Linked Open Data-based Explanations for Transparent Recommender Systems

Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco de Gemmis, Giovanni Semeraro

PII: S1071-5819(18)30094-6
DOI: 10.1016/j.ijhcs.2018.03.003
Reference: YIJHC 2193

To appear in: *International Journal of Human-Computer Studies*

Received date: 15 May 2017
Revised date: 9 March 2018
Accepted date: 15 March 2018

Highlights

- We design the main components of an algorithm-agnostic framework to generate natural language explanations;
- We propose a methodology to extract descriptive (direct and indirect) properties about the items, and we use these properties to feed a graph-based explanation model;
- We define a scoring function to rank these explanation patterns and we use the most relevant ones to generate a template-based natural language explanation;
- We validate our methodology by carrying out a large user study (N=680) in three different domains, as movies, books and music;
- We integrate our methodology in a conversational recommender system implemented as a Telegram Bot.

Linked Open Data-based Explanations for Transparent Recommender Systems

Cataldo Musto, Fedelucio Narducci,
Pasquale Lops, Marco de Gemmis, Giovanni Semeraro

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

Abstract

In this article we propose a framework that generates *natural language explanations* supporting the suggestions generated by a recommendation algorithm.

The cornerstone of our approach is the usage of Linked Open Data (LOD) for explanation aims. Indeed, the descriptive properties freely available in the LOD cloud (e.g., the *author* of a book or the *director* of a movie) can be used to build a graph that connects the recommendations the user received to the items she previously liked via the properties extracted from the LOD cloud. In a nutshell, our approach is based on the insight that properties describing the items the user previously liked as well as the suggestions she received can be effectively used *to explain* the recommendations.

Such a framework is both *algorithm-independent* and *domain-independent*, thus it can generate a natural language explanation for every kind of recommendation algorithm, and it can be used to explain a single recommendation (*Top-1* scenario) as well as a group of recommendations (*Top-N* scenario). It is worth noting that the algorithm-independent characteristic does not mean that the framework is able to explain to the user how the recommendations have been generated and how the recommendation algorithm works. The framework explains to users why they might like the recommended items, independently from the recommendation algorithm that generated the recommendations.

In the experimental evaluation, we carried out a user study (N=680) aiming to investigate the effectiveness of our framework in three different domains, as *movies*, *books* and *music*. Results showed that our technique leads to transparent explanations for all the domains, and such explanations resulted independent of the specific recommendation algorithm in most of the experimental settings. Moreover, we also showed the goodness of our strategy when an entire group of recommendations has to be explained.

As a case study, we integrated the framework in a real-world application, a *conversational recommender system* implemented as a Telegram Bot. The idea is to use the explanation for supporting both the training phase (when the user expresses her preferences) and the recommendation step (when the user receives the recommendations). Interesting outcomes emerge from these preliminary experiments.

Email address: `name.surname@uniba.it` (Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco de Gemmis, Giovanni Semeraro)

1. Introduction

Recommender systems (RS) are intelligent systems which typically acquire users' needs, interests and preferences and tailor their behavior by personalizing users' experience and by supporting people in several decision-making tasks [23]. These systems proved to have a strategic role on consumers' habits, since many people use them to buy products on *Amazon*, to listen to music on *Spotify*, to choose the restaurants picked up by *Foursquare* or even to read the posts *Facebook* has ranked at the top of their feed. Moreover, as reported by several articles [31], recommender systems proved to have a significant impact on both sales volumes and click-trough rates. As an example, 35% of Amazon's revenues are generated through its recommendation engine[1], and many companies frequently report claims that RS contribute from 10% to 30% of total revenues [16].

Unfortunately, most of the researches in the literature are focused on how to improve the performance of a recommender system in terms of its capability of predicting items the user would probably like. Accordingly, evaluation metrics usually reward systems that maximize the predictive accuracy and neglect the user experience, even though some researchers recently pointed this aspect out [14].

This is a common problem for most of the state-of-the-art techniques. As an example, latent factor models and matrix factorization [28] are very accurate methods which learn a set of latent factors modeling users' preferences and item characteristics, and use this representation to predict the most relevant items for the user. However, such latent information is hidden in the data, thus it is very complex to explain *why* a certain item is recommended. A recent attempt [46] tried to explain recommendations generated through latent factor models, but the results are still very preliminary. Similarly, deep learning techniques [30] are based on complex neural networks whose internal representation exploits several hidden layers. Even if these techniques obtained very good results also in RS-related tasks [9], such representations are very difficult to understand and interpret. To sum up, the recent advances in RS research are facing a sharp dichotomy between the need for *effective* and precise recommendation techniques and the development of *transparent* algorithms.

In this work, we tackle the problem of providing a justification for a recommended item by proposing an *algorithm-agnostic* framework able to generate *natural language explanations* supporting the suggestions provided by a generic recommendation algorithm. Our explanations strongly rely on the information available in the Linked Open Data (LOD) [6] cloud, a huge set of interconnected semantic datasets which encode in `RDF` format the information covering many topical domains, such as people, books, scientific publications, films, music, and so on. The *nucleus* of such data is commonly represented by `DBpedia` [1], the `RDF` mapping of Wikipedia. Therefore, the *algorithm-agnostic* characteristic is related to the capability of justifying recommendations generated, for example, through a collaborative algorithm, even though the explanation is based on content features associated to the items. Our effort is thus to discover a *similarity* between the user preferences (e.g., items in the past) and the generated recommendations, even when the

[1]http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

recommendation algorithm does not take into account content features (e.g., the director of a movie, the music composer, the writer of a book, etc.).

In a nutshell, our approach works in four steps. First, all the available items are uniquely *mapped* to a `URI` in the LOD cloud, in order to gather properties describing the items. Next, we build a graph-based representation connecting the items the user *liked* with those in her *recommendation set* via the properties previously extracted. In our representation we model both direct relationships between the items (e.g., sharing the same *director*) as well as indirect relationships connecting them (e.g., two different actors sharing the same nationality). Finally, the explanation patterns connecting the items are ranked, and the most relevant ones are exploited to fill in a template which is presented to the user as *explanation* of the recommendation she received. Our framework can be exploited to generate an explanation for a *single* recommendation as well as for an entire *group* of recommendations, without any effort.

In the experimental sessions we extensively tested the framework by carrying out a user study in three different domains as *movies, books* and *music*, aiming at evaluating how transparent, trustful and engaging our explanations are. We also evaluated to what extent our framework was able to build effective explanations *regardless of the specific recommendation algorithm* used to generate the suggestions, and the results provided us with encouraging findings. Moreover, the experiment also showed that the users actually enjoyed the ability of the framework of aggregating several recommendations in a single explanation.

Given the positive impact of the experimental results, we integrated the explanation framework in a COnversational Recommender System (CORS) implemented as a Telegram Bot. There is a renewed interest of the RS research community on conversational assistants, capable of interacting with the users during the recommendation process [35] to lead them towards items of interest. In this context explanations play a crucial role, in order to have more and more *transparent recommender systems*, able to provide effective user-recommender interactions, besides the classical accuracy.

To sum up, in this paper we provide the following contributions:

- We design the main components of an *algorithm-agnostic* framework to generate *natural language explanations*;
- We propose a methodology to extract descriptive (direct and indirect) properties about the items, and we use these properties to feed a graph-based explanation model;
- We define a scoring function to rank these explanation patterns and we use the most relevant ones to generate a template-based natural language explanation;
- We validate our methodology by carrying out a user study involving 680 users in three different domains, as movies, books and music;
- We integrate our methodology in a conversational recommender system implemented as a Telegram Bot.

The paper is organized as follows. Section 2 discusses the related work in the area and emphasizes the distinguishing aspects of this work. In Section 3 we thoroughly describe the main components of our framework, while in Section 5 we describe the implementation of a Telegram Bot which integrates explanation functions. In Section 4 all the details about the experimental evaluation as well as the discussion of the results are provided. Finally, in Section 6 we draw some conclusions and sketch the future directions of our research.

2. Related Work

Most of the researches in the information filtering and retrieval areas are principally focused on the improvement of the system accuracy by exploiting various strategies such as the introduction of semantics [41, 48, 32, 43, 49, 33], the definition of new models for generating the recommendations [4, 34, 47], the combination of different paradigms for recommending or retrieving relevant items [29, 2]. Although less extensively, the problem of improving the user interaction with machine-learning systems has been investigated in several researches. If the user is able to *communicate* with machine learning systems, the users' understanding and trust of the system could improve and the system accuracy could be improved as well [50]. This enriched user experience, in the particular case of RS, can be obtained by allowing the user to give feedback on the received recommendations in order to improve the next recommendation cycles, and at the same time, by making the recommendation process more transparent to the user. The first goal can be pursued by designing systems which are able to acquire the user feedback and to adapt their behavior consequently; the second goal can be achieved by explaining to the user the motivation behind a given recommendation.

The importance of providing information systems with *explanation* facilities was established in several researches [15, 27, 37] already in the early 90s [24]. However, the first attempt towards the exploitation of such facilities in RS was proposed in 2000 by Herlocker et al. [20], who presented a user study based on 21 different explanation interfaces aiming at investigating how different types of interfaces impact on the users' acceptance of the recommendations. More recently, Gedikli et al. [15] have extended this study and compared their explanation technique based on the definition of a *personalized tag cloud*, with a subset of Herlocker's explanation styles [20] on different explanation goals. These explanation goals are inspired by the work of Tintarev and Masthoff [53], who defined a set of seven possible aims for explanation, namely: *transparency, scrutability, trust, effectiveness, persuasiveness, efficiency, satisfaction.* Pu and Chen [45] define an *organization interface* where recommendations are grouped according to their tradeoff properties. An example of explanation produced is: *Here are laptops that are cheaper and lighter but with a slower processor.* The authors demonstrated that this kind of interface increases the users' trust in the RS and their intention to return to the RS.

Differently from such researches, we focused more on the algorithmic aspects of our model, and we only investigated the role of the user interface as a case study. Specifically, we aimed at developing *an algorithm-agnostic explanation framework* totally independent of the underlying recommendation model. According to the taxonomy of explanation strategies provided by Friedrich et al. [13], our technique can be classified as a *black box* methodology. On the other side, *white box* methodologies are also aware of the underlying recommendation model and exploit this information to generate the explanations.

The development of a *black box* explanation algorithm is one of the distinguishing aspects of this work. Indeed, most of the works presented in the literature fall into the other category. As an example, in [5, 20] information about the neighborhood to justify the recommendation generated by the system are used. In that direction, in the information retrieval domain, Coyle and Smyth [11] exploit the search histories of a community of online users as a source of text explanations. In that work, most of the users viewed the additional explanation as having a positive impact on the search results. Cleger et al. [10] use neighbors' opinions about items previously rated by the user to learn a regression model from the given explanations when items are recommended. This model is used to change the recommendation for a target item.

Other attempts provided explanations on the ground of the knowledge model encoded in a knowledge-based RS [22]. As regards content-based approaches, Symeonidis et al. [51] proposed a model to generate explanations which exploits the overlap between the features of the profile of the target user and the features describing the suggestion. In [38], opinionated explanations were proposed. The idea is to generate rich and compelling explanations mined from user-generated reviews. In [8], the system provided a tag-based explanation where tags were used as intermediary entities to relate target users to the recommended items and to understand users' intents as well. Another content-based explanation approach is provided in [12], where the model lists properties the recommended artworks had in common with artworks the user had previously rated positively. The authors established that explaining why a recommendation was provided increases its acceptance.

Even if the main insight of those researches is common to our framework, i.e. explanations based on the overlapping properties describing the recommendations and items liked by a user, our methodology acts in an *algorithm-independent way*, since the graph-based data model underlying the explanation can be provided on the ground of the suggestions generated by any recommendation algorithm. The use of graphs for explanation-related tasks was also investigated by Knijnenburg et al. [25], who demonstrated that a graph-based explanation results in a better user experience. However, in this case the authors focused on the usage of graphs for visualization purposes. An approach based on graph for explaining collaborative filtering recommendations is presented in [21]: the graph connects rated items with recommendations, and the system is able to generate explanations like *The user has rated the film 'Taxi Driver' with high value.* In our approach the graph connections are item properties, and the framework generates more complex explanations based on the item properties liked by the user.

One of the most interesting *black box* explanation techniques is due to Vig et al. [54], which used *tags* to generate explanations. Specifically, they provided explanations by combining the *tag preference* (how likely the target user is interested in a certain tag) with *tag relevance* (how likely a tag describes an item). Even if this technique produces algorithm-independent explanations as our algorithm does, the novelty of our work with respect to both the previous work and [51] lies in the inclusion of non-overlapping properties able to generalize the explanation by also considering indirect relationships. Up to our knowledge this was never investigated in literature.

Another distinguishing aspect of our work lies on the exploitation of the properties coming from the LOD cloud. Even if the usefulness of injecting the information gathered from knowledge graphs to improve the accuracy of recommendations is taken for granted [39, 44, 42, 3], their adoption for explanation-related tasks is a poorly investigated research line. The only similar attempt was due to Wang et al. [55], who exploited properties encoded in RDF to support artwork recommendation. Differently from that work, our model is based on the whole `DBpedia` and can provide explanations for every domain covered by Wikipedia, provided that some descriptive properties about the items are available in the LOD cloud.

Tintarev and Masthoff [52] state that justifying recommendations to the user is half of the solution towards a more transparent system. The second half is making the system *scrutable* by allowing the user to make changes. This goal is reached by conversational systems which allow users to elaborate their requirements trough a dialog. Indeed, a conversational approach generally improves the interaction time required for obtaining a useful recommendation and can have a positive impact on the efficiency of the RS. Even tough in this work we mainly focused on the evaluation of the explanation framework, we preliminarily tested how explanation facilities are used and accepted by users in a conversational RS implemented as Telegram Bot.

A recent research tried to explain matrix factorization recommendation algorithms [19]. The aspect of interpretability is addressed by enabling the detection of overlapping co-clusters that can be easily visualized and transcribed into a textual description.

An interesting model for generating natural language explanations in the movie domain has been recently proposed in [7]. The personalized explanations take into account the user past activities (e.g. ratings, clicks, purchases), and exploit relevant quotes extracted from user reviews. Compared to our work, this model is strictly dependent on the human activity since the the relevant quotes have been synthesized into explanations by crowdworkers.

Finally, this paper continues our research in the area of explanation [40]. Differently from our previous work, we extended our model by introducing *indirect* explanation patterns and by defining more dynamic explanation templates. Moreover, we also further validate the outcomes of our study by evaluating the framework in different domains and by providing explanations based on the suggestions generated by different recommendation algorithms as well.

3. Description of the Explanation Framework

Figure 1 shows the general architecture of our framework to generate natural language explanations. In a nutshell, our system works as a *black box* which takes as input a *profile* (that is to say, a set of items the user previously liked) and a *set of recommendations*, and returns as output a *natural language explanation*, which is built regardless the specific recommendation algorithm which generated the recommendations.

The framework is actually split in four main building blocks. In the following, we will describe the characteristics of each component.

3.1. Mapper

The `Mapper` is the first module involved in the workflow. Its goal is to create an *entry point* to the information available in the LOD cloud (and specifically to the data which are available in `DBpedia`) to feed our model with the features necessary to generate the explanation.

Generally speaking, the goal of the mapping procedure is to identify, for each item in the u*ser profile* or in the *recommendation list*, its corresponding `URI` in the LOD cloud. As an example, we associate the book *The Shining* with its corresponding resource in

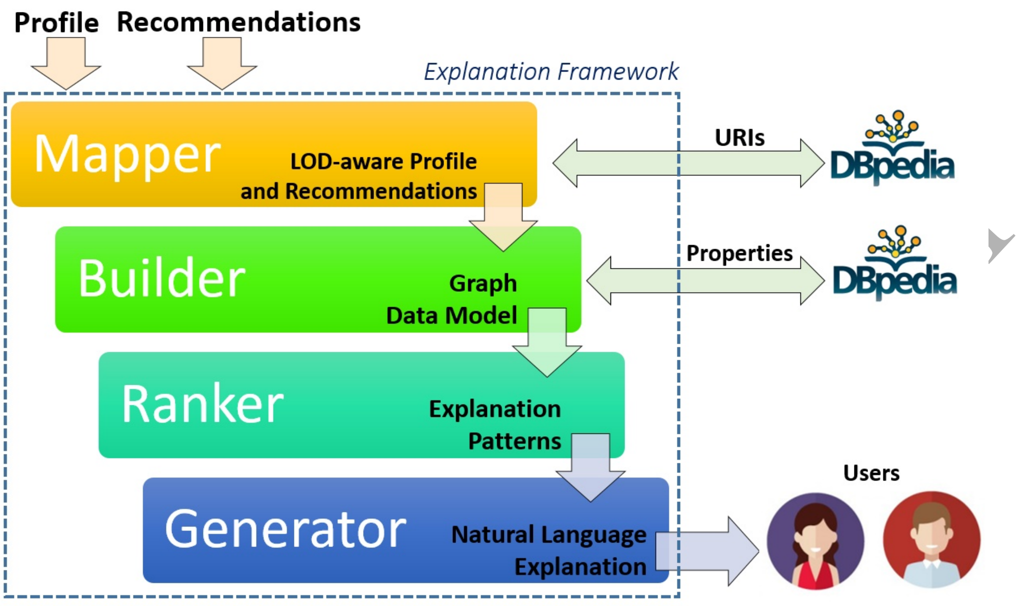


Figure 1: Workflow carried out by our framework.

DBpedia[2]. Formally, let $I_p = \{I_{p1}, I_{p2} \ldots I_{pn}\}$ be the set of the n items the user previously liked and let $I_r = \{I_{r1}, I_{r2} \ldots I_{rm}\}$ be the set of the m recommendations she received, we define the set of the items to be mapped as $I = I_p \bigcup I_r$.

Next, for each $i \in I$, we define a mapping function *map*(i) that returns the URI of the resource. For the sake of simplicity, we label the output of the mapping as i_{LOD}, to identify the LOD-aware representation of the item i. In order to carry out this process, each item has to be equipped with some textual *metadata*. In our setting, such metadata are the *title of a movie*, the *title of the book* or the *name of the artist*, but more complex mapping procedures that also consider different metadata can be easily implemented. In this implementation we did not employ any form of disambiguation and we mapped the entity with the first result returned.

In a nutshell, the function *map*(i) is implemented as a SPARQL query[3] based on the textual metadata available for the item i, which is submitted to a SPARQL endpoint and returns as output the corresponding URI of the resource. As an example, to get the mapping for the movie *The Shining*, we submit a SPARQL query which returns as output the resource whose name corresponds to the *title* of the movie.

However, it may happen that no exact matching to the query is returned. Thus, we also exploited the Levenshtein distance[4] to deal with approximate matching. In this case, we mapped the item with the resource having the minimum distance from the original query. As an example, thanks to the Levenshtein distance it is possible to correctly

[2]http://dbpedia.org/resource/The_Shining_(film). From now on, we will use the acronym dbr as abbreviation of http://dbpedia.org/resource/.

[3]https://www.w3.org/TR/rdf-sparql-query/

[4]https://en.wikipedia.org/wiki/Levenshtein_distance

map the movie Trainspotting[5] to the `DBpedia` resource it refers to, even if the name of the resource does not exactly match the title of the movie. Further details about the mapping procedure will be provided in Section 4.

The mapping is a mandatory step to get an *entry point* to the LOD cloud. Once the mapping is completed, it is possible to gather all the features describing the items and model our *graph-based representation* accordingly. This process is carried out by the `Builder`.

3.2. Builder

The output returned by the `Mapper` is a set of LOD-aware representations i_{LOD} for all the items $i \in I$. `URIs` of those items are used to feed the `Builder`, whose goal is to build a graph-based data model, which is the main *backbone* of the whole explanation process. Using this graph we model all the patterns that connect the items the user liked to the items in her recommendation lists via the properties gathered from the LOD cloud.

In this work we propose two different implementations of this module, referred to as `Basic-Builder` and `Extended-Builder`, respectively. The main difference lies in the nature of the patterns modeled in the graph. In the former, we only model *direct* connections between the items in the profile and those in the recommendation list (e.g., a book I liked has the same author of the book I received as recommendation). In the latter, we enrich the graph by also modeling *indirect* connections between the properties describing the items (e.g., a book I liked does not have the same author of the book I received as recommendation, but both of them share the same *nationality*, or their books have the same *genre*), in order to define more interesting and more general patterns which can *explain* a recommendation.

Formally, the process carried out by the `Basic-Builder` can be defined as follows: let I_u be the set of items in the user profile, let I_r be the set of recommendations and let $exists(s, o)$ be a predicate returning *true* if an RDF triple having s as subject and o as object exists in the LOD cloud. We define $P_r = \{p|exists(i, p), i \in I_r\}$ as the properties available in the LOD cloud describing the items the user received as recommendations and $P_u = \{p|exists(i, p), i \in I_u\}$ as the properties describing the items in the profile. Given such a formalization, we define $P = P_r \cap P_u$ as the set of the properties[6] describing both the items.

Given such a representation, the module builds a bipartite graph $G = (N, E)$, where $N = I_u \cup I_r \cup P$ and $E = \{(i, p)|i \in I_u \cup I_r \wedge p \in P\}$. A toy example of a graph-based data model built by the `Basic-Builder` is provided in Figure 2. Elements in I_p, I_r and P are reported in grey, blue and red, respectively. In this case, the user liked *Psycho IV*, *Sherlock Holmes* and *The Prestige*, and received as recommendation the movies *Memento* and *Inception*. Accordingly, four properties connecting the movies in the profile to the recommendation are added as nodes in the graph[7]. We label them as *explanation patterns*, since they represent the possible *patterns* that can be used to explain the recommendation received by the user.

[5]`dbr:Trainspotting_(film)`

[6]In this work, *properties* and *resources* are used as synonyms. Actually, we model in the graph the resources connected to the item via some property encoded in `RDF`.

[7]The LOD cloud contains many more overlapping properties. Due to space reasons we just reported a small subset of them.

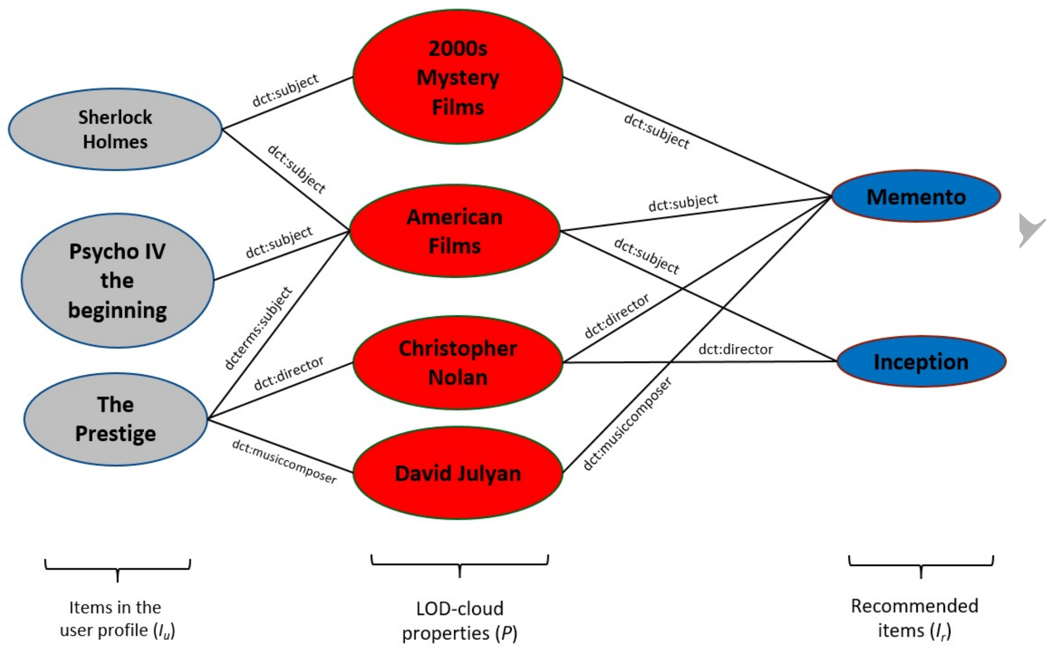


Figure 2: Toy example of the *basic* builder.

Such a basic data model can be further enriched by also including indirect relationships between the items. This implementation is referred to as `Extended-Builder`. Let P_u and P_r be again the set of the properties describing the items in the profile and those in the recommendation list. From now on, we will refer to such properties as *basic* properties.

The `Extended-Builder` introduces a set of *broader* properties P_b as the set of the properties available in the LOD cloud which are in turn connected to the properties describing the items in the profile or to the properties describing the recommendations. Formally, $P_b = \{p | exists(p, p_r) \wedge exists(p, p_u), p_r \in P_r, p_u \in P_u\}$. As an example, the set of *broader* properties connected to the resource `dbr:Christopher_Nolan` includes resources as `dbr:English_film_director` and `dbr:Edgar_award_winner`, just to name a few.

Also these broader properties are freely available in the LOD cloud, and can be extracted again through `SPARQL` queries. In Figure 3 we provide another toy example showing how our `Extended-Builder` works. In this case, we have extended the example in Figure 2 by introducing (some of the) *broader* properties available in the LOD cloud. Broader properties are reported as *orange* nodes. Such an extension, which makes larger the whole representation, creates many new interesting *explanation patterns*. Indeed, the introduction of such properties leads to two advantages: first, different properties can be generalized to the same *broader* property. As an example, both *Guy Ritchie* and *Christopher Nolan* won the *Edgar Award*, and this information, which is encoded in `DBpedia`, can be exploited to provide a more general pattern connecting *Memento* to the

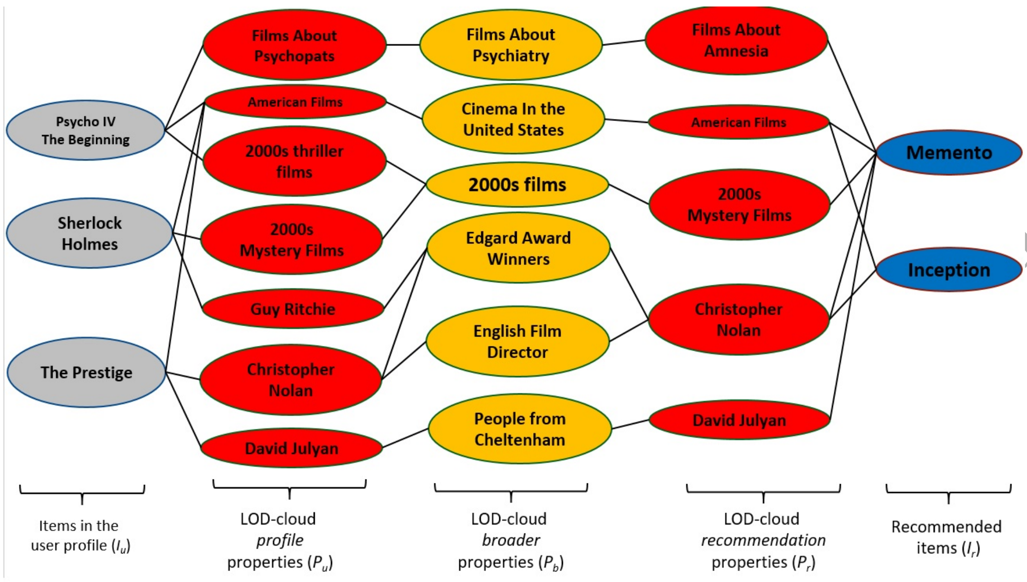


Figure 3: Toy example of the *extended* builder. For the sake of readability we reported only a small part of the *broader* properties available in the LOD cloud.

items in the user profile. Similarly, properties as *2000s thriller movie* and *2000s mystery films* can be generalized to infer that the user is interested in *2000s movies*. Thanks to broader properties, also nodes that did not appear in the basic data model, as *Guy Ritchie*, are now modeled and can be exploited for the explanation as well.

The main advantage of introducing *broader* properties lies in the fact that also indirect relationships can be modeled: in our example, a new pattern connecting *Memento* and *Psycho IV* is created, thanks to the fact that both of them are *movies about psychiatry*. This is a very interesting *explanation pattern*, which would have not emerged if only basic properties were modeled in the graph.

Clearly, the different implementations of the builder give rise to different graphs, which in turn create different *explanation patterns*. Thanks to these patterns based on broader properties, it is likely that our framework can also explain the recommendations generated by algorithms as *collaborative filtering*, whose suggestions are not based on the overlapping properties which are shared by the user profile and recommendations. In the experimental evaluation we will also evaluate this conjecture, by evaluating the effectiveness of the explanations built by exploiting both implementations of the builder.

3.3. Ranker

The `Ranker` takes as input the *explanation patterns* modeled by the `Builder` and identifies the most relevant ones by providing each property with a *relevance score*. We implemented two different scoring formulas: one for the properties modeled in the *basic* graph and another one for the *broader* properties built in the *extended* graph-based data model. Both the formulas are based on the insight that a *good* explanation should emphasize those properties which can describe the current recommendation on the grounds of the items the user liked.

The formula for calculating the score of each *basic* property follows. Formally, given a property p extracted from an explanation pattern, the RANKER calculates its score as:

$$score(p, I_u, I_r) = (\alpha \frac{n_{p,I_u}}{|I_u|} + \beta \frac{n_{p,I_r}}{|I_r|}) * IDF(p) \tag{1}$$

where n_{p,I_u} is the number of edges connecting p with the items in the user profile, n_{p,I_r} is the number of edges connecting p with the items in the recommendation set, α and β are two weighting factors and $IDF(p)$ is an adaptation over `DBpedia` of the classical Inverse Document Frequency [36], which calculates how many items over all the dataset are described by that property. Formula (1) gives a higher score to those properties which are highly connected to the items in I_u and I_r and which are not so common as well. The need for the IDF component of the formula emerged by analyzing the data model presented in Figure 2. Without the adoption of the IDF, very common properties as *American Films*, would often appear in the explanations. Thanks to the introduction of the IDF, the pure frequency of a property is weighted with the inverse of its popularity, thus more interesting and less common explanation patterns may likely emerge.

By following the same insight, the formula for calculating the relevance score of each *broader* property is defined as the *sum* of the scores of the basic properties each extended property is connected to. We preferred to exploit the *sum* over other aggregation strategies (*e.g.*, the average) since we wanted to weigh more those properties that typically generalize several single basic properties (e.g., *Film Directors from Chicago*, *1960s movies*, *Italian actors*, etc.). Formally, given a broader property b, we define the set of the basic properties it connects to as $P_c(b) = \{p|exists(b,p), p \in (P_r \cup P_u)\}$. Next, the score of this property b can be defined as:

$$score(b, I_u, I_r) = \sum_{i=1}^{|P_c(b)|} score(p_i, I_u, I_r) * IDF(b) \tag{2}$$

In this case we use again the IDF (of the broader property) with the same purpose, since we want to emphasize the role of less common properties. Through this calculation, all properties extracted from the explanation patterns of both the *basic* and *extended* graph are provided with a relevance score and the *top-K* are used to generate the explanation. Formally, the `Ranker` returns a set of explanation patterns $E = \{e_1 \dots e_k\}$ where each e_i is a quadruple $< i_u, p, t_p, i_r >$, where $i_u \in I_u$, $p \in P \bigcup P_b$, t_p is the type of the property p, and $i_r \in I_r$.

By referring again to the example in Figure 3, it is likely that the top-3 properties of such a graph would be `dbr:2000s_films`, `dbr:Edgar_Awards_Winner` (since both of them generalize two items) and `dbr:Film_about_Psychiatry`, since it is supposed to be less common than the other properties encoded in the graph, thus thanks to the IDF it should be ranked at the top of the list. Due to the IDF, it is also likely that `dbr:Cinema_in_the_United_States` is not ranked in the top-3 properties as well.

3.4. Generator

In the final step, the most relevant properties returned by the `Ranker` are exploited by the `Generator` module, whose goal is to create the natural language explanation supporting the suggestion.

Generally speaking, this module implements a *template-based structure* which is dynamically filled in using the *explanation patterns* extracted from the graph. In this case we have to distinguish between two different scenarios, depending on the number of recommendations to *explain* and on the kind of *properties* to use.

As regards the number of recommendations, our framework can be used to explain a single recommendation (*top-1 scenario*) or an entire set of suggestions (*top-N scenario*). As regards the properties, we can fill in the template using only the *basic* properties $p \in P$ identified by the `Ranker` (in this case we refer to as `Basic-Generator`) as well as the top *broader* properties $p \in P_b$ (`Extended-Generator`).

Each explanation can include a different number of properties (k): the higher the value of k, the more patterns are included in the explanations and the more sentences are generated.

Formally, for each explanation pattern $e_i \in E$, the explanation is generated using the following template, described using the Backus-Naur Form (BNF):

⟨*explanation*⟩ ::= ⟨*sentence*⟩
  | ⟨*sentence*⟩ ⟨*adverb-continue*⟩ ⟨*sentence*⟩

⟨*sentence*⟩ ::= ⟨*verb*⟩ ⟨*item_rec*⟩ 'since you' ⟨*adverb*⟩ like ⟨*item-type*⟩ whose ⟨*property-type*⟩ is ⟨*property-value*⟩ as ⟨*item_profile*⟩ '.'

⟨*adverb-continue*⟩ ::= 'Moreover'| 'Furthermore' (etc.)

⟨*adverb*⟩ ::= 'always', 'often'| 'sometimes' (etc.)

⟨*verb*⟩ ::= 'I suggest you' | 'I recommend you' | 'I propose you' (etc.)

⟨*item-type*⟩ ::= 'movie' | 'book' | 'artist'

<item_profile>, <property_value>, <property_type> and <item_rec> are filled in with the values of the *explanation pattern* e_i. We lexicalized each *<property_type>* with a natural language expression. As an example, the property type DBO:DIRECTOR is mapped to the expression *'directed by'*. Next, *<adverbs>* are dynamically defined by computing the normalized occurrences of that property in the data model and by mapping each adverb to a different range of the score. Moreover, when different sentences shared the same <property-value>,(e.g., two movies directed by the same director) we merged the sentences by using the conjunction *'and'*. Finally, the other elements of the template were *randomly* generated on the ground of the alternatives shown in the grammar. As an example, the *basic* explanation generated using the top-3 properties ($k = 3$) for the data model provided in Figure 2 is: *"I recommend you* MEMENTO *since you sometimes like movies whose director is* CHRISTOPHER NOLAN *as* THE PRESTIGE*. Moreover, I suggest it because you like* 2000S MYSTERY FILMS *as* SHERLOCK HOLMES *and* AMERICAN FILMS *as* PSYCHO IV - THE BEGINNING*"*.

On the other side, the *broader* explanation built on the ground of the data model shown in Figure 3 is: *"I suggest you* MEMENTO *since you sometimes like movies as* THE PRESTIGE *and* SHERLOCK HOLMES*, whose director was an* EDGAR AWARD WINNER*. Moreover, I recommend it because you like* FILMS ABOUT PSYCHIATRY *as* PSYCHO IV

- THE BEGINNING *and* 2000S FILMS *as* PSYCHO IV THE BEGINNING *and* SHERLOCK HOLMES*".*

The set of templates used to generate different explanations can be easily extended, by building different (and more complex) grammars. In our framework, we implemented several different templates having a different structure, which are randomly chosen at run-time in order to differentiate as much as possible the explanations presented to the users. Moreover, our framework can be used to provide a *single* natural language explanation for an entire set of recommendations. We refer to this case as *Top-N* scenario. To this end, we define an *aggregation* strategy which takes as input the explanations generated for each *single* item and returns a single explanation that (potentially) covers all of them.

Our aggregation process is based on the idea of merging the sentences supporting the items sharing some common explanation patterns. Basically, if two items share some properties (in our running example, both *Inception* and *Memento* share the same *director*, i.e. *Christopher Nolan*) we merge the sentences explaining them into a single sentence via the property they share. This process is iterated over all the items in the recommendation set until no overlap between the properties exists anymore.

In our *toy* example the aggregation is very simple, since both the items can be covered through a single property, as the *director*, and we can explain all the recommendation set through a single sentence. In other scenarios, more than one sentence may be needed to explain larger recommendation sets. An extreme scenario is represented by the case where no overlap among the properties describing the items in the recommendation list exists. In that case, the single explanation for the whole group will correspond to the explanations for the single items.

4. Experimental Evaluation

In the experimental evaluation we tried to answer to the following research questions:

- **R1:** What is the the impact of broader properties gathered from the LOD cloud on the overall effectiveness of the explanations? *(Experiment 1)*
- **R2:** Is our explanation framework able to produce algorithm-independent explanations, i.e. explanations whose effectiveness does not differ on the recommendation algorithm used to generate the suggestions? *(Experiment 2)*
- **R3:** Is our explanation framework able to produce effective explanations for an entire group of recommendations (Top-N scenario) compared to those built for a single recommendation? (*Experiment 3*).

To this end, we designed a user study involving 680 subjects (male=64.3%, degree or PhD=36.4%, already used a RS=87.7%) in three different domains, i.e. *movies*, *books* and *music*. Interest in *movies* was indicated as *medium* or *high* by 72.46% of the sample, while the interest in *books* and *music* was indicated as as *medium* or *high* by 52.52% and 71.52%, respectively. As in [53], we evaluated the following explanation aims: *transparency*, *persuasiveness*, *engagement*, *trust* and *effectiveness*.

4.1. Experimental Design

To run the experiment, we deployed a web application[8] implementing the previously described framework. The platform was designed to run a *between-subject* experiment, i.e. we tested two different explanation styles with the recommendations generated by three different recommendation algorithms. Each user was randomly assigned to an explanation style and to a recommendation algorithm, and evaluated the explanations for all the domains. The order the domains were presented to the user was randomized as well.

As regards the setting of the framework, the MAPPER was able to map to `DBpedia` 8,121 movies, 2,161 books and 3,405 artists. As previously explained, mapping was carried out by matching the title of the movie or of the book and the name of the artist to `DBpedia` resources via a `SPARQL` query. The mappings are available online[9]. Next, the BUILDER queried `DBpedia` to extract the properties describing movies, books and artists. All the available properties were extracted, without performing any filtering. Some statistics about the datasets are provided in Table 1. RDF triples refer to the whole number of triples extracted from `DBpedia` by the *basic* and *broader* builder, while basic and broader properties refer to the number of *distinct* properties which could be potentially used as *explanation patterns*, as CHRISTOPHER NOLAN or MOVIES ABOUT PSYCHIATRY, respectively. Finally, the ranking formula was set by choosing 0.5 as weighting factor for α and β and the top-3 properties were returned.

We used two versions of the explanations styles: *(i)* one exploiting onbly *basic* properties and *(ii)* one including *broader* ones as well. We did not evaluate any other baseline since in our previous research [40] we already showed that our framework provides better explanations than non-personalized and popularity-based explanation styles.

As *recommendation* algorithms, we exploited Personalized PageRank (PR) [17], a content-based recommender system (CBRS) based on the Vector Space Model, and a User-to-User collaborative filtering (CF) technique. To implement the recommendation algorithms we exploited Jung framework[10], Lucene[11] and Mahout[12], respectively. PR was run by using the default settings (80% of the weight distributed on the nodes the user liked), while the CBRS was implemented by modeling the profile of the user as the centroid vector of the items liked and by identifying the most similar documents returned by Lucene through the *cosine similarity.* As content-based features we used the *plots* of the movies and of the books, and a *description* of the artist, extracted from Wikipedia. Finally, CF was run by setting the neighborhood size to 100 for all the domains.

Given that CF algorithms need some *ratings* to provide recommendations, we exploited the data available in MovieTweetings[13], DBbook[14] and Last.fm[15] datasets. Specifically, we extracted the ratings available in these datasets *only* for the subset of items we mapped, and used them to generate the neighborhoods. Overall, 558,539 ratings provided by 30,639 users were exploited for the *movie* domain, while 37,499 ratings provided

[8]http://193.204.187.192:8080/WebLodrecsys/
[9]http://www.di.uniba.it/%7Eswap/datasets/explanation-mapping.zip
[10]http://jung.sourceforge.net/
[11]https://lucene.apache.org/core/
[12]http://mahout.apache.org/
[13]https://github.com/sidooms/MovieTweetings
[14]http://challenges.2014.eswc-conferences.org/index.php/RecSys
[15]http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip

	Movies	**Books**	**Music**
#ITEMS	8,121	2,161	2,161
#RDF TRIPLES (BASIC)	139,737	18,625	50,627
#RDF TRIPLES (BROADER)	138,059	55,691	52,585
#BASIC PROPERTIES	36,585	6,163	14,440
#BROADER PROPERTIES	34,724	20,162	14,203

Table 1: Statistics about the datasets

by 6,046 users and 32,626 ratings provided by 1,876 users were used for the *books* and *music* domains, respectively.

4.2. Experimental Protocol

Each user involved in the experiment carried out the following steps:

(1) Collection of Demographic Data. Users were asked to provide basic demographic data and their interest in movies, books and music domains.

(2) Preference Elicitation and Generation of the Explanations. Users were asked to explicitly rate at least three items for each domain, chosen among a randomly generated subset of 20 movies, 20 books and 20 artists extracted from the datasets. The selection was completely random from the whole dataset in order to avoid to introduce bias in the experimental evaluation. Once the profiles were built, top-5 recommendations were generated by running a recommendation algorithm randomly chosen in the experimental run.

(3) Top-1 Evaluation through Questionnaires. Users were asked to fill in a questionnaire to evaluate the quality of the explanation built for the Top-1 Recommendation. To this end, the user was asked to evalute the previously mentioned *explanation aims* through a 5-point rating scale and to evaluate how much she liked the suggestion. This was only done on the top-1 recommendation. The questions the users had to answer are listed in Table 2. Next, each user had to enjoy a trailer of the movie, an excerpt of the book, or to listen to some songs of the artist, and had to evaluate again the movie and the book after watching the trailer, reading the excerpt or listening to the song.

(4) Top-5 Evaluation through Questionnaires. Finally, users were asked to choose the *explanation style* they preferred for the top-5 recommendation list. As shown in Figure 4, on the left side of the screen they were provided with five different explanations, each one explaining a single item, while on the right part a single explanation for the whole group of recommendations was shown. Group explanations were generated by following the strategy we previously described. In this case, the user was asked to explicitly choose the explanation style she preferred for each of the metrics we took into account.

The whole experiment took less than 5 minutes. A screen recording showing a demo of the experiment is available online[16].

Evaluation Metrics. We evaluated *transparency*, *persuasiveness*, *engagement* and *trust* of the recommendation as the average score collected through the user questionnaires, while the *effectiveness* was calculated as the normalized difference between the

[16]http://bit.ly/2r4elgM

aim	**question**
transparency	"I understood why this movie was recommended to me"
persuasion	"The explanation made the recommendation more convincing"
engagement	"The explanation helped me discover new information about this movie"
trust	"The explanation increased my trust in the recommender system"
effectiveness	"I like this recommendation"

Table 2: Details of the Questionnaire

pre- and post-item consumption. Item consumption was done by watching a trailer, by reading an excerpt of the book and by listening to some songs of the artist, according to the specific domain.

In this case, we followed the evaluation protocol exploited by Bilgic et al., [5], whose insight is that an *effective* explanation may help the user to evaluate to what extent she would like the recommendation, even before enjoying it. For each explanation style at least 75 observations were collected. Given that in [26] the minimum acceptable sample size for each experimental condition was set as 73, we can state that our experiment guaranteed the significance of the results. Statistical significance was assessed by adopting *Mann-Whitney Test*, with $p < 0.05$.

4.3. Discussions of the Results

In the next tables, we refer to *transparency*, *persuasion*, *engagement*, *trust*, and *effectiveness* as TRA, PER, ENG, TRU and EFF, respectively.

4.3.1. Experiment 1

In Experiment 1, we evaluated the impact of *broader* properties on the overall performance of our explanation framework. Results of our user study are reported in Table 3, 4, and 5. As regards the MOVIE domain, we noted a positive impact of such properties on *transparency* and *trust* of the explanations, since we obtained an improvement for CF and PR. Improvement is statistically significant for the *trust*, and this is an interesting outcome of the experiment since one of the goals of providing explanations is actually to increase the *trust* of the users in the RS, and it seems that the adoption of more general and broader properties makes some improvements in this direction. Significant improvements were also obtained on the *engagement* and the *effectiveness* of the explanations, when PR is used as recommendation algorithm. Results in terms of effectiveness are particularly interesting, since when *broader* properties are exploited for the explanation only a 0.40 gap between the pre- and post-trailer ratings was obtained. We obtained a good improvement also on CF. The only algorithm that did not benefit from the information coming from broader properties is CBRS, which slightly improved only the *transparency* of the explanations. A further analysis is needed to better investigate this behavior.

These outcomes were confirmed by the results on the BOOKS domain, reported in Table 4. In this case, results are even better, since the use of *broader* properties led to an

Property	**Algor.**	Tra	Per	Eng	Tru	Eff
Basic	PR	3.80	3.53	3.24	3.40	0.56
Broader	PR	**3.82**	3.51	**3.60(*)**	**3.73(*)**	**0.40**
Basic	CBRS	3.56	3.58(*)	3.42	3.63	0.47(*)
Broader	CBRS	**3.64**	3.13	3.33	3.56	0.71
Basic	CF	3.62	3.08	3.05	3.13	0.59
Broader	CF	**3.63**	**3.11**	2.87	**3.32(*)**	0.68
Basic	(all)	3.66	3.41	3.24	3.39	0.54
Broader	(all)	**3.70**	3.26	**3.29**	**3.55(*)**	0.59

Table 3: Results of the user study in the Movie domain. Improvement obtained by *broader* properties are reported in bold. Statistically significant gaps are emphasized with (*).

Property	**Algor.**	Tra	Per	Eng	Tru	Eff
Basic	PR	3.80	3.59(*)	3.41	3.32	0.57
Broader	PR	3.79	3.42	3.37	**3.47(*)**	0.63
Basic	CBRS	3.60	3.38	3.29	3.44	0.56
Broader	CBRS	**3.70**	3.33	3.13	3.23	**0.53**
Basic	CF	3.13	2.88	2.93	3.05	0.78
Broader	CF	**3.39(*)**	**3.34(*)**	**3.08(*)**	**3.32(*)**	**0.53(*)**
Basic	(all)	3.54	3.28	3.20	3.25	0.65
Broader	(all)	**3.60**	**3.38**	**3.25**	**3.41(*)**	**0.57**

Table 4: Results of the user study in the Books domain. Improvement obtained by *broader* properties are reported in bold. Statistically significant gaps are emphasized with (*).

average improvement for all the metrics we took into account. As we noted for the Movie domain, *transparency* and *trust* are the metrics which benefit the most from the generation of more general explanations based on *broader* properties. Specifically, we noted an improvement in terms of *transparency* on CBRS and an improvement in terms of *trust* in PR. Moreover, it is particularly interesting that our explanation framework got a *significant* improvement for all the metrics when CF is used as recommendation algorithm. This is a very important outcome, since CF is undoubtedly the most challenging recommendation algorithm we evaluated in our user study. Indeed, as previously explained, *collaborative* recommendations are not based on overlapping properties, thus one of the conjectures behind this experiment was that the introduction of *broader* properties would have been helpful to create indirect explanation patterns able to generate satisfying explanations also for this class of recommendation algorithms. The results emerging from this experimental session validated this conjecture, and showed that thanks to *broader* properties our framework can provide more effective explanations also for CF recommendations.

Finally, as shown in Table 5, we got encouraging results also for the Music domain. In this case, broader properties have a positive impact on four out of five metrics, with a statistical significant improvement in terms of *trust*. In this case, the best results are

Property	**Algor.**	TRA	PER	ENG	TRU	EFF
BASIC	PR	3.63	3.28	3.17	3.24	0.43
BROADER	PR	**3.89(*)**	**3.67(*)**	**3.41(*)**	**3.74(*)**	0.63
BASIC	CBRS	3.56	3.49	3.35	3.51	0.63
BROADER	CBRS	3.55	3.40	3.19	3.43	**0.45**
BASIC	CF	3.03	3.08	2.90	2.97	0.56
BROADER	CF	**3.32(*)**	3.05	**3.13(*)**	**3.16(*)**	0.71
BASIC	(ALL)	3.42	3.29	3.15	3.25	0.54
BROADER	(ALL)	**3.60**	**3.40**	**3.25**	**3.46(*)**	0.60

Table 5: Results of the user study in the MUSIC domain. Improvement obtained by *broader* properties are reported in bold. Statistically significant gaps are emphasized with (*).

obtained when PR is used as recommendation algorithm, since all the metrics (with the exception of the *effectiveness*) obtained a significant increase when indirect connections are modeled in our graph. This outcome confirms again the importance of introducing such properties in our data model, since the novel connections induced by broader properties can improve the efficacy of the explanations.

Moreover, it is particularly interesting to emphasize the goodness of the results obtained when the explanations are based on CF recommendations. In this case, we obtained a significant improvement in terms of *transparency, engagement and trust*. This means that the data model built upon broader properties can significantly help the users to discover new information about the items and to better understand the motivations behind the recommendations. In turn, this leads to an increase in the overall trust of the user towards the system, which is a very important finding of this experiment.

As we already noted for the BOOKS domain, also in the MUSIC domain the explanations built on the ground of CBRS did not benefit of *broader* properties. In this case, all the metrics obtained a (non-significant) decrease.

This behavior was somehow expected, and it is probably due to the nature of our explanation model. Indeed, when CBRS is used as recommendation algorithm, suggestions are mainly based on the overlap between the features describing the items and the features stored in the *profile* of the user. Indeed, recommended items typically share a high number of features with the user profile. Given that also our explanation strategy is based on the same insight, it is likely that by only exploiting *basic* properties a good overlap between the features encoded in the profile with those describing the content-based recommendation already exists. Thus, it is reasonable that the introduction of *broader* properties does not produce any benefit in this particular scenario.

On the other side, when CF is used as recommendation algorithm such overlap is not trivial, since recommendation are based on the preferences of the neighborhood of the target user, and content-based features are not taken into account. Indeed, it may happen that the recommendation does not share any common property with the items in the user profile, thus in this case the introduction of *broader* connections can increase the likelihood of an overlap between the properties describing the items, which can help our framework to generate a good explanation.

This conjecture is empirically validated in Table 6, where we report the average

		Basic	Broader	Gap
Movies	PR	2.98	3.00	+0.67%
	CBRS	2.97	3.00	+1.00%
	CF	2.40	2.91	**+21,25**
Books	PR	2.78	2.78	0.0%
	CBRS	3.00	2.92	*-2,67%*
	CF	1.83	2.32	**+26,78**
Music	PR	2.51	2.56	+1,99%
	CBRS	2.96	2.79	*-5,74%*
	CF	1.67	1.84	**+10,18%**

Table 6: Average number of properties included in the explanations, split by *domain* and by *recommendation* algorithm. The best increase are highlighted in **bold**, while the decreases are reported in *italics*.

number of properties used in the explanations (we recall that the maximum number of properties is set to 3) when *basic* and *broader* properties are exploited, respectively. As we expected, the explanations based on CF recommendations obtained the best improvement in terms of overlap, ranging from +10.18% in the Music domain to +26.78% in the Books domain. This means that, when only basic properties are exploited, an explanation supporting a book recommendation generated on the ground of a CF algorithm contains on average 1.83 properties. This value is increased up to 2.32 when also broader properties are encoded in the model, thus it is clear that such an increase in the overlap in terms of properties lead also to an improvement on the overall effectiveness of explanations.

It is not by chance that the best improvement was actually noted in the Books domain, that is the one which increased the most from the number of overlapping properties. On the other side, if we take into account the explanations supporting the recommendations generated by PR, the increase is pretty tiny for all the domains. The values are really close to 3, even when only the basic properties are taken into account. This means that the introduction of broader properties does not create any new connection, but it only provides an explanation based on more general and abstract concepts (e.g., *American directors* instead of *Quentin Tarantino*). As a consequence, it is not by chance that for several metrics we did not note any significant difference when broader properties are introduced.

Finally, we noted an interesting behavior for the explanations based on content-based algorithms which exploit broader properties, since this is the only case where a decrease in terms of overlap is noted. This behavior is probably due to the fact that some of the basic properties we used in the explanation did not have any broader counterpart, and this can produce a decrease in the average number of overlapping properties. This behavior also explains the performance of broader properties based on CBRS, in both Books and Music domains: given that number of properties which are included in the explanations is lower, it makes sense that the overall quality of the explanations based on broader properties is lower as well.

Now we can answer to the first research question:

R1: What is the the impact of broader properties gathered from the LOD cloud on the overall effectiveness of the explanations?

We demonstrated that there is a connection between the average number of overlapping properties we have in the explanations with the general behavior of broader properties. When the introduction of these properties produces an increase in terms of overlap, as it happens when recommendations generated from CF algorithms, it is likely that also the overall quality of the explanations increases as well. On the other side, when the increase is tiny (or there is even a decrease), it is likely that also the metrics do not particularly benefit from the introduction of indirect connections. In conclusion, the broader properties positively affect the effectiveness of the explanations.

4.3.2. Experiment 2

Next, in Experiment 2 we evaluated the *algorithmic-agnostic* nature of our framework, that is to say, we analyzed to what extent the explanations generated by our framework were independent from the specific recommendation algorithm they relied on. The analysis is based on the results reported in Tables 3, 4 and 5.

Specifically, for each of the metrics we evaluated, we compared the results obtained on varying of the different recommendation algorithms and we assessed whether the gap among the algorithms was significant or not. As an example, to evaluate whether the transparency of the explanations for the Movie domain based on the *basic* properties does not differ on varying of the recommendation algorithm, we compared the scores obtained by PR, CBRS and CF for that specific configuration, that is to say, 3.80, 3.56 and 3.62. The overall results are presented in Tables 7, 8 and 9 for Movies, Books and Music domains, respectively.

As shown in the Tables, we compared the algorithms two by two: when the difference between the scores they obtained is significant, we put an **'X'** at the intersection between the algorithms. Otherwise, if the *p-value* was above 0.05 we left the cell empty. To sum up, if all the cells under a specific metrics are empty, it means that the behavior of the explanation framework is *algorithm-agnostic* with respect to that metric.

As regards the Movie domain, the explanations resulted as *algorithm-independent* for most of the comparisons. Indeed, both the *transparency* and the *effectiveness* of the recommendations does not change on varying of the recommendation algorithm. The result is confirmed for both basic and broader properties. This is a very interesting and important insight, which shows that our framework can provide effective and transparent explanations regardless the specific algorithm which is used to generate the recommendation. Similarly, also the other metrics often resulted as algorithm-independent. Specifically, when *basic* properties are used, we did not note any significant difference in terms of *engagement*, while a similar outcome emerged for *persuasion* and *trust* with *broader* properties.

The outcomes emerging for the Books domain are even better, since we were able to validate the *algorithmic-agnostic* nature of our framework for most of the metrics. The only differences we noted regard the comparison between CF and PR in terms of *transparency* and *persuasion*. As we already discussed in the previous Section, this difference is probably due to the poor overlap in terms of features that characterizes the explanations based on CF algorithms. However, the introduction of *broader* properties is able to significantly tackle also this issue: indeed, our tests showed that when *broader* properties are used to explain book recommendations, the gaps among the algorithms is

Movie Domain - *Basic* Properties

	Transparency			Persuasion			Engagement			Trust			Effectiveness		
	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB
CF															
PR				X											
CB				X						X					

Movie Domain - *Broader* Properties

	Transparency			Persuasion			Engagement			Trust			Effectiveness		
	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB
CF															
PR							X								
CB							X								

Table 7: Results of Experiment 2 for the Movie domain. Statistically significant differences between the algorithms are reported with **X**. If no 'X' appears, the *p-value* is above 0.05, thus the behavior of the framework is *algorithm-agnostic*.

never statistically significant. This is another encouraging outcome, which shows that the introduction of *broader* properties can further improve the algorithm-independence of the framework, thus providing effective explanations for all the algorithms we compared.

Finally, by analyzing the results reported in Table 9, very good outcomes emerge also for the Music domain. In this case, the only significant differences were noted when broader properties are used, in terms of *trust* and *persuasion* of the explanations. Moreover, this only happens when recommendations are generated through CF techniques. On the other side, no difference between PR and CB emerged in all the comparisons.

This result further confirmed our hypothesis and showed that the ability of the framework of generating *satisfying* explanations regardless the specific recommendation algorithm is valid for all the domains.

To sum up, the second experiment provided us with several interesting findings: indeed, one of the claim of this work was that our framework was able to generate algorithm-independent explanations, and this session empirically validated this hypothesis. Results were particularly good for the Music and Books domain, but also in the Movie domains most of the comparisons confirmed the *algorithmic-agnostic* nature of the framework.

The *effectiveness* of the explanations is never influenced by the specific recommendation algorithm. This is a fundamental outcome of this experimental session, since it definitely proved that our framework can help the users to correctly perceive the goodness of the recommendations she received, regardless the specific algorithm that generated them.

Now we can answer to the second research question:

R2: Is our explanation framework able to produce algorithm-independent explanations, that is to say, explanations whose effectiveness do not differ on varying of the recommendation algorithm which generated the suggestion?

Yes, our conjecture is confirmed and the experiments showed that our system can be exploited to generate personalized explanations built on the ground of different recommendation techniques.

Books Domain - *Basic* Properties

	Transparency			Persuasion			Engagement			Trust			Effectiveness		
	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB
CF	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
PR	X	▨	▨	X	▨	▨		▨	▨		▨	▨		▨	▨
CB			▨			▨			▨			▨			▨

Books Domain - *Broader* Properties

	Transparency			Persuasion			Engagement			Trust			Effectiveness		
	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB
CF	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
PR		▨	▨		▨	▨		▨	▨		▨	▨		▨	▨
CB			▨			▨			▨			▨			▨

Table 8: Results of Experiment 2 for the Books domain. Statistically significant differences between the algorithms are reported with **X**. If no 'X' appears, the *p-value* is above 0.05, thus the behavior of the framework is *algorithm-agnostic*.

Music Domain - *Basic* Properties

	Transparency			Persuasion			Engagement			Trust			Effectiveness		
	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB
CF	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
PR		▨	▨		▨	▨		▨	▨		▨	▨		▨	▨
CB			▨			▨			▨			▨			▨

Music Domain - *Broader* Properties

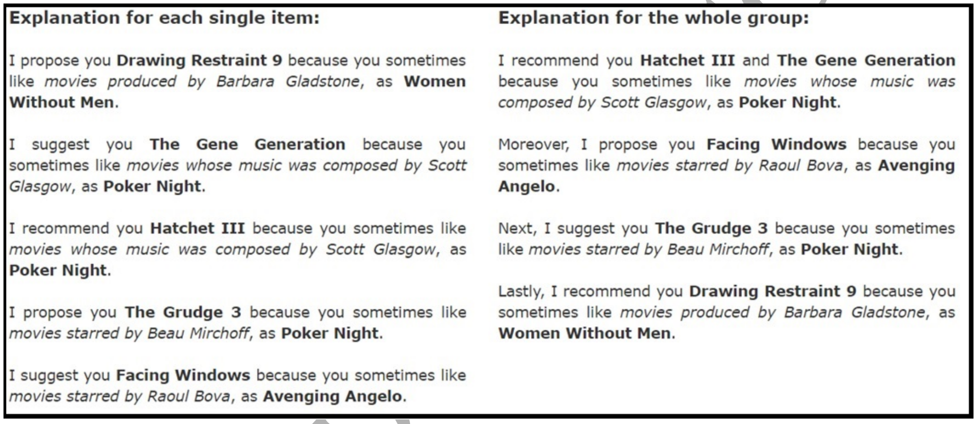
	Transparency			Persuasion			Engagement			Trust			Effectiveness		
	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB	CF	PR	CB
CF	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
PR		▨	▨	X	▨	▨		▨	▨	X	▨	▨		▨	▨
CB			▨			▨			▨			▨			▨

Table 9: Results of Experiment 2 for the Music domain. Statistically significant differences between the algorithms are reported with **X**. If no 'X' appears, the *p-value* is above 0.05, thus the behavior of the framework is *algorithm-agnostic*.

4.3.3. Experiment 3

Finally, in the third experiment we evaluated the ability of our framework of explaining an entire *group* of recommendations through a *single* explanation, by following the strategy described in the previous section. Given that most of the real-world recommender systems work as top-N algorithms (they provide a set of recommendations, not just one) this is a very interesting feature of our framework, thus we decided to evaluate its impact on the users involved in the experiment.

Results of the Experiment are provided in Tables 10. Specifically, each user was asked to explicitly select the explanation strategy she preferred by comparing the alternatives (explanation for a *single item* vs. explanation for a *group of items*) on the screen. Tables are split by domains and report the percentage of the users who selected a certain configuration. An example of the screen the users had to interact with is reported in Figure 4.


Explanation for each single item:

I propose you **Drawing Restraint 9** because you sometimes like *movies produced by Barbara Gladstone*, as **Women Without Men**.

I suggest you **The Gene Generation** because you sometimes like *movies whose music was composed by Scott Glasgow*, as **Poker Night**.

I recommend you **Hatchet III** because you sometimes like *movies whose music was composed by Scott Glasgow*, as **Poker Night**.

I propose you **The Grudge 3** because you sometimes like *movies starred by Beau Mirchoff*, as **Poker Night**.

I suggest you **Facing Windows** because you sometimes like *movies starred by Raoul Bova*, as **Avenging Angelo**.

Explanation for the whole group:

I recommend you **Hatchet III** and **The Gene Generation** because you sometimes like *movies whose music was composed by Scott Glasgow*, as **Poker Night**.

Moreover, I propose you **Facing Windows** because you sometimes like *movies starred by Raoul Bova*, as **Avenging Angelo**.

Next, I suggest you **The Grudge 3** because you sometimes like *movies starred by Beau Mirchoff*, as **Poker Night**.

Lastly, I recommend you **Drawing Restraint 9** because you sometimes like *movies produced by Barbara Gladstone*, as **Women Without Men**.

Figure 4: Preference acquisition between *single explanation* and *explanation for groups*

As shown in Table 10, in all the domains the explanations for *groups* of recommendations were preferred when PR and CBRS are used as recommendation algorithms. Results are particularly significant since almost 80% of the sample involved in the experiment liked the ability of our framework of aggregating several items into a single explanation. Moreover, we carried out a Chi-Square test on the data that showed how the difference between the configurations is statistically significant.

This interesting finding confirmed the goodness of our framework, since the capability of providing a single explanation for a set of recommendations is a very suitable and useful feature. The only exception to this result is represented by the explanations built on the ground of CF recommendations. Indeed, for all the domains the majority of the users labeled as *indifferent* the choice between the single explanations and the aggregate one.

This outcome was quite expected, and is directly related to the nature of our *aggregation* strategy. Indeed, the sentences in the explanation are aggregated on the ground of the properties that are common among the items in the recommendation set. Thus it is likely that when the *diversity* among the items in the recommendation set is *higher*, as it happens when CF algorithms are used, it is likely that no overlapping property among the items exists, so no aggregation can be carried out.

		Group	Single	Indifferent
Movies	PR	**75.7% (*)**	9.4%	14.9%
	CBRS	**78.7% (*)**	15.0%	6.3%
	CF	28.2%	28.0%	**44.8%**
Books	PR	**58.9% (*)**	21.9%	19.2%
	CBRS	**76.2% (*)**	15.7%	8.1%
	CF	41.1%	14.7	**44.7%**
Music	PR	**79.7% (*)**	10.8%	9.5%
	CBRS	30.9%	7.3%	**61.8% (*)**
	CF	**83.7% (*)**	10.0%	6.3%

Table 10: Results of Experiment 3. In each cell we reported the percentage of users who stated to prefer a specific explanation style. The explanation style obtaining the highest preferences is highlighted in **bold**, while statistical significance is further emphasized with **(*)**

In this particular situation, which we empirically noted in our study, the single explanations resulted almost the same as the group explanations, thus it is absolutely straightforward that the users labeled the choice as *indifferent*. An example of such behavior is reported in Figure 4: in this case only two of out five items were aggregated, thus we expect that the user did not experience any difference between the alternatives.

Finally, we can answer to the third and last research question:

R3: Is our explanation framework able to produce effective explanations for an entire group of recommendations (Top-N scenario) compared to those built for a single recommendation?

The experiments demonstrated that our framework is able to provide effective and satisfying explanations also when more than one recommendation is provided to the user. As we already stated, this is an important feature which is definitely of interest for any real-world recommender system, since they typically suggest more than one item and it is important to explain at the same time all the suggestions put in the recommendation list.

5. A Case Study: a Telegram Chatbot integrating explanation function

The main goal of this work was to extensively test our explanation framework with different algorithms and in different domains. However, in order to preliminarily investigate the aptitude of the framework to be easily integrated in a real-world system and its impact on end users, we implemented a conversational movie recommender system as Telegram Bot (@MovieRecSysBot) which offers explanation facilities through our explanation framework.

Chatbots are a kind of bots which emulate user conversations. The main advantages of using a Telegram Bot[17] lie in the simplicity of the interaction, which is due a clean and well-known user interface (the same that people daily use for chatting on their

[17] https://core.telegram.org/bots/api

smartphones) as well as the ease of access, since each account is identified in Telegram by the phone number, thus no registration is needed.

As the explanation framework, the bot is based on the LOD cloud as well, and more specifically on the properties encoded in `DBpedia`. These properties are exploited by the bot for eliciting user preferences, for providing recommendations as well as for generating personalized natural language explanations through our framework.

The workflow carried out by the Bot is depicted in Figure 5.

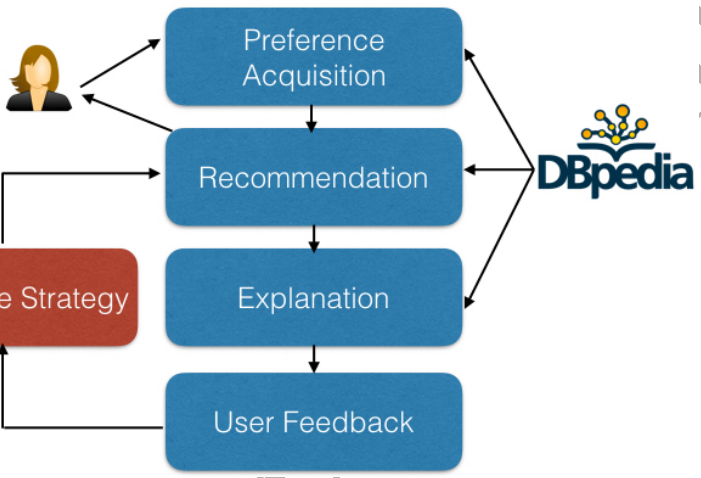


Figure 5: The Bot workflow

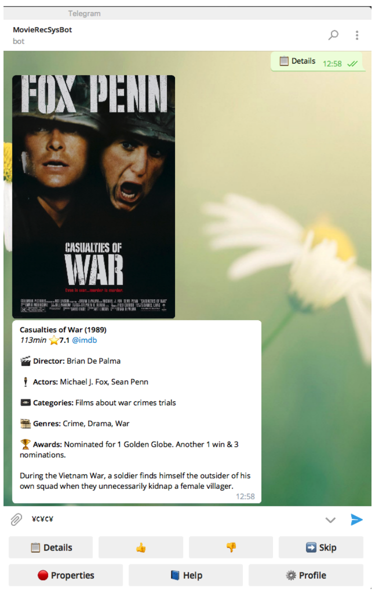
In the first step, *Preference Acquisition*, the Bot asks the user to express her interests. It asks questions related to entities (e.g, movies, persons) and their properties in `DBpedia` (e.g, genre, role).

When the user starts the interaction, her profile is empty, so the recommender system needs to address a classical cold-start problem. The system offers the user two different strategies to express her preferences: (i) rating a set of *items* or *properties* proposed by the system; (ii) typing the entities or properties she is willing to rate. The first option allows the user to express the preferences by tapping buttons. The second option implements an entity recognizer based on the Levenshtein distance [56] by means of a *Did you mean* function, so that, if the user makes typos, the system is anyway able to recognize the right entity or property. The second step is the *Recommendation*. The Bot currently implements PageRank with Priors [18], the same algorithm used for testing the explanation framework.

Even if several RS implemented as Telegram Bots already exist[18], a distinguishing aspect of the bot we implemented is the integration of the explanation facility borrowed from our framework.

As pointed out by Tintarev and Masthoff [53], explaining a recommendation is generally intended as *justifying* the suggestion, but it might be also intended as *providing a detailed description* that allows the user to understand the quality of the recommended

[18]https://telegram.me/imdb


Telegram

MovieRecSysBot
bot

Details 12:58

Casualties of War (1989)
113min ⭐7.1 @imdb

Director: Brian De Palma

Actors: Michael J. Fox, Sean Penn

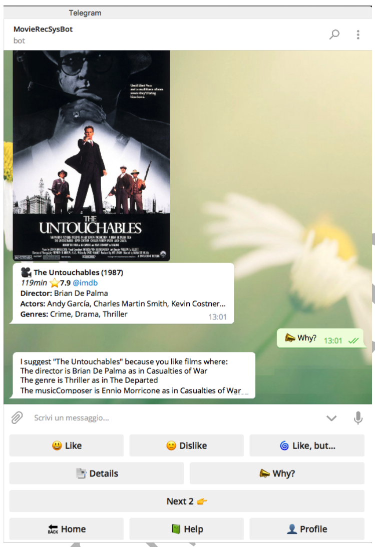
Categories: Films about war crimes trials

Genres: Crime, Drama, War

Awards: Nominated for 1 Golden Globe. Another 1 win & 3 nominations.

During the Vietnam War, a soldier finds himself the outsider of his own squad when they unnecessarily kidnap a female villager.
12:58

Figure 6: A detailed description of a movie on the training phase


Telegram

MovieRecSysBot
bot

The Untouchables (1987)
119min ⭐7.9 @imdb
Director: Brian De Palma
Actors: Andy García, Charles Martin Smith, Kevin Costner...
Genres: Crime, Drama, Thriller
13:01

Why? 13:01

I suggest "The Untouchables" because you like films where:
The director is Brian De Palma as in Casualties of War
The genre is Thriller as in The Departed
The musicComposer is Ennio Morricone as in Casualties of War

Figure 7: A personalized explanation based on our framework

item. According to this definition, the Bot implements both the visions since it provides a detailed description of the movie (see Figure 6) as well as a personalized explanation powered by our framework (Figure 7).

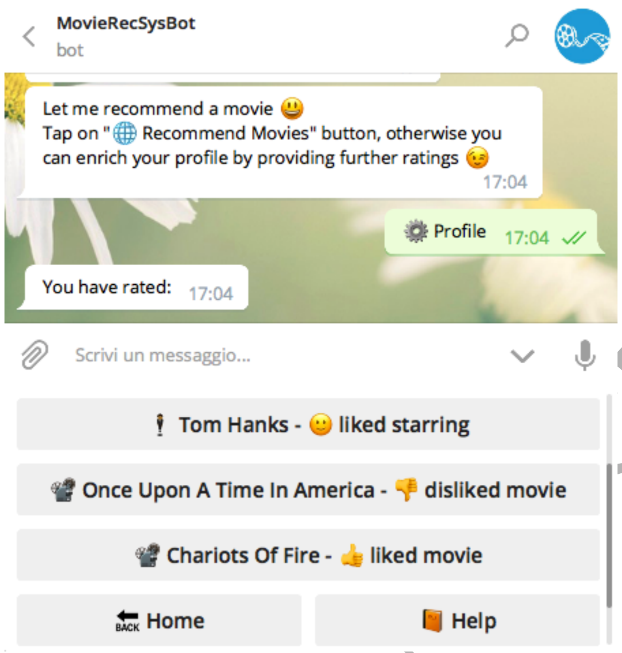
The detailed description of an item can be obtained by tapping on the *Details* button which shows information extracted from IMDB on a given movie. This function could be useful when the user does not know the movie and wants to get more information on it (e.g., director, starring, runtime, genre, etc.). This function is available both in the training phase, when the user expresses her preferences, and in the recommendation phase, when she receives suggestions.

In the recommendation phase, in addition to *Details* is available the *Why?* button that offers a personalized explanation function. This function invokes our framework which uses the connections available in the LOD cloud to explain why a given item has been recommended. Accordingly, as opposed to the explanation based on the detailed description, *Why?* function is tailored on the user preferences.

An example of natural-language explanation provided by the system is: "I suggest you *The Untouchables* because you like movies where: the director is *Brian De Palma* as in *Causality of War*, the genre is *Thriller* as in *The Departed*, the music composer is *Ennio Morricone* as in Causality of War." In this case the system used the connections, extracted from DBpedia, between the recommended movie *The Untouchables* and the user preferences (consisting of *Causality of War* and *The Departed*). In this case adverbs are not used since the user has a very small number of preferences in her profile.

By tapping on the *Profile* button (Figure 8) the user can also explore her profile, and update her preferences.

As a very preliminary follow up of this case study, we analyzed the logs of 415 users


MovieRecSysBot
bot

Let me recommend a movie 😃
Tap on "🌐 Recommend Movies" button, otherwise you can enrich your profile by providing further ratings 😉
17:04

⚙ Profile 17:04

You have rated: 17:04

Scrivi un messaggio...

Tom Hanks - 🙂 liked starring

Once Upon A Time In America - 👎 disliked movie

Chariots Of Fire - 👍 liked movie

Home

Help

Figure 8: A screenshot of the user profile exploration

of the Telegram Bot and we extracted some simple statistics about the usage of the explanation facilities.

In the training phase, the *Details* button was tapped for 2.67% of the rated movies. In the recommendation phase, the *Details* button was tapped for 7.82% of the recommendations, while the *Why?* button for 12.20%. From this very preliminary analysis emerged that in a real-world application the explanation facilities are more used in the recommendation phase than in the training phase. This is probably due to the fact that in the training phase the user rates movies she already knows, on the other side during the recommendation phase it is more likely that the user in interested in better understanding the behavior of the recommender system thus it asks for some *explanation*. In this case, the users also showed that a personalized explanation is typically preferred over a simple detailed description of the movie. This a very interesting outcome which we will better investigate in the future.

6. Conclusions and Future Work

In this work we presented a framework which exploits the information available in the LOD cloud to generate domain-independent explanations supporting the suggestions generated by a recommendation algorithm. We carried out a user study in the movies, books, and music domains, aiming at evaluating the effectiveness of our explanations and at assessing its algorithmic-agnostic nature. Results provided us with encouraging findings, since the usage of broader properties led to significant improvement especially in the books and music domains, and showed that CF techniques particularly benefit from the exploitation of indirect explanation patterns. Moreover, our explanations resulted as algorithm-independent in most of the experimental settings, and this definitely confirmed the insights behind this research. Finally, we showed that the ability of our framework of generating also a single explanation for several items was appreciated by the users, thus paving the way to many research directions. As a case study, we integrated the framework in a *conversational recommender system*.

As future work, we will carry out an in-depth study aiming at better understanding the role of different properties (e.g. *director*, *genre*, *subject*, etc.) on the different metrics we evaluated, in order to filter noisy properties and provide users with more effective explanations. Moreover, we will extend our user study by also evaluating the effectiveness of our explanations with different and more recent recommendation algorithms, as matrix factorization and deep learning-based techniques. Finally, we will investigate the role which plays the explanation in the interaction between the user in a conversational recommender system.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A nucleus for a web of open data, The semantic web (2007) 722–735.

[2] K. Balasubramaniam, Hybrid fuzzy-ontology design using fca based clustering for information retrieval in semantic web, Procedia Computer Science 50 (2015) 135–142.

[3] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, G. Semeraro, Aggregation strategies for linked open data-enabled recommender systems, in: European Semantic Web Conference (Satellite Events), volume 475, Springer, Heidelberg, 2014.

[4] R. Bell, Y. Koren, C. Volinsky, Modeling relationships at multiple scales to improve accuracy of large recommender systems, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 95–104.

[5] M. Bilgic, R. Mooney, Explaining recommendations: Satisfaction vs. promotion, in: Beyond Personalization, IUI WS, volume 5.

[6] C. Bizer, The emerging web of linked data, IEEE Intelligent Systems 24 (2009) 87–92.

[7] S. Chang, F.M. Harper, L.G. Terveen, Crowd-based personalized natural language explanations for recommendations, in: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, ACM, New York, NY, USA, 2016, pp. 175–182. URL: `http://doi.acm.org/10.1145/2959100.2959153`. doi:10.1145/2959100.2959153.

[8] W. Chen, W. Hsu, M.L. Lee, Tagcloud-based explanation with feedback for recommender systems, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, ACM, New York, NY, USA, 2013, pp. 945–948. URL: `http://doi.acm.org/10.1145/2484028.2484108`. doi:10.1145/2484028.2484108.

[9] H.T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ACM, pp. 7–10.

[10] S. Cleger, J.M. Fernández-Luna, J.F. Huete, Learning from explanations in recommender systems, Information Sciences 287 (2014) 90–108.

[11] M. Coyle, B. Smyth, Explaining search results, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI'05, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005, pp. 1553–1555. URL: `http://dl.acm.org/citation.cfm?id=1642293.1642552`.

[12] H. Cramer, V. Evers, S. Ramlal, M. Van Someren, L. Rutledge, N. Stash, L. Aroyo, B. Wielinga, The effects of transparency on trust in and acceptance of a content-based art recommender, User Modeling and User-Adapted Interaction 18 (2008) 455–496.

[13] G. Friedrich, M. Zanker, A taxonomy for generating explanations in recommender systems, AI Magazine 32 (2011) 90–98.

[14] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of the fourth ACM conference on Recommender systems, ACM, pp. 257–260.

[15] F. Gedikli, D. Jannach, M. Ge, How should I explain? A comparison of different explanation types for recommender systems, International Journal of Human-Computer Studies 72 (2014) 367–382.

[16] J. Grau, Personalized product recommendations: Predicting shoppers' needs, 2009.

[17] T.H. Haveliwala, Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search, IEEE Trans. Knowl. Data Eng. 15 (2003) 784–796.

[18] 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.

[19] R. Heckel, M. Vlachos, T. Parnell, C. Duenner, Scalable and interpretable product recommendations via overlapping co-clustering, in: Data Engineering (ICDE), 2017 IEEE 33rd International Conference on, IEEE, pp. 1033–1044.

[20] J.L. Herlocker, J.A. Konstan, J. Riedl, Explaining collaborative filtering recommendations, in: Proceedings of the 2000 ACM conference on Computer Supported Cooperative Work, pp. 241–250.

[21] A. Hernando, J. Bobadilla, F. Ortega, A. GutiéRrez, Trees for explaining recommendations made through collaborative filtering, Information Sciences 239 (2013) 1–17.

[22] D. Jannach, Advisor suite-a knowledge-based sales advisory system, in: Proceedings of the 16th European Conference on Artificial Intelligence, IOS Press, pp. 720–724.

[23] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender systems: an introduction, Cambridge University Press, 2010.

[24] H. Johnson, P. Johnson, Explanation facilities and interactive systems, in: Proceedings of the 1st international conference on Intelligent user interfaces, ACM, pp. 159–166.

[25] B. Knijnenburg, S. Bostandjiev, J. O'Donovan, A. Kobsa, Inspectability and control in social recommenders, in: RecSys 2012, pp. 43–50.

[26] B. Knijnenburg, M. Willemsen, Evaluating recommender systems with user experiments, in: Recommender Systems Handbook., Springer, 2015, pp. 309–352.

[27] B.P. Knijnenburg, M.C. Willemsen, Z. Gantner, H. Soncu, C. Newell, Explaining the user experience of recommender systems, User Modeling and User-Adapted Interaction 22 (2012) 441–504.

[28] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009).

[29] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, L. Getoor, Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems, in: Proceedings of the 9th ACM Conference on Recommender Systems, ACM, pp. 99–106.

[30] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[31] D. Lee, K. Hosanagar, Impact of recommender systems on sales volume and diversity, in: ICIS 2014 proceedings, online: http://aisel.aisnet.org/icis2014/proceedings/EBusiness/40/.

[32] P. Lops, M. de Gemmis, G. Semeraro, C. Musto, F. Narducci, M. Bux, A semantic content-based recommender system integrating folksonomies for personalized access, A Semantic Content-Based Recommender System Integrating Folksonomies for Personalized Access, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 27–47. URL: https://doi.org/10.1007/978-3-642-02794-9_2. doi:10.1007/978-3-642-02794-9_2.

[33] P. Lops, M. de Gemmis, G. Semeraro, F. Narducci, C. Musto, Leveraging the linkedin social network data for extracting content-based user profiles, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, ACM, New York, NY, USA, 2011, pp. 293–296. URL: http://doi.acm.org/10.1145/2043932.2043986. doi:10.1145/2043932.2043986.

[34] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decision Support Systems 74 (2015) 12–32.

[35] T. Mahmood, F. Ricci, Improving recommender systems with adaptive conversational strategies, in: Proceedings of the 20th ACM conference on Hypertext and hypermedia, ACM, pp. 73–82.

[36] C. Manning, P. Raghavan, H. Schütze, Scoring, term weighting and the vector space model, Introduction to Information Retrieval 100 (2008) 2–4.

[37] D. McSherry, Explanation in recommender systems, Artificial Intelligence Review 24 (2005) 179–197.

[38] K.I. Muhammad, A. Lawlor, B. Smyth, A live-user study of opinionated explanations for recommender systems, in: Proceedings of the 21st International Conference on Intelligent User Interfaces, ACM, pp. 256–260.

[39] C. Musto, P. Basile, P. Lops, M. de Gemmis, G. Semeraro, Introducing linked open data in graph-based recommender systems, Information Processing & Management 53 (2017) 405–435.

[40] C. Musto, F. Narducci, P. Lops, M. De Gemmis, G. Semeraro, Explod: A framework for explaining recommendations based on the linked open data cloud, in: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, pp. 151–154.

[41] C. Musto, F. Narducci, P. Lops, G. Semeraro, M. de Gemmis, M. Barbieri, J. Korst, V. Pronk, R. Clout, Enhanced semantic tv-show representation for personalized electronic program guides, in: J. Masthoff, B. Mobasher, M.C. Desmarais, R. Nkambou (Eds.), User Modeling, Adaptation, and Personalization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 188–199.

[42] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, F. Narducci, Leveraging social media sources to generate personalized music playlists, in: C. Huemer, P. Lops (Eds.), E-Commerce and Web Technologies, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 112–123.

[43] F. Narducci, M. Palmonari, G. Semeraro, Cross-language semantic retrieval and linking of e-gov services, in: Proceedings of the 12th International Semantic Web Conference - Part II, ISWC '13, Springer-Verlag New York, Inc., New York, NY, USA, 2013, pp. 130–145. URL: `http://dx.doi.org/10.1007/978-3-642-41338-4_9`. doi:`10.1007/978-3-642-41338-4_9`.

[44] V.C. Ostuni, T. Di Noia, E.D. Sciascio, R. Mirizzi, Top-n recommendations from implicit feedback leveraging linked open data, in: Proceedings of the ACM Conference on Recommender Systems, ACM, 2013, pp. 85–92.

[45] P. Pu, L. Chen, Trust-inspiring explanation interfaces for recommender systems, Knowledge-Based Systems 20 (2007) 542–556.

[46] M. Rossetti, F. Stella, M. Zanker, Towards explaining latent factors with topic models in collaborative recommender systems, in: Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on, IEEE, pp. 162–167.

[47] N. Rubens, M. Elahi, M. Sugiyama, D. Kaplan, Active learning in recommender systems, in: Recommender systems handbook, Springer, 2015, pp. 809–846.

[48] G. Semeraro, P. Lops, M. De Gemmis, C. Musto, F. Narducci, A folksonomy-based recommender system for personalized access to digital artworks, J. Comput. Cult. Herit. 5 (2012) 11:1–11:22. URL: `http://doi.acm.org/10.1145/2362402.2362405`. doi:`10.1145/2362402.2362405`.

[49] G. Semeraro, P. Lops, M. Degemmis, Wordnet-based user profiles for neighborhood formation in hybrid recommender systems, in: Fifth International Conference on Hybrid Intelligent Systems (HIS'05), pp. 6 pp.–. doi:`10.1109/ICHIS.2005.109`.

[50] S. Stumpf, V. Rajaram, L. Li, W.K. Wong, M. Burnett, T. Dietterich, E. Sullivan, J. Herlocker, Interacting meaningfully with machine learning systems: Three experiments, International Journal of Human-Computer Studies 67 (2009) 639–662.

[51] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, MoviExplain: a recommender system with explanations, in: Proceedings of the third ACM conference on Recommender systems, ACM, pp. 317–320.

[52] N. Tintarev, J. Masthoff, A survey of explanations in recommender systems, in: Data Engineering Workshop, 2007 IEEE 23rd International Conference on, IEEE, pp. 801–810.

[53] N. Tintarev, J. Masthoff, Evaluating the effectiveness of explanations for recommender systems, UMUAI 22 (2012) 399–439.

[54] J. Vig, S. Sen, J. Riedl, Tagsplanations: explaining recommendations using tags, in: Proceedings of the 14th international conference on Intelligent user interfaces, ACM, pp. 47–56.

[55] Y. Wang, L.M. Aroyo, N. Stash, L. Rutledge, Interactive user modeling for personalized access to museum collections: The rijksmuseum case study, in: International Conference on User Modeling, Springer, pp. 385–389.

[56] L. Yujian, L. Bo, A normalized levenshtein distance metric, IEEE transactions on pattern analysis and machine intelligence 29 (2007) 1091–1095.