ExpLOD: a Framework for Explaining Recommendations based on the Linked Open Data Cloud

Cataldo Musto Fedelucio Narducci Pasquale Lops
Marco De Gemmis Giovanni Semeraro

Dept. of Computer Science - University of Bari 'Aldo Moro - Italy
name.surname@uniba.it

ABSTRACT

In this paper we present ExpLOD, a framework which exploits the information available in the Linked Open Data (LOD) cloud to generate a *natural language explanation* of the suggestions produced by a recommendation algorithm. The methodology is based on building a graph in which the items liked by a user are connected to the items recommended through the properties available in the LOD cloud. Next, given this graph, we implemented some techniques to rank those properties and we used the most relevant ones to feed a module for generating explanations in natural language.

In the experimental evaluation we performed a user study with 308 subjects aiming to investigate to what extent our explanation framework can lead to more transparent, trustful and engaging recommendations. The preliminary results provided us with encouraging findings, since our algorithm performed better than both a non-personalized explanation baseline and a popularity-based one.

1. BACKGROUND AND MOTIVATIONS

The performance of a recommender system is generally evaluated on the ground of its capability of predicting items the user would probably like. Accordingly, evaluation metrics usually reward systems that maximize the predictive accuracy. However, especially when unknown items are proposed, the system should help the user to make an informed choice instead of just proposing a list of items based on an obscure reasoning mechanism [4].

Tintarev and Masthoff [8] point out that *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 item. Accordingly, they define seven possible aims for explanation, namely: *transparency, scrutabilty, trust, effectiveness, persuasiveness, efficiency, satisfaction.* The authors also demonstrate that a personalized explanation strategy generally works better than a generic one. Typically, the explanations are grouped in three classes according to the information they exploit: preferences of similar users (e.g. *customers who bought this item also bought...*), similar items (e.g., *this item is similar to other items you liked*), attributes of interest (e.g., *this item has attributes you prefer*). However, there is not a general assessment on which approach works best, since a strategy can maximize an aspect rather than another.

In this work we investigated five of the aforementioned aims by implementing a personalized explanation strategy exploiting both *similar items* and *attributes of interest* through a graph-based recommendation model. Our explanation framework is based on the Linked Open Data cloud, and more specifically on the properties encoded in DBpedia[1] to link the items the user liked with the recommended ones. These properties are thus used for generating textual explanations. A similar attempt is due to [5], who demonstrated that a graph-based explanation results in a better user experience, However, they focused on the usage of graphs for visualization purposes, while our contribution exploits graphs for modeling the properties describing the items and identifying the most relevant ones to be used in a natural language explanation.

In the experimental session we evaluated the effectiveness of our framework against several baselines through a user study. Moreover, we also investigated the impact of the single features on the overall explanation. This research line has been already investigated in the past, and several authors [2, 4] showed a clear relationship between some features (as the favorite *actor* in a movie recommendation scenario) and the effectiveness of the explanation.

The rest of the article is organized as follows: in Section 2 describes the framework designed to generate personalized explanations. The description of the experiments carried out to evaluate the effectiveness of our strategy is given in Section 3. Finally, Section 4 concludes this work by discussing the main findings and defining the future research directions.

2. DESCRIPTION OF THE FRAMEWORK

The workflow carried out by ExpLOD is depicted in Figure 1. The framework is organized in four main modules: Data Mapper, Graph Builder, a property ranker and a generator.

First, the Data mapper implements mechanisms to map each item with an element in the LOD cloud. As an example,

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

RecSys '16, September 15-19, 2016, Boston , MA, USA
© 2016 ACM. ISBN 978-1-4503-4035-9/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2959100.2959173

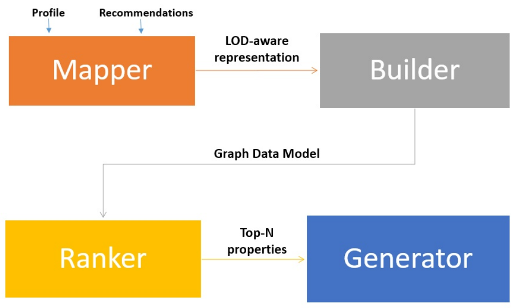

Figure 1: Organization of the framework

if a user is interested in the movies *The Matrix* and *Cloud Atlas*, the module will find the `DBpedia` nodes they refer to. In our case, the mapping is carried out by implementing a *shallow* matching of the name of the item with the title of the Wikipedia page the `DBpedia` node is linked to, but more complex mapping methods may be easily implemented. The output of this step is a list in which all the items in the user profile and all the items in her recommendation list are mapped with an URI (if any) in the LOD cloud. This step is mandatory since the explanations generated by the system will rely on a subset of the properties available in the LOD cloud.

Next, those LOD-aware item representations are used to feed the graph BUILDER, which exploits the information available in the LOD cloud to build a graph-based data model. Formally, let I_p 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. Next, we define $P = \{p | i \in I_r \wedge exists(i, p) = true\}$ as the set of the values of the properties available in the LOD cloud describing the items the user received as recommendations. Given such a representation, the module builds a bipartite graph $G = (N, E)$, where $N = I_p \cup I_r \cup P$ and $E = \{(i, p) | i \in I_p \cup I_r \wedge p \in P\}$. Basically, this graph connects the items the user liked and those in her recommendation list through the values of the properties describing those items in the LOD cloud. A visual explanation of the data model is provided in Figure 2. Elements in I_p, I_r and P are reported in blue, red and green, respectively. In this case, the user liked *Saving Private Ryan*, *The Matrix* and *The Da Vinci Code*, and received as recommendation the movie *Cloud Atlas*. Accordingly, four properties connecting the movies in the profile to the recommendation are added as nodes in the graph[2]. We label them as *candidate properties*, since they represent the preliminary set of properties which *could* explain the recommendation received by the user.

Next, the property RANKER analyzes such a data model to identify the subset of *candidate properties* which are likely to explain the recommendation. To this end, the module assigns a *relevance score* to each property, based on the insight that a *good* explanation should emphasize those properties which can describe the current recommendation on the ground of the items the user liked. Given a candidate

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

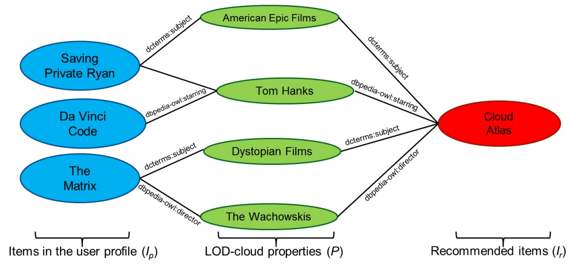

Figure 2: ExpLOD Data Model.

property c, the property RANKER calculates its score as:

$$score(c, I_p, I_r) = (\alpha \frac{n_{c,I_p}}{|Ip|} + \beta \frac{n_{c,I_r}}{|Ir|}) * IDF_c \qquad (1)$$

where n_{c,I_p} is the number of edges connecting c with the items in the user profile, n_{c,I_r} is the number of edges connecting c with the items in the recommendation set, α and β are two weighting factors and IDF_c is an adaptation over `DBpedia` of the classical Inverse Document Frequency [7], 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_p and I_r and which are not so common as well. Through this calculation, all the candidate properties are provided with a relevance score and the *top-K* are used to generate the explanation. By following the example in Figure 2, *Tom Hanks* is the most relevant property explaining the recommendation since it occurs in two out of three movies in the user profile. Next, it is likely that the IDF component of the formula would help to rank as second property *Dystopian Movies* which is surely a less common one than *The Wackowski* and *American Epic Films.* It is worth to note that our ranking formula is independent of the size of the recommendation list, so it can produce a single explanation also for a larger set of recommendations. When $|I_r| > 1$, the algorithm will rank first those properties which are common to many items in the profile and can also describe many items in the recommendation list.

Finally, once the *top-K* properties have been returned, the GENERATOR module builds a natural-language explanation supporting the recommendation. The basic idea is to use the properties returned by the RANKER to fill a *template-based* structure which generates the final explanation. As an example, the explanation generated for the data model provided in Figure 2 is: *"I recommend you* CLOUD ATLAS *since you often like movies starred by* TOM HANKS *as* DA VINCI CODE *and* SAVING PRIVATE RYAN. *Moreover, I recommend it because you sometimes like* DYSTOPIAN MOVIES *as* THE MATRIX *and* AMERICAN EPIC FILMS *as* SAVING PRIVATE RYAN*"*.

Text in *small caps* refers to the elements coming from the LOD cloud. Adverbs as *often* or *sometimes* 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, expressions as *starred by* are obtained by mapping each property in the LOD cloud (in this case, DBPEDIA:OWL:STARRING) with a natural language expression. This is done for all the properties available in the LOD cloud, as DBPEDIA:OWL:DIRECTOR, DBPEDIA:OWL:MUSICCOMPOSER and so on. Due to space reasons it is not possible to provide further details about this module. However, even this description showed how the combination of techniques for generating natural-language expressions with the richness of the information available in the LOD cloud let the algorithm produce very meaningful explanations supporting the recommendation set.

3. EXPERIMENTAL EVALUATION

The goal of the experimental evaluation was twofold: to understand to what extent our explanation framework may lead to more transparent, trustful or engaging recommendations (Experiment 1) and to investigate whether a correlation exists between the choice of the properties used to generate the explanation and the effectiveness of the explanation itself (Experiment 2). To this end, we designed a user study in the movie domain by involving 308 subjects (male=70%, degree or PhD=62%, medium interest in movies). We evaluated the following explanation aims: *transparency*, *persuasiveness*, *engagement*, *trust* and *effectiveness*, as in [8].

Experimental Protocol. To run the experiment, we deployed a web application[3] implementing the previously described framework. The platform was designed to run a *between-subject* experiment, i.e. we tested four different explanation styles and each user was randomly assigned to one of them. As explanations styles we defined two versions of the previously presented natural language-based explanations (by ranking the properties with and without IDF) and two baselines, a *popularity-based explanation* and a *non-personalized explanation style* based on the presentation of the movie properties extracted from `DBpedia`. Those baselines were already adopted in literature [8] for similar experimental protocols. In a nutshell, each user involved in the experiment carried out the following steps:

(1) Collection of Demographic Data and Preference Elicitation. We asked the users to provide some basic demographic data. Then, each user provided her preferences in the movie domain: a small portion of the users (around 5%) authorized our application to extract preferences from their Facebook profiles, while the other explicitly rated a randomly generated subset of 20 movies extracted from the top-100 popular movies in IMDB. Once the profiles were built, recommendations were generated by running Personalized PageRank [3] as recommendation algorithm.

(2) Generation of the Explanations. We used the preferences of the users and the Top-1 recommendation generated by the algorithm to feed our framework. When the *popularity-based* baseline was picked as explanation style, each user was provided with a simple explanation as *"We suggest this item since it is very popular among people who like the same movies as you."*. On the other side, as *non-personalized explanation style* we exploited the properties encoded in `DBpedia` to build a structured description of the movie used as explanation (e.g. starring, director, music composer, subject, and so on) without any filtering or ranking of the properties. Finally, when ExpLOD was randomly picked, we run our framework as previously shown. In this setting, the Data Mapper was able to map to `DBpedia` 90,318 movies gathered from IMDB. The mapping is available online[4]. Next, Graph Builder queried `DBpedia` to extract the properties describing the movies. It is worth to note that all the available properties were extracted, without performing any filtering. Finally, the ranking formula was set by choosing 0.5 as weighting factor for α and β and the top-3 properties were returned. All these values were roughly set through some simple heuristics. In the future we will also investigate the impact of a proper tuning of the parameters on the overall results.

(3) Evaluation through Questionnaires. Finally we asked the users to fill a questionnaire to evaluate the quality of the different explanation styles. Each user was asked to evaluate the previously presented *explanation aims* through a five-point rating scale and to evaluate how much she liked that suggestion. The questions the users had to answer are presented in Table 1. Finally, in the last part of the experiment, each user had to enjoy a trailer of the movie and had to evaluate again the movie after watching the trailer. The whole experiment took less than 3 minutes. A screen recording showing a demo of the experiment is available online[5].

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 pre- and post-trailer ratings the user provided for the recommendation. This is an evaluation protocol which is very common in literature [1], 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 77 observations were collected. Given that in [6] 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.

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 1: Details of the Questionnaire

4. DISCUSSION AND CONCLUSIONS

Results of Experiment 1 are reported in Table 2. Results show that our framework is able to significantly outperform both the baselines (statistical significance has been assessed through a *Mann-Whitney U Test*) for most of the explanation aims we evaluated ($p < 0.001$). Specifically, configurations based on ExpLOD are able to provide users with more *transparent*, *persuasive* and *trustful* recommendations. Our framework also obtained the best results in terms of *engagement*, but a statistically significant improvement was noted only over the simple popularity-based baseline. The only metrics that did not benefit of the adoption of the explanation generated by our framework is the *effectiveness*, since in this case the best results are obtained by

[3]http://193.204.187.192:8080/WebLodrecsys/Benvenuto.jsp
[4]http://www.di.uniba.it/ swap/explod/mapping.txt

[5]http://bit.ly/1WEPhc4

the non-personalized explanation based on the structured presentation of LOD properties. This result may be due to the fact that also a simple summary reporting the most relevant properties of the movies may be helpful to understand whether the user will like it or not. However, we did not notice any statistical significance, so we can state that our framework can be as effective as the non-personalized baseline on this task. On the other side, when more complex aspects as the trust or the persuasion are taken into account, our strategy is the only one which can significantly outperform the baseline. Finally, as regards the use of the IDF, experiments did not show any significant difference so it is possible to state that its use does not influence the overall performance of the explanation.

Table 2: Results of Experiment 1. The best-performing configuration is in bold. Significant improvements over the baselines are in italics and bold.

	transp.	pers.	engag.	trust	eff.
popularity	3.01	2.59	2.31	2.67	0.93
LOD properties	3.04	2.84	3.28	2.81	**0.66**
ExpLOD-noIDF	*4.00*	3.39	***3.48***	***3.39***	0.72
ExpLOD	***4.18***	**3.41**	*3.31*	*3.36*	0.75

In Experiment 2, we investigated whether some relationship between the choice of particular properties and the overall performance of our explanation framework exists. Indeed, it is likely that in some scenarios it would be useful to generate explanations able to maximize a specific explanation aim, thus it is worth to know whether to use (or to ignore) a specific property may optimize a specific aspect. As an example, in a movie recommendation scenario it would be good to *persuade* the user watching that movie. To this end, we split the data we collected during the first experiment on the ground of the properties selected by *ExpLOD*. Next, we calculated again the average transparency, persuasion, engagement, trust and effectiveness by *only* considering those explanations which contained a specific property, as the *director* of the movie, the *producer*, the *music composer*, and so on.

Table 3: Results of Experiment 2: average improvement (or average decrease) over the results presented in Table 1 are reported in parenthesis.

	positive	**negative**
transparency	topic (+12%) director (+3%)	distributor (-11%) composer (-13%)
persuasion	won (+15%) director (+12%)	location (-8%) producer (-9%)
engagement	writer (+25%) director (+20%)	producer (-13%) distributor (-3%)
trust	won (+21%) composer (+5%)	producer (-4%) topic (-9%)
effectiveness	director (+77%) writer (+43%)	location (-71%) composer (-50%)

Due to space reasons, we reported in Table 3 only the most interesting findings emerging from this experiment. The results provided several insights, since they showed that some interesting correlations which can drive the choice of the most promising properties actually exist. Specifically, data showed that the usage of particular properties as the *director* of the movie or its *writer* positively affects most of the explanations aims. On the other side, properties as the *producer* or the *distributor* do not improve the engagement nor the persuasion of the explanation. Moreover, it is very interesting that both the persuasion and the engagement of the explanation are improved when the information about which prizes the movie has won was reported in the explanation. Finally, it is also worth to note that including information about the general topic of the movie (encoded in the `DBpedia` properties DCTERMS:SUBJECT) may lead to more transparent explanations, but it negatively affects the trust. This is a very interesting insight, since it shows that a different choice of the properties may lead to the maximization of a different *explanation aim*, thus our framework may be properly tuned in order to use (or to avoid) a specific property according to the goals of the explanation.

To sum up, in this paper we presented a framework for generating natural language explanations based on the information available in the LOD cloud. Preliminary results showed the goodness of the framework, since our strategy significantly outperformed all the baselines we took into account. Moreover, we also showed that the choice of specific properties may influence the behavior of the framework, leading to explanations able to maximize some specific aspects. As future work, we will evaluate the effectiveness of the framework in different domains, as music and books, but we will also investigate the impact of different data points gathered from the LOD cloud. Indeed, we plan to extract also properties not directly connected to the items to be recommended, in order to build explanations containing more interesting and unexpected patterns connecting the items the user liked to those she got as recommendation, hopefully leading towards more engaging and effective explanations.

5. REFERENCES

[1] M. Bilgic and R. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization, IUI WS*, volume 5, 2005.

[2] G. Carenini and J. Moore. An empirical study of the influence of user tailoring on evaluative argument effectiveness. In *IJCAI 2001*, pages 1307–1314, 2001.

[3] T. H. Haveliwala. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

[4] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW 2000*, pages 241–250, 2000.

[5] B. Knijnenburg, S. Bostandjiev, J. O'Donovan, and A. Kobsa. Inspectability and control in social recommenders. In *RecSys 2012*, pages 43–50, 2012.

[6] B. Knijnenburg and M. Willemsen. Evaluating recommender systems with user experiments. In *Recommender Systems Handbook.*, pages 309–352. Springer, 2015.

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

[8] N. Tintarev and J. Masthoff. Evaluating the effectiveness of explanations for recommender systems. *UMUAI*, 22(4-5):399–439, 2012.