Decision Support Systems 131 (2020) 113250

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Conversational Recommender Systems and natural language: A study through the ConveRSE framework

Andrea Iovine[a], Fedelucio Narducci[b,*], Giovanni Semeraro[a]

[a] Department of Computer Science, University of Bari Aldo Moro, Via E. Orabona 4, Bari I-70125, Italy
[b] Politecnico di Bari, Via E. Orabona 4, Bari I-70125, Italy

ARTICLE INFO

Keywords:
Conversational Recommender Systems
Domain-independent recommender systems
Natural language processing

ABSTRACT

Digital Assistants (DA) such as Amazon Alexa, Siri, or Google Assistant are now gaining great diffusion, since they allow users to execute a wide range of actions through messages in natural language. Even though DAs are able to complete tasks such as sending texts, making phone calls, or playing songs, they do not yet implement recommendation facilities. In this paper, we investigate the combination of Digital Assistants and Conversational Recommender Systems (CoRSs) by designing and implementing a framework named ConveRSE (Conversational Recommender System framEwork), for building chatbots that can recommend items from different domains and interact with the user through natural language. Since a CoRS architecture is generally composed of different elements, we performed an in-vitro experiment with two synthetic datasets, to investigate the impact that each component has on the CoRS in terms of recommendation accuracy. Additionally, an in-vivo experiment was carried out to understand how natural language influences both the cost of interaction and recommendation accuracy of a CoRS. Experimental results have revealed the most critical components in a CoRS architecture, especially in cold-start situations, and the main issues of the natural-language-based interaction. All the dialogues have been collected in a public available dataset.

1. Introduction

Digital Assistants (DAs) are systems that can interact with users via natural language, assisting them in performing daily tasks, such as memorizing appointments, booking plane tickets or, more generally, finding information.

Digital assistants debuted on smartphones, and then shifted towards more specialized contexts, in which traditional user interfaces are not adequate, or completely out of the question (e.g. smart homes, cars). One of the most common uses of DAs is for finding new media products to enjoy. It is reasonable then to think that DAs are suitable platforms for providing recommendations through the integration of a Conversational Recommender System (CoRS). CoRSs are characterized by the capability of interacting with the user during the recommendation process [1]. CoRSs differ from traditional recommender systems because they do not require that the users provide all the information in one step. Instead, they guide them through an interactive, human-like dialog [2]. For this reason, they are suitable in contexts in which we cannot interact with a classical user interface, but we can dialog with the system using, for example, the voice.

In this paper, we propose a domain-independent, configurable framework for building CoRSs named ConveRSE (Conversational Recommender System framEwork). ConveRSE exploits different interaction mechanisms: natural language, buttons, and a combination of the two. We developed ConveRSE to understand how the use of natural language influences the accuracy and the quality of the user experience in a CoRS [3]. This investigation is conducted in different domains, in order to capture possible differences among them.

This paper aims to answer the following Research Questions:

- RQ1: Can natural language improve a Conversational Recommender System in terms of cost of interaction?
- RQ2: Can natural language improve a Conversational Recommender System in terms of quality of the recommendations?
- RQ3: What is the impact of each component of a Conversational Recommender System on the accuracy of the recommendations?
- RQ4: What are the most critical aspects to consider when modeling a natural language-based dialog for a Conversational Recommender System?

We conducted an in-vitro study on two synthetic datasets (bAbI Movie Dialog Dataset and ConvRecSysDataset) to evaluate the system's

* Corresponding author.
E-mail addresses: andrea.iovine@uniba.it (A. Iovine), fedelucio.narducci@poliba.it (F. Narducci), giovanni.semeraro@uniba.it (G. Semeraro).

https://doi.org/10.1016/j.dss.2020.113250
Received 17 June 2019; Received in revised form 14 January 2020; Accepted 14 January 2020
Available online 21 January 2020

overall recommendation accuracy and the impact caused by each component on said accuracy. We also evaluated the general user satisfaction level over different aspects by asking users to answer to a questionnaire derived from the ResQue model [4].

2. Relatedwork

Conversational Recommender Systems (CoRSs) provide a strategy for overcoming the limitations of traditional recommenders by changing the way the users input their preferences and receive the recommendations. Unlike traditional recommender interfaces, in which all the user data must be collected in advance, CoRSs make use of an interactive process that can span over multiple iterations. In He et al. [5] , the recommendation process of an interactive recommender system is described. CoRSs make the interaction more efficient and natural [6], as they are able to help user navigate a complex product space by iteratively selecting items for recommendation [7]. Recently, Rafailidis and Manolopoulos [8] discussed the idea of putting together Virtual Assistants and Recommender Systems, highlighting the gap between the two technologies. The authors sustain that a Virtual Assistant can improve the recommendation process because it can learn the users' evolving, diverse and multi-aspect preferences.

Digital Assistants can be seen as a particular case of Goal-Oriented Conversational Agent [9]. A Goal-Oriented CA has the objective of communicating through natural language to reach the user's objectives. In the case of the ConveRSE framework, the objective is recommending items that are interesting for the user. To develop a Virtual Assistant, or any other form of CA, two main architectures are available: *modular* and *end-to-end* [10]. Modular (or *pipeline-based*) Conversational Agents are made up of a series of modules, each with its own specific function [10-13]. These modules are connected to each other to form a pipeline. Usually, a modular system is composed of [11, 12]: a *Natural Language Understanding* component, that maps the utterances of the user in a semantic representation; a *Dialog State Tracker*, that manages an estimate of the state of the conversation; a *Dialog Policy*, that selects the appropriate action given the current dialog state; a *Natural Language Generator*, that generates the final response. An example of conversational agent that uses a modular architecture can be found in Rudnicky et al. [14]. Works on end-to-end systems started with Ritter et al. [15]. They then popularized thanks to the diffusion of Deep Learning techniques [10]. End-to-end systems replace modules with a single model that is directly trained on conversational data, and that handles all the steps: from understanding the user's message to generating a response. End-to-end systems are a promising solution, because they do not require the development of specific components, which makes them easier to port to a new domain. However, there are still some challenges to face in order to enable them to handle goal-oriented conversations [10]. For example, end-to-end systems require very large datasets for training which are not always easy to find. This is one of the reasons we decided to adopt a modular architecture for ConveRSE.

An example of CoRS in the movie domain is ACORN, presented in Warnestaal [6], which uses natural language dialog in order to acquire user preferences, update them, and present personalized item recommendations. The system can interview the user in order to understand their preferences, answer questions about the movies, and provide personalized and motivated recommendations during the dialog. A user study concluded that ACORN's dialog strategy allows for efficient dialogs and high user satisfaction. Our work extends this idea by measuring how natural language impacts the efficiency of the conversation and the accuracy of the recommendations in different domains. Goker and Thompson [16] and Jannach et al. [17] developed two CoRSs based on constraint-satisfaction search. Goker and Thompson [16] use a natural language interface, while Jannach et al. [17] do not. In fact, the authors believed that using natural language is detrimental because users may not be able to keep the conversation running. In our work, we investigate these claims, to determine the effect of natural language in terms of flexibility, efficiency, and effectiveness of a CoRS. Mori et al. [18] developed a movie recommender system with a natural language-based interface and compared two interfaces: *system-initiated*, that presents all the details of the recommended movie, and *user-initiated*, that only gives details if the user asks for them. A user experiment is conducted to evaluate the simplicity and completeness of the interaction. Our work focuses not only on the presentation of the details, but also on the profile acquisition step. We also compare the natural language interface with other interaction modes.

Lee and Choi [19] propose a study on the effect of self-disclosure and reciprocity in the interaction between users and a conversational agent that recommends movies. Self-disclosure is defined as " the process of revealing thoughts, opinions, emotions, or personal information to others" while reciprocation measures how much the disclosure is done in a bi-directional way. A user experiment concluded that users were satisfied more when both the self-disclosure and the reciprocity were high. Zhao et al. [20] introduced MOLI, a conversational agent for technical customer service. MOLI is able to solve customer questions by providing answers from a Knowledge Base. The system was tested against several baseline models and achieved a higher accuracy in giving answers. The natural language dialog model introduced in our framework is partly inspired by Moore [21] as it features a *recommendation sequence* that can be enriched by asking for details or explanation, and a *dialog state* that keeps track of the conversation. Vahidov et al. [22] present an investigation of the negotiation between humans and software agents. In that study, a human and a software agent interact to obtain an agreement. This is relevant to our study since the interaction between a user and a CoRS can definitely be seen as a negotiation task. However, our study focuses more on the exploitation of natural language for interacting with the agent, rather than the negotiation tactics.

Another peculiarity of a CoRS compared to a traditional recommender system is related to the evaluation metrics. Indeed, the metrics generally adopted for evaluating recommender systems, like Precision, Recall, *F*-measure, MAE, RMSE, can only determine whether a recommendation is relevant or not. However, they give no information on how much the recommendation is novel and useful to the user and how difficult it was to get that recommendation. For this reason, research has also focused the attention on user-centric approaches.

In Konstan and Reidl [23], the authors underline the necessity of going beyond the accuracy in terms of prediction and the importance of taking into account the user experience during the design of both algorithms and systems. In a study proposed in Swearingen and Sinha [24], the authors observed the users' interaction with 11 different recommender systems (movie, book, and music domains), and highlighted the most important aspects of the interaction. For example, in order to make a trustworthy recommender system, it must be able to explain to the users why the recommended items are suitable for them. In Pu et al. [25], a list of design guidelines for developing RSs that generate a good UX is presented. The authors explain that a good RS should minimize user effort, provide explanation interfaces, generate diverse recommendations, and give the user the opportunity to refine their preferences. In order to evaluate these aspects, Pu et al. [4] presented ResQue (Recommender Systems' Quality of User Experience). It is a framework for the evaluation of Recommender Systems from a user-centric perspective. ResQue does not only evaluate how much the recommendations match the interests of the users, but also tests how much users believe that the system is easy to use, how much the system satisfies users, and how much suggestions are trusted.

In summary, several works analyzed different aspects of CORSs. However, to the best of our knowledge, an extensive study on the impact of natural language on the UX and accuracy of a CoRS is not yet available in the literature. With this work, we aim to bridge this gap by highlighting the critical components in a CoRS architecture and the crucial steps in the interaction with the user. A preliminary evaluation

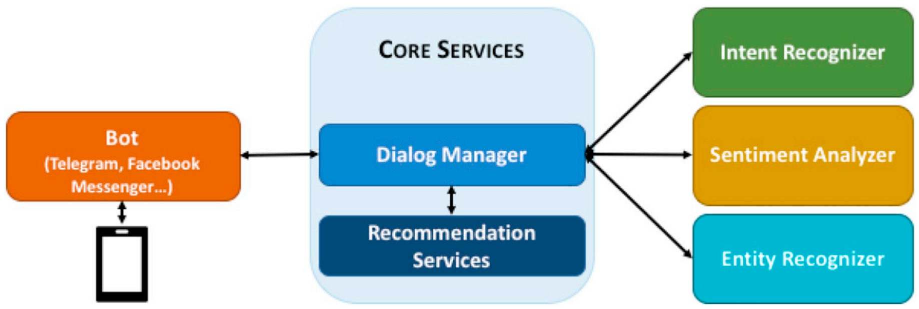


Fig. 1. The architecture of ConveRSE.

on a single domain and with a simpler interaction mode has been conducted in Refs. [26, 27] .

3. The architecture of the framework

The recommendation process performed by a recommender system follows some well-defined steps [28, 29]: preference acquisition, profile building, generation of a list of recommendations, and user evaluation of the recommendations. As said before, these steps do not follow a precise order in a CoRS. For example, users can ask for recommendations, express new preferences, and then they can ask for recommendations again.

In ConveRSE, the interaction follows a *slot-filling* model, where a set of features need to be filled in order to accomplish the user goal. To be more precise, the recommendation step requires that the user has already provided a minimal set of preferences. Accordingly, the implementation of a natural language-based CoRS requires at least five components: a Dialog Manager, an Intent Recognizer, a Sentiment Analyzer, an Entity Recognizer, and a set of Recommendation Services. Some of these components need to be designed and tailored to a specific domain. It is important to note that the components and the recommendation steps are not the same thing: the former identify different moments in the user-recommender interaction, while the latter represent different functions from an architectural point of view. For example, in order to perform the preference acquisition step, we need to invoke in the following order: Intent Recognizer, Entity Recognizer, and Sentiment Analyzer. The task of invoking the single components in the right order is in charge of the Dialog Manager. The running example in Fig. 2 helps to understand how the components are invoked according to the user sentence. The architecture of our framework is depicted in Fig. 1. These components are easily mapped with those of the modular architecture described in Williams et al. [12]: the Dialog Manager performs the functions of the Dialog State Tracker, Dialog Policy, and Natural Language Generator, while the Intent Recognizer, the Sentiment Analyzer and the Entity Recognizer assume the role of the Natural Language Understanding component.

Dialog Manager. This is the core component of ConveRSE, whose responsibility is to supervise the whole recommendation process. The *Dialog Manager* (DM) is the component that keeps track of the dialog state. It is the orchestrator of the system. The DM receives the user message, invokes the components needed for answering the request, and returns the final response to the user. This response can be either a question (e.g. when a disambiguation is needed, or a recommended entity is shown), or a feedback (e.g. when a preference is acquired). Responses are built by filling template messages with contextual information.

Intent Recognizer. This component has the goal of detecting the intent of the user. When the user sends a message, the recommender system must first understand what is the goal; this means recognizing the *intent* of the user. In our scenario, there are four main intents to be recognized:

- *preference:* The user is rating a new or recommended item;
- *recommendation:* The user is asking to receive a recommendation;
- *show profile:* The user is asking to look (and edit) the profile;
- *help:* The user is asking for instructions on how to continue the interaction.

Sentiment Analyzer. The Sentiment Analyzer (SA) is based on the Stanford CoreNLP Sentiment Tagger[1]. The Sentiment Tagger takes the user sentence as input and returns the identified sentiment tags. Afterwards, the SA assigns the sentiment tags to the entities identified in the sentence. For example, given the sentence *I like The Matrix, but I don't like Keanu Reeves*, the Sentiment Tagger identifies a positive sentiment (i.e. *like*) and a negative one (i.e. *don't like*). The SA associates the positive sentiment to the entity *The Matrix* and the negative sentiment to the entity *Keanu Reeves*. The sentiment-entity association is performed by computing the distance between the sentiment tag and the entity into the sentence. This distance is expressed 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 *don't like* and *Keanu Reeves*. The sentiment tag identified by CoreNLP is thus associated to the closest entity in the sentence.

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[2].

We chose to develop a custom Entity Recognizer for two reasons: 1) existing entity recognizer/linking algorithms are hard to customize to a specific domain; 2) entity recognizers included in existing Natural Language Understanding platforms 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. The task is challenging since a concept may have more than one surface form (e.g. *Barry Eugene Carter*, *Barry White* can both refer to *Barry_White:singer*) and the same surface form *Barry White* can refer to multiple concepts (i.e. *Barry_White:singer* and *Barry_White:songwriter*) 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.

Recommendation Services. This component handles the functions strictly related to the recommendation process. The *recommendation algorithm* implemented is the PageRank with Priors [30], also known as Personalized PageRank. It works on a graph where the nodes are the entities handled by the recommender (e.g., movies, actors, producers, writers). These entities are extracted from Wikidata. The algorithm has been effectively used in other recommendation environments [31].

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

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

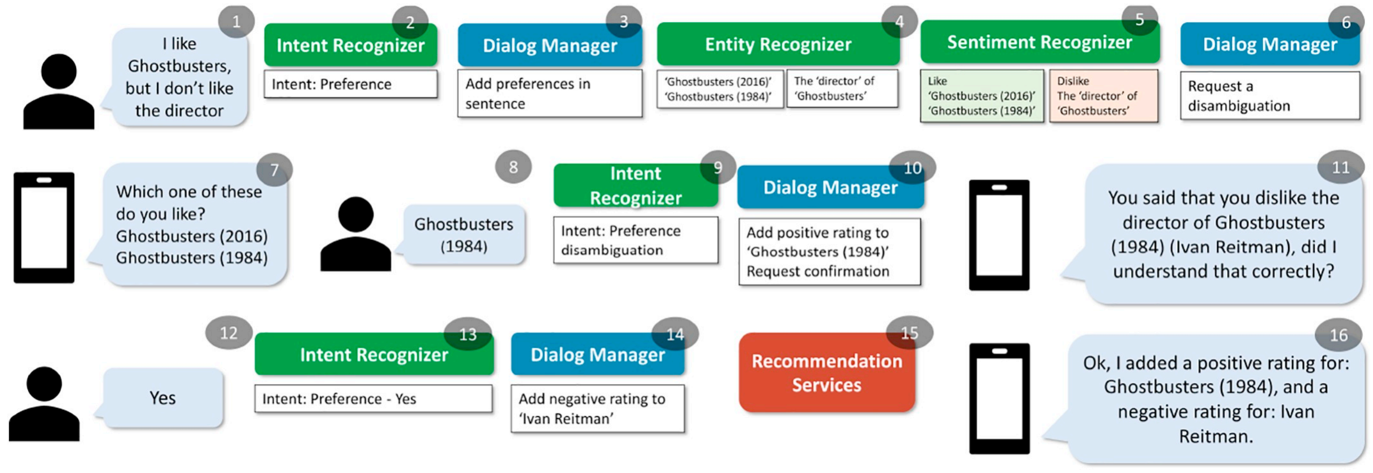

1
I like Ghostbusters, but I don't like the director
2
Intent Recognizer
Intent: Preference
3
Dialog Manager
Add preferences in sentence
4
Entity Recognizer
'Ghostbusters (2016)' 'Ghostbusters (1984)'
The 'director' of 'Ghostbusters'
5
Sentiment Recognizer
Like 'Ghostbusters (2016)' 'Ghostbusters (1984)'
Dislike The 'director' of 'Ghostbusters'
6
Dialog Manager
Request a disambiguation
7
Which one of these do you like? Ghostbusters (2016) Ghostbusters (1984)
8
Ghostbusters (1984)
9
Intent Recognizer
Intent: Preference disambiguation
10
Dialog Manager
Add positive rating to 'Ghostbusters (1984)' Request confirmation
11
You said that you dislike the director of Ghostbusters (1984) (Ivan Reitman), did I understand that correctly?
12
Yes
13
Intent Recognizer
Intent: Preference - Yes
14
Dialog Manager
Add negative rating to 'Ivan Reitman'
15
Recommendation Services
16
Ok, I added a positive rating for: Ghostbusters (1984), and a negative rating for: Ivan Reitman.

Fig. 2. The framework in action.

Another recommendation service offered by the framework is the *explanation* feature. The framework implements an explanation algorithm inspired by Musto et al. [32, 33]. 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 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 feature is *critiquing*. It allows the user to provide a valuable feedback on a recommended item and its properties (e.g. *I like this song (Rocket Man), but I don't like the producer (Gus Dudgeon)*). 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

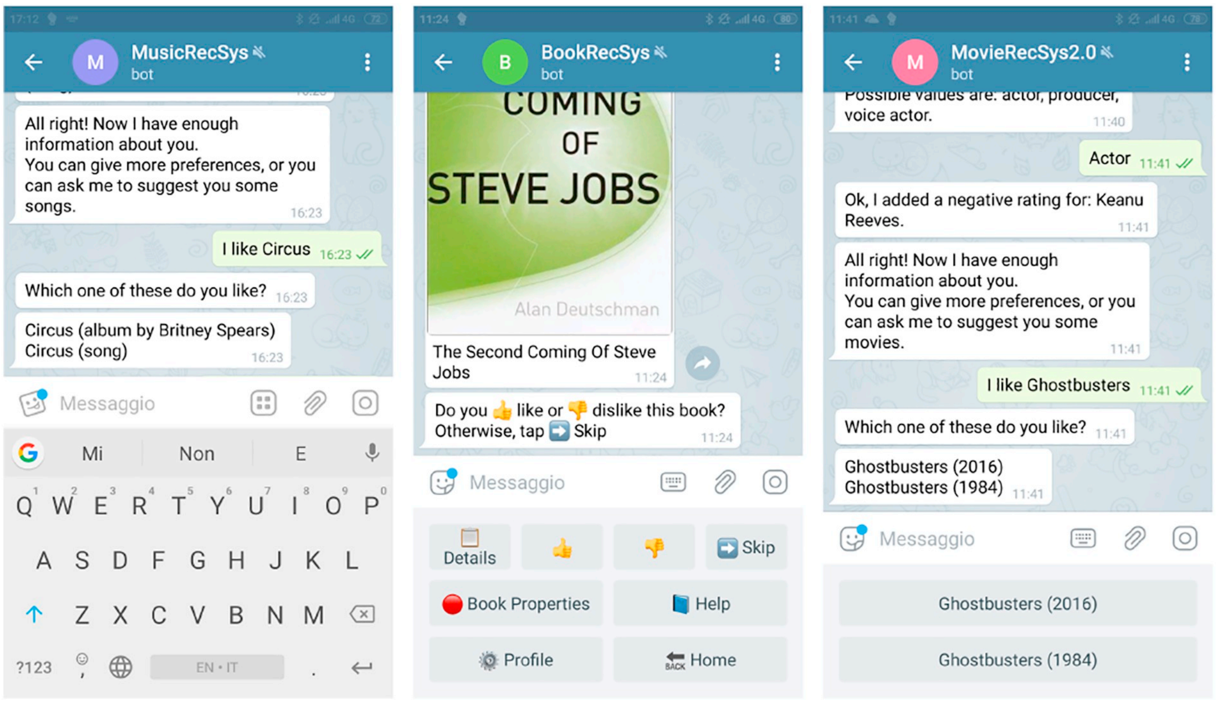
Fig. 2 proposes a running example in order to explain how each component effectively participates in the execution of the recommendation steps. For brevity reasons, the example will only cover the preference acquisition step, because it is the most complex step both in terms of interaction between components and in terms of user involvement. Let us suppose that the system receives the message *I like Ghostbusters, but I hate the director*. This sentence expresses both a positive and a negative preference. The positive preference is related to a movie (i.e. *Ghostbusters*), while the negative preference is on one of its properties (i.e. the director). After the user expresses a preference to the bot (step 1), the first system action is to identify the intent (step 2). The Intent Recognizer is invoked, which understands that the user wants to express a *preference*. Now, the Dialog Manager implements all the actions required to complete the job (step 3). First, it asks the Entity Recognizer to identify and link the entities in the sentence (step 4). Next, the Sentiment Analyzer associates a sentiment (positive or negative) to each retrieved entity (step 5). However, in our example, the retrieved entity *Ghostbusters* can either refer to the 1984 movie directed by Ivan Reitman or the 2016 movie directed by Paul Feig. Hence, the disambiguation process starts (step 6). The system asks the user which entity he/she refers to (step 7), and the answer (step 8) is passed to the Intent Recognizer which recognizes the new *preference-disambiguation* intent (step 9). Now, the Dialog Manager has to solve the last ambiguity before adding the preferences to the user profile (step 10). Indeed, the user expressed a negative preference for the director of the movie which is not explicitly referenced. Therefore, the system retrieves the entity the user refers to (i.e., *Ivan Reitman*) and asks the user to confirm (steps 11–12). Now, the Intent Recognizer recognizes the user's confirmation via the *preference-yes* intent (step 13) and the Dialog Manager can store the preferences in the user profile by invoking the Recommendation Services component (steps 14–15). We chose this case as it best shows how the system works since it features two preferences, and two different system prompts. It also introduces the main challenges that a natural language-based needs to overcome, e.g. disambiguating a preference.

3.2. Interaction modes

Our framework implements three different interaction modes: buttons, natural language, and mixed. Figs. 3 and 4 show some screenshots of the framework in action using each interaction mode: natural language mode (left), button-based mode (center), and mixed mode (right). Different domains are shown for each screenshot, to illustrate the configurability of ConveRSE.

Button-based interaction. The user interacts with the system by tapping buttons. This interface is derived from the system described in Narducci et al. [27], with several modifications applied in order to make it domain-independent. This interface is used mainly as the control group for the user study described in Section 4.2. The interaction is divided into two phases: a *preference acquisition* phase and a *recommendation* phase. During the preference acquisition phase, a new user can rate a series of popular items in order to build its own profile. For each proposed item, the user can directly like or dislike it (Fig. 3), or can rate its properties (e.g. for the movie domain, directors, actors, etc.). When the system has acquired enough information, the user can ask for recommendations, thus initiating the second phase. During this phase, the system presents a list of recommended items, that the user can either like, dislike, skip, or *criticize* (with the option "I like it, but… "). When this option is selected, the system shows the list of properties associated to the item and allows the user to rate one or more of them. The user can also ask for details about the item or for an explanation. When the list of recommendations is over, the user can start again, by giving more ratings, or getting a new set of recommendations. In this case, the dialog is completely driven by the system, which is in charge of asking all the information required to complete the recommendation. If the user wants to change the flow of the dialog (e.g., exit the recommendation phase), she needs to navigate the menu and choose the corresponding button.

Natural language-only interaction. Users interact with the system exclusively via text messages. The flow of the dialog does not follow a rigid path. At any time, the user can decide to start a new action. The system will recognize her intent and activate the necessary components. Like in the button-based interaction, we can distinguish a preference acquisition phase and a recommendation phase. However, in the first



All right! Now I have enough information about you.
You can give more preferences, or you can ask me to suggest you some songs. 16:23

I like Circus 16:23

Which one of these do you like? 16:23

Circus (album by Britney Spears)
Circus (song) 16:23

The Second Coming Of Steve Jobs 11:24

Do you 👍 like or 👎 dislike this book?
Otherwise, tap ➡️ Skip 11:24

Possible values are: actor, producer, voice actor. 11:40

Actor 11:41

Ok, I added a negative rating for: Keanu Reeves. 11:41

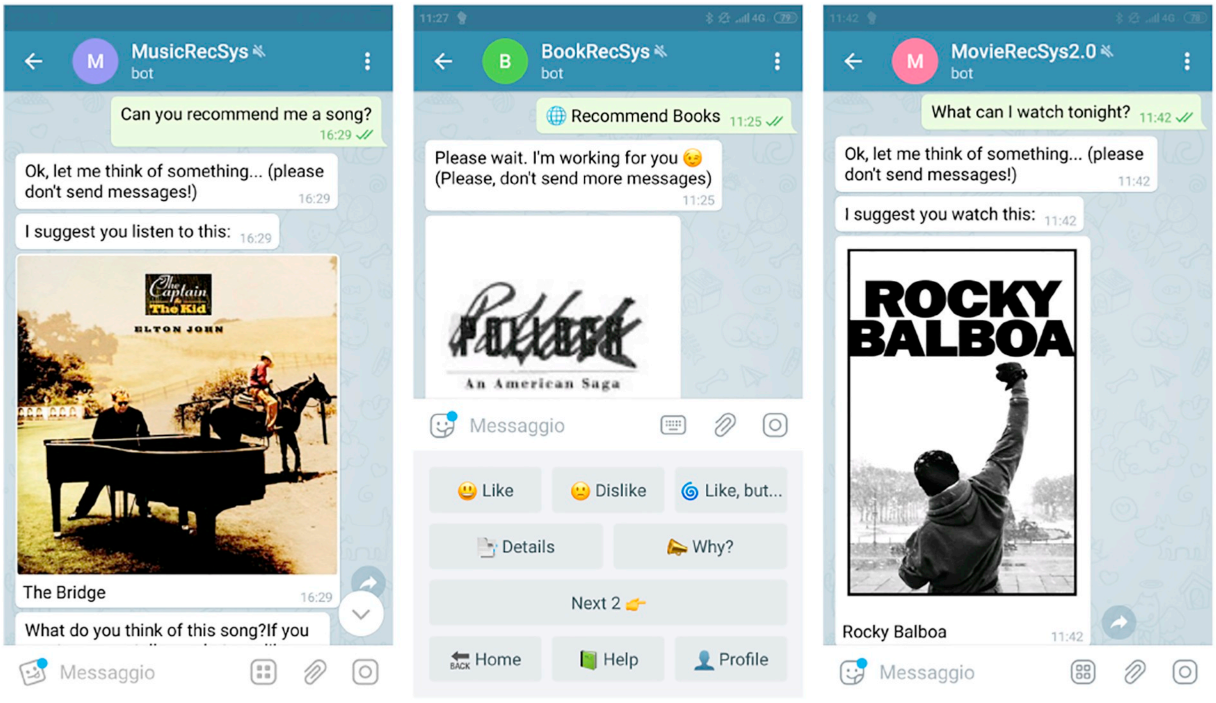
All right! Now I have enough information about you.
You can give more preferences, or you can ask me to suggest you some movies. 11:41

I like Ghostbusters 11:41

Which one of these do you like? 11:41

Ghostbusters (2016)
Ghostbusters (1984) 11:41

Fig. 3. Preference acquisition phase for the three interaction modes.



Can you recommend me a song? 16:29

Ok, let me think of something... (please don't send messages!) 16:29

I suggest you listen to this: 16:29

The Bridge 16:29

What do you think of this song?If you

Recommend Books 11:25

Please wait. I'm working for you 😉
(Please, don't send more messages) 11:25

What can I watch tonight? 11:42

Ok, let me think of something... (please don't send messages!) 11:42

I suggest you watch this: 11:42

Rocky Balboa 11:42

Fig. 4. Requesting a new recommendation in the three interaction modes.

phase, the user is free to provide her preferences, rather than evaluating items proposed by the system. The preferences are provided via natural language messages. Multiple preferences can be provided within a single message (e.g. step 1 in Figs. 2, and 3). 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. Ghostbusters (1984)). After enough information has been collected, the user can request a recommendation (Fig. 4) and the recommendation phase starts. The system will then propose a set of recommended items that the user can evaluate, skip, or criticize (e.g. "I like this book, but I don't like the genre", or "I don't like this song, but I love Michael Jackson"). Just like in the button-based interaction, the user can also request an explanation or more details about the recommended item.

Mixed interaction. The last interaction mode is basically the same as the previous one, except for the disambiguation stage. In this case, when a disambiguation is required (step 6 in Fig. 2), the system will show a button for each possible option. Hence, the user is not required to write the correct entity, but she will only tap the corresponding button.

The button-based interaction offers most of the functionalities of the natural language-based and mixed interactions. The main difference is due to the fact that the dialog is mainly system-driven in the former one, and more user-driven in the last two.

4. Experimentalevaluation

We organized two different experimental sessions. The first one is an in-vitro experiment on two synthetic datasets. The goal of this session is to evaluate the impact of each component of the architecture on the recommendation accuracy. The second session is a user study with the goal of assessing the user experience with the different interaction mode. The in-vitro experiment was developed to answer RQ3, while the user study was developed to answer RQ1, RQ2 and RQ4.

4.1. Session 1: in-vitro experiment

In this section, we describe the process used to evaluate each component or subsets of components of the framework, so that we can verify their impact on the recommendation process (e.g., intent, entity, and property recognition). The experiment was carried out on two synthetic datasets, which are the *bAbI Movie Dialog Dataset*[3] and *ConvRecSysDataset*[4]. For the sake of readability, in this section, we report only the most significant results. The complete results for the in-vitro experiment can be found here[5].

4.1.1. Description of the datasets

4.1.1.1. bAbI Movie Dialog Dataset. The bAbI dataset has been developed by Facebook Research with the objective of learning and evaluating automatic text comprehension and reasoning systems. It is divided into 20 tasks, each one devoted to the training of a particular text comprehension skill. We used the Movie Dialog Dataset part, in particular, the part dedicated to the recommendation sub-task. Each item in the dataset is composed of a list of movies that are liked by the user followed by a movie recommendation request, and finally, by the title of a recommended movie. For example:

```
U: I like Psycho, The Diving Bell and the Butterfly,
   Vertigo, Show Me Love, Raising Arizona, Rio Bravo, and
   La Femme Nikita.
U: Would you suggest a movie?
B: Let the Right One In
```

This particular dataset is divided into training, test, and dev (development) sets. The training set is composed of a million of instances, while test and dev sets are composed of 10,000 instances each. Due to the excessive size of the training set, we decided to use only the test and dev sets. We then filtered those sets by excluding the instances in which the recommended movie is not part of our movie recommender KB, and those in which none of the movies in the preference set were found in our KB. The test set was reduced to 6667 instances, while the dev set was reduced to 6733 instances.

4.1.1.2. ConvRecSysDataset. ConvRecSysDataset is a dataset made for training CoRSs in the movie domain. It was generated automatically from data extracted from two data sources: MovieLens1 M and MovieTweetings. The dataset is divided in two parts, one for each source, and both parts are divided into training, test, and validation sets. Each instance represents a whole conversation and each conversation contains all the messages exchanged between the user and the bot. For this test, we only considered the MovieTweetings portion of the dataset. Some modifications were required in order to fit the data into our framework. The resulting conversations were then composed only of user preferences, system recommendations, and user feedback. Each conversation was split into sessions, each of which contains one or two recommendation messages. The training set is composed of 2535 conversations, 24,578 sessions, and 26,691 individual recommendations. The testing set is composed of 2516 conversations, 14,461 sessions, and 15,166 recommendations. The validation set is composed of 2405 conversations, 10,730 sessions, and 11,132 recommendations. The following is an example of a session in ConvRecSysDataset. For the purposes of the experiment, only the user utterances (U:) were used.

```
B: Which movies do you like?
U: I love Lovelace, The Conjuring, The Green Hornet,
   Psycho
B: I suggest you Chasing Amy, Leaving Las Vegas, Teenage
   Mutant Ninja Turtles II: The Secret of the Ooze, The
   Apparition, The Omen. Do you like them?
U: I do not like them
B: What are your favourite actors?
U: I like Scott Mosier
B: I suggest you Chasing Amy, Leaving Las Vegas, Kick-Ass
   2, Dark Skies. Do you like them?
U: I love them
B: I am glad that you like them
```

4.1.2. Experimental procedure

The test was conducted through four system configurations:

Upper bound: For each dataset instance, all entity preferences were added programmatically to the system and a list of the first 100 recommended movies was returned.

Intent Recognizer Test: For each instance (for each session in ConvRecSysDataset), all entity preferences were added programmatically, but only if the Intent Recognition component correctly recognized the intents. In case of erroneous recognition, no recommendation was generated. Otherwise, we generated a list of the first 100 movies recommended by the system.

Entity Recognizer Test: For each instance, we added a preference for all the entities that were recognized by the ER component, forcing the correct sentiment for each one. We then generated a list of the first 100 movies recommended by the system.

Sentiment Recognizer Test: For each instance, we added the preferences to the entities with the recognized sentiment (or the correct rating if an entity was not recognized by the Entity Recognizer). We then generated a list of the first 100 recommended movies.

Hence, the goal of each configuration is to test each component of the system architecture independently from each other. The Upper bound is the best recommendation result ConverRSE is able to achieve on a given dataset since intents, entities, and sentiments are provided as input. Conversely, the other configurations individually evaluate each component. For example, for the Entity Recognizer Test given the sentence:

I love Lovelace, The Conjuring, The Green Hornet, Psycho

the system will receive as input the sentence as well as the following information:

`sentiment: positive, intent: preference`

while the Entity Recognizer will be used to identify the entities (Lovelace, The Conjuring, The Green Hornet, Psycho). The assumption made is that errors in the recognition of entities will negatively affect the recommendation accuracy. In this way, we are able evaluate the impact of the Entity Recognizer on the recommendation process.

[3] https://research.fb.com/downloads/babi/.

[4] https://github.com/swapUniba/ConvRecSysDataset.

[5] https://drive.google.com/open?id=106H24x_0CINfZQEqknULNoonh4FJECst.

Table 1
bAbI Dataset recommendation test results (hit rate loss relative to the upper bound).

	HR@5	HR@10	HR@20	HR@50	HR@100
Upper Bound	1.00%	1.59%	2.50%	4.46%	6.97%
	Loss@5	**Loss@10**	**Loss@20**	**Loss@50**	**Loss@100**
Intent Test	19.40%	24.30%	23.21%	**23.00%**	**21.11%**
Entity Test	**58.21%**	**39.25%**	**32.74%**	14.67%	11.94%
Sentiment Test	4.48%	4.67%	8.93%	5.67%	5.54%

4.1.3. Metrics of the in-vitro experiment

In order to test the recommendation accuracy, we calculated the percentage of *hits* (correct recommendations) in the list of recommended movie (*HitRate*). Hence, the *HitRate@k* metric describes the rate of correct recommendations within the first k results from the system. We calculated the loss of *HitRate@k* relative to the upper bound, to determine the impact of each component to the overall recommendation accuracy. During the evaluation, the values used for k were 5, 10, 20, 50, and 100.

4.1.4. bAbI Movie Dialog Dataset results

Table 1 reports the recommendation accuracy for the bAbI dataset. For the sake of readability, we report only the results on the dev set, since also the results on the test dataset shows the same pattern.

The recommendation task on this dataset is very difficult: for each dialog there is only one recommended item (true positive), but we do not know whether other items in the list could be liked by the user. Another limit is the low quantity of information on which each recommendation is based: in some cases the system had to provide a recommendation with less than three preferences, thus working in extreme cold-start conditions. For $k = 5$, the component with the most negative impact on the recommendation accuracy is the Entity Recognizer, for which we observe a 58.21% loss. The ER component has a ~ 85% accuracy, which means that there is a ~ 15% error in recognizing entities. This error is then responsible for the large loss in the recommendation accuracy. In second place, we find the Intent Recognizer, with a 19.4% loss. The Sentiment Recognizer is the component with the least impact on the recommendation accuracy for which we report ~ 4.5% loss. This loss is then justified by a ~ 17% error in recognizing sentiments. For $k = 100$, we can see that the Intent Recognizer reports the highest loss (21.11%). This result mirrors the Intent Recognizer accuracy which is around 77%. Due to how the Intent Recognizer test was performed (see Section 4.1.2), a recommendation was generated only if both intents were correctly recognized. It makes sense then that the accuracy is around 77% of that of the upper bound.

From these results, we can gather that the Entity Recognition task is crucial in order to ensure that the first recommendation results are relevant, as small errors in recognizing entities cause great loss in recommendation accuracy. This becomes even more apparent when the number of user preferences is limited. The Sentiment Recognition task is also important, but less critical to the final result. By increasing the value of k the Intent Recognizer has a stronger impact than the other components. This is due to the fact that recognizing the wrong intent results in not generating a recommendation. Thus, this component does not benefit from the increased probability (with a large k value) of recommending the right item by chance.

Table 2
ConvRecSysDataset recommendation test results (hit rate loss relative to the upper bound).

	HR@5	HR@10	HR@20	HR@50	HR@100
Upper Bound	14.22%	17.82%	21.34%	25.57%	28.26%
	Loss@5	**Loss@10**	**Loss@20**	**Loss@50**	**Loss@100**
Intent Test	0.03%	0.02%	0.06%	0.10%	0.12%
Entity Test	1.88%	2.14%	1.77%	1.80%	2.28%
Sentiment Test	**4.07%**	**3.77%**	**3.86%**	**3.89%**	**4.07%**

Table 2 contains the results of the test on ConvRecSysDataset. Also in this case, we decided to show only the results obtained from the validation set since the trend is the same in all three sets. ConvRecSysDataset shows definitely better results than the ones measured on the bAbI dataset (Table 2): the Upper Bound measures a minimum of 14% to a maximum of 31%. This happens because each session provides several preferences and this means that the recommendation engine can work better. The impact of each component to the recommendation accuracy is also lower. In this case the Sentiment Recognizer component causes the greatest loss, with ~3–4% loss. This loss is in turn caused by a ~ 7% sentiment recognition error. A plausible explanation for this change is that in this dataset the ratings are much more complicated: unlike bAbI in which all ratings were positive, in ConvRecSysDataset each sentence can contain both positive and negative ratings which obviously makes the Sentiment Recognition task more challenging. The Entity Recognizer drops to second position, with a ~ 2% maximum loss over all sets, caused by a ~ 4% recognition error. Lastly, the Intent Recognizer produces the smallest impact, with a maximum loss of ~ 0.12%, caused by an error of around 1–2% in the intent recognition task. This is in part due to the fact that in this test the IR component had to recognize only one intent (preference) rather than two (preference and recommendation) as in the bAbI dataset. When the user is able to provide more complex preferences, recognizing the correct rating for each entity is just as important as correctly recognizing the entities themselves. These results are consistent also when increasing the k value.

4.1.5. Discussion

From the analysis of the in-vitro results, we can assume that the Entity Recognition is a crucial step in order to ensure that the first recommendation results are relevant, as small errors in recognizing entities cause great loss in recommendation accuracy. This becomes even more apparent when the number of user preferences is limited, i.e. in cold-start situations. The Sentiment Recognition task is also important, but less critical to the final result.

4.2. Session 2: user study

In this section, we describe the experimental evaluation we carried out with real users. We designed three separate within-subject experiments. The goal of the experiments was to compare different interaction modes, in order to assess their impact on the recommendation accuracy and interaction cost of a CoRS. Each experiment was tested on different domains, such as movie, book, and music. We decided to use these three domains since they are the most used for testing recommender systems. The movie recommender system test involved a group of 50 participants (medium–high interest in movies = 90.31%). The book recommender system involved a group of 55 users (medium–high interest in books = 74.55%). The music recommender system involved a group of 54 users (medium–high interest in music = 92.59%). Each subject tested the system using each interaction mode (see Section 3.2).

Before starting the experiment, users were able to use the system and learn the main functions. This helps reduce the ordering effect of the different configurations. During the experiment, ConveRSE performs the four steps defined in Section 3: i) preference acquisition, ii) profile building, iii) generation of a list of recommendations, iv) and user evaluation of the recommendations. The main differences between the interaction modes are found in steps i) and iv). During step iv), the user is presented with 5 recommended items. He/she can directly accept or reject each recommended item, or can provide more detailed feedback (triggering the critiquing strategy). We assumed that a list of 5 recommended items was a right compromise between the significance of the results and user effort. It is really important to consider that users are allowed to ask for a new set of recommendations, if they want. At the end of the experiment, the users answered a questionnaire based on the ResQue model [4]. In order to carry out the experimental evaluation, we built three instances of ConveRSE, suited to recommend items in the movie, book, and music domains. The knowledge base for the movie-based instance contains 15,954 movies and 41,704 properties, for the book-based instance contains 7592 books and 7172 properties, and for the music-based instance contains 12,926 songs and 27,029 properties.

4.2.1. Metrics of the user study

In this work, we adopted objective metrics (i.e. independent from the user's point of view), combined with a questionnaire that elicitates the subjective experiences of the users. The aim of the metrics is to evaluate the accuracy of the system and the efficiency of the interaction.

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

Mean Average Precision (MAP): The average precision of the recommendation lists, calculated by taking into account the items liked by the user and their ranking. The MAP is based on the Average Precision (AP), which is computed as follows:

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

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

Interpretation: The MAP is used to evaluate the quality of the ranking of the recommended items. The higher the MAP, the better the ranking.

Accuracy: The ratio between liked items and recommended items.

Interpretation: The higher the accuracy, the higher is the percentage of recommended items that were liked by the users.

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.

Interpretation: Usually, the more questions the user needs to answer, the less efficient the interaction is.

Time Per Question (TPQ): The total time required by the user to answer a question by the system. The time is calculated from the time stamp of the message sent to the user.

Interpretation: It is useful to analyze the effort required by the user for understanding and answering questions. A lower TPQ is better, as it means that the user is able to respond quicker to a system prompt.

Interaction Time (IT): The time required for completing the experiment. It is calculated from the first message sent to the user, to the start of the questionnaire administration.

Interpretation: A lower IT is better, since it means that the user was able to find a satisfactory item in less time.

Query Density (QD): An adaptation of the Query Density proposed in Glass et al. [35]. It measures the amount of concepts the user can introduce for each message. In this experiment, a concept is an entity (e.g. movies, artists) or a property (e.g. music genres). The QD is computed by the following formula:

$$QD = \frac{1}{N_d} \sum_{i=1}^{N_d} \frac{N_u(i)}{N_m(i)}, \tag{2}$$

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 addition of a new preference.

Interpretation: The higher the QD, the better, as it means that the user is able to provide more information in a dialog turn.

4.2.2. Results

Table 3 summarizes the results of the *interaction-cost* metrics. The first observation is related to the NQ. A system should be as effective as it reduces the number of questions, preserving a good recommendation accuracy. We clearly see that the interaction based on buttons asks a higher number of questions compared to the NL and mixed interaction modes. This result is probably due to the fact that an interaction based only on buttons is completely driven by the system, thus it requires much more questions in order to obtain the information needed for generating recommendations.

Looking at the TPQ and IT measures, we can see that the NL interaction generally requires a longer time than the button-based interaction, while the mixed interaction shows a lower (or comparable) time than both NL and button-based interactions in all the three domains. Since the only difference between the NL and mixed modes is in the disambiguation phase, this suggests us that the most cumbersome part of the NL interaction is indeed the disambiguation. Pressing a button rather than typing the full entity name makes this process easier, and once this obstacle is removed, the mixed interaction helps users complete the task in less time than the button-based interface. As regards the QD, when the user cannot express a preference with a single sentence (i.e. when a disambiguation is needed), the QD value will be penalized (see Formula (2)). We can see that in the book and music domains the mixed interaction mode performs better than the pure NL one, which may be due to the fact that it makes disambiguation easier. Our expectation that the NL and mixed modes would produce a higher QD than the button-based interaction was not met. It can be explained by the fact that the users might have not taken advantage of the capability to express more than one preference in a single sentence (e.g. *"I like The Doors and Pink Floyd"*).

Table 4 reports the results for the accuracy metrics. All the metrics show best results for the mixed mode. Since all the configurations share the same recommendation algorithm, the only motivation for this outcome is that the user can express her preferences more effectively through the mixed interaction. The recommendation lists generated in the mixed interaction are better both in terms of accuracy and ranking.

4.2.3. Statistical tests

In order to understand whether there are significant differences between the three interaction modes, we conducted a MANOVA statistical test (p-value < 0.05) on all the metrics. The null hypotheses of the test are of the form $H_{0,d}$: *The multivariate means of the interaction modes are equal in domain d, for d* $\in$*{Movie,Book,Music}*. The MANOVA concluded that the null hypothesis was rejected in all domains:

	p	Pillai's Trace	F	num Df	den Df	Critical value
Movie	< 0.001	0.76	15.27	12	300	1.78
Book	< 0.001	0.90	21.75	12	316	1.78
Music	< 0.001	0.88	20.44	12	310	1.78

We then proceeded to analyze in depth these differences by performing a one-tailed Mann-Whitney Test (p-value < 0.05) for each pair of configurations, and for each metric. The hypotheses of the Mann-Whitney test are of the form $H_{0,m,I_1,I_2,d}$: There is no difference between the values of m for the I_1 and I_2 interaction modes in domain d, for m $\in${*NQ,IT,TPQ,QD,Accuracy,MAP*}, I_1,I_2 $\in${*NL,Buttons,Mixed*}, d $\in${*Movie,Book,Music*}. In order to counter the effect of multiple comparisons, Bonferroni correction has been applied [36]. The complete results of the Mann-Whitney statistical tests are available here[6]. The main outcome of the test is that, even though we might guess that an

[6] https://drive.google.com/open?id=1ch-RHlLmS2LAAMaYhCtIPud86IHO3rHR.

Table 3
Results for the interaction cost metrics (best results for each metric are in bold).

	Movie			Book			Music		
	NL	**Buttons**	**Mixed**	**NL**	**Buttons**	**Mixed**	**NL**	**Buttons**	**Mixed**
Number of Questions (NQ)	2.86	24.49	**2.8**	2.09	15.25	**1.29**	4.52	18.39	**2.63**
Interaction Time (sec) (IT)	681.69	659.78	**444.7**	437.44	272.35	**237.69**	545.87	688.41	**318.52**
Time per Question (sec) (TQ)	28.35	**14.06**	16.24	17.13	**10.2**	**10.2**	19.04	24.28	**12.36**
Query Density (QD)	**0.84**	0.74	0.75	0.65	0.75	**0.78**	0.57	**0.77**	0.72

Table 4
Results for the accuracy metrics (best results for each metric are in bold).

	Movie			Book			Music		
	NL	**Buttons**	**Mixed**	**NL**	**Buttons**	**Mixed**	**NL**	**Buttons**	**Mixed**
Accuracy	0.45	0.56	**0.63**	0.57	0.46	**0.69**	0.55	0.54	**0.65**
Mean Average Precision	0.36	0.43	**0.55**	0.52	0.4	**0.69**	0.46	0.44	**0.58**

interaction completely based on natural language can facilitate the user of a Conversational Recommender System, the best solution is actually a combination of NL and buttons. Indeed, the mixed interaction is significantly better than NL and buttons in terms of interaction cost. Therefore, a NL-based interface needs to be supported by traditional interactive elements for more complex tasks (e.g. the disambiguation). Developing an interaction completely guided by NL requires great effort, whereas a button-driven approach is much less difficult to implement. However, this effort is not always repaid by an increase in the recommendation performance. In some cases, in fact, it may even result in a decrease. For each metric in which the NL interface performs worse than the button-driven interface, the mixed mode closes the gap (resulting in no significant difference), or even surpasses both (resulting in a significant improvement in performance). This means that the mixed interaction mode that was developed during this experiment does not decrease performance in the worst case, and instead, increases performance in the best case. Also, the fact that this observation is valid in all three domains gives more significance to our statement. Regarding the accuracy metrics, the mixed interaction is significantly better than NL and buttons for most of the domains.

4.2.4. *Questionnaire*

At the end of the experiment, the system proposes a questionnaire to the user. This questionnaire is a combination of the short version of the ResQue model[7] and the questionnaire proposed in [37]. It can be found here[8]. The questions are organized into constructs, which are derived from Pu et al. [4]. The constructs are: *Perceived Ease of Use, Control, Interaction Adequacy, Transparency, Use Intentions, Overall Satisfaction, Recommendation Accuracy*, and *Perceived Usefulness*. The Likert scales are represented by their numerical value, with 1 signifying “Strongly Disagree”, and 5 signifying “Strongly Agree”. A median of 4.5 means that there is a tie between “Agree” and “Strongly Agree” answers. For Question 18 (*The recommender adapts its behaviour to my critiquing*), we ignored users that chose “I didn’t do any critiquing”. Medians and averages of the answers to the questionnaire can be found here[9]. For all the questions, the median is the “Agree” value, thus the users are generally satisfied with the interaction with the system. To understand whether there are differences between the answers for each interaction mode, we conducted a MANOVA statistical test. *The null hypotheses of the test are of the form $H_{0,d}$: The multivariate means of the answers to the questionnaire are equal for all three interaction modes in domain d, for d ∈{Movie,Book,Music}*. The multivariate test did not prove any significant difference in the answers:

	p	Pillai's Trace	*F*	num Df	den Df	Critical value
Movie	0.20	0.20	1.21	32	348	1.48
Book	0.52	0.19	0.97	32	296	1.48
Music	0.94	0.13	0.64	32	290	1.48

By analyzing the individual questions, the most interesting result is the answer to Q22 (*The preference in my profile correspond to the preferences expressed to the recommender*). This answer has great importance in our experiment since it is an indicator of the system's ability to understand the preferences expressed through the conversation. Before asking this question, the system shows the user's profile, to let him/her verify its correspondence with the preferences expressed during the conversation. If the system did not correctly understand the preferences, this question should have a negative answer. Overall, the answers to this question range from Agree to Strongly agree, which means that all three interaction modes were able to understand the users' preferences correctly, with only minor differences between the interaction modes.

4.2.5. *In-depth analysis and interpretations*

After the collection of the results, we decided to do a more in-depth investigation of the conversation logs, in order to obtain further insights. We categorized users in terms of experience in using computers (Question 4 on the questionnaire), and interest in the relative domain (Question 6 on the questionnaire). With respect to the level of expertise, we also divided the users by interest in the domain. In all three domains, we observed that the mixed interaction greatly reduces the IT and TPQ metrics compared to the NL-only mode, and the gap between them is particularly great for low-interest users. Since the only difference between the two modes is in the disambiguation phase, a possible reason is that this task is particularly burdensome for users that are less knowledgeable about the entities of that domain. Even though the user is presented with a list of possible answers, just the action of typing the correct one is much more demanding and error-prone for this kind of users.

Another analysis was conducted by calculating the Pearson coefficients for each pair of metrics, in order to discover noteworthy relations

[7] http://hci.epfl.ch/research-projects/resque/.
[8] https://drive.google.com/open?id=1HclB65iiheU9ZMKdJwhNJLDE0WVemNXQTw1IihnuAvc.
[9] https://drive.google.com/open?id=1rWS3PW8vIxpjNm2HbiA_zDpqVUCzXcVX.

between them. The book and music domains show a weak to moderate negative correlation between QD and IT (ranging from -0.29 to -0.46) for the NL and mixed interactions. In both cases, the correlation is slightly stronger than the one measured in the button-based interaction (-0.25 for books and -0.15 for music). Even though the ability to give multiple preferences per message has been underutilized during the user study, we can speculate that the NL-based interfaces allow to increase the QD and this makes the interaction more efficient in terms of IT and this explains the aforementioned correlation. However, further experiments are needed to confirm this hypothesis.

4.3. Answers to the Research Questions

Now, we try to answer the Research Questions formulated in Section 1:

RQ1: Can natural language improve a Conversational Recommender System in terms of cost of interaction?

A pure natural language-based interface poses several obstacles that hinder the efficiency of the interaction. What we gathered from the experiment is that one of these obstacles appears when a precise input is needed. By assisting the interface in these weak points, natural language can definitely make the interaction more efficient.

RQ2: Can natural language improve a Conversational Recommender System in terms of quality of the recommendations?

The interaction based only on natural language did not perform consistently better than the button-based mode. The mixed interaction mode, on the other hand, never performed worse than the other two, and mostly performed better.

RQ3: What is the impact of each component of a Conversational Recommender System to the accuracy of the recommendations?

The accuracy in recognizing both entities and ratings has great effect on the quality of the recommendation, which was expected. More interesting is the fact that this effect is less noticeable the more preferences are added in the system. This means that the recognition accuracy is more crucial in cold-start conditions.

RQ4: What are the most critical aspects to consider when modeling a natural language-based dialog for a CoRS?

The first step is to identify the most stressful activities for the user. Indeed, even though NL makes the interaction more natural, it can introduce some difficulties for accomplishing particular tasks. In the context of a CoRS, choosing among a predefined set of options is definitely an example of such task. This is a frequent step in the recommendation task. Therefore, in domains where users are asked to answer to a set of predefined answers, this activity needs to be facilitated.

5. A dataset with real dialogs for CoRSs

During the experiment, we collected all the dialogs between the users and the system for all three domains. We collected ~6500 user messages for the NL interaction and ~2800 for the mixed interaction [38]. For each message, we collected the user ID, the message, the timestamps, and the recognized intent. We also collected all the recognized entities with the corresponding sentiment. For privacy reasons, the message logs are anonymized, and the user IDs are replaced with a numerical index. The dataset is in JSON format and it is available here[10]. To the best of our knowledge, this is the first dataset based on real data in the context of conversational movie, music, and book recommender systems.

[10] https://drive.google.com/drive/folders/1rIOeQUkoi4S6VRvfQ_M04zQfWjxe8Xbf?usp=sharing.

6. Conclusion

In this paper, we proposed ConveRSE, a framework for the development of Conversational Recommender Systems. We conducted both *in-vitro* and *in-vivo* experiments to evaluate it. Even though it might be intuitive that a dialog in natural language can improve the general user experience of a recommender system, the results of the user study show that pure NL interfaces need to be supported by more traditional interaction strategies. More specifically, when the user has to choose among a set of possible options, the integration of buttons drastically improves the interaction. This is also a valuable result also in the design of voice-based systems, for example by integrating a touch screen to support some operations. There is reason to believe that one of the main advantages of both natural language-based modes is the ability to provide personalized preferences. Even though providing custom preferences is not exclusive to the natural language interface, this could be an effective way to make the user profile acquisition task more efficient. Consequently, we suggest to expand research on the effects of a chat-based interface to the acquisition of the user requirements.

The experiments conducted on the synthetic datasets, described in Section 4.1, demonstrated that entity and sentiment recognition play a crucial role, in particular when few preferences have been expressed by the user. In this situation, it is imperative to understand what the user said as precisely as possible, even if this means asking for additional feedback. As future research, further analysis is required to understand the effectiveness of the components that make up the architecture of ConveRSE in each domain.

We also need to focus on isolating each aspect of the interaction and run an experiment that measures the effect of each one. Such a test would involve at least two factors: input mode (natural language/mixed/buttons), and initiative (user initiative — the user provides the preferences, system initiative — the system provides the items to rate). There is also much room for improvement in the dialog model: by further streamlining and optimizing the conversation (especially by improving the recognition accuracy, and reducing disambiguation requests), we believe that a lower interaction cost can be achieved, while preserving a good accuracy. We also need to investigate the extension towards a voice-based interaction. Even though it requires only the integration of an additional component (i.e. a speech synthesizer/recognizer), it might have a strong impact on the interaction and recommendation accuracy.

CRediT authorship contribution statement

Andrea Iovine: Software, Conceptualization, Methodology, Validation, Investigation, Writing - original draft, Writing - review & editing, Resources, Data curation, Visualization. **Fedelucio Narducci:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Supervision. **Giovanni Semeraro:** Funding acquisition, Project administration, Supervision, Conceptualization, Methodology, Investigation, Writing - review & editing.

References

[1] T. Mahmood, F. Ricci, Improving recommender systems with adaptive conversational strategies, Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, ACM, 2009, pp. 73–82.
[2] M. Jugovac, D. Jannach, Interacting with recommenders: overview and research directions, ACM Transactions on Intelligent Systems 7 (3) (2017) 10:1–10:46.
[3] S. Moller, K.-P. Engelbrecht, C. Kuhnel, I. Wechsung, B. Weiss, A taxonomy of quality of service and quality of experience of multimodal human-machine interaction, Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on, IEEE, 2009, pp. 7–12.
[4] P. Pu, L. Chen, R. Hu, A user-centric evaluation framework for recommender systems, Proceedings of the Fifth ACM conference on Recommender systems, ACM, 2011, pp. 157–164.
[5] C. He, D. Parra, K. Verbert, 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.
[6] P. Wärnestål, User evaluation of a conversational recommender system, Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2005, pp. 32–39.
[7] B. Smyth, L. McGinty, An analysis of feedback strategies in conversational recommenders, the Fourteenth Irish Artificial Intelligence and Cognitive Science Conference (AICS 2003), Citeseer, 2003.
[8] D. Rafailidis, The Technological Gap Between Virtual Assistants and Recommendation Systems, arXiv preprint arXiv:1901.00431 (2018).
[9] A. Bordes, Y.-L. Boureau, J. Weston, Learning End-to-End Goal-Oriented Dialog, arXiv:1605.07683 cs, (2016) arXiv: 1605.07683.
[10] J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, J. Weston, Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems, arXiv preprint arXiv:1511.06931 (2015).
[11] T. Zhao, M. Eskenazi, Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning, (2016) arXiv:1606.02560 [cs].
[12] J. Williams, A. Raux, M. Henderson, The dialog state tracking challenge series: a review, Dialogue & Discourse 7 (3) (2016) 4–33.
[13] B. Liu, I. Lane, An End-to-End Trainable Neural Network Model with Belief Tracking for Task-Oriented Dialog, Interspeech 2017 2017, pp. 2506–2510 arXiv:1708.05956.
[14] A.I. Rudnicky, E. Thayer, Paul Constantinides, C. Tchou, R. Shern, Kevin Lenzo, W. Xu, A. Oh, Creating Natural Dialogs in the Carnegie Mellon Communicator System, (1999).
[15] A. Ritter, C. Cherry, W.B. Dolan, Data-driven response generation in social media, Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011, pp. 583–593.
[16] M. Goker, C. Thompson, The adaptive place advisor: a conversational recommendation system, Proceedings of the 8th German Workshop on Case Based Reasoning, Citeseer, 2000, pp. 187–198.
[17] D. Jannach, G. Kreutler, Rapid development of knowledge-based conversational recommender applications with advisor suite, Journal of Web Engineering 6 (2007) 165–192.
[18] H. Mori, Y. Chiba, T. Nose, A. Ito, Dialog-Based Interactive Movie Recommendation: Comparison of Dialog Strategies, 82 Springer International Publishing, 2018, pp. 77–83.
[19] S. Lee, J. Choi, Enhancing user experience with conversational agent for movie recommendation: effects of self-disclosure and reciprocity, International Journal of Human-Computer Studies 103 (2017) 95–105.
[20] G. Zhao, J. Zhao, Y. Li, C. Alt, R. Schwarzenberg, L. Hennig, S. Schaffer, S. Schmeier, C. Hu, F. Xu, MOLI: smart conversation agent for mobile customer service, Information 10 (2019) 63.
[21] R.J. Moore, A natural conversation framework for conversational UX design, Studies in Conversational UX Design, Springer, 2018, pp. 181–204.
[22] R. Vahidov, G. Kersten, R. Saade, An experimental study of software agent negotiations with humans, Decision Support Systems 66 (2014) 135–145.
[23] J.A. Konstan, J. Riedl, Recommender systems: from algorithms to user experience, User Modeling and User-Adapted Interaction 22 (2012) 101–123.
[24] K. Swearingen, R. Sinha, Interaction design for recommender systems, Designing Interactive Systems, 6 2002, pp. 312–334.
[25] P. Pu, L. Chen, R. Hu, Evaluating recommender systems from the users perspective: survey of the state of the art, User Modeling and User-Adapted Interaction 22 (2012) 317–355.
[26] F. Narducci, B. Pierpaolo, M. de Gemmis, P. Lops, G. Semeraro, An investigation on the user interaction modes of conversational recommender systems for the music domain, User Model and User-Adapted Interaction (2019).
[27] F. Narducci, M. de Gemmis, P. Lops, G. Semeraro, Improving the user experience with a conversational recommender system, AI*IA 2018 - Advances in Artificial Intelligence - XVIIth International Conference of the Italian Association for Artificial Intelligence, 2018, pp. 528–538, , https://doi.org/10.1007/978-3-030-03840-3_39 Trento, Italy, November 20–23, 2018, Proceedings.
[28] L. Chen, P. Pu, Critiquing-based recommenders: survey and emerging trends, User Modeling and User-Adapted Interaction 22 (2012) 125–150.
[29] F. Ricci, Q.N. Nguyen, Acquiring and revising preferences in a critique-based mobile recommender system, IEEE Intelligent Systems 22 (2007) 22–29.
[30] T.H. Haveliwala, Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search, IEEE Transactions on Knowledge and Data Engineering 15 (2003) 784–796.
[31] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, G. Semeraro, Content-based recommender systems + DBpedia knowledge = semantics-aware recommender systems, Semantic Web Evaluation Challenge, Springer, 2014, pp. 163–169.
[32] C. Musto, F. Narducci, P. Lops, M. De Gemmis, G. Semeraro, ExpLOD: a framework for explaining recommendations based on the linked open data cloud, Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 151–154.
[33] C. Musto, F. Narducci, P. Lops, M. de Gemmis, G. Semeraro, Linked open data-based explanations for transparent recommender systems, International Journal of Human Computer Studies 121 (2019) 93–107.
[35] J. Glass, J. Polifroni, S. Seneff, V. Zue, Data collection and performance evaluation of spoken dialogue systems: the MIT experience, Sixth International Conference on Spoken Language Processing, 2000.
[36] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (2006) 1–30.
[37] A. Silvervarg, A. Jönsson, Subjective and objective evaluation of conversational agents in learning environments for young teenagers, Proceedings of the 7th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2011.
[38] A. Iovine, F. Narducci, M. de Gemmis, A dataset of real dialogues for conversational recommender systems, Proceedings of the Sixth Italian Conference on Computational Linguistics, 2019 Bari, Italy, November 13–15, 2019.

Andrea Iovine is a Ph.D. student 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. During his Ph.D., he is investigating several aspects related to Conversational Recommender Systems and Dialog Agents in general, under the supervision of Prof. Giovanni Semeraro and Dr. Fedelucio Narducci. His primary research interests lie in the areas of natural language processing, dialog agents, conversational recommender systems, user modeling, and personalization.

Fedelucio Narducci is Assistant Professor at Politecnico di Bari, 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 served as a reviewer and co-reviewer for international conferences and journals in the areas of AI, recommender systems, user modeling, and personalization.

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 on the Board of Directors till 2018. From 2006 to 2011 he was on the Board of Directors of AI*IA (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) and of the Steering Committee of the ACM Conference Series on Recommender Systems.