Post-hoc recommendation explanations through an efficient exploitation of the DBpedia category hierarchy

Yu Du, Sylvie Ranwez, Nicolas Sutton-Charani, Vincent Ranwez

Knowledge-Based Systems 245 (2022) 108560

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Post-hoc recommendation explanations through an efficient exploitation of the DBpedia category hierarchy

Yu Du [a], Sylvie Ranwez [a,*], Nicolas Sutton-Charani [a], Vincent Ranwez [b]

[a] EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France
[b] UMR AGAP Institut, Univ Montpellier, CIRAD, INRAE, Institut Agro, F-34398 Montpellier, France

ARTICLE INFO

Article history:
Received 4 August 2021
Received in revised form 16 December 2021
Accepted 9 March 2022
Available online 15 March 2022

Keywords:
Linked Open Data (LOD)
Knowledge graph
Recommender system
Recommendation explanation
DBpedia
Ontology

ABSTRACT

Leveraging knowledge graphs for post-hoc recommendation explanations has been investigated in recent years. Existing approaches rely mainly on the overlap properties (encoded by knowledge graphs) that characterize both user liked items and the recommended ones. These approaches, however, do not fully leverage the property hierarchy of knowledge graphs which may lead to flawed explanations. In this paper we introduce an approach that takes the whole property hierarchy into account. This is done with a limited computation time overhead thanks to efficient algorithmic optimizations relying on sub-ontology extraction. The hierarchical relationships among properties are also considered to avoid redundant properties for explanation. We carried out a user study of 155 participants in the movie recommendation domain and used both offline and online metrics to assess the proposed approach. Significant improvements, in terms of *informativeness* (by 39%), *persuasiveness* (by 22%), *engagement* (by 29%) and *user trust* (by 26%), are suggested by the obtained results, as compared to the state-of-the-art property-based explanation model. Our findings indicate the superiority of accounting for the whole property hierarchy when dealing with post-hoc recommendation explanations.

1. Introduction

Recommender systems (RSs) are efficient solutions to overcome the information overload in product offering. Recommendation models have the ability to predict accurately users' preferences for items they have not yet consumed or evaluated and to recommend the top-N most relevant ones to them [1]. Despite the efficiency of RSs in terms of accuracy, showing only the recommendation lists (without any justification) could make it difficult for users to make decisions based on those recommendations [2]. Recently, recommendation explanations have become an important functionality of RS, aiming at providing justifications for users' recommended items, which increases the trust users have in the RS and helping them decide which item to consume next [3–5].

In the literature, recommendation explanation approaches can be roughly divided into two categories: model-based approaches and post-hoc ones [4]. Model-based approaches aim at investigating how the recommending process works in order to make it easier to interpret the recommendation model. Nevertheless, the capacity to provide explainable recommendations depends heavily on the recommendation models themselves, which leads to the fact that some RS models will have particular advantages over others. Previously, to explain recommendations based on association rule mining [6] or neighborhood-based collaborative filtering [7] was quite easy, because those RS models are human understandable and self-explainable, *e.g.*, "*People who bought product A also bought product B*". But recent models, such as matrix factorization [8] and artificial deep neural networks [9–11], while being highly accurate, are based on opaque latent spaces, hard to interpret. To address this problem, additional information such as user reviews can be used. For example, Zhang et al. [12] proposed EFM (Explicit Factor Model), which aligns each latent dimension of matrix factorization with an explicit item feature extracted from user reviews data. The proposed model can thus provide personalized explanations for the recommended products, *e.g.*, "*The product is recommended because you are interested in {a particular feature}, and the product performs well on that feature*".

Following a quite different perspective, post-hoc recommendation explanation approaches consider the RS models as black-boxes, and proceed in either a non-personalized or a personalized way. In the non-personalized case, explanations are only based on the recommended items' descriptions. This absence of personalization allows the pre-computing of the explanation for each catalog item and thus avoids the optimization challenges of personalized explanations. Tintarev and Masthoff [3] discussed

* Corresponding author.
E-mail addresses: yu.du@mines-ales.fr (Y. Du), sylvie.ranwez@mines-ales.fr (S. Ranwez), nicolas.sutton-charani@mines-ales.fr (N. Sutton-Charani), vincent.ranwez@supagro.fr (V. Ranwez).

https://doi.org/10.1016/j.knosys.2022.108560

several non-personalized explanation styles such as popularity-based styles, *e.g.*, "*This movie is recommended to you because it is a popular movie*". The authors of [13] proposed another non-personalized, post-hoc approach that summarizes positive user reviews on items. The authors used natural language processing (NLP) and sentiment analysis to extract relevant features for explanations. Meanwhile, the users' reviews data needed to implement such an approach are not available for all types of items (*e.g.*, scientific papers, radio podcasts) and their quantity may also vary greatly between catalog items (*e.g.*, a US blockbuster as compared to a Swedish art house film). In the case of personalized explanations, the relations between the recommended items and those liked by users (user profile items) are usually explored to draw up the explanations, *e.g.*, "*This movie is recommended to you because you have liked comedy movies*".

Semantic web resources such as Linked Open Data (LOD) and knowledge graphs (KG) are useful tools for enhancing post-hoc recommendation explanations as they are capable of representing facts and domain knowledge in a formal, machine-readable way. Existing LOD-based explanation approaches rely mainly on the overlap features (*i.e.*, the KG's entities) between user liked items and the recommended ones. However, the main drawback of existing methods is that neither the KG's hierarchy nor the hierarchical relationships among entities are considered carefully, which may lose the inference power of knowledge graphs and lead to irrelevant or redundant explanations. To address these issues, we proposed in this paper a generic method that allows to exploit efficiently the entire entity hierarchy and to select the most relevant entities for explanation.

The main contributions of this paper are the followings:

- we account for the whole property hierarchy with low computation time overhead thanks to algorithmic optimizations relying on sub-ontology extraction [14];
- we limit irrelevant properties in explanations by favoring those that directly annotate (describe) the catalog items.
- we propose a simple criterion to eliminate fully redundant explanations;
- we propose a parameter-free scoring function to rank properties.

The rest of the paper is organized as follows. In Section 2 we present related works and our research motivations before detailing our proposed approach in Section 3. The assessment protocol is presented in Section 4. The results based on the carried-out online user study are detailed in Section 5. Finally, we present the main conclusions and perspectives in Section 6.

2. Related works and motivation

In Linked Open Data, domain knowledge is encoded by large-scale knowledge graphs (KGs) [15]. LOD has facilitated the development of some famous KGs such as DBpedia [16] or Freebase [17], encoding heterogeneous facts and knowledge in a standardized, machine-readable structure relying on the RDF[1] schema of W3C. The authors of [18] proposed a model based on entities extracted from DBpedia to enhance explanations of recommendations in the tourism domain. For a recommended visit tour, the authors compared four different explanation styles including EN (entities), NL (natural language sentences containing entities), PC (pure classes of entities) and CC (entities accompanied by their classes). Based on an online user study, they found that NL and CC styles performed better than EN and PC ones. The explanations based on their proposed method are non-personalized as they only focus on the descriptions of the recommended tours while ignoring user preferences.

Musto et al. [19] proposed a personalized explanation framework named ExpLOD, which is based on the set of object properties[2] P describing both a given user u's liked items I_u (*i.e.*, the user profile) and the items I_r recommended to that user. Let P_u (resp. P_r) be the set of properties describing items in I_u (resp. I_r), then $P = P_u \cap P_r$. The scoring function for a given property $p \in P$ regarding I_u and I_r is defined by Eq. (1), where n_{p,I_u} (resp. n_{p,I_r}) represents the number of items in I_u (resp. I_r) that are described by the property p, α and β are weighting factors and the $IDF(p)$ term stands for the *Inverse Document Frequency* of the property p. The latter term aims at penalizing those properties describing most of the catalog items, *e.g.*, *American Films*.

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

After the ranking step, the top-k properties are used to generate natural language explanations based on a predefined template. As an example, the explanation for the recommended movie *Memento* to a user who liked *Sherlock Holmes* and *The Prestige*, can be (with $k = 2$): "We recommend *Memento* because you liked *2000s mystery films* as *Sherlock Holmes* and *Christopher Nolan directed films* as *The Prestige*".

The scoring function favors properties frequent in both I_u ($\frac{n_{p,I_u}}{|I_u|}$) and I_r ($\frac{n_{p,I_r}}{|I_r|}$) while rare in the catalog ($IDF(p)$). The part favoring properties appearing several times in the recommended list (*i.e.*, $\frac{n_{p,I_r}}{|I_r|}$) is a specificity of the ExpLOD method. Indeed, ExpLOD was designed to provide compact explanations for a group of items. Assigning a bonus to properties describing multiple items in I_r helps to favor explanations that rely on the same properties for different recommended items. The explanations are then factorized by sentences such as: "We recommend movies m_1 and m_2 because they are *romantic comedy films* as most of the ones you liked". The counterpart is that an item $i_r \in I_r$ is not necessarily as good as it could have been if alone (*i.e.*, if $I_r = \{i_r\}$). It could even occur that none of the top-k group properties are relevant to explain some items of I_r, the larger $|I_r|$ the higher the risk of such problematic cases.

In [20], the authors extended ExpLOD by considering a *broader* set of properties P_b that are direct subsumers of the properties in $P_u \cup P_r$. More formally,

$$P_b = \{p | parent(p, p_u) \wedge parent(p, p_r), p_u \in P_u, p_r \in P_r\}.$$

For instance, the broader set of properties subsuming directly the property *2000s mystery films* (*i.e.*, its parent properties) will contain *2000s Films*, *mystery films by decade* and *21st-century mystery films*. The *broader* properties $p_b \in P_b$ are scored by summing up the scores of properties they subsume (Eq. (2)). The top-k ranked p_b are then used to generate the explanation.

$$score(p_b, I_u, I_r) = \sum_{p \in children(p_b) \cap (P_u \cup P_r)} score(p, I_u, I_r) * IDF(p_b) \quad (2)$$

Note that the same item may contribute multiple times to p_b's score if it is described by several children of p_b. For instance, consider a user profile with two films *set in USA*; one is described as *set in New York* while the other is described as both *set in New York* and *set in Washington*; then the second film will have more

[1] https://www.w3.org/TR/rdf-schema/.

[2] Note that in the knowledge engineering domain, the terminology "object property" refers to a relationship that links individuals to other individuals, *e.g.*, "dct:subject" that associates a category to a film. To avoid possible ambiguity, please note that by an abuse of language, and for sake of simplicity, the term "property" we are using below refers to the individual itself rather than the property, *i.e.*, it designates "dbc:American_sports_comedy_films" that may characterize "dbr:Cool_Runnings" rather than "dct:subject".

impact on the scoring of the *set in USA* broader property than the first one as it will contribute twice in the sum of Eq. (2). Note also that the score of the broader property not only takes into account its *IDF* value but also the *IDF* values of a subset of its children – which might seem questionable – indeed Eq. (2) can be rewritten as:

$$score(p_b, I_u, I_r) = \sum_{p \in children(p_b) \cap (P_u \cup P_r)} \left(\alpha \frac{n_{p,I_u}}{|I_u|} + \beta \frac{n_{p,I_r}}{|I_r|} \right) * IDF(p) * IDF(p_b) \quad (3)$$

The authors evaluated ExpLOD through an online user study on three datasets in different recommendation domains (movie, music and book). The results have shown that their proposed model performed well in terms of *transparency*, *persuasiveness*, *engagement* and *trust* which are classic evaluation metrics for recommendation explanations [3,4]. In addition, it seems that their model allows to generate effective post-hoc explanations, regardless of the used RS model. Finally, their evaluations confirmed the superiority of the *broader* ExpLOD extension and the relevance of accounting for subsumer properties.

This latter observation is the main motivation of our work: it aims at taking the *broader* variant of ExpLOD a step further by efficiently taking into account the whole property hierarchy. Though the approach proposed in this article is generic we will focus on the DBpedia category hierarchy to provide the reader with a familiar framework. The DBpedia category hierarchy is an ontology encoding subsumption relationships among categories within the DBpedia knowledge graph. In this hierarchy, categories are linked to those generalizing them via the "skos:broader" relationship.

The experiments comparing the *basic* and *broader* variants of ExpLOD proved the value of considering subsumer properties [20]. However, there is no obvious reason for restricting this broadening only to direct subsumers. Meanwhile, subsumer properties should be considered carefully to: (i) avoid using very abstract and irrelevant properties in the explanation (ii) limit redundancy in the explanation and (iii) keep the computation time low.

An item explicitly described by the property p is also implicitly described by all properties subsuming p. Most KGs have at least one strong property backbone defined by a transitive subsumption relationship such as "rdf:type" (is-a), "dbo:isPartOf" or "skos:broader". This is a key feature of KGs that enables semantic reasoning without which most of the KG value would be lost. Fig. 1 illustrates the importance of the property hierarchy through simple examples inspired by the movie categories in DBpedia. In all these examples the aim is to explain a recommended movie i_r based on a set of the user's liked movies I_u. If i_r is described by *French romantic comedy* (denoted by a green star) and some items of I_u are also described by this property (denoted by a red triangle), then it can be used to explain the recommendation, as illustrated in Fig. 1A. This is the idea behind the *basic* version of ExpLOD. However, this basic version would fail when i_r is annotated as *French romantic comedy* while I_u contains some *Italian romantic comedy* movies, *i.e.*, Fig. 1B. The *broader* variant of ExpLOD overcomes this problem while keeping the computation time reasonable, by considering direct subsumers of each annotating property. This allows to explain the recommended movie using the *European romantic comedy* property which is a direct subsumer of both *Italian romantic comedy* and *French romantic comedy* properties.

Note also that the two ExpLOD variants proposed in [19, 20] were considered as two separate approaches with somehow different scoring functions. As a result, the *basic* variant only considers direct properties for explanations whereas the *broader* one only considers the direct subsumers of those properties. Thus, the *French romantic comedy* in Fig. 1A will be considered by the *basic* version but not by the *broader* one whereas *European romantic comedy* in Fig. 1B will be considered by the *broader* variant of ExpLOD but not the *basic* one.

Though the *broader* variant is clearly a meaningful improvement, the last three examples, *i.e.*, Fig. 1C, D and E, illustrate its limitation and the new problems it raises. In Fig. 1C, i_r is annotated as *French romantic comedy* while one of the I_u items is annotated as *US romantic comedy* and the other one *Romantic comedy*. The problem here is that the direct subsumer of *French romantic comedy* is *European romantic comedy*; it follows that even the *broader* version of ExpLOD fails to infer that the three movies are *Romantic comedy* and will hence be unable to use this key property to explain the recommendation. Note also that any subsumer of *Romantic comedy*, *e.g.*, *Comedy*, though shared by these three movies' annotations, would be irrelevant for the explanation as it would be redundant with the explanation provided using the more specific *Romantic comedy* property. Fig. 1D provides a minimalist example of this redundancy problem. In DBpedia, the *Romantic comedy* property has about 1,710 subsumers including *Romance*, *Films* or *Creative works*, all of which would (implicitly) annotate the two items of Fig. 1D. It may seem more reasonable to base the explanation on the more specific *European romantic comedy* property than on its subsumers, *e.g.*, *Creative works* is meaningless for user explanation. Finally, adding (direct) subsumers of annotating properties raises another problem: some of those subsumers may be relevant for organizing the property hierarchy but meaningless for explanation as illustrated in our last example, Fig. 1E. Here, i_r is annotated as *Comedy*, while an item in I_u is annotated as *Drama*. Since both properties are directly subsumed by the property *Films by genre*, the following surrealistic explanation could be proposed by the *broader* variant of ExpLOD: "*This movie is recommended to you because it is a Films by genre and you usually liked Films by genre*".

In this paper we propose a generic, item Property-based Explanation Model (PEM) which efficiently exploits the whole property hierarchy of the KG. The next section presents the proposed approach and details the way we exploit this full hierarchy in a reasonable computing time while providing explanations which avoid redundancy and meaningless properties.

3. Proposed Property-based Explanation Model (PEM)

In this section we detail the proposed approach to handle the full property hierarchy for post-hoc recommendation explanations efficiently.

3.1. Definitions and notations

First, we denote as $\mathcal{C}$ the set of catalog items and as I_u the set of items liked by the user u (user profile). The target item to explain is denoted as i_r. We denote as $\mathcal{H}(\mathcal{N}, \mathcal{E})$ the hierarchy of DBpedia categorical entities, with $\mathcal{N}$ being the entities prefixed by "dbr:Category" and $\mathcal{E}$ being the "skos:broader" relationship among those entities. $\mathcal{P}(i)$ provides the set of properties directly annotating the item i, *i.e.*, $\mathcal{P}(i) = \{p | p \in \mathcal{N}$ and $\langle i,$ "dct:subject", $p\rangle\}$. The predicate "dct:subject"[3] is commonly used in LOD to link a resource to its topics, *e.g.*, link a movie (i) to its genres (properties p annotating i). By extension we define the $\mathcal{P}(\cdot)$ function on a set I of items as $\mathcal{P}(I) = \bigcup_{i \in I} \mathcal{P}(i)$. It follows that $\mathcal{P}(\mathcal{C})$ is the set of properties explicitly used to annotate at least one item of the catalog.

[3] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms/subject.

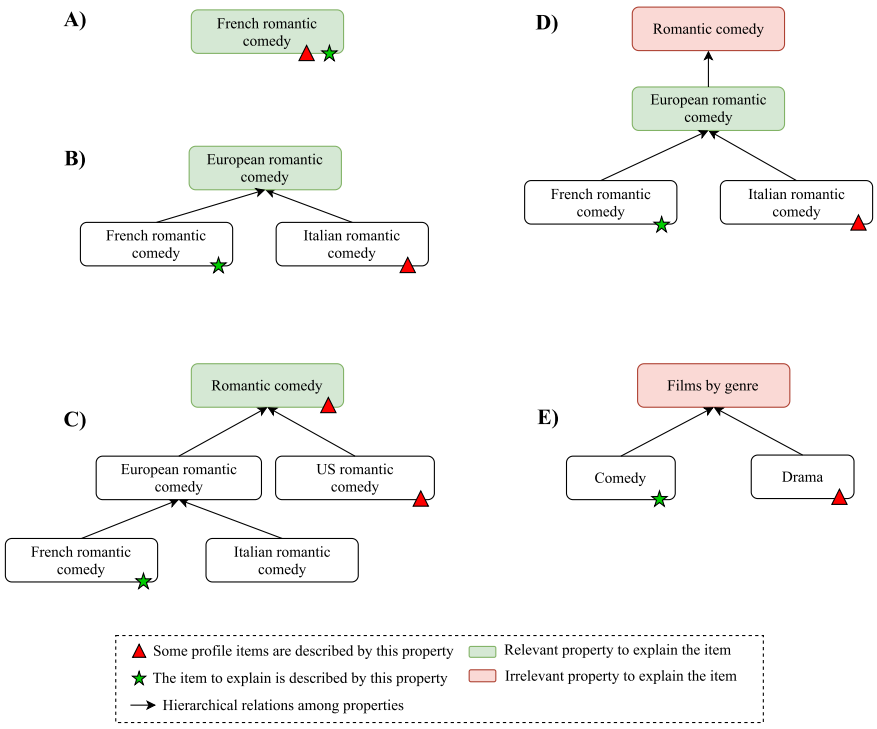

Fig. 1. Motivating examples illustrating the importance of considering the property hierarchy within the explanation process in order to emphasize semantic relationships between properties that are valued by the user and the ones that annotate recommended items.

Second, we denote as $\overline{\mathcal{P}(i,\mathcal{H})}$ the set of all the properties that directly or indirectly *describe* the item i regarding the hierarchy $\mathcal{H}$, *i.e.*, $\overline{\mathcal{P}(i,\mathcal{H})} = \{p | \exists p_x$ such that p subsumes p_x in $\mathcal{H}$ and $p_x \in \mathcal{P}(i)\}$. Again, we extend this notation to a set of items using union: $\overline{\mathcal{P}(I,\mathcal{H})} = \bigcup_{i\in I} \overline{\mathcal{P}(i,\mathcal{H})}$. For simplicity's sake, we will omit the $\mathcal{H}$ parameter and use $\overline{\mathcal{P}(i)}$ and $\overline{\mathcal{P}(I)}$ if there is no ambiguity regarding the hierarchy used. Similarly, we denote as $\mathcal{I}(p,I)$ (resp. $\overline{\mathcal{I}(p,I)}$) the subset of items of I directly annotated (resp. directly or implicitly described) by p.

3.2. Scoring properties

In the PEM method, we aim at taking the whole property hierarchy into account while avoiding irrelevant properties that are only there to organize the property hierarchy (*e.g.*, *Films by genre*). To this end, we consider all properties p that are simultaneously: (i) a subsumer of a property of i_r, (ii) a subsumer of a property of an item in I_u and (iii) directly annotate at least one item of the catalog, *i.e.*, $p \in \overline{\mathcal{P}(i_r)} \bigcap \overline{\mathcal{P}(I_u)} \bigcap \mathcal{P}(\mathcal{C})$. By doing so, we take the whole property hierarchy into account thus handling the potential problems illustrated by Fig. 1B and C. The benefit of this intersection is that it is no longer necessary to consider all properties of the hierarchy, but only those directly annotating an item of the considered catalog, which allows to have a low-latency solution while considering the whole hierarchy. The score of the properties in this intersection is built upon a simple fold change, as detailed in Eq. (4). The fold change metric is widely used to detect over-represented properties in a subset of elements [21]. Here, we use it to detect properties of the recommended item that are over-represented in the user profile as compared to the whole item catalog.

$$score(p, I_u, i_r) = \log_{10}\left(|\mathcal{I}(p,\mathcal{C})|\right) * \left(\frac{\left|\overline{\mathcal{I}(p,I_u)}\right| \Big/ |I_u|}{\left|\overline{\mathcal{I}(p,\mathcal{C})}\right| \Big/ |\mathcal{C}|} \right) \tag{4}$$

For each scored property p, we introduce a factor, $\log_{10}\left(|\mathcal{I}(p,\mathcal{C})|\right)$, which strongly penalizes the score when p is rarely used to describe the catalog items directly. The scores of properties directly annotating less than 10 items will be reduced while the scores of properties directly annotating more than 10 items will be increased (without making much difference between properties directly annotating 300 or 320 items, thanks to the logarithmic transformation). The benefit of this factor is that it avoids properties that are relevant to organize the hierarchy but not to describe the items, *i.e.*, the problem illustrated in Fig. 1E. Finally, for all properties to be considered, our scoring is based on a fold change ratio that highlights properties of i_r describing more items in the user profile than expected by chance should the profile be a random sampling from the catalog. Note that in this ratio, a property is considered as describing an item if itself or one of its descendants is used to describe the item. The full hierarchy information is hence taken into account to infer implicit item descriptions.

3.3. Efficient scoring calculation using sub-ontology extraction

In large KGs such as DBpedia, a property may have thousands of subsumers (*e.g.*, *Romantic comedy* having about 1700 subsumers). While an item is usually directly annotated by a handful of properties, it is implicitly described by thousands of their subsumers. Fig. 2A depicts a toy example where direct item descriptions are represented by a small colored geometric shape

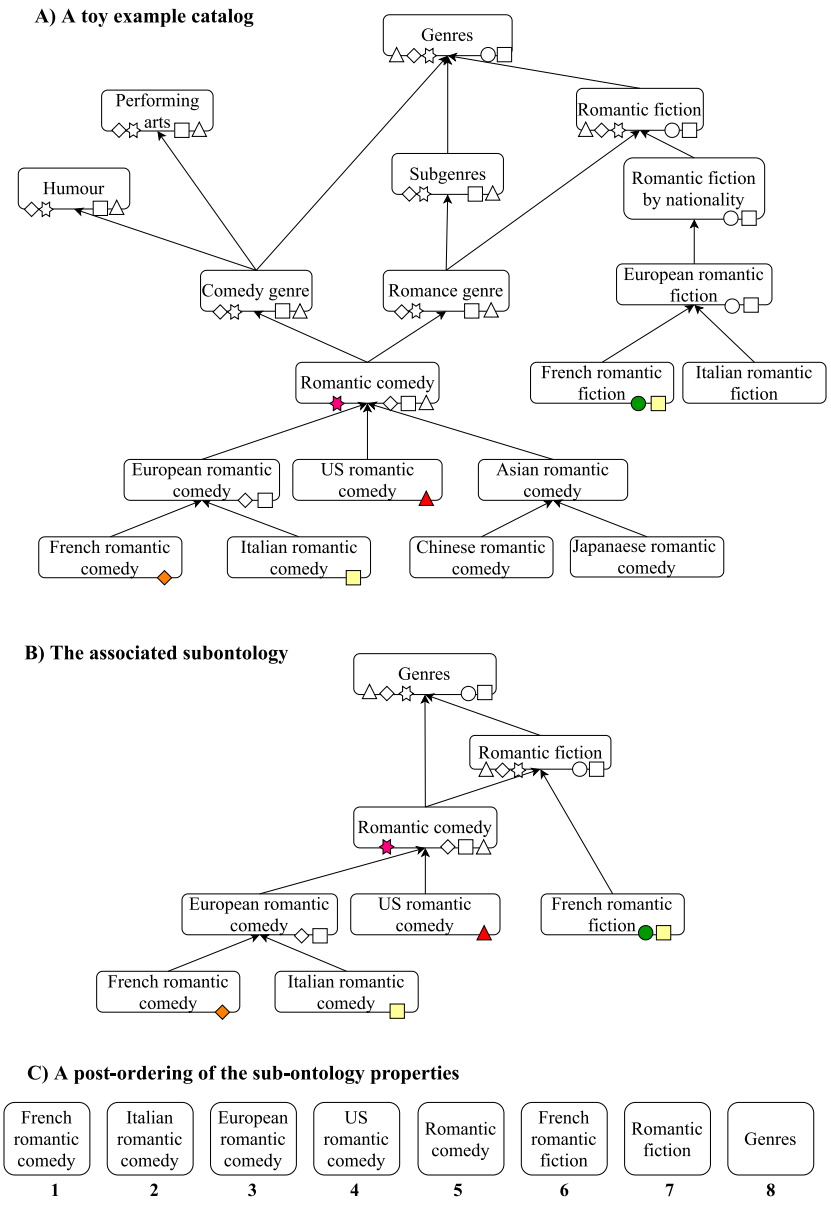

Fig. 2. Toy examples illustrating the sub-ontology extraction process: (A) provides an extract of the full property hierarchy; (B) provides the associated sub-ontology and (C) a post-ordering of this latter, that supports algorithmic optimization.

(using one shape per catalog item) and implicit descriptions are depicted by the corresponding white geometric shape. Even in this toy example, inspired by the DBpedia property hierarchy, it is obvious that there is many more implicit descriptions than direct ones. Considering those implicit descriptions is thus key to getting the most out of the LOD data, but it poses an algorithmic challenge as the scoring computation should be able to be scaled up.

The scoring calculation we propose here ensures the scalability thanks to two key considerations. First, only properties directly annotating at least one item of the catalog have a score above 0 and are worth evaluating. In practice, this considerably reduces the number of properties for which the score has to be evaluated. Computing the score of each property independently is inefficient as it would require to consider the same properties multiple times. For instance, in order to estimate (the cardinality of) $\overline{\mathcal{I}(\textit{Romantic comedy}, \mathcal{C})}$ one needs, among other things, to identify all *French romantic comedy* movies of the catalog, something that has to be done also to compute $\overline{\mathcal{I}(\textit{French romantic comedy}, \mathcal{C})}$ and $\overline{\mathcal{I}(\textit{European romantic comedy}, \mathcal{C})}$. Given a relatively stable catalog, it could be argued that those counts can be precomputed once and only need updating when the catalog changes. However, such an argument does not hold for the user profiles counts $\overline{\mathcal{I}(p, I_u)}$ or to identify properties within $\left(\overline{\mathcal{P}(i_r)} \bigcap \overline{\mathcal{P}(I_u)}\right)$, for example in an online study setting.

As $\mathcal{H}$ encodes the subsumption relationships it should be a directed acyclic graph (or DAG for short). The absence of cycles allows to order nodes in a consistent way so that a node always appears after all its descendants (post-ordering) or its ascendants (pre-ordering). Fig. 2B depicts a small property hierarchy and together with a possible post-ordering of its properties in Fig. 2C. Using a post-ordering of the properties allows to factorize the computation of $\overline{\mathcal{I}(p, I_u)}$ and $\overline{\mathcal{I}(p, \mathcal{C})}$ efficiently. Indeed, for any item set I, all $\overline{\mathcal{I}(p, I)}$ can be obtained by processing all properties

in the post-order and using the following recursive formula:

$$\overline{\mathcal{I}(p,I)} = \mathcal{I}(p,I) \bigcup \Big(\bigcup_{p_c \in children(p)} \overline{\mathcal{I}(p_c,I)} \Big) \quad (5)$$

Note that for our scoring function Eq. (4), only the set sizes are needed; the actual set are only temporarily needed to avoid counting the same item multiple times. Thus, once all fathers of a property p have been processed, the actual set $\mathcal{I}(p,I)$ can be deleted to free some memory as long as its cardinality is saved. This can be done efficiently by assigning a counter to each property that is initially set to the number of its direct parents and decreased every time one of them is processed (see Section 4.2.3 of [14] for more details on this optimization).

So far we mentioned two solutions to optimize the scoring calculation: the first one computing scores only for properties directly annotating at least one item of the catalog (*i.e.*, those within $\mathcal{P}(\mathcal{C})$) and the second one using the recursive formula of Eq. (5). The problem is that they seem to exclude each other since there is no guaranty that a child of properties directly annotating items is also directly used to annotate items. However, these two optimization ideas can be combined thanks to the sub-ontology extraction algorithm described in [14]. Extracting from $\mathcal{H}$ the sub-DAG induced by $\mathcal{P}(\mathcal{C})$ allows to obtain a much smaller graph $\mathcal{H}_{\mathcal{C}}$. This sub-DAG is the smallest one that contains all properties of $\mathcal{P}(\mathcal{C})$ and the least common ancestors of any subset of the kept properties that are useful to factorize $\overline{\mathcal{I}(p,I_u)}$ and $\overline{\mathcal{I}(p,\mathcal{C})}$ computations efficiently. Fig. 2 illustrates the construction of $\mathcal{H}_{\mathcal{C}}$ on a toy example. Moreover, note that all properties of $\mathcal{H}_{\mathcal{C}}$ that are not in $\mathcal{P}(\mathcal{C})$ would be useful if they have an ancestor in $\mathcal{P}(\mathcal{C})$. Hence, $\mathcal{H}_{\mathcal{C}}$ can be further reduced by pruning all its properties that are not in $\mathcal{P}(\mathcal{C})$ and have no ancestors in $\mathcal{P}(\mathcal{C})$. Consider, Fig. 2B as an instance. While *Romantic fiction* is useful to grasp the relationship existing between *Romantic comedy* and *French romantic fiction* and hence kept by the sub-ontology extraction, it is not useful in our case since neither it nor its only subsumers (*i.e.*, *Genres*) directly annotate an item of the toy catalog. Both properties can thus be safely removed from $\mathcal{H}_{\mathcal{C}}$ without impacting our algorithm. On the contrary, *European romantic comedy*, while not in $\mathcal{P}(\mathcal{C})$ of this toy example, needs to be kept in order to evaluate efficiently the score of *Romantic comedy* (*i.e.*, a property that directly annotates some items and can thus be relevant for explanation). Identifying properties that are safely removable (or not) can be achieved by processing all properties in pre-order using the following recursive formula:

$$NotRemovable(p,\mathcal{C}) = \big(p \in \mathcal{P}(\mathcal{C})\big) \text{ or } \big(\exists p_p \in parents(p) | NotRemovable(p_p,\mathcal{C})\big) \quad (6)$$

3.4. Avoiding full redundancy among selected properties

Let us first clarify what we call redundancy and full redundancy among properties used for an explanation. We say that a property p is *redundant* with respect to p_1, if and only if, p subsumes p_1 in $\mathcal{H}$. For instance, *Romantic comedy* is redundant with respect to *French romantic comedy*. However, it could be worthwhile to use both those properties to explain the recommendation of a *French romantic comedy* movie if the user profile contains few *French romantic comedy* movies and many *Romantic comedy* movies. In such a case to compare the two arguments is not obvious. Indeed, would it be more meaningful for the user to explain the recommendation with "we recommend this movie because it is a *French romantic comedy* such as 3 of the movies you liked" or with "we recommend this movie because it is a *Romantic comedy* such as 8 of the movies you liked"? On the other hand, if all the *Romantic comedy* of the user profile are French ones, then the *Romantic comedy* property is *fully redundant* with respect to the *French romantic comedy* property and using it in the explanation is meaningless since it does not carry new information. More formally, we say that a property p is *fully redundant* with respect to a user profile I_u and a property p_1 if, and only if, p subsumes p_1 in $\mathcal{H}$ and $\left|\overline{\mathcal{I}(p,I_u)}\right| = \left|\overline{\mathcal{I}(p_1,I_u)}\right|$. Note that if p_1 does not directly annotate any item of the catalog (*i.e.*, $p_1 \notin \mathcal{P}(\mathcal{C})$) then p_1 will never be used within an explanation and p can thus be used without fearing redundancy with p_1. To identify full redundancy, we need to know the maximum number of user profile items annotated by a property subsumed by p:

$$M_u(p,I_u) = \max_{p_1 \in \{descendants(p) \cap \mathcal{P}(\mathcal{C})\}} \left|\overline{\mathcal{I}(p_1,I_u)}\right|.$$

Any property p such that $M_u(p,I_u) = \left|\overline{\mathcal{I}(p,I_u)}\right|$ is not considered for explanations since this equality implies that p is fully redundant with a relevant and more specific property. The $M_u(p,I_u)$ values can be efficiently computed using a post-order traversal of $\mathcal{H}_{\mathcal{C}}$ using the following recursive formula:

$$m_u(p,I_u) = \max_{p_c \in children(p)} M_u(p_c,I_u) \quad (7)$$

$$M_u(p,I_u) = \begin{cases} max\Big(m_u(p,I_u), \left|\overline{\mathcal{I}(p,I_u)}\right|\Big) & \text{if } p \in \mathcal{P}(\mathcal{C}) \\ m_u(p,I_u) & \text{otherwise} \end{cases} \quad (8)$$

3.5. The PEM method

The PEM method provides a tractable solution to identify and rank properties that can be used to explain a recommended item with respect to a user profile while considering the whole property hierarchy. The first steps of the PEM method precompute things that only depend on the item catalog $\mathcal{C}$ and the hierarchical property DAG $\mathcal{H}$ used to describe them, namely:

1. identify the set of relevant properties $\mathcal{P}(\mathcal{C})$
2. build the sub-DAG $\mathcal{H}_{\mathcal{C}}$
3. compute the number of catalog items directly ($\mathcal{I}(p,\mathcal{C})$) or indirectly ($\overline{\mathcal{I}(p,\mathcal{C})}$) described by any property $p \in \mathcal{P}(\mathcal{C})$.

Then any time an explanation should be provided for an item i_r and a user profile I_u, the following steps are carried out:

1. count the number of items of I_u that are directly ($\mathcal{I}(p,I_u)$) or indirectly ($\overline{\mathcal{I}(p,I_u)}$) described by any property $p \in \mathcal{P}(\mathcal{C})$, cf. Section 3.3
2. filter out fully redundant properties of $\mathcal{P}(\mathcal{C})$ with respect to I_u, cf. Section 3.4
3. compute the scoring of each remaining properties of $\mathcal{P}(\mathcal{C})$ and rank them accordingly, cf. the Section 3.2

An algorithm description of the proposed PEM method is provided in Appendix. The low-latency is guaranteed by the algorithmic optimization related to the sub-ontology extraction. For instance, it required less than 10 min to preprocess (cf. Algorithm 1) the whole DBpedia knowledge graph (1,639,815 properties) and the corpus of 18,983 items that we used during our experimentation. Having done once this preprocessing, each explanation is then generated in about 1 s. The above computation times were obtained on a MacBook Pro (2019, 16 GB RAM and 1.4 GHz Quad-Core Intel Core i5 Processor).

4. Experimental evaluation

This section details the assessment of our proposed property-based explanation model (PEM). As the baseline, we chose to compare PEM with the ExpLOD model which leverages *broader* properties [20]. Indeed, in [19], the authors have shown that the *basic* variant of ExpLOD outperformed other baseline approaches

and that the broader version of ExpLOD is even more effective since more indirect properties can be discovered [20]. For the sake of brevity, in the evaluation section we will use the term ExpLOD to refer to its *broader* variant.

The most common way to assess and compare recommendation explanation approaches is to carry out an online user study [22]. Our experimental evaluation is strongly inspired by those used to test most recommendation explanation systems, including ExpLOD [2,3,18]. We will first discuss the experimental protocol (*i.e.*, an overview of the conducted user study), and then we will discuss the experimental design including the used annotated item catalog and the recommendation model. We will then discuss the evaluation metrics used in our experimentation. In Section 5 the results will be presented and discussed.

4.1. Experimental protocol

To conduct our user study, we deployed a movie recommendation web application implementing the proposed PEM model and the ExpLOD model. All source code implementing these approaches and the web application can be found at this GitHub repository.[4] For demonstration purpose, we also deployed a demo web application[5] illustrating the proposed PEM method.

In a nutshell, each participant involved in the user study took the following steps (a two-stage evaluation):

1. **Collection of user demographic data**. We first asked the participant to fill a form to collect user's demographic data such as first/last name, gender and email.
2. **Generation of recommendations and explanations**. The participant was asked to select at least 5 movies from the database in order to build his/her profile (*i.e.*, I_u). Then, the content-based recommender was used to generate the top-1 most relevant movie according to that specific profile. Simultaneously, one of the two explanation models (ExpLOD and PEM) was chosen at random. The explanation for the recommended movie was then drawn based on the assigned explanation model. The number of properties used for the explanation was set to $k = 3$ for both approaches. The authors of ExpLOD discussed the impact of the number of properties used in the explanation. As the natural language explanation is based on a pre-defined template, the more properties we would consider, the longer the explanation sentence would be, which may require more user efforts and decrease his/her satisfactions. For this point, we also chose to use the top-3 ranked properties for generating the explanations.
3. **Stage-I evaluation through questionnaire**. For the recommended movie along with its explanation, the participant was asked to evaluate different evaluation metrics through a questionnaire (cf. Table 1), *i.e.*, a 5-point scale (1 = strongly disagree, 5 = strongly agree). During the stage-I, only the movie title and poster were shown to the user and the user was asked to rate how much he/she liked the movie (*i.e.*, stage-I of *effectiveness*) and other four metrics.
4. **Stage-II evaluation through questionnaire**. During the second phase, the user was asked to rate the recommended movie again after watching a trailer of the recommended movie. The closeness of the two ratings assigned to the recommended movie can then be used as a proxy for the metric effectiveness [2,3].

[4] https://github.com/lgi2p/web_app.

[5] https://rse.ceris.mines-ales.fr.

4.2. Experimental design

4.2.1. Annotated item catalog

To implement the web application for our user study, we chose the MovieTweetings dataset[6] as our item catalog, that compiles movie ratings contained in well-structured tweets on Twitter [23]. It is a daily updated dataset and has been widely used in the recommender system community. In our experimentation we only used MovieTweetings to assemble a broad movie catalog containing 35,944 movies, and did not exploit the user ratings.

As both ExpLOD and PEM require item properties from DBpedia for generating explanations, one needs first to map movie items into DBpedia entities. We applied the method described in [24] to create mappings, which consists in measuring the Levenshtein distance between movie titles and labels of DBpedia entities (rdfs:label). Items were then mapped with corresponding DBpedia entities identified by their URIs. As a result, we have mapped 18,983 movies, which has constituted the item catalog of the web application we deployed.

To collect the properties that directly annotate movies in the catalog, we chose to retrieve the genres (*i.e.*, properties indicating movie categories) for all of the mapped movies via the "dct:subject" predicate in the DBpedia knowledge graph. Note that in the experiments we did not consider other predicates such as "dbo:director", "dbo:musicComposer", etc. This was done for the following 3 main reasons. First, movie genres are crucial features that best represent the characteristics of movies and are considered a key criterion impacting users' satisfaction and experiences when using a movie recommendation system [25]. The second reason is that the properties of other predicates within DBpedia such as "dbo:director" and "dbo:writer" are often literal nodes (represented as string values) that are not useful. The last reason is that the semantics of the properties of other predicates are often repeated via the "dct:subject" predicate, *e.g.*, ⟨*The Intouchables*,[7] "dbo:musicComposer", Ludovico_Einaudi⟩ and ⟨*The Intouchables*, "dct:subject", Films_scored_by_Ludovico_Einaudi⟩.

By extracting from DBpedia the movie genres for all the movies of the catalog, we gathered 22,720 direct movie descriptions. We downloaded the dump file from the official website[8] that contains 1,639,815 nodes and used it to build a local version of the property hierarchy. A quick check revealed that this graph has several roots and contains a few unexpected cycles (*e.g.*, **Pakistan Tehreek-e-Insaf**, *"skos:broader"*, **Pakistan Tehreek-e-Insaf politicians**, *"skos:broader"*, **Imran Khan**, *"skos:broader"*, **Pakistan Tehreek-e-Insaf**). We used a basic approach to get rid of those cycles by removing some of the $4,084,504$ edges of this graph. The resulting DAG has $4,080,682$ edges. To simplify the treatment of this graph, we linked its 61,017 distinct root nodes to a single broader node labeled "Thing".

Note that our goal in this paper is to propose a fast and generic LOD-based approach providing a personalized explanation to users; in our case the movie benchmark is nothing more than a very convenient benchmark to test our approach. For reproducibility purpose, the movie mappings that we generated as well as the used local version of DBpedia category hierarchy are available in this public Zenodo [26] repository.[9]

[6] https://github.com/sidooms/MovieTweetings.

[7] https://dbpedia.org/page/The_Intouchables.

[8] http://downloads.dbpedia.org/repo/dbpedia/generic/categories/.

[9] https://doi.org/10.5281/zenodo.5122902.

4.2.2. The recommendation model

As both models (ExpLOD and PEM) are post-hoc ones that are algorithm-independent, the RS model is not the focus here. So we chose to use a content-based recommendation model based on item semantic similarities, leveraging knowledge graphs embedding techniques (see the CBF recommender of [27] for more details on the implementation of the recommendation model).

4.3. The evaluation metrics

The quality of explanation approaches can be assessed partly by some offline metrics (that do not require users' assessment), but online metrics (based on users assessment) are usually more informative.

4.3.1. Offline metrics

Using a rating dataset one could try to assess automatically the quality of the explanations that would be provided to a user based on the user's profile. Having no user-centric feedbacks, the assessment is limited, however some insights can be gained from such offline metrics. For instance, a method that often fails to provide any explanation or provides very uninformative ones is clearly not ideal.

For the capacity of a method to produce explanations, we refer to the *explainability precision* metric [28]. It represents typically the proportion of explainable items within the top-n recommendation list relative to the number of recommended items (*i.e.*, n) and is denoted as *EPrec@n*.

Another aspect that we would like to consider here is the information content conveyed by the properties used in the explanation. In general, the information content (IC) of a concept (or property in our case) provides an estimation of its degree of generality/concreteness, that is used to enable a better understanding of the concept's semantics [29]. The more general a property is, the less information it conveys, *e.g.*, *Films*, *Creative works*. The IC of a property is usually defined based on its position in the ontology hierarchy (intrinsic definition) or based on the number of items it indirectly describes (extrinsic definition). As our approach relies on the number of items indirectly described by a property, we prefer to rely on an intrinsic IC definition to avoid bias. Following Eq. (9) of [29] we thus use the following IC metric:

$$IC(p) = -\log\left(\frac{|\text{leaves}(p)+1|}{|\text{leaves}(\mathcal{H})|}\right) \tag{9}$$

where leaves(p) is the set of properties subsumed by p, which are leaves; and leaves($\mathcal{H}$) is the total number of leaves in the property DAG.

4.3.2. Online metrics

The recommendation explanations should increase the trust users have in the RS and favor their interactions. A good explanation should lead users to (i) better understand the recommendation (*Transparency*); (ii) be more convinced by the recommendation (*Persuasiveness*); (iii) learn new information (*Engagement*); (iv) trust the RS more (*Trust*) and (v) better assess the quality of the recommended item (*Effectiveness*), according to [2,3]. It has been proposed to evaluate these metrics through a classic user-study protocol executed in two stages as stated in Section 4.1. Given the poster of the recommended movie and the proposed explanation, the user is asked to use a 5-point scale (1 = strongly disagree, 5 = strongly agree) to answer the five questions listed in Table 1. The user is then asked to rate the recommended movie again after having watched its trailer. The difference between the movie assessments before and after seeing the trailer is considered to be a proxy of the *Effectiveness* of the explanation. A change of rating might indicate that the trailer provides new relevant information that was missing from the explanation, whereas a stable rating would indicate that the explanation was comprehensive enough. This protocol has since been used in numerous studies [2,3,18–20].

5. Results

In this section we will examine the results of the user study carried out on a panel of 155 subjects (male = 55%), each one evaluating the explanation of one movie described by DBpedia properties. First, we will discuss some indicators concerning the sub-ontology optimization; we will then provide the evaluation metrics measured during the online user study and finish by detailing a toy example that sheds light on the differences between ExpLOD and PEM in terms of selecting properties.

5.1. The efficiency of the sub-ontology optimization

While the DBpedia knowledge graph contains 1,639,815 properties, only 22,720 (1.38%) of them are describing one of the 18,983 movies of our benchmark dataset directly. Given this whole ontology and the set of movie annotations that we extracted, we then applied the method proposed in our previous work [14] to extract a sub-ontology that relating the movie domain. The resulting sub-ontology contains 28,424 (1.73%) properties. Excluding from this sub-ontology those without any ancestors (including themselves) that directly annotate a movie allows us to reduce the property hierarchy to be considered, leading to a DAG of 28,016 (1.70%) properties. For the ExpLOD model, the whole ontology was used when generating the explanations. Using the sub-ontology allowed us to develop a low-latency solution while dealing with the very large DBpedia property hierarchy. Note that this approach is not restricted to the case where properties annotating items directly are the sole ones to be considered. Indeed, if a property p is not in the sub-DAG $\mathcal{H}_{\mathcal{C}}$, then it can be proven that an alternative property $p' \in \mathcal{H}_{\mathcal{C}}$ exists such that (i) p' describes all items described by p and (ii) p' is more informative than p. By definition of the sub-ontology it is guaranteed that the least common ancestor of any subset of $\mathcal{P}(\mathcal{C})$ is in $\mathcal{H}_{\mathcal{C}}$. Considering the subset of $\mathcal{P}(\mathcal{C})$ made of p's descendants, the least common ancestor p' of this set then describes all the items described by p while being more informative (*i.e.*, the fold change ratio of p' in Eq. (4) is at least as good as the one of p).

The source code implementing the sub-ontology extraction process is available at the GitHub repository.[10]

5.2. Performance comparison of the ExpLOD and PEM methods on a user study

Among those 155 explanations, 82 have been generated using the ExpLOD approach and 73 using the proposed PEM method. These 155 user profiles and explanations were also used to evaluate offline metrics.

Let us first consider the comparison results for offline metrics. In terms of the *explainability precision* (see Section 4.3.1), both approaches have an optimal *EPrec*@1 = 1. If a larger set of items has to be explained then *EPrec*@5 = 0.7 decreases for ExpLOD whereas it remains equals to 1 for PEM. In other words, if each of the 82 subjects (associated with ExpLOD) was shown with a top-5 recommendation list (*i.e.*, 410 items to be explained in total), then only 287 out of the 410 items were explainable. This essentially reflects the difference of goals of the two methods; when several items have to be explained simultaneously, ExpLOD focuses on factorizing explanations rather than on producing the best

[10] https://github.com/lgi2p/Sub-Ontology_Extraction.

Table 1
Questionnaire details.

Metric (Stage)	Corresponding question in user study
Transparency (I)	"I understood why this item was recommended to me"
Persuasiveness (I)	"The explanation made the recommendation more convincing"
Engagement (I)	"The explanation helped me discover new information about this item"
Trust (I)	"The explanation increased my trust in the recommender system"
Effectiveness (I, II)	"I like this recommendation"

Table 2
Results of the user study. Better metrics are bolded and statistically significant improvements (with $p < 0.05$) are emphasized with (*).

	Transparency	Persuasiveness	Engagement	Trust	Effectiveness	IC
ExpLOD	3.91	2.83	2.45	2.62	0.40	7.49
PEM	**4.21**	**3.45**(*)	**3.16**(*)	**3.30**(*)	**0.34**	**10.42**(*)

explanation for individual items. As this group explanation task is beyond the scope of this paper, we will focus our remaining comparisons on explanations produced by ExpLOD for a single item (*i.e.*, $I_r = \{i_r\}$), as such, both approaches have the same objective to produce the best explanation for i_r.

In terms of the *information content* (IC) of properties, we computed the corresponding values using Eq. (9). The average IC of the 246 properties used within the 82 ExpLOD explanations (as each explanation used 3 properties) is 7.49 whereas it is 10.42 for the 219 properties used within the 73 PEM explanations, as illustrated in Table 2. The results demonstrate that the explanations of our proposed approach are more informative.

Considering the result comparison for online metrics. For each of the 5 online metrics listed in Table 1, we measured the average value of user feedback collected over the 82 ExpLOD user evaluations and the 73 PEM ones. All those results are provided in Table 2. On average, PEM was better evaluated than ExpLOD on all metrics. To assess the significance of those average metric differences, we used *Mann–Whitney U Test* to compare the 6 metrics values among all participants. The differences are significant (with $p < 0.05$) for *Persuasiveness*, *Engagement*, *Trust* as well as for the average property *information content* (IC). The result files of this user study are available in an open-access Zenodo repository.[11]

5.3. A concrete comparison example

In this subsection, we use a toy example to shed light on the differences between ExpLOD and PEM (Fig. 3). The recommended movie (*Bitter Moon*[12]) has to be explained based on a user profile designed from three movies, *The Intouchables*,[13] *Welcome to the Sticks*[14] and *Amélie*.[15] The top-3 properties according to our proposed PEM method are *French comedy films*, *Films shot in France* and *French-language films*. None of those properties is in the top-3 properties ranked by ExpLOD. The first one is considered by ExpLOD but not ranked in the top-3 properties whereas the last two properties are not even considered since they are not part of the *broader* property set considered by ExpLOD. The top-3 properties selected by ExpLOD on this example are *Film scores by composer*, *Films by director* and *Films by French directors*. The first two properties are quite uninformative (and ignored by the PEM method since they do not directly annotate any catalog items). Note that the *Films by director* property is also *fully redundant* with respect to the third property *Films by French directors*, selected by ExpLOD.

5.4. Discussion and limitations

In this article, we proposed a post-hoc explanation model that generates explanations based on item properties available in the DBpedia knowledge graph. The proposed method improves existing post-hoc approaches by considering the entire property hierarchy and eliminating irrelevant and redundant properties in the explanation. Moreover, taking advantage of the sub-ontology extraction makes the calculations more efficient and our model more flexible. However, the proposed model has the following main limitations.

As mentioned in Section 5.2, the goal of the proposed PEM explanation model is to focus on one single explanation at a time, which might represent a major limitation when multiple (or a group of) items need to be explained simultaneously. In such a scenario, PEM would compute sequentially the explanations item by item, thus it would not be efficient in terms of the calculation time. One possible way to tackle this issue is to provide users with the possibility of asking for explanations by their own needs. The idea is that not each of the recommended items would need an explanation for the user, *i.e.,* some of the recommendations might be already familiar for the user. The compared ExpLOD approach treats the recommendation list as a whole and uses the top-k most relevant properties to explain at the same time the items within the entire list. However, as discussed in Section 5.2, the drawback is that the *explanation precision* decreases when the length of the recommendation list gets larger, *i.e.,* the number of explainable items of the list decreases for ExpLOD.

From a more general point of view, the post-hoc explanation approaches based on item properties share a common limitation: they rely solely on the quality of knowledge encoded in the knowledge graphs such as DBpedia used in our approach. The completeness and the correctness of the knowledge and facts encoded in these KGs are essential for identifying relevant properties for explanations. However, as pointed out by Färber et al. in [30], the famous knowledge graphs such as DBpedia may contain a large amount of missing facts and sometimes incorrect ones (*e.g.*, the circles in the DAG as shown in Section 4.2.1). To alleviate

11 https://doi.org/10.5281/zenodo.5122902.
12 https://dbpedia.org/page/Bitter_Moon.
13 https://dbpedia.org/page/The_Intouchables.
14 https://dbpedia.org/page/Welcome_to_the_Sticks.
15 https://dbpedia.org/page/Am%C3%A9lie.

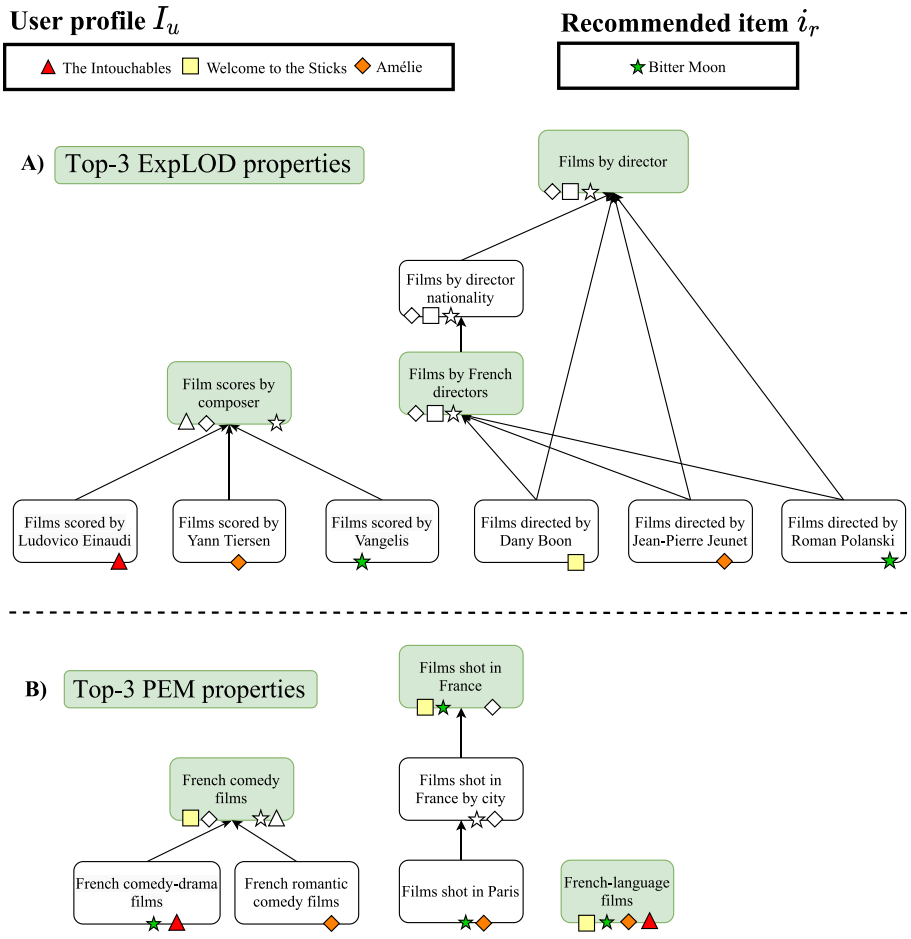

Fig. 3. A concrete example comparing the *broader* variant of the ExpLOD (A) and PEM (B) approaches. User profile movies are represented by red, yellow and orange symbols while the green star refers to the recommended movie to explain. White geometric symbols represent indirect movie descriptions induced by the property hierarchy. The top-3 ranked properties for each approach are highlighted in green.

this issue related to the quality of the KGs, one may consider to take advantage of other types of data and information other than the item properties, such as the user–item interactions, when generating the explanations. It is also one of our perspectives, as illustrated in the following section.

6. Conclusions & perspectives

The explanation of recommendations is becoming an important feature of recommender systems as it provides users with extra information and helps them make faster and better decisions.

In this paper, we propose a generic, personalized post-hoc approach exploiting Linked Open Data. To be more specific, Linked Open Data generally relies on a backbone hierarchy organizing a huge number of concepts so that the *broader* or *is-a* relationship among these concepts can be explored. Thanks to algorithmic optimizations relying on the sub-ontology extraction and post-order graph traversal, we demonstrate that this whole hierarchy can be efficiently exploited. We also provide simple solutions to limit explanation redundancy in the proposed approach as well as the use of properties that are helpful to organize the hierarchy but irrelevant for explanation. The user study carried out confirms the advantage of our proposed approach as compared to existing LOD-based approaches such as ExpLOD which does not consider the whole property hierarchy and does not explicitly take into account the property relevance and possible redundancy issues.

Another type of personalized explanation approaches exist that ignore item contents and only take advantage of the user–item rating matrix [31]. Those methods generate post-hoc explanations based on user–item interactions (collaborative data), leading to explanations such as "we recommend the item x because you liked the item y and 80% of users liking y also like x".

We believe that the two types of approaches are not exclusive. Depending on the context, the user could be more convinced by either a LOD-based explanation or by a collaborative-based approach. If the LOD-based explanation is based upon very generic properties it would be better to rely on a collaborative one; conversely if the collaborative explanation relies on a very low percentage the LOD-based explanation is probably preferable. One can therefore consider switching between those strategies depending on their output or even combining their output into a hybrid explanation mixing both points of view.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank Pierre Jean (IMT Mines Ales) for deploying the web applications (both for the user study and the demonstration purpose) on the IMT Mines Ales, server.

Appendix. Algorithm description of the proposed PEM method

Algorithm 1: Property_frequency (auxiliary method)

Input : an item set I, a post-ordered property list *postO*, an annotation function $\mathcal{P}(\cdot)$
Output: *I_expl*, *I_impl* // dictionaries providing the number of items in I that are explicitly, resp. implicitly, described by properties of *postO*

// collect the set of items annotated by each property
I_impl = *I_expl* = EmptyDictionnary;
for *item i in I* **do**
  for *property p in* $\mathcal{P}(i)$ **do**
    I_expl[p].add(*i*);
  end
end
for *property p in postO* **do**
  I_impl[p] = *I_expl*[p];
  for *property* p_c *in p.children*() **do**
    I_impl[p].add(*I_impl*[p_c]); // Eq. (5)
  end
end
// count the number of items annotated by each property
for *(p, p_item_set)* ***in*** *I_expl* **do**
  I_expl[p] = *p_item_set*.size();
end
for *(p, p_item_set)* ***in*** *I_impl* **do**
  I_impl[p] = *p_item_set*.size();
end
return *I_expl*, *I_impl*;

Algorithm 2: PEM_preprocessing (This is done only once)

Input : an item catalog $\mathcal{C}$, a property hierarchy $\mathcal{H}$, an annotation function $\mathcal{P}(\cdot)$
Output: $\mathcal{H}_\mathcal{C}$ // the sub-hierarchy of properties
$\mathcal{H}_\mathcal{C}$_po // list of the sub-hierarchy properties in post ordering
C_expl, *C_impl* // dictionaries providing for each property of $\mathcal{H}_\mathcal{C}$_po, the number of items in $\mathcal{C}$ that it explicitly, resp. implicitly, describes

// identify properties annotating explicitly(directly) at least 1 item of $\mathcal{C}$
$\mathcal{P}_\mathcal{C}$ = EmptySet;
for *item i in* $\mathcal{C}$ **do**
  for *property p in* $\mathcal{P}(i)$ **do**
    $\mathcal{P}_\mathcal{C}$.add(*p*);
  end
end
// compute the sub-hierarchy sufficient to fully handle properties of $\mathcal{P}_\mathcal{C}$
$\mathcal{H}_\mathcal{C}$ = subOntology($\mathcal{H}$, $\mathcal{P}_\mathcal{C}$); // Algorithm 5 of [14]
// get a list of its properties in post order
$\mathcal{H}_\mathcal{C}$_po = postOrderTraversal($\mathcal{H}_\mathcal{C}$); // Algorithm 3 of [14]
// get explicit and implicit annotation counts at the catalog level
C_expl, *C_impl* = Property_frequency($\mathcal{C}$, $\mathcal{H}_\mathcal{C}$_po, $\mathcal{P}(\cdot)$);
return $\mathcal{H}_\mathcal{C}$, $\mathcal{H}_\mathcal{C}$_po, *C_expl*, *C_impl*

Algorithm 3: PEM_explanation (proposed method of selecting the top-*k* properties — Run-time computation, *i.e.*, this is called whenever a user request is received)

Input : a user profile I_u, a target item i_r, an item catalog $\mathcal{C}$, an annotation function $\mathcal{P}(\cdot)$, the number *k* of top properties to select, and the four outputs of Algorithm 2 (PEM_preprocessing) *i.e.*, $\mathcal{H}_\mathcal{C}$, *C_expl*, *C_impl* and $\mathcal{H}_\mathcal{C}$_po
Output: the top-*k* properties for the explanation of i_r

// identify properties annotating the target item i_r
i_r_expl, i_r_impl = Property_frequency({i_r}, $\mathcal{H}_\mathcal{C}$_po, $\mathcal{P}(\cdot)$);
// identify properties annotating at least 1 item of I_u
I_u_expl, I_u_impl = Property_frequency(I_u, $\mathcal{H}_\mathcal{C}$_po, $\mathcal{P}(\cdot)$);
// identify the set of candidate properties
CandidateProperties = setIntersection(i_r_impl.keys(), I_u_impl.keys(), *C_expl*.keys());
// remove fully redundant properties // Eq. (8)
mu = Mu = emptyDictionnary;
for *property p in* $\mathcal{H}_\mathcal{C}$*_po* **do**
  mu[p] = 0;
  for *property* p_c *in p.children*() **do** mu[p] = max(mu[p], Mu[p_c]);
  if *mu[p]* == I_u*_impl*[p] **then**
    CandidateProperties.remove(*p*);
  if *p in C_expl.keys()* **then**
    Mu[p] = max(mu[p], I_u*_impl*[p]);
  else
    Mu[p] = mu[p]
  end
end
for *property p in CandidateProperties* **do**
  fold_change = (I_u*_impl*[p]/I_u.size())/(*C_impl*[p]/*C*.size());
  p.score = $\log_{10}$(*C_expl*[p]) * fold_change; // Eq. (4)
end
CandidateProperties.sortByScore();
return the top-*k* elements of CandidateProperties

References

[1] F. Ricci, L. Rokach, B. Shapira, Recommender systems: introduction and challenges, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 1–34, http://dx.doi.org/10.1007/978-1-4899-7637-6_1.
[2] F. Gedikli, D. Jannach, M. Ge, How should i explain? A comparison of different explanation types for recommender systems, Int. J. Hum.-Comput. Stud. 72 (4) (2014) 367–382, http://dx.doi.org/10.1016/j.ijhcs.2013.12.007.
[3] N. Tintarev, J. Masthoff, Evaluating the effectiveness of explanations for recommender systems, User Model. User-Adapt. Interact. 22 (4-5) (2012) 399–439, http://dx.doi.org/10.1007/s11257-011-9117-5.
[4] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, Found. Trends Inf. Retr. 14 (1) (2020) 1–101, http://dx.doi.org/10.1561/1500000066.
[5] P. Lops, D. Jannach, C. Musto, T. Bogers, M. Koolen, Trends in content-based recommendation, User Model. User-Adapt. Interact. 29 (2) (2019) 239–249, http://dx.doi.org/10.1007/s11257-019-09231-w.
[6] W. Lin, S.A. Alvarez, C. Ruiz, Collaborative recommendation via adaptive association rule mining, Data Min. Knowl. Discov. 6 (1) (2000) 83–105.
[7] J.L. Herlocker, J.A. Konstan, J. Riedl, Explaining collaborative filtering recommendations, in: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, in: CSCW '00, 2000, pp. 241–250, http://dx.doi.org/10.1145/358916.358995.
[8] Y. Koren, R. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 77–118, http://dx.doi.org/10.1007/978-0-387-85820-3_5.
[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, in: WWW '17, 2017, pp. 173–182, http://dx.doi.org/10.1145/3038912.3052569.

[10] M.F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? A worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, in: RecSys '19, 2019, pp. 101–109, http://dx.doi.org/10.1145/3298689.3347058.

[11] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38, http://dx.doi.org/10.1145/3285029.

[12] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, S. Ma, Explicit factor models for explainable recommendation based on phrase-level sentiment analysis, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, 2014, pp. 83–92, http://dx.doi.org/10.1145/2600428.2609579.

[13] C. Musto, M. de Gemmis, P. Lops, G. Semeraro, Generating post hoc review-based natural language justifications for recommender systems, User Model. User-Adapt. Interact. 31 (2021) 629–673, http://dx.doi.org/10.1007/s11257-020-09270-8.

[14] V. Ranwez, S. Ranwez, S. Janaqi, Subontology extraction using hyponym and hypernym closure on is-a directed acyclic graphs, IEEE Trans. Knowl. Data Eng. 24 (12) (2011) 2288–2300, http://dx.doi.org/10.1109/TKDE.2011.173.

[15] C. Bizer, T. Heath, K. Idehen, T. Berners-Lee, Linked data on the web (LDOW2008), in: Proceedings of the 17th International Conference on World Wide Web, in: WWW '08, 2008, pp. 1265–1266, http://dx.doi.org/10.1145/1367497.1367760.

[16] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The Semantic Web, Springer, 2007, pp. 722–735, http://dx.doi.org/10.1007/978-3-540-76298-0_52.

[17] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, in: SIGMOD '08, 2008, pp. 1247–1250, http://dx.doi.org/10.1145/1376616.1376746.

[18] V. Lully, P. Laublet, M. Stankovic, F. Radulovic, Enhancing explanations in recommender systems with knowledge graphs, Procedia Comput. Sci. 137 (2018) 211–222, http://dx.doi.org/10.1016/j.procs.2018.09.020.

[19] 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, in: RecSys '16, 2016, pp. 151–154, http://dx.doi.org/10.1145/2959100.2959173.

[20] C. Musto, F. Narducci, P. Lops, M. de Gemmis, G. Semeraro, Linked open data-based explanations for transparent recommender systems, Int. J. Hum.-Comput. Stud. 121 (2019) 93–107, http://dx.doi.org/10.1016/j.ijhcs.2018.03.003.

[21] M.I. Love, W. Huber, S. Anders, Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2, Genome Biol. 15 (12) (2014) 1–21, http://dx.doi.org/10.1186/s13059-014-0550-8.

[22] I. Nunes, D. Jannach, A systematic review and taxonomy of explanations in decision support and recommender systems, User Model. User-Adapt. Interact. 27 (3) (2017) 393–444, http://dx.doi.org/10.1007/s11257-017-9195-0.

[23] S. Dooms, T. De Pessemier, L. Martens, Movietweetings: a movie rating dataset collected from twitter, in: Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys 2013, 2013, p. 43.

[24] T. Di Noia, R. Mirizzi, V.C. Ostuni, D. Romito, M. Zanker, Linked open data to support content-based recommender systems, in: Proceedings of the 8th International Conference on Semantic Systems, in: I-SEMANTICS '12, 2012, pp. 1–8, http://dx.doi.org/10.1145/2362499.2362501.

[25] T. Nanou, G. Lekakos, K. Fouskas, The effects of recommendations' presentation on persuasion and satisfaction in a movie recommender system, Multimedia Syst. 16 (4–5) (2010) 219–230, http://dx.doi.org/10.1007/s00530-010-0190-0.

[26] European organization for nuclear research, OpenAIRE, zenodo, 2013, http://dx.doi.org/10.25495/7GXK-RD71, URL https://www.zenodo.org/.

[27] Y. Du, S. Ranwez, N. Sutton-Charani, V. Ranwez, Is diversity optimization always suitable? Toward a better understanding of diversity within recommendation approaches, Inform. Process. Manage. 58 (6) (2021) 102721, http://dx.doi.org/10.1016/j.ipm.2021.102721.

[28] B. Abdollahi, O. Nasraoui, Using explainability for constrained matrix factorization, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, in: RecSys '17, 2017, pp. 79–83, http://dx.doi.org/10.1145/3109859.3109913.

[29] D. Sánchez, M. Batet, D. Isern, Ontology-based information content computation, Knowl.-Based Syst. 24 (2) (2011) 297–303, http://dx.doi.org/10.1016/j.knosys.2010.10.001.

[30] M. Färber, F. Bartscherer, C. Menne, A. Rettinger, Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago, Semantic Web 9 (1) (2018) 77–129, http://dx.doi.org/10.3233/SW-170275.

[31] G. Peake, J. Wang, Explanation mining: Post hoc interpretability of latent factor models for recommendation systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, in: KDD '18, New York, NY, USA, 2018, pp. 2060–2069, http://dx.doi.org/10.1145/3219819.3220072.