Insight-Report-2017.pdf.

[3] https://techcrunch.com/2019/01/15/pandora-launches-a-personalized-voice-assistant-on-ios-and-android/.

improving the user experience. The interaction mode plays a crucial role in a CoRS: It not only influences the usability of the system, but it deeply conditions the flow of the dialog. For example, an interaction based on buttons follows a definite flow and no ambiguity can occur in the conversation. The system decides the question at any time, providing a definite set of possible answers. Conversely, an interaction based on natural language offers a free flow in the dialog, but it can generate ambiguity due to misunderstanding.

In this paper, we propose a CoRS for the music domain that exploits different interaction modes: natural language, buttons, and a combination of the two. Furthermore, our CoRS implements most of the capabilities that a recommender should offer, such as preference acquisition, profile exploration, critiquing strategies, and explanation.

We can summarize the contributions of this paper as the answers to the following research questions:

- RQ1: To what extent can conversational interfaces support the music recommendation?
- RQ2: What is the best conversational interface in terms of cost of interaction?
- RQ3: What is the best conversational interface in terms of recommendation accuracy?
- RQ4: Is the disambiguation step particularly strenuous for the user of a conversational music recommender system?

For this aim, we analyzed and studied the criticisms for developing a conversational recommender system in the music domain. Thus, we defined a general architecture of the system and we proposed a solution for implementing its components. After that, we focused our attention on the impact of different interaction modes on the user satisfaction in terms of recommendation accuracy and cost of interaction. We carried out a user study involving 110 users; 54 users completed the experiment by testing three interaction modes (natural language, buttons, and mixed). We also evaluated the general user satisfaction over different aspects and dimensions by asking users to answer to a set of questions according to the ResQue model (Pu et al. 2011).

2 Related work

Music recommendation is a very challenging research area since it has many distinguishing characteristics compared to other domains (Narducci et al. 2013a, b, c; Musto et al. 2009, 2010; Basile et al. 2014a). Therefore, music recommendation is not a simple adaptation of collaborative filtering methods to the music domain (Schedl et al. 2017). Some peculiar properties of the music domain are, for example, automatic playlist generation, context-aware music recommendation, and ubiquitous consumption. As regards the latter, music involves several daily tasks such as running, cooking, cleaning, working, and relaxing (Sloboda and ONeill 2001). Thus, music should be always available. Naturally, ubiquitous consumption needs also to focus on the interaction with the devices that provide music.

In He et al. (2016a), it is described how the recommendation process in an interactive recommender system works. The main difference compared to a canonical recom-

mender system is that during the interaction, the behavior of the recommender changes according to the data that user provides. In several works, this goal is achieved through an interactive visualization (O'Donovan 2008; Gretarsson et al. 2010; Bostandjiev et al. 2012, 2013; Schaffer et al. 2015). In MusiCube Saito and Fukusako (2011), for example, the interaction takes place by integrating user data and recommendations in a single view. The idea is that this visualization allows to discover relationships between rated songs and recommended items. Hence, users can freely explore the connections. An innovative interaction methodology is proposed in Barthet et al. (2015), where a component called Mood Conductor facilitates audience feedback during improvised live music performances. Through Mood Conductor, audience members can send emotional directions using their mobile devices; then, emotion coordinates are aggregated and clustered to create a video projection. Mood Conductor is a collaborative system that allows users to control music and lighting effects to express desired emotions. Another system that is able to provide a mood-based recommendation is proposed in Andjelkovic et al. (2019), where a graphic interface allows users to explore music collections by musical mood dimensions arranged through a hierarchy rather than the traditional Circumplex model. The recommendation is performed by exploiting mood associated with artists mood profile, and evaluation results show that visualization based on mood improves user acceptance and understanding of recommendations. In our study, we focus on an interaction based on natural language and buttons. Since music is nowadays mostly enjoyed on smartphone, we decided to investigate an interaction mode that is as natural as possible on this device. Indeed, smartphone users are already used to interact with other applications through buttons and natural language.

As regards exploitation of natural language processing (NLP) for music recommendation, most of works proposed in the literature apply NLP for analyzing music content and discovering similarities among items. In McFee and Lanckriet (2011), NLP is exploited for generating music playlists. The analysis of lyrics is also adopted in Laurier et al. (2008) for extracting the mood related to a given song. Similarly, NLP for sentiment analysis of lyrics is proposed in Yi et al. (2003). One of the first approaches that tries to apply dialogue strategies in music domain is that proposed in Wärnestål et al. (2007), where two dialogue strategies are investigated: interview and delivery. The authors model these strategies by analyzing a corpus of dialogues collected in a movie recommendation task and then implement them in CORESONG, a content-based recommender system for suggesting music. A more recent work that applies NLP for interacting with a music recommender system is Zhou et al. (2018). The authors describe MusicRoBot, a conversational music recommender system based on a music knowledge graph built by exploiting data from Xiami, a database of Chinese songs. The proposed system integrates classical dialogue modules, and it is able to provide explanation by using the user knowledge graph built during the dialogue with user. However, the paper does not provide technical details on the implemented components since it is a demonstration summary. The main difference compared to our work is that MusicRoBot does not provide several interaction modes as our system.

A detailed survey on music recommender systems highlights an interesting aspect of the music domain (Schedl et al. 2015): rarity of explicit rating data, while implicit positive feedback is common and related to listening events. In this scenario, trying to collect user preferences through a dialogue could be effective. On the other hand,

content-based approaches can be applied in the music domain by leveraging several characteristics associated with musical content. For example, metadata can be used to alleviate the cold-start problem in collaborative filtering (Bogdanov and Herrera 2012; McFee et al. 2012), or social tags coming from Last.fm are used to compute similarity between items (Green et al. 2009; Levy and Sandler 2008). Another strategy for extracting user preferences leveraging social media sources is proposed in Musto et al. (2012).

Some commercial devices such us Amazon Echo and Google Home allow to access music content through voice, but generally they only provide functionalities for accessing music or listening to music associated with a particular mood or genre. Hence, these systems cannot be properly defined as CoRSs.

A relevant work in the context of CoRSs is the one proposed in Bridge (2002), in which a programming model for natural language is proposed for modeling CoRSs by using an approach based on conversational analysis. In that paper, a model based on rules able to program the dialogue flow between user and agent is described. That approach is currently used by several dialogue managers, and we exploit a similar methodology in our system.

Other interesting CoRSs are proposed in other domains. In Sun et al. (2016), deep learning is used for combining personalized recommendation and dialogue in a virtual sales agent. In particular, the proposed system is trained without using labeled data, but exploiting unlabeled data collected by a mobile app which provides personal assistance services to thousands of users. Another deep learning approach is proposed in Li et al. (2018), where an architecture based on a general purpose sentence representations and hierarchical encoder–decoder architectures is extended with an autoencoder-based recommendation engine. Moreover, authors provide a dataset of 10,000 conversations in the movie domain that are exploited to train their model. The authors report that good performance can be achieved only when a large number of conversations are available for training, and this is the reason why we do not adopt a deep learning approach in the music domain since a large dataset of dialogs is not available. In Christakopoulou et al. (2018), an interactive recommender system for YouTube videos is designed by exploiting RNNs. In particular, RNNs are used for predicting questions that user might ask based on her recent behaviors and predict responses. The relevant aspect of this approach is the way in which RNN is trained. Since no conversation is available, the system exploited surrogate tasks by leveraging user data and other signals available from traditional interfaces. An interesting work about both search and recommendation functionalities provided by conversational systems is proposed in Zhang et al. (2018). Here, a multi-memory network architecture able to integrate attention mechanisms is used for building a framework based on the system ask–user respond (SAUR) paradigm. Moreover, the paper describes an interesting approach used to train the network by exploiting a large-scale collections of user reviews. A hierarchical deep learning architecture for CoRS is proposed in Basile et al. (2018). A hierarchical reinforcement learning framework splits dialogue into more manageable tasks corresponding to the goals of the user. The system is trained by using an automatically generated corpus of conversations (Suglia et al. 2017) in movie domains. All the systems based on deep learning approaches show promising results, but underline that good performance can only be achieved when a large

corpus of conversations is available. Some approaches deal with this lack of data by automatically building data, or augmenting data from other domains and tasks.

Moreover, there are several works in the literature that tried to improve various aspects of the conversational recommendation process (Jugovac and Jannach 2017). In Grasch et al. (2013), the authors demonstrated that a speech-based interaction model produces higher user satisfaction and needs less interaction cycles. In Nguyen and Ricci (2017), 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 He et al. (2016b), the authors present an interactive visualization framework that combines recommendation and visualization techniques to support human–recommender interaction. Several researchers developed integrated frameworks for CoRS (Goker and Thompson 2000; Ricci and Del Missier 2004) by combining conversational functionalities with adaptive and recovery functions. A key factor in a conversational recommender system is to understand "what to ask" and "how to ask" in order to elicit user preferences in an effective way. In Christakopoulou et al. (2016), the authors describe a preference elicitation framework able to identify which questions to ask a new user to quickly learn their preferences. Their approach exploits a probabilistic latent factor model able to discover latent structure in the recommendation space.

Regarding different interaction mechanisms, an interesting study proposed in Swearingen and Sinha (2002) proves that trust in the context of music recommender systems is affected by several aspects of the user interactions. In particular, in addition to accuracy, other relevant aspects are: transparency of system logic, familiarity of items recommended, and the process for receiving recommendations. This suggests that a dialogue mechanism that integrates explanation can induce trust in the user; moreover, the ability of a dialogue system to interact with users can produce more familiar recommendations. This intuition is corroborated by another important analysis provided in Konstan and Riedl (2012), where the authors underline the necessity of going beyond accuracy in terms of prediction and the importance of taking into account user experience during the design of both algorithms and systems.

In our previous work Narducci et al. (2018), we preliminary investigated the use of an interface based only on buttons for a movie recommender system. That work gave us the possibility to understand that the interaction with a conversational recommender system is a challenging research field to be investigated. However, compared to this research, that work is extremely simple, since there are not Intent Recognizer, Entity Recognizer, Sentiment Analyzer, and all the messages sent by users were not ambiguous and did not need neither natural language understanding nor generation.

3 The architecture of a conversational recommender system for the music domain

Given a user profile composed of songs, entities (albums, singers, songwriters, producers), and properties (music genres) on which the user expressed a binary rating (like/dislike), the music recommendation problem addressed by our recommender consists in selecting a list of songs that best match user preferences.

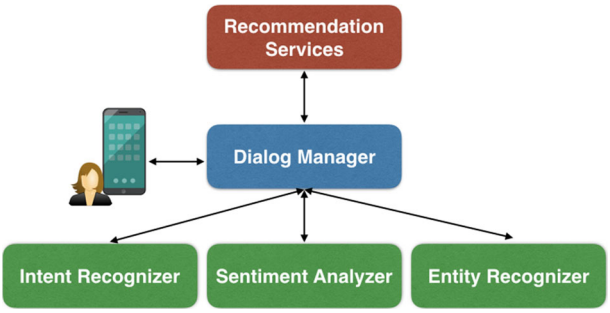

Fig. 1 The framework architecture

The recommendation process performed by a recommender system follows some well-defined steps: preference acquisition, profile building, generation of a list of recommendations, and user evaluation of recommendations. When the interaction with the recommender is provided through a conversational interface, the above-mentioned steps need to be handled by specific components that have to extract the required information (preferences, evaluations, etc.) by analyzing natural language sentences.

Accordingly, the implementation of a CoRS requires at least five components: a Dialog Manager, an Intent Recognizer, a Sentiment Analyzer, an Entity Recognizer, and a set of Recommendation Services. Furthermore, two other components are generally considered as optional: a voice synthesizer that transforms text into voice (text to speech) and an automatic speech recognizer that transforms voice into text (speech to text). We can state that the architecture of a CoRS is therefore quite standard. Indeed, our contribution on this aspect is not the definition of an original architecture for a CoRS, but to provide some ideas for implementing its components. Indeed, some components need to be designed and tailored to the specific domain they have to serve.

The architecture of our framework is depicted in Fig. 1. In the following, we analyze each component in detail.

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. It can be viewed as the orchestrator of the system and strictly depends on the task the dialog agent is dealing with. The 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 the information for fulfilling the user request is available, the *Dialog Manager* returns the message to the client.

So, for example, if a user asks for recommendations, but her profile is still empty, the Dialog Manager should ask a set of questions in order to acquire the user preferences. Hence, it implements the logic behind each task the CoRS can address. When a specific recommendation step is running and some additional information is required to complete it, the DM asks the user to provide the missing information. Indeed, it implements a *slot-filling* model, where a set of features need to be filled in order to accomplish the user goal. For generating messages to be sent to the user, we adopted a strategy based on a set of standard sentences that we fill in with the contextual informa-

tion. For example, there is a standard sentence like *Ok, I added a <sentiment> rating for: <entity>.* for confirming the preference acquisition. The sentiment (positive or negative) is identified by the Sentiment Analyzer, whereas entity is identified by the Entity Recognizer. More details on those components are reported in the next sections.

Intent recognizer This component has the goal of defining the intent of the user formulated as a natural language sentence. When the user sends a message, the recommender system must first understand what the goal of the user is and what she wants to express and get by that message. Hence, the recommender, as first step, needs to identify the intent of the user. In our scenario, there are four main intents to be recognized:

- *preference* the user is providing a preference. The preference can be expressed on a new song or on a recommended song. In the latter case, the preference can also express a judgment (e.g., *"I like this song, but I don't like its producer"*).
- *recommendation* the user asks to receive a recommendation. This intent is the condition for other sub-intents, such as *explanation* (when the user asks the motivation for a given recommendation), *critiquing* (when the user expresses a critique on one or more features of the recommended item), and *more info* (when the user asks more details on the recommended item, e.g., the official video clip from YouTube, the genre, the producer).
- *show profile* user asks to visualize (and modify) the preferences stored in her profile.
- *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.

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 which acts as a hub of several structured data coming from Wikimedia sister projects.[4] We choose to develop a custom ER for three reasons: (1) existing entity recognizer algorithms are hard to customize to a specific domain; (2) entity recognizers included in existing dialog manager toolkits usually 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; (3) we need to map recognized entities to concepts in the KB exploited by the recommendation algorithm. We need to develop an ER algorithm that is customizable according to the entities involved in a specific domain, in this case, the music domain. A classical entity linking approach is composed of two steps: *spotting* and *disambiguation*. In the spotting step, the algorithm analyzes the text in order to discover candidates entities, and in particular, the algorithm detects sequences of words (*surface form*) representing an entity and retrieves all the concepts that can be associated with the surface form.

[4] Wikipedia, Wikivoyage, Wikisource, and others.

The disambiguation step tries to select for each surface form the correct concept from a list of candidate concepts. For the spotting step, we exploit a strategy developed in our previous work Basile et al. (2015a, b) able to deal with noisy text, since this is an important feature to take into account during a conversation.

For the disambiguation step, we propose a new approach based on a graph embedding technique for leveraging information about relations between concepts in the knowledge graph used in the specific domain. The disambiguation task is challenging, since, for example, both the surface forms *Barry Eugene Carter* and *Barry White* can refer to *Barry_White:singer*. Furthermore, the same surface form *Barry White* can refer to more than one concept, like *Barry_White:singer* and *Barry_White:songwriter*. The first step is to create the subgraph G_d of the KB according to the list of concepts L_e and properties L_p provided during the framework setup. G_d is built by querying Wikidata and retrieving all the triples that have entities in L_e as subject and properties in L_p as predicate. In our case, we consider only entities and properties related to the music domain.

For spotting entities in the user request, we index for each entity all the possible aliases provided by Wikidata. For example, for the concept *Q213647* (*Barry White*), we store the aliases *Barry Eugene Carter* and *Barry White*. The index is exploited by the spotting module as described in Basile et al. (2015a, b). The output of the spotting is a list of candidate concepts assigned to each surface form.

The last step consists in selecting the correct concept for each surface form. 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, in the same sentence, tends to refer to entities that are in some way related. Let us suppose the user writes the following sentence: *I like Michael Jackson and Beat It*. *Beat It* is an ambiguous entity since it might be referred to the candidate concepts *Beat It by Michael Jackson* and *Beat It by Sean Kingston*. Hence, this surface form needs to be disambiguated. In order to accomplish this task, the ER computes the similarity between the candidate entities (i.e., *Beat by Michael Jackson* and *Beat It by Sean Kingston*) and the other entities in the context (i.e., Michael Jackson). In this case, the similarity will be higher between *Michael Jackson* and *Beat It by Michael Jackson* than *Michael Jackson* and *Beat It by Sean Kingston*. Therefore, *Beat It by Michael Jackson* will be chosen.

More formally, the ER computes the score $s(c_j)$ assigned to each candidate concept c_i for the chunk e_j. It is computed according to Eq. 1, where E is the set of the other surface forms in the text (in the example above Michael Jackson). The score $s(c_j)$ is the sum for each surface form e_k in E of the maximum similarity score between all the candidate concepts c_i of e_k and c_j.

$$s(c_j) = \sum_{e_k \in E} \text{argmax}_{c_i \in C_{e_k}} \text{sim}(c_j, c_i) \tag{1}$$

The idea is to select a set of concepts that are in some way strongly related between them. In order to compute the score $s(c_j)$, we need to define a similarity (relatedness) function between concepts. In our approach, we rely on graph embeddings that have recently gained considerable attention (Nickel et al. 2016a). 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) (Nickel et al. 2016b), 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 can compute the similarity between two entities using 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 spotting component and the list of candidate concepts is re-ranked in descending order. The first element of each candidate list is the concept assigned to each surface form in the text. Sometimes the score assigned to the first concept in the candidate list can be low since the text is too ambiguous or there is not enough context for the disambiguation. In this case, the Dialog Manager asks the user which concept she refers to.

Furthermore, the ER implements a *Property-Type Recognizer* that is able to identify property-mentions in the sentence. Given the sentence *I like Rocket Man, but I hate its producer*, the ER identifies the property *producer* and assigns the negative sentiment to it. Afterward, the ER will retrieve the entity associated with that property, namely *Gus Dudgeon*, in the given example. The list of properties is provided in the configuration file.

Sentiment analyzer The Sentiment Analyzer (SA) has the goal of assigning the right sentiment (i.e., positive and negative) to the entities mentioned in sentences (e.g., singer, song, genre) and identified by the ER module. We developed a new component able to combine the results provided by the ER module and the output of a sentiment analyzer tool. We adopt the Stanford CoreNLP Sentiment Tagger[5] as sentiment analyzer tool. It takes as input the user sentence and returns the identified sentiment tags. The CoreNLP Sentiment Tagger solves the problem of assigning the sentiment tags to each sub-tree of the parsing tree of the sentence. The difficulty is that a sentence can contain different sentiment tags (also opposite) as well as different entities. For example, given the sentence *I like Rocket Man, but I don't like Gus Dudgeon*, the Sentiment Tagger identifies a positive sentiment (i.e., like) and a negative one (i.e., do not like). SA should assign the positive sentiment to the entity *Rocket Man* and the negative sentiment to the entity *Gus Dudgeon*. For solving this issue, we combine the output of the ER component with the sentiment tags assigned by the CoreNLP Sentiment Tagger. The sentiment–entity association is performed by computing the distance between the sentiment tag and the entity recognized by the ER module in the sentence. This distance is expressed in terms of the number of tokens that separate the sentiment tag from the entity. Given the aforementioned example, the distance between *like* and *Rocket Man* is zero, as well as the distance between *don't like* and *Gus Dudgeon*. The sentiment tag identified by CoreNLP is thus associated with the closest entity in the sentence.

Recommendation services This component collects the services strictly related to the recommendation process. The *recommendation algorithm* implemented is the PageRank with Priors Haveliwala (2003), also known as Personalized PageRank.

[5] https://stanfordnlp.github.io/CoreNLP/.

The algorithm works on a graph in which the nodes are the entities the recommender deals with (e.g., songs, artists, producers, genres, etc.). These entities are extracted from Wikidata, and their connections (edges in the graph) are extracted from DBpedia.[6] Hence, for example, the song *Rocket Man* is connected to the producer node *Gus Dudgeon*, to the genre node *soft rock*, and to the performer *Elton John*. The Personalized PageRank assigns different weights to different nodes to get a bias toward some nodes (in our case, the preferences of a specific user). The algorithm is run for each user, and at the end of the execution, each node has a probability score assigned. The nodes representing songs with the highest probability will be recommended.

The algorithm has been effectively used in other recommendation environments (Basile et al. 2014b). Another recommendation service offered by the framework is the *explanation* feature. The framework implements an explanation algorithm inspired by Musto et al. (2016). 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 song *Rocket Man* because in this song the composer is *Elton John*, and you like *Elton John*, the genre is *soft rock* and you like *soft rock*.". In this case, the system used the connections between the recommended song *Rocket Man* (and its properties) and the user preferences (*Elton John* and *soft rock*). The last recommendation service implemented is the *critiquing*. This service allows to acquire a valuable feedback on a recommended item and its properties (e.g., *I like the song Rocket Man, but I don't like the producer Gus Dudgeon*), and this feedback will be used in the next recommendation cycle, by properly setting the weights of the nodes in the PageRank graph.

3.1 The framework at work

Figure 2 shows a toy example in order to explain how the single components work. Let us suppose that the system receives the message *I like Circus, but I hate its producer*. This sentence expresses both a positive and a negative preference. The positive preference is related to a song or an album (i.e., *Circus*), while the negative preference is on one of its properties (i.e., the producer). The first step (1) is the identification of the intent. The Intent Recognizer is invoked, and it identifies the intention of the user, that is, to express a *preference*. Now, the Dialog Manager implements all the actions required to complete the job (step 2). First, it asks the Entity Recognizer to identify and link the entities in the sentence (step 3). Next, the Sentiment Analyzer associates a sentiment (positive or negative) to each retrieved entity (step 4). However, in our example, the retrieved entity *Circus* can be associated with a song or to an album. Hence, the disambiguation process starts (step 5). The system asks the user which entity she refers to (step 6), and the answer (step 7) is passed to the Intent Recognizer, which recognizes the new intent *preference-disambiguation* (step 8). Now, the Dialog Manager has to solve the last ambiguity before adding the preferences to the user profile (step 9). Indeed, the user expressed a negative preference for the producer of the song without specifying the name. Hence, the system retrieves the entity the user

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

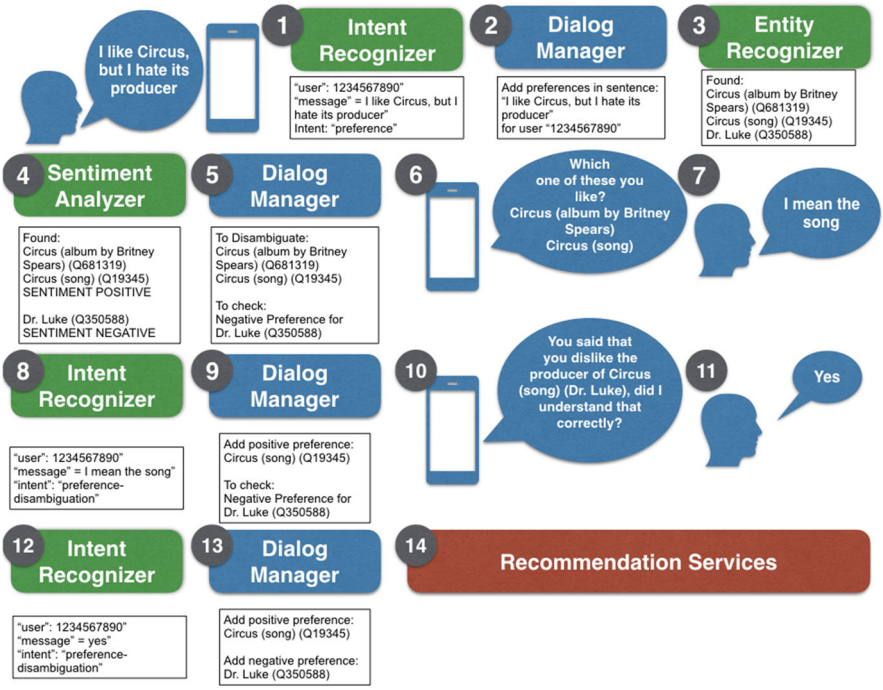

Fig. 2 The framework in action

refers to (i.e., *Dr. Luke*) and asks the user to confirm (step 10–11). Now, the Intent Recognizer identifies the *preference-disambiguation* intent again (step 12) and the Dialog Manager can store the preferences in the user profile by invoking the correct Recommendation Service (step 13–14). We report this example that shows better how the system works with two preferences that both require disambiguation. Please note that the disambiguation at step 7 is different from the disambiguation performed by the Entity Recognizer and described in Sect. 3. Indeed, there are two disambiguation steps in our recommender system: One is the responsibility of the ER, and another is delegated to the user. However, these two disambiguation tasks are different. The first disambiguation performed by the ER has the goal of reducing the space of candidate entities for a specific surface form (in this example *Circus*). However, when the ER is not yet able to associate the surface form (*Circus*) to the right entity (album or song), the recommender asks the user to choose what option she refers to.

The dialog will be handled differently on the basis of the implemented interaction mode. The next section provides more details on this aspect.

3.2 Interaction modes

As described in Sect. 3, the Dialog Manager sends a JSON message to the client that contains all the information to be visualized on the interfaces. This allows to have a single API for the different interaction modes.

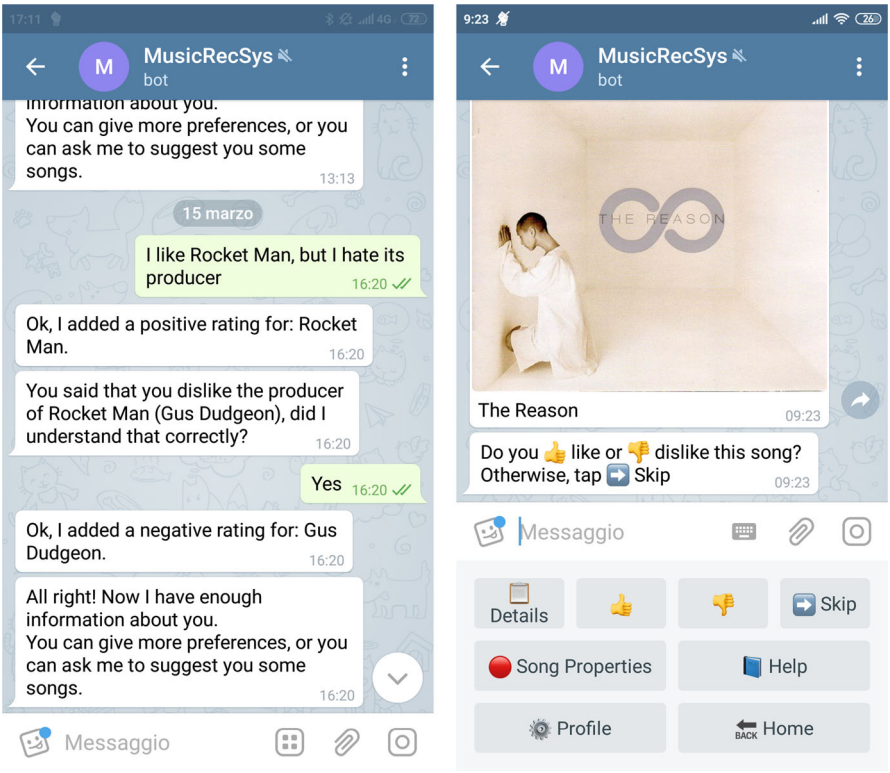

Fig. 3 Preference acquisition for the NL (left) and button-based (right) interaction modes. NB: for this screen, both NL and mixed interactions are the same

There are three different interaction modes: only buttons, only natural language, and mixed (Figs. 3, 4, 5, 6, 7, 8).

The capabilities of the recommender remain unchanged independently from the interaction mode implemented, and thus, each interface basically offers the same functionalities. What changes between an interface and another is how a functionality is offered.

Through the first mode—buttons—the user interacts with the system by tapping buttons. Hence, the dialog is completely driven by the system, which asks the user the information required to complete the recommendation. The user can only type the name of an entity she wants to rate. If the user wants to change the flow of the dialog (e.g., she is receiving the recommendation, but she wants to show or change the preferences in her profile), she needs to navigate in the menu and choose the corresponding button.

The second interaction mode is completely based on natural language. The flow of the dialog does not follow a rigid path. At each time, the user can decide to start a new action and the system will recognize her intent and activate the necessary components. When a disambiguation is required during the dialog (e.g., step 6 in Fig. 2), the user has to answer by writing the chosen entity [e.g., Circus (song)].

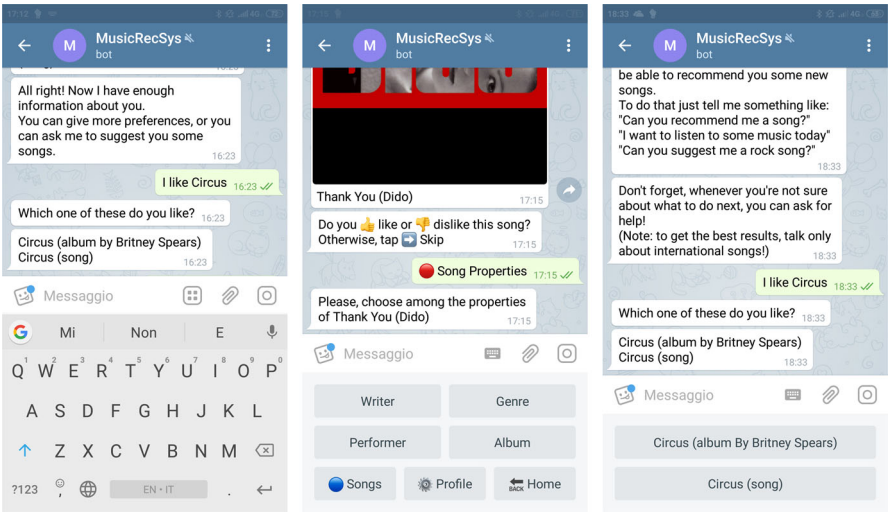

Fig. 4 Disambiguation prompt for the NL (left) and mixed (right) interaction modes. The button-based mode (center) does not require disambiguation, but instead uses several sub-menus for rating song properties

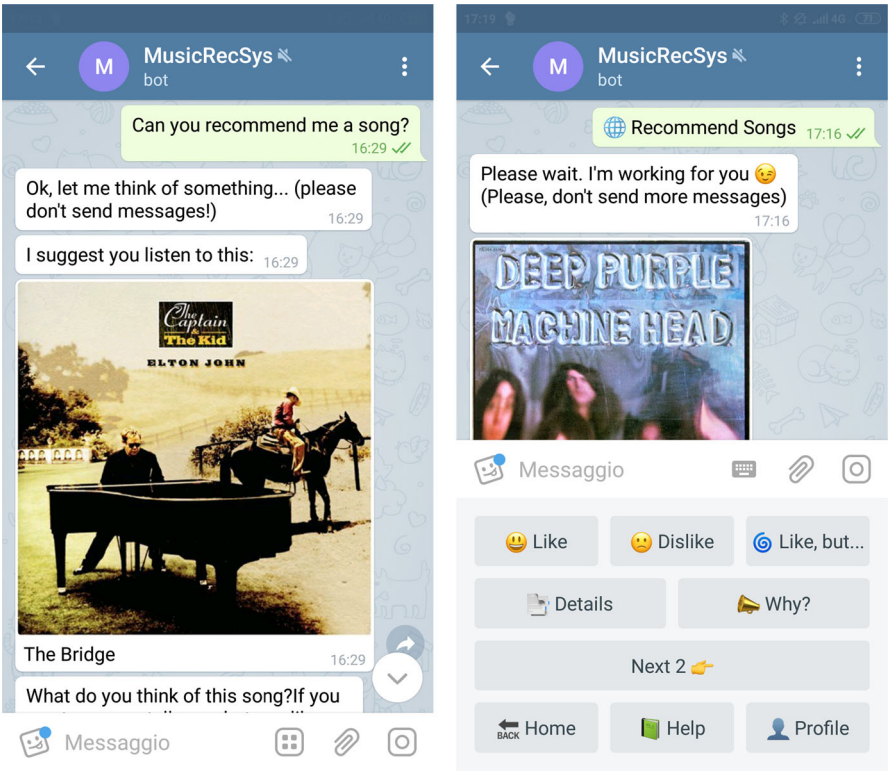

Fig. 5 Requesting a recommendation in the NL (left) and button-based (right) interaction modes. NB: the mixed mode is the same as the NL mode

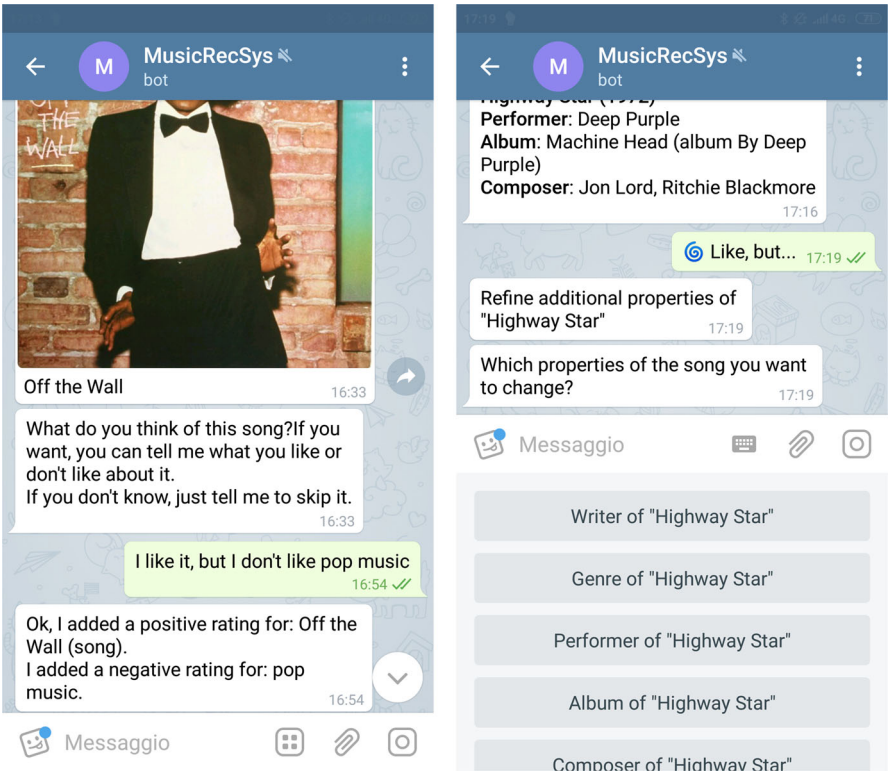

Fig. 6 Critiquing example for the NL (left) and button-based (right) interaction modes. NB: the mixed mode is the same as the NL mode

The last interaction mode (i.e., mixed) is basically the same as the previous one, except for the disambiguation stage. In this case, when a disambiguation is required (again, the step 6 in Fig. 2), the system will show a button for each possible option. Hence, there is no possibility that the user makes a mistake in writing the entity, since she only taps the corresponding button.

Figure 3 shows the preference acquisition step through the NL and button-based interfaces. Through the NL mode, the user can freely express her preference by writing sentences which can be more or less complex. Conversely, the button-based mode proposes songs, singers, writers on which the user can express a rating. This interface also allows to type the name of an entity to rate (e.g., Rocket Man). For the preference acquisition step, the NL and mixed modes have the same behavior.

Figure 4 shows the disambiguation step. As mentioned before, the disambiguation step is required when the ER is not able to associate a unique entity to a given surface form. (In Fig. 4, the ambiguous entity is *Circus*.) The NL and mixed interactions have a similar behavior: The only difference is that the latter shows a button for each option the user can choose. In Fig. 4, if the user wants to choose *Circus (song)* through the NL interface (left side), she has to exactly write *Circus (song)*, whereas through the mixed interface she only taps a button. The button-based interaction does not provide a

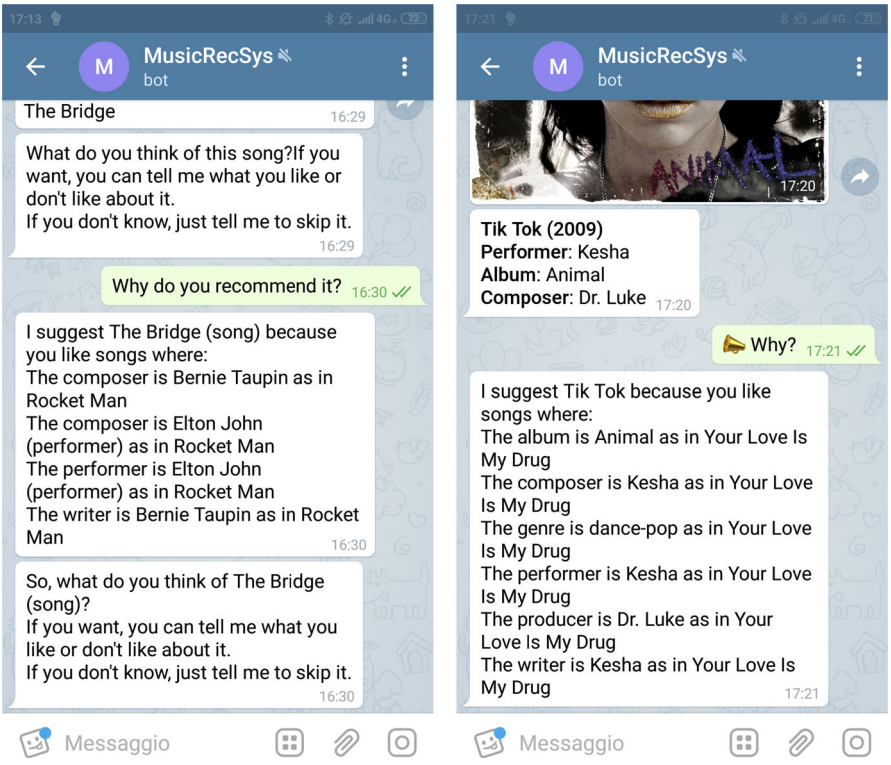

Fig. 7 Requesting an explanation in the NL (left) and button-based (right) interaction modes. NB: the mixed mode is the same as the NL mode

disambiguation step since it is inherently not ambiguous. However, for some entities, the user has also to choose the property associated with. As an example, in Fig. 4 for Thank You (Dido), the system asks the user whether she refers to the Writer, the Genre, the Performer, or the Album.

The recommendation request (Fig. 5) is performed by writing a sentence like *What can I listen to?* through the NL interface and by tapping the button *Recommend Songs* through the button-based interface. Also, for this step, there are no differences between NL and mixed modes.

During the recommendation step, the user can send a feedback to the recommender in order to improve the next recommendations (Fig. 6). Through the NL modes, the user sends a message similar to *I like it, but I don't like pop music*. In the button-based interaction, the process is more articulated. When the user taps the button *I like but..*, the recommender proposes a set of options for identifying what is the characteristic on which the user desires to express a feedback.

The explanation (Figs. 7, 8) is provided by the recommender in two different ways: by providing more details on the recommended song or by discovering connections between the user preferences and the characteristics of the recommended song. Also, in this case, the request is performed by writing a sentence for the NL mode (e.g.,

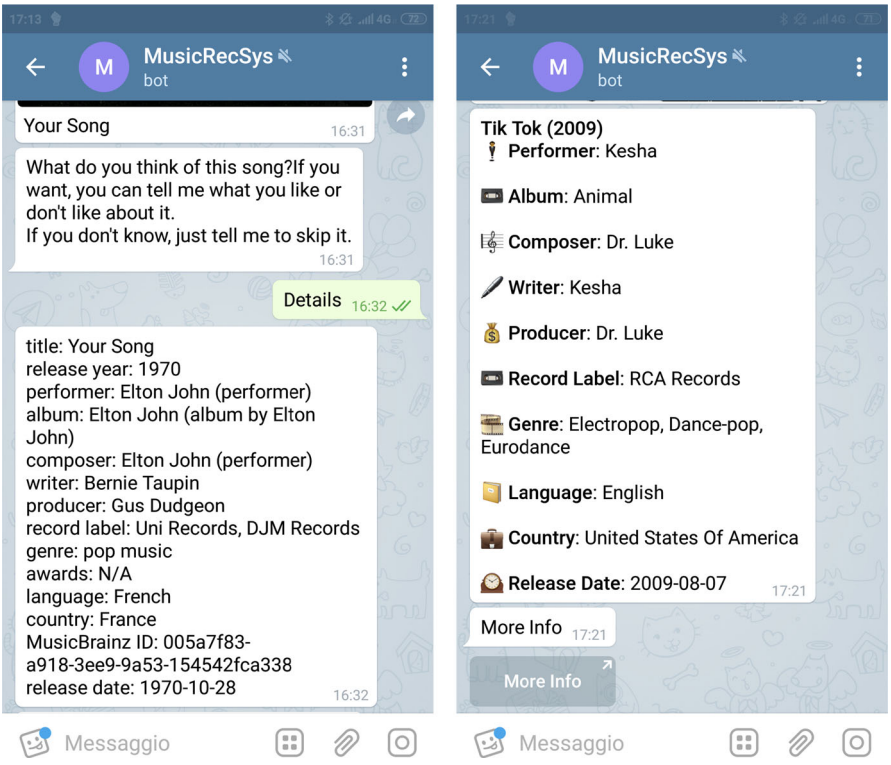

Fig. 8 Requesting song details in the NL (left) and button-based (right) interaction modes. NB: the mixed mode is the same as the NL mode

Please, show more details on this song or *Why did you recommend this song?*) or by tapping a button (e.g., *More details* or *Why?*)

4 Experimental evaluation

We designed a within-subject experiment by involving 110 subjects (female = 11.85%, high school = 75.33%, medium–high interest in music = 82.59%). The participants were recruited from students enrolled on the first year of the Bachelor's Degree in Computer Science, and this is the motivation of a strong bias toward male users. Users did not receive any incentive to participate in our study. The goal of our experiment was to compare different interaction modes in order to answer to the research questions defined in Sect. 1.

Each subject tested three system configurations (see Sect. 3.2):

– NL: in this configuration, the interaction is completely based on natural language. The user formulates her requests via natural language sentences, and the system answers in the same way.

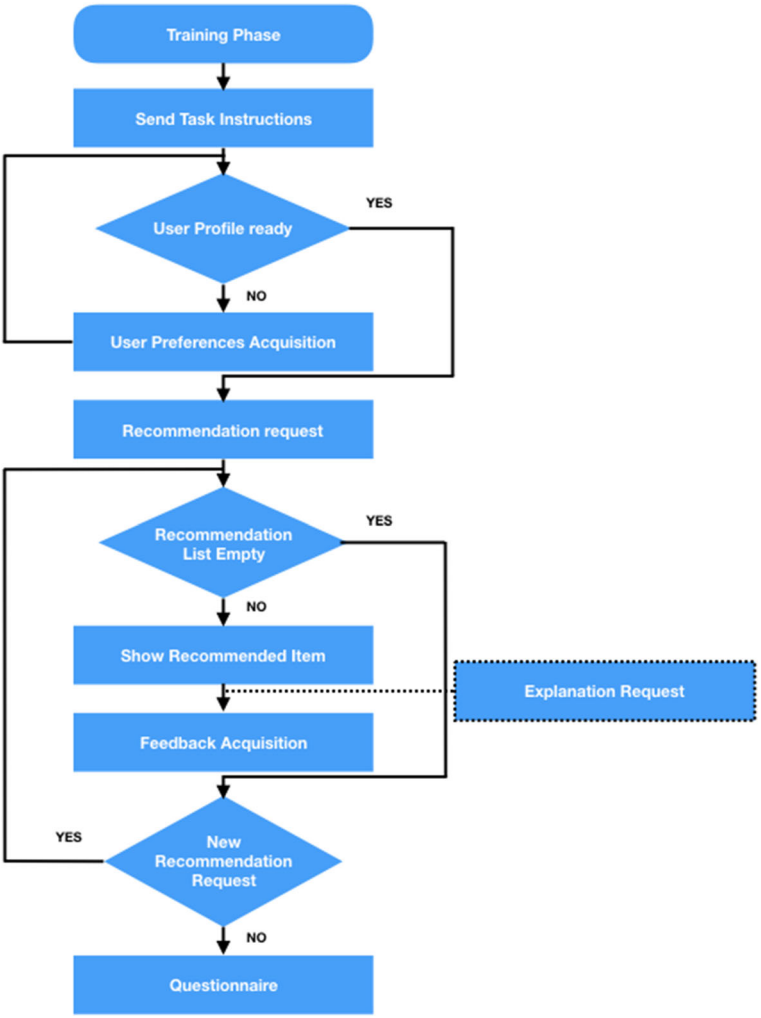

Fig. 9 The experiment process

– Button: in this configuration, the interaction is completely based on buttons. The dialog follows a rigid path since the user actions depend on the options the system offers at any time. In this case, there are no possible misunderstandings between the system and the user, since for each question, the recommender system provides a set of possible answers.
– mixed: in this configuration, the interaction is mostly based on natural language. However, when the system needs a disambiguation, the possible options are shown through a list of buttons. Hence, the user chooses the right option by tapping the corresponding button.

Each configuration was randomly assigned to each subject. The experiment followed the step as depicted in Fig. 9. There was a training phase where users freely

interacted with the system and became familiar with the different interfaces. After this step (the duration was about 30 min), the experiment started. The whole experiment takes about 2 h. The system provided an introduction with some instructions on the experiment. The first step was the preference acquisition. Users had to express at least three preferences. When the profile was ready, the user could ask for recommendations. The system proposed a list of five songs, and for each song, users gave a feedback (like, dislike, or she can skip the recommendation without providing a feedback). It is worth noting that, during the experiment, the songs in the user profile are filtered out from the set of potentially recommendable songs. Optionally, users could ask for an explanation of the recommended item. The feedback could be also related to a specific property of the song (e.g., the singer, the producer, the songwriter, the genre, etc.). When users enjoyed the first set of recommendations, they could decide to request a new set of recommended songs or to stop the experiment and answer to the questionnaire based on the ResQue model (Pu et al. 2011). Fifty-four out of 110 subjects completed the experiment by testing all the configurations. Naturally, each step of the experiment was performed according to the interaction mode the user was evaluating at that time, as defined in Sect. 3.2.

4.1 Dataset

In order to carry out the experimental evaluation, we built a dataset composed of 12,926 entities, 27,029 properties, and 105,372 entity-property connections extracted from Wikidata. The user can express her preferences in terms of song, album, genre, and producer. The sources we exploited for building the dataset are:

- Wikidata: we extracted all the items belonging to the class song (Q7366) with release date not null and after 1950. For each song, we extracted: title, singer, album, songwriter, composer, label, country, language, release year, producer, prize, MusicBrainz ID.
- Billboard.com: in order to ensure that the recommender contains the latest most popular songs, we extracted the list of 100 hot songs for the year 2017.
- MusicBrainz and CoverArtArchive: we used these Web sites for retrieving the album cover for each song.
- YouTube and Last.fm: we used these Web sites for extracting the video for each song.

4.2 Metrics

In this work, we adopted objective metrics that are independent of the user opinions, combined with a questionnaire that assesses the subjective experience of the user. The aim is to evaluate the accuracy of the system and the interaction efficiency.

4.2.1 Accuracy metrics

The metrics adopted for evaluating the accuracy of the system are:

– *Accuracy* the ratio between the liked items and the recommended items.[7]
– *Mean average precision (MAP)* the average precision of the recommender lists which takes into account the items liked by the user and their ranking. MAP is based on the average precision (AP), computed as follows:

$$\text{AP@}k = \frac{\sum_{l=1}^{k} P@l \cdot \text{rel}(l)}{k}, \tag{2}$$

where $P@l$ is the precision considering the first l positions in the recommended list, and rel(l) is a function equal to 1 if the item at rank l is liked, 0 otherwise. In our experiment, $k = 5$. The MAP is the average of AP@k for all the recommended lists.

4.2.2 Interaction-cost metrics

The metrics adopted for evaluating the cost of interaction are:

– *Number of questions (NQ)* the number of questions the chatbot asked the user. It includes disambiguation requests, confirmation requests, and item evaluations.
– *Time per question (TPQ)* the total time required by the user to answer a system question. The time is calculated from the time stamp of the message sent to the user. It is useful for analyzing the effort required to the user for understanding and answering a system request.
– *Interaction time (IT)* the time required for completing the experiment. It is calculated from the first message sent to the user, to the questionnaire administration.
– *Query density (QD)* an adaptation of the query density proposed in Glass et al. (2000). It measures the amount of concepts the user can introduce for each message. In this experiment, a concept is an entity (e.g., a song, an artist) or a property (e.g., the genre). QD is computed by the following formula:

$$\text{QD} = \frac{1}{N_d} \sum_{i=1}^{N_d} \frac{N_u(i)}{N_m(i)}, \tag{3}$$

where $N_u(i)$ is the number of distinct concepts introduced by the user u in the dialog i, $N_m(i)$ is the number of messages of the user in the dialog i, and N_d is the number of dialogs. In this experiment, $N_m(i)$ only counts the messages related to the preferences through which the user can introduce new concepts. For NQ, TPQ, IT values, lower are better; on the contrary for QD, higher is better.

4.3 Results

Figure 10 summarizes the results of the interaction-cost metrics. The first observation is related to the NQ. A system should reduce the number of questions, preserving a

[7] Please not be confused with MAP.

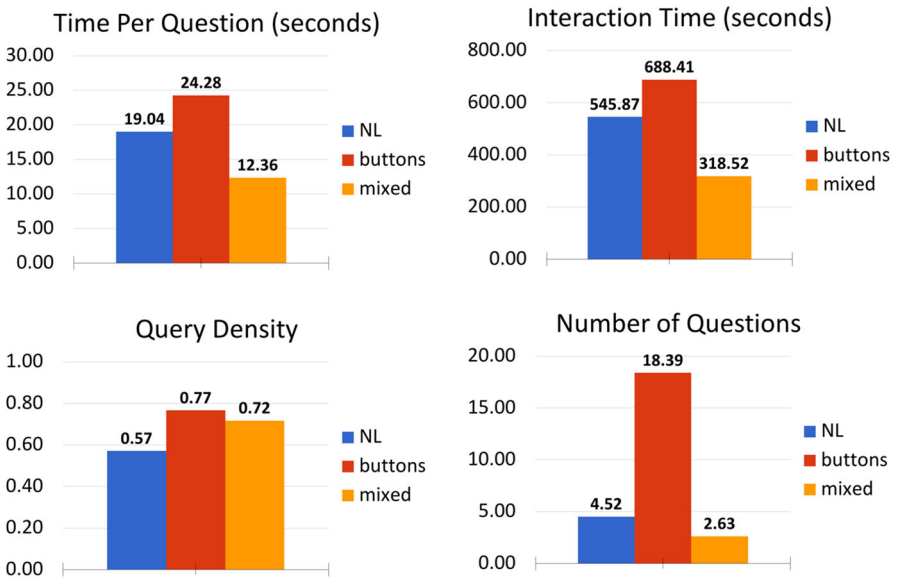

Fig. 10 Results for the interaction-cost metrics

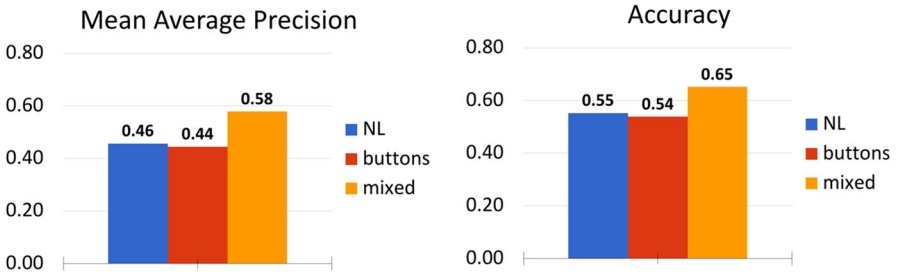

Fig. 11 Results for the accuracy metrics

good recommendation accuracy. We can see that the interaction based on buttons has a number of questions larger than the interactions based on NL and mixed. This result is probably due to the fact that an interaction based only on buttons is completely driven by the system, and thus, it requires a larger number of questions in order to obtain the information needed for generating recommendations. Hence, the button-based interaction needs to ask more questions than the other modes. For example, if a user wants to give a feedback on a recommendation writing I like the recommended song, but I don't like its genre, the NL-based interface does not need any further information. To do the same in the button-based interface, the user first taps the button I like but; then, the system proposes the set of properties to rate and subsequently asks the rating to be assigned.

As regards TPQ, which measures the time required by the user for answering to the system questions, we can see that the TPQ is drastically reduced for the mixed interaction compared to the other two. Indeed, the TPQ for the mixed interaction is half

than the button-based interaction and 36% less than the NL interaction. Furthermore, the NL-based interactions (i.e., NL and mixed) generally require less time than the button-based mode. Such minor effort is likely due to the rigid path that the interaction based on buttons follows, with a set of predefined answers to each question. Hence, the user spends more time for identifying the right answer in the predefined set proposed by the system. By comparing NL and mixed, the only difference between them is the use of buttons for the disambiguation questions. Hence, the fact that the disambiguation is a difficult task in terms of cost of interaction is indisputable. This is also confirmed by the IT (interaction time) value that shows the lowest result with the mixed interaction.

With regard to QD, we observe that it has the lowest value with the NL interaction. Also, this result might depend on the disambiguation task. Indeed, when the user is not able to express her preferences with a single sentence, the denominator in Formula 3 grows up, whereas the number of concepts does not increase (i.e., the numerator). Indeed, the system asks the user to disambiguate as long as she correctly writes the name of the ambiguous entity. This task is drastically improved when the user only has to tap a button for disambiguating an entity. A further confirmation of this interpretation is that the QD value is very similar for the button-based and mixed interactions that manage in the same manner the disambiguation. The low score can also be explained by the fact that users did not take advantage of the ability to give multiple preferences in the same message, which was possible in both the NL and mixed interactions.

Figure 11 reports the results for the accuracy metrics. Both metrics show that the best result is achieved by the mixed mode. Since for all the configurations, the recommendation algorithm is the same, the only motivation for this result is that the user can express more effectively her preferences through the mixed interaction. The recommendation lists generated in the mixed interaction are better both in terms of accuracy (Accuracy) and ranking (MAP).

4.4 Discussion and statistical analysis

In order to answer the research questions defined in Sect. 1, we performed some statistical analyses on the results discussed in the previous section. We first performed a MANOVA statistical test on all the accuracy and cost of interaction metrics, and results demonstrated that there is a statistically significant difference among the three interaction modes (p value $= 0.001$, F-score $= 29.126$, $Df = 2$, Wilks $= 0.21943$, $\eta^2 = 0.5315711$).

Next, we performed a Wilcoxon test in order to investigate the behavior of each couple of interaction modes for each metric. Tables 1 and 2 report the results of the Wilcoxon test for the interaction-cost and accuracy metrics, respectively. (The same results are confirmed by the UNIVARIATE ANOVA whose results are reported in "Appendix.") Since for each metric, we performed multiple independent comparisons, we applied the Bonferroni correction on the p value (Demšar 2006). Each cell reports the p and Z value, and the symbol + or −: + means that the interaction mode on the row is better than the interaction mode on the column for that specific metric (respectively, − worse). For example, for the metric Number of Questions (Table 1), in the cell mixed-NL, the value + ($< .001$) means that mixed showed a better performance (+),

Table 1 Results of the Wilcoxon test on the interaction-cost metrics (sample size = 54)

Number of questions	NL	Buttons	Mixed
NL		+	−
		$p < .001$	$p < .001$
		$Z = 26$	$Z = 203.50$
Buttons	−		−
	$p < .001$		$p < .001$
	$Z = 26$		$Z = 2.5$
Mixed	+	+	
	$p < .001$	$p < .001$	
	$Z = 203.50$	$Z = 2.50$	

Time per question	NL	Buttons	Mixed
NL			−
			$p < .001$
			$Z = 168$
Buttons			−
			$p < .001$
			$Z = 376$
Mixed	+	+	
	$p < .001$	$p < .001$	
	$Z = 168$	$Z = 376$	

Interac. time	NL	Buttons	Mixed
NL			−
			$p < .001$
			$Z = 225$
Buttons			−
			$p < .001$
			$Z = 254$
Mixed	+	+	
	$p < .001$	$p < .001$	
	$Z = 225$	$Z = 254$	

Query density	NL	Buttons	Mixed
NL		−	−
		$p < .001$	$p < .001$
		$Z = 350$	$Z = 286.50$
Buttons	+		
	$p < .001$		
	$Z = 350$		
Mixed	+		
	$p < .001$		
	$Z = 360.5$		

In the upper right corner, there are the metrics. + means that the interface on the row is better than the interface on the column (respectively, − means worse) and the differences on that matric are statistically significant

Table 2 Results of the Wilcoxon test on the accuracy metrics (sample size = 54)

MAP	NL	Buttons	Mixed	Accur.	NL	Buttons	Mixed
NL			−	NL			−
			$p = 0.009$				$p = .009$
			$Z = 335.50$				Z=275.50
Buttons			−	Buttons			−
			$p = .015$				$p < .001$
			$Z = 369$				$Z = 264.50$
Mixed	+	+		Mixed	+	+	
	$p = .009$	$p = .015$			$p = .009$	$p < .001$	
	$Z = 335.50$	$Z = 369$			$Z = 275.50$	$Z = 264.50$	

In the upper right corner, there are the metrics. + means that the interface on the row is better than the interface on the column (respectively, − means worse) and the differences on that metric are statistically significant

namely a lower number of questions, compared to NL and the difference is statistically significant (p value< 0.001). Empty cells mean that the difference is not statistically significant for that couple of interaction modes for that metric.

Table 1 emerges that the mixed interaction shows statistically significant differences compared to the other two interactions. Hence, we can certainly state that this interaction mode is the best in terms of cost of interaction for the user of a music recommender system. Through the NL-based interaction, generally, we do not have any improvement compared to the button-based interface. This is not an obvious outcome since one would think that an interaction completely based on natural language might facilitate the user of a conversational music recommender system.

In terms of accuracy metrics, the situation is quite similar. Indeed, the mixed interaction shows the best performance compared to NL and button-based interfaces. Accordingly, a lower cost of interaction finally results in a higher recommendation accuracy. Conversely, there is no statistically significant difference between the NL-based interface and the button-based mode.

4.5 Questionnaire

At the end of the experiment, the system proposed a set of questions to the user. This questionnaire is a combination of the short version of the ResQue model[8] and the questionnaire proposed in Silvervarg and Jönsson (2011). The medians and averages for each answer are reported in Table 3. The Likert scale is represented with numerical values, with 1 meaning "Strongly Disagree" and 5 meaning "Strongly Agree." A median of 4.5 means that there is a draw between 4 and 5 answers. For Question 18, we ignored users that chose "I didn't do any critiquing." We can see that for all questions, the median is the "Agree" value, and thus, the users have been generally satisfied with the interaction with the system. We performed a MANOVA statistical test for assessing the significance of the differences, and generally, there are no statistically significant differences between the different modes (p value$= 0.9374$). However, by analyzing the single questions, Q19 *It is easy for the recommender to understand what I said* shows a statistically significant difference among the interaction modes (p value$= 0.03$, F-score$= 0.9774$, $Df = 2$, Wilks$= 0.81694$, $\eta^2 = 0.096154$). The interaction with the lowest average is the NL-based one. This is a further confirmation that the mixed mode drastically improves a NL-based interaction by helping the user to be understood by the system. Overall, the NL interaction has lower results than the button and mixed interfaces.

Another interesting result is the answer to the question Q22. This answer has great importance in our experiment, since it is a thermometer of the capability of the system in understanding the preferences expressed through the conversation. Indeed, before asking this question, the system shows the profile of the user and thus asks her to verify its correspondence with the preferences expressed during the conversation. If the system did not correctly understand the user preferences, this question should have a negative answer. For the NL and button-based interactions, the median is *Strongly agree/Agree*, and for the mixed, interaction is *Strongly agree*. Also, in this case, the

[8] http://hci.epfl.ch/research-projects/resque/.

Table 3 Medians and averages of the answers to the questionnaire

Question	NL		Buttons		Mixed	
	Med	Avg	Med	Avg	Med	Avg
7. I became familiar with the recommender system very quickly	4	4.02	4	4.24	4	4.10
8. I feel in control of modifying my taste profile	4	3.93	4	4.06	4	4.07
9. I found it easy to tell the system what I like/dislike	4	3.85	4	4.17	4	4.02
10. I understood why the items were recommended to me	4	3.89	4	4.22	4	4.09
11. I will use this recommender again	4	3.70	4	3.94	4	3.80
12. It is easy for me to get a new set of recommendations	4	4.00	4	4.07	4	4.06
13. It is easy for me to inform the system if I like/dislike the recommended items	4	3.91	4	4.20	4	4.06
14. It is easy to learn to tell the system what I like	4	3.87	4	4.13	4	3.98
15. Overall, I am satisfied with the recommender	4	3.63	4	4.02	4	3.93
16. The items recommended to me match my interests	4	3.74	4	4.06	4	4.07
17. The recommender helped my find the ideal item (strongly matches my preferences)	4	3.67	4	3.81	4	3.85
18. The recommender adapts its behavior to my critiquing	4	3.93	4	3.96	4	4.02
19. It is easy for the recommender to understand what I said	4	3.52	4	4.00	4	3.87
20. I could fix misunderstandings if I wanted to	4	3.57	4	3.85	4	3.87
21. It is easy to understand how to dialogue so that the recommender should understand	4	3.85	4	4.09	4	4.00
22. The preferences in my profile correspond to the preferences expressed to the recommender	4.5	4.46	4.5	4.44	5	4.41

system effectively understood and acquired the user preferences, with a slightly better result for the mixed configuration.

We also performed some analyses in order to discover correlations between the characteristics of the subjects involved in the experiment (Q1–Q6) and the results related to the assessment of the system (Q7–Q22). (A). Generally, there are no significant differences among the groups. An interesting outcome is related to the experience as a computer user (Question 4). In terms of *cost of interaction*, and in particular for TPQ and IT, the beginners (13 out of 54) show results comparable to those achieved by the advanced (22 out of 54) and average (19 out of 54) users through the mixed interaction. Conversely, the difference is statistically significant with the button-based interaction, and beginner users show a larger TPQ compared to the other groups of users. This is a very interesting result since it demonstrates that a NL-based interface, even though it does not improve the accuracy of the recommender system, is actually more natural and facilitates users with little IT experience.

4.6 Answers to the research questions

Now, we can answer to the research questions.

- **RQ1: To what extent can conversational interfaces support the music recommendation?**
  Conversational interfaces are a valid alternative for a music recommender system. The interaction cost is drastically reduced through an interaction mode based on natural language.
- **RQ2: What is the best conversational interface in terms of cost of interaction?**
  The best conversational interface in terms of cost of interaction is based on natural language supported by buttons when the user has to choose among multiple options.
- **RQ3: What is the best conversational interface in terms of recommendation accuracy?**
  NL-based interface supported by buttons has the best performance in terms of accuracy.
- **RQ4: Is the disambiguation step particularly strenuous for the user of a conversational music recommender system?**
  A step that is particularly strenuous for the user of a CoRS is confirmed to be the disambiguation. When the user has to choose among a set of multiple options, buttons associated with the different options make this task easier.

5 Limitations and future work

In this paper, we proposed an investigation on the interaction modes of a conversational music recommender system. We guess that this study provides concrete answers in designing a conversational interface in a domain where the interaction is moving toward voice commands. Naturally, this work shows some limitations. More specifically, even though we demonstrated that the disambiguation is a stressful step for the user, we cannot give an answer on whether the disambiguation effectively impacts the recommendation accuracy. Furthermore, the single components of the interface (in particular for the button-based one) would be separately analyzed in order to verify the impact they have in the interaction. Finally, our study does not provide an answer to the utility of the explanation in a music recommender system, also because this function has been used by a small number of users. As regards the framework, we provide a solution that we demonstrated to work fine. However, we need to evaluate the impact of each component on the whole conversational recommendation process.

By summing up, a natural language interface drastically improves the interaction cost and the accuracy compared to a button-based interface. It is crucial that activities that require to choose among a set of predefined answers have to be supported by buttons. Indeed, the natural language-based interaction can make difficult some tasks that increase the interaction cost and, consequently, decrease the accuracy of the recommender. Another interesting outcome is that users who do not show a great experience in the use of computers are supported well by an interface based on natural language.

We plan to extend this model in order to verify whether these results are also confirmed in other domains and we will perform additional in vitro experiments. Furthermore, we want to test the single components on the dataset built during this work, in order to evaluate the impact of each of them on the whole conversational recommendation process.

Acknowledgements This work is partially funded by Project Electronic Shopping & Home delivery of Edible goods with Low environmental Footprint (ESHELF), under the Apulian INNONETWORK programme - Italy, and by OBJECTWAY SPA under the project Unified Wealt Management Platform. The authors also want to thank Andrea Iovine for the support given to this research.

A Questionnaire

See Table 4.

Table 4 Questionnaire

ID	Question	Answer
1	Please choose your gender	Male
		Female
2	Please select your age group	–20
		21–30
		31–40
		41–50
		51–60
		61–
3	Please choose your education level	Primary school
		High school
		College
		Graduate school
4	How would you rate yourself as a computer user?	No experience
		Beginner
		Average
		Advanced
5	Have you ever used a recommender system before?	Yes
		No
		Maybe
6	How much are you interested in music?	High
		Medium
		Low
7	I became familiar with the recommender system very quickly	5-point Likert scale:
		1-Strongly disagree …
		5-Strongly agree

Table 4 continued

ID	Question	Answer
8	I feel in control of modifying my taste profile	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
9	I found it easy to tell the system what I like/dislike	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
10	I understood why the items were recommended to me	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
11	I will use this recommender again	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
12	It is easy for me to get a new set of recommendations	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
13	It is easy for me to inform the system if I like/dislike the recommended items	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
14	It is easy to learn to tell the system what I like	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
15	Overall, I am satisfied with the recommender	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
16	The items recommended to me match my interests	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
17	The recommender helped me find the ideal item	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
18	The recommender adapts its behavior to my critiquing	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree or I didn't do any critiquing
19	It is easy for the recommender to understand what I said	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree

Table 4 continued

ID	Question	Answer
20	I could fix misunderstandings if I wanted to	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
21	It is easy to understand how to talk so that the recommender should understand	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree
22	The preferences in my profile correspond to the preferences expressed to the recommender	5-point Likert scale: 1-Strongly disagree … 5-Strongly agree

B Anova test results

See Tables 5, 6, and 7.

Table 5 Degrees of freedoms, F-score, p value, partial η^2 for each metric

Metric	df	F	p	η^2
MAP	2	3.06	0.049	0.04
Accuracy	2	2.5	0.086	0.03
Time per question	2	10.56	0	0.12
Interaction time	2	13.28	0	0.14
Query density	2	12.9	0	0.14
Number of questions	2	120.44	0	0.6

Table 6 Average value for each metric and interaction mode

Metric	Buttons	Mixed	NL
MAP	0.44	0.58	0.46
Accuracy	0.54	0.65	0.55
Time per question	1851.81	12364.06	19039.82
Interaction time	688407.41	318518.52	545870.37
Query density	0.77	0.72	0.57
Number of questions	18.39	2.63	4.52

Table 7 Standard deviation for each metric and interaction mode

Metric	Buttons	Mixed	NL
MAP	0.26	0.33	0.33
Accuracy	0.26	0.3	0.3
Time per question	14288.11	5881.33	9541.18
Interaction time	99864.25	415951.09	302935.89
Query density	0.29	0.13	0.16
Number of questions	9.29	2.08	3

References

Andjelkovic, I., Parra, D., ODonovan, J.: Moodplay: interactive music recommendation based on artists mood similarity. Int. J. Hum. Comput. Stud. **121**, 142–159 (2019)

Barthet, M., Fazekas, G., Allik, A., Sandler, M.: Moodplay: an interactive mood-based musical experience. In: Proceedings of the Audio Mostly 2015 on Interaction with Sound, p. 3. ACM (2015)

Basile, P., Caputo, A., Semeraro, G.: Uniba: combining distributional semantic models and sense distribution for multilingual all-words sense disambiguation and entity linking. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, Colorado, pp. 360–364 (2015)

Basile, P., Caputo, A., Semeraro, G., Narducci, F.: Uniba: exploiting a distributional semantic model for disambiguating and linking entities in tweets. Mak. Sense Microposts (# Microposts2015) (2015)

Basile, P., Greco, C., Suglia, A., Semeraro, G.: Deep learning and hierarchical reinforcement learning for modeling a conversational recommender system. Intell. Artif. **12**(2), 125–141 (2018)

Basile, P., Musto, C., de Gemmis, M., Lops, P., Narducci, F., Semeraro, G.: Aggregation strategies for linked open data-enabled recommender systems. In: 11th ESWC (2014)

Basile, P., Musto, C., de Gemmis, M., Lops, P., Narducci, F., Semeraro, G.: Content-based recommender systems + dbpedia knowledge=semantics-aware recommender systems. In: Semantic Web Evaluation Challenge, pp. 163–169. Springer (2014)

Bogdanov, D., Herrera, P.: Taking advantage of editorial metadata to recommend music. In: 9th International Symposium on Computer Music Modeling and Retrieval (CMMR), pp. 618–632 (2012)

Bostandjiev, S., O'Donovan, J., Höllerer, T.: Tasteweights: a visual interactive hybrid recommender system. In: Proceedings of the Sixth ACM Conference on Recommender Systems, pp. 35–42. ACM (2012)

Bostandjiev, S., O'Donovan, J., Höllerer, T.: Linkedvis: exploring social and semantic career recommendations. In: Proceedings of the 2013 International Conference on Intelligent User Interfaces, pp. 107–116. ACM (2013)

Bridge, D.G.: Towards conversational recommender systems: a dialogue grammar approach. In: ECCBR Workshops, pp. 9–22 (2002)

Christakopoulou, K., Beutel, A., Li, R., Jain, S., Chi, E.H.: Q&r: a two-stage approach toward interactive recommendation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 139–148. ACM (2018)

Christakopoulou, K., Radlinski, F., Hofmann, K.: Towards conversational recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 815–824. ACM (2016)

Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(Jan), 1–30 (2006)

Glass, J., Polifroni, J., Seneff, S., Zue, V.: Data collection and performance evaluation of spoken dialogue systems: the MIT experience. In: Proc. 6th Int., pp. 1–4 (2000)

Goker, M., Thompson, C.: The adaptive place advisor: a conversational recommendation system. In: Proceedings of the 8th German Workshop on Case Based Reasoning, Limmerbuckel, pp. 187–198 (2000)

Grasch, P., Felfernig, A., Reinfrank, F.: Recomment: towards critiquing-based recommendation with speech interaction. In: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pp. 157–164. ACM, New York (2013)

Green, S.J., Lamere, P., Alexander, J., Maillet, F., Kirk, S., Holt, J., Bourque, J., Mak, X.W.: Generating transparent, steerable recommendations from textual descriptions of items. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 281–284. ACM (2009)

Gretarsson, B., O'Donovan, J., Bostandjiev, S., Hall, C., Höllerer, T.: Smallworlds: visualizing social recommendations. In: Computer Graphics Forum, vol. 29, pp. 833–842. Wiley (2010)

Haveliwala, T.H.: Topic-sensitive PageRank: a context-sensitive ranking algorithm for Web search. IEEE Trans. Knowl. Data Eng. **15**(4), 784–796 (2003)

He, C., Parra, D., Verbert, K.: Interactive recommender systems: a survey of the state of the art and future research challenges and opportunities. Expert Syst. Appl. **56**, 9–27 (2016)

He, C., Parra, D., Verbert, K.: Interactive recommender systems: a survey of the state of the art and future research challenges and opportunities. Expert Syst. Appl. **56**, 9–27 (2016)

Jugovac, M., Jannach, D.: Interacting with recommenders—overview and research directions. ACM Trans. Interact. Intell. Syst. **7**(3), 10:1–10:46 (2017)

Konstan, J.A., Riedl, J.: Recommender systems: from algorithms to user experience. User Model. User Adapt. Interact. **22**(1–2), 101–123 (2012)

Laurier, C., Grivolla, J., Herrera, P.: Multimodal music mood classification using audio and lyrics. In: Seventh International Conference on Machine Learning and Applications, 2008. ICMLA'08, pp. 688–693. IEEE (2008)

Levy, M., Sandler, M.: Learning latent semantic models for music from social tags. J. New Music Res. **37**(2), 137–150 (2008)

Li, R., Kahou, S.E., Schulz, H., Michalski, V., Charlin, L., Pal, C.: Towards deep conversational recommendations. In: Advances in Neural Information Processing Systems, pp. 9748–9758 (2018)

Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, pp. 73–82. ACM (2009)

McFee, B., Barrington, L., Lanckriet, G.: Learning content similarity for music recommendation. IEEE Trans. Audio Speech Lang. Process. **20**(8), 2207–2218 (2012)

McFee, B., Lanckriet, G.R.: The natural language of playlists. ISMIR **11**, 537–542 (2011)

Musto, C., Narducci, F., de Gemmis, M., Lops, P., Semeraro, G.: A Tag Recommender System Exploiting User and Community Behavior. Recommender Systems and The Social Web, Menlo Park (2009)

Musto, C., Narducci, F., Lops, P., de Gemmis, M., Semeraro, G.: ExpLOD: a framework for explaining recommendations based on the linked open data cloud. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 151–154. ACM (2016)

Musto, C., Narducci, F., Lops, P., de Gemmis, M.: Combining collaborative and content-based techniques for tag recommendation. In: International Conference on Electronic Commerce and Web Technologies, pp. 13–23. Springer (2010)

Musto, C., Semeraro, G., Lops, P., de Gemmis, M., Narducci, F.: Leveraging social media sources to generate personalized music playlists. In: International Conference on Electronic Commerce and Web Technologies, pp. 112–123. Springer (2012)

Narducci, F., de Gemmis, M., Lops, P., Semeraro, G.: Improving the user experience with a conversational recommender system. In: International Conference of the Italian Association for Artificial Intelligence, pp. 528–538. Springer (2018)

Narducci, F., Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Leveraging encyclopedic knowledge for transparent and serendipitous user profiles. In: International Conference on User Modeling, Adaptation, and Personalization, pp. 350–352. Springer (2013)

Narducci, F., Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Exploiting big data for enhanced representations in content-based recommender systems. In: International Conference on Electronic Commerce and Web Technologies, pp. 182–193. Springer (2013)

Narducci, F., Palmonari, M., Semeraro, G.: Cross-language semantic retrieval and linking of e-gov services. In: International Semantic Web Conference, pp. 130–145. Springer (2013)

Nguyen, T.N., Ricci, F.: Dynamic elicitation of user preferences in a chat-based group recommender system. In: Proceedings of the Symposium on Applied Computing, SAC '17, pp. 1685–1692. ACM, New York (2017)

Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016)

Nickel, M., Rosasco, L., Poggio, T.A., et al.: Holographic embeddings of knowledge graphs. In: The Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), pp. 1955–1961 (2016)

O'Donovan, J., Smyth, B., Gretarsson, B., Bostandjiev, S., Höllerer, T.: Peerchooser: visual interactive recommendation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1085–1088. ACM (2008)
Pu, P., Chen, L., Hu, R.: A user-centric evaluation framework for recommender systems. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 157–164. ACM (2011)
Ricci, F., Del Missier, F.: Supporting travel decision making through personalized recommendation. In: Designing Personalized User Experiences in eCommerce, pp. 231–251. Springer (2004)
Saito, Y., Fukusako, T.: A novel low-profile electrically small meander line antenna for gain improvement. In: Korea–Japan Microwave Conference TH2, Cited by 2 (2011)
Schaffer, J., Höllerer, T., O'Donovan, J.: Hypothetical recommendation: a study of interactive profile manipulation behavior for recommender systems. In: FLAIRS Conference, pp. 507–512 (2015)
Schedl, M., Knees, P., Gouyon, F.: New paths in music recommender systems research. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, pp. 392–393. ACM, New York (2017)
Schedl, M., Knees, P., McFee, B., Bogdanov, D., Kaminskas, M.: Music recommender systems. In: Recommender Systems Handbook, pp. 453–492. Springer (2015)
Silvervarg, A., Jönsson, A.: Subjective and objective evaluation of conversational agents in learning environments for young teenagers. In: Proceedings of the 7th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems (2011)
Sloboda, J.A., ONeill, S.A.: Emotions in everyday listening to music. In: Music and Emotion: Theory and Research, pp. 415–429 (2001)
Suglia, A., Greco, C., Basile, P., Semeraro, G., Caputo, A.: An automatic procedure for generating datasets for conversational recommender systems. In: CLEF (Working Notes) (2017)
Sun, Y., Zhang, Y., Chen, Y., Jin, R.: Conversational recommendation system with unsupervised learning. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 397–398. ACM (2016)
Swearingen, K., Sinha, R.: Interaction design for recommender systems. Des. Interact. Syst. **6**, 312–334 (2002)
Wärnestål, P., Degerstedt, L., Jönsson, A.: Interview and delivery: dialogue strategies for conversational recommender systems. In: Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007), pp. 199–205 (2007)
Yi, J., Nasukawa, T., Bunescu, R., Niblack, W.: Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques. In: Third IEEE International Conference on Data Mining, pp. 427–434. IEEE (2003)
Zhang, Y., Chen, X., Ai, Q., Yang, L., Croft, W.B.: Towards conversational search and recommendation: system ask, user respond. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 177–186. ACM (2018)
Zhou, C., Jin, Y., Zhang, K., Yuan, J., Li, S., Wang, X.: Musicrobot: towards conversational context-aware music recommender system. In: International Conference on Database Systems for Advanced Applications, pp. 817–820. Springer (2018)

Fedelucio Narducci is a PostDoc Research Fellow at the Department of Computer Science, University of Bari Aldo Moro, Italy. He is also member of the SWAP (Semantic Web Access and Personalization) research group of University of Bari Aldo Moro. His primary research interests lie in the areas of machine learning, recommender systems, user modeling, and personalization. He completed his Ph.D. in 2012 with a dissertation on Knowledge-enriched Representations for Content-based Recommender Systems. From April 2012 to July 2014 he worked as a PostDoc Researcher at the university of Milano-Bicocca on the SMART (Services & Meta-services for smART eGovernment) project with a specific focus on cross-language information retrieval and filtering. He has published more than 60 papers and he is one of the authors of the textbook "Semantics in Adaptive and Personalized Systems: Methods, Tools and Applications", edited by Springer. He served as a reviewer and co-reviewer for international conferences and journals in the areas of AI, recommender systems, user modeling, and personalization.

Pierpaolo Basile is an Assistant Professor at the Department of Computer Science, University of Bari Aldo Moro, Italy. He received the PhD in Computer Science from the University of Bari in 2009 with

the dissertation "Word Sense Disambiguation for Intelligent Information Access". His research interests include natural language processing, machine learning and information retrieval. He was the principal investigator of the project "Multilingual Entity Linking" funded by the Apulia regional program Future In Research and he was a visiting researcher at the Alan Turing Institute. He authored several scientific papers and he gave a tutorial on "Linked Open Data for Semanticsaware Recommender Systems" at the ESWC 2016 conference. He is a member of the board of the Italian Association of Computational Linguistics.

Marco de Gemmis is Assistant Professor at the Department of Computer Science, University of Bari Aldo Moro, Italy, where he received his PhD in Computer Science in 2005. His primary research interests include content-based recommender systems, natural language processing, information retrieval, text mining, and personalized information filtering. He authored over 150 scientific articles published in international journals and collections, proceedings of international conferences and workshops, and book chapters. He was program committee member for international conferences, including: ACM Recommender Systems; User Modeling, Adaptation and Personalization (UMAP), and served as a reviewer for several international journals, including: User Modeling and User Adapted Interaction; ACM Transactions on Internet Technologies. He served as Doctoral Consortium Chair and Workshop Chair at UMAP and organized several international workshops. He was editor of the book "Emotions and Personality in Personalized Services - Models, Evaluation and Applications", Springer, 2016. He was invited speaker at the 1st ACM Summer School on Recommender Systems in Bolzano (2017) and at the Workshop on Semantics-Enabled Recommender Systems (IEEE ICDM), 2016.

Pasquale Lops is Associate Professor at the University of Bari, Italy. He received the Ph.D. in Computer Science from the University of Bari in 2005 with a dissertation on "Hybrid Recommendation Techniques based on User Profiles". His research interests include recommender systems and user modelling, with a specific focus on the adoption of techniques for semantic content representation. He authored over 200 scientific articles, and he is one of the authors of the textbook "Semantics in Adaptive and Personalized Systems: Methods, Tools and Applications", edited by Springer. He was Area Chair of User Modelling for Recommender Systems at UMAP 2016, and co-organized more than 20 workshops related to user modeling and recommender systems. He gave a tutorial on "Semantics-Aware Techniques for Social Media Analysis, User Modeling, and Recommender Systems" at UMAP 2016 and 2017, he was a speaker at two editions of the ACM Summer School on Recommender Systems. He was a keynote speaker at the 1st Workshop on New Trends in Content-based Recommender Systems (CBRecSys) at RecSys 2014. Finally, he gave the interview "Beyond TF-IDF" in the Coursera MOOC on Recommender Systems.

Giovanni Semeraro is full professor of computer science at University of Bari Aldo Moro, Italy, where he teaches "Intelligent Information Access and Natural Language Processing", and "Programming languages". He leads the Semantic Web Access and Personalization (SWAP) "Antonio Bello" research group. In 2015 he was selected for an IBM Faculty award on Cognitive Computing for the project "Deep Learning to boost Cognitive Question Answering". He was one of the founders of AILC (Italian Association for Computational Linguistics) and he was on the Board of Directors till 2018. From 2006 to 2011 he was on the Board of Directors of A|*|A (Italian Association for Artificial Intelligence). He has been a visiting scientist with the Department of Information and Computer Science, University of California at Irvine, in 1993. From 1989 to 1991 he was a researcher at Tecnopolis CSATA Novus Ortus, Bari, Italy. His research interests include machine learning; AI and language games; recommender systems; user modelling; intelligent information mining, retrieval, and filtering; semantics and social computing; natural language processing; the semantic web; personalization. He has been the principal investigator of University of Bari in several European, national, and regional projects. He is author of more than 400 publications in international journals, conference and workshop proceedings, as well as of 2 books, including the textbook "Semantics in Adaptive and Personalized Systems: Methods, Tools and Applications", under publication by Springer. He regularly serves in the PC of the top conferences in his areas and is Program Co-Chair of CLiC-it 2019. Among others, he served as Program Co-chair of CLiC-it 2016, ACM RecSys 2015 and as General Co-chair of UMAP 2013. From 2013, he is the coordinator of the 2nd Cycle Degree Program in Computer Science at University of Bari. He is the coordinator of the 1st edition of the Master in Data Science at University of Bari. He is a member of the Steering Committee of the National Laboratory of Artificial Intelligence and Intelligent Systems (AIIS) of the National Interuniversity Consortium for Informatics (CINI).