Schema-summarization in Linked-Data-based feature selection for recommender systems

Azzurra Ragone[1], Paolo Tomeo[2], Corrado Magarelli[2], Tommaso Di Noia[2], Matteo Palmonari[1]
Andrea Maurino[1], Eugenio Di Sciascio[2]
[1] University of Milan Bicocca, [2] Polytechnic University of Bari
[1] firstname.lastname@unimib.it, [2] firstname.lastname@poliba.it

ABSTRACT

Recommender systems are emerging as an interesting application scenario for Linked Data (LD). In fact, by exploiting the knowledge encoded in LD datasets, a new generation of semantics-aware recommendation engines have been developed in the last years. As Linked Data is often very rich and contains many information that may result irrelevant and noisy for a recommendation task, an initial step of feature selection is always required in order to select the most meaningful portion of the original dataset. Many approaches have been proposed in the literature for feature selection that exploit different statistical dimensions of the original data. In this paper we investigate the role of the semantics encoded in an ontological hierarchy when exploited to select the most relevant properties for a recommendation task. In particular, we compare an approach based on schema summarization with a "classical" one, i.e., Information Gain. We evaluated the performance of the two methods in terms of accuracy and aggregate diversity by setting up an experimental testbed relying on the Movielens dataset.

1. INTRODUCTION

In the last years we have witnessed a flowering of semantics-aware solutions for Recommender Systems (RSs) exploiting information held in knowledge graphs, as the ones in the Linked Data (LD) Cloud. Several approaches using LD to build RSs have been proposed in literature, however, almost no one tackles the issue of *automatically* select the best subset of LD-based features. Usually, the feature-selection process is done manually by choosing the properties more "suitable" for the scenario taken into account. For example, in a scenario related to movies, properties as `dbo:starring` or `dbo:director` look more relevant than `dbo:releaseDate` or `dbo:distributor`. As well as for the music domain, properties as `dbo:genre` and `dbo:writer` look more important than `dbo:producer` or `dbo:recordedIn`. However, without an automatic feature selection process, the human intervention is required every time a new domain is chosen, while it could be good to have a general way to select properties regardless of the domain. Moreover, selecting the top-K features to use in a recommendation scenario is somehow equivalent to discover which properties in a LD-dataset (e.g., DBpedia) encode the knowledge useful in the recommendation task and which ones are just noise[15]. In many machine learning tasks, as in recommendation systems, there is the need to perform a selection of features and this feature selection could not be straightforward when attributes are embedded in a knowledge graph, as it is the case of LD.

Today we have a huge amount of ontological data available on the Web as the one in the LD cloud. The importance of the ontological schema in such knowledge bases is often not fully exploited in data management tasks, indeed many times only the *extensional* part is used, while ignoring all the *intensional* part, which has a primary importance, being custodian of the meaning of data. In many graph-based recommendation systems the knowledge exploration starts from the data and goes on following the fact graph, without taking into account the knowledge lying in the ontology and then in its class hierarchy. In this paper we investigate if ontological schema summarization can be used as a feature selection technique for LD-based recommender systems and compare the results with other "*well-know*" techniques of feature selection. We performed an experimental evaluation on the Movielens dataset* in order to analyze how the choice of a particular feature selection technique may influence the performance of the recommendation algorithm in terms of *accuracy* and *aggregate diversity*.

The remainder of the paper is organized as follows: Section 2 reviews related literature while in Section 3 we analyze the feature selection process and the feature selection technique we take as a baseline in our evaluation. Section 4 describes the LD summarization process we perform with ABSTAT. The graph-based kernel methods used in our recommendation system are described in Section 5 together with the metrics used to evaluate our recommendation algorithm. The experimental evaluation of the two feature selection techniques is presented in Section 6. Discussion and future work close the paper.

2. RELATED WORK

*Available at http://grouplens.org/datasets/movielens

SAC'17, April 3-7, 2017, Marrakesh, Morocco

http://dx.doi.org/xx.xxxx/xxxxxxx.xxxxxxx

Recommender systems can be divided into two main classes: *Collaborative Filtering* and *Content-Based* systems. The former try to predict the users interests exploiting the statistical information about the ratings of all the users. The underlying assumption is that users with similar ratings have similar tastes, and similarly rated items may be of interest for the same users. However, collaborative filtering recommender systems suffer from the data sparsity problem. Conversely, content-based recommender systems only use the descriptive content of the item - such as tags, genre, textual description, etc. - in order to recommend items similar to the ones the user liked in the past. Such systems do not suffer from data sparsity, since they do not need to compare the ratings of different users[13]. However, descriptive content information about the items is not always available or sufficient. A solution to such problem is represented by the exploitation of Linked Open Data sources like `DBpedia`[3]. Many approaches have been proposed for exploiting information extracted from Linked Open Data in recommendation tasks. Heitmann and Hayes [9] proposed one of the first approach for using Linked Open Data in a recommender system. A system for recommending artists and music using DBpedia was presented in [19]. Several other approaches have been proposed afterwards, such as a knowledge-based framework leveraging DBpedia for cross-domain recommendation task [4], a content-based context-aware method able to adopt a semantic representation based on a combination of distributional semantics and entity linking techniques [16], a hybrid graph-based algorithm based on learning-to-rank method and path-based features extracted from heterogeneous information networks built upon DBpedia and collaborative information [17]. To the best of our knowledge, the only approach proposing an automatic selection of LOD features is the one in [15] where seven different techniques for automatic selection of LOD-based features are compared. Differently from[15], we are not interested in the best performing techniques for feature selection. Here we want to investigate if the knowledge encoded at ontological level can be used to select the most significant properties in a LD-dataset for recommendation purposes.

3. FEATURE SELECTION

Feature selection is a process to automatically select the attributes in a dataset that are most relevant to the predictive model at hand. Feature selection is useful to remove irrelevant or redundant attributes that do not contribute to the accuracy of the predictive model or that can, indeed, decrease the accuracy of the model itself. Feature selection, simplifying the problem, reduces the *overfitting* risk. The objective of feature selection is three-fold: (i) to improve the prediction performance of the predictors, (ii) to provide faster and more cost-effective predictors, (iii) to give a better understanding of the process that generate the data [8]. Feature selection has a prominent role when we have noise data; lots of low frequent features or, conversely, few popular features assuming always different values; too many feature comparing to samples, or complex model. A good feature selection technique should exclude features that give no, or little, information contribution. There are three typical measures of feature selection, the first are known as "**filters**", they apply a statistical measure to assign a score to each feature (e.g. *Information gain*, *Entropy*, *Mutual information*, *KL divergence*, *Gini index*, *Chi-square test*, ecc.). Here the feature selection process is a preprocessing step and can be independent from learning[5]. The second are called "**wrapper**", here the problem is modeled as a search problem, the learning system is used as a black box to score subsets of features (e.g. *forward selection* and *backward elimination*)[12]. Finally, there are the **embedded methods** that perform the selection within the process of training (e.g. *Nested subset methods*)[8]. When dealing with recommender systems the feature selection process may vary depending on the purpose of the recommendation. We may want to maximize **accuracy** (to recommend the most similar items to the user) or **diversity** (e.g. in music domain to recommend music from different genres). A relevant task is to determine the impact of a particular feature selection technique on the behavior of the recommendation system algorithm. Indeed, some techniques can improve the accuracy of the recommendation, some improves the diversity, others can provide a good trade-off between diversity and accuracy. Among all the different feature selection techniques mentioned before we selected the *Information Gain*, *Information Gain Ratio*, *Chi-squared test* and *Principal Component Analysis* as their computation can be adapted to categorical features, as the LOD ones. Then, the features selected from each technique have been used as an input of the *graph kernel-based* algorithm we use in the recommendation scenario (See Section 5). Finally, we chose the *Information Gain* as a baseline to compare with as it was the best performing techniques among the four analyzed[†].

3.1 Information Gain

Information Gain (IG) is defined as the expected reduction in entropy occurring when a feature is *present* versus when it is *absent*. For a feature f_i, IG is computed as [15]:

$$IG(f_i) = E(I) - \sum_{v \in dom(f_i)} \frac{|I_v|}{|I|} * E(I_v)$$

where $E(I)$ is the value of the entropy of the data, I_v is the number of items in which the feature f_i (e.g. *starring for movies*) has a value equal to v (e.g. *Al Pacino* in the movie domain), and $E(I_v)$ is the entropy computed on data where the feature f_i assumes value v. The IG of a feature f_i is higher as the lower is the value of the entropy $E(I_v)$. Features are ranked according to their IG and the top-k ones are returned.

4. SCHEMA SUMMARIZATION WITH ABSTAT

Linked data summarization is the process of extracting a summary of an input linked data set, such that this summary is smaller (in size) than the input data, but retains information useful for certain tasks. Relevance-oriented summaries capture subsets of the input data sets and/or ontologies. These subsets are estimated to be more relevant for the

[†]The interested reader can see the results obtained with other feature selection techniques here: https://github.com/sisinflab/SAC2017/FeatureSelection

users according to multidimensional relevance criteria [24]. Vocabulary-oriented summaries describe the usage of vocabularies, e.g., ontologies, used in a dataset. These summaries are usually defined so as to be complete, i.e., to provide information about every element of the vocabulary/ontology used in the data set [22]. Compression-oriented summaries are proposed to compress the original data so as to support certain computational tasks, e.g., autocomplete of queries [11]. Vocabulary-oriented summaries that provide complete descriptions of vocabulary usage may support feature selection by providing relevant information about every possible feature, i.e., property, in the data set. One advantage of using such approach is that summaries can be accessed via web interfaces. If summaries are proved useful for feature selection, features can be selected before processing the entire data set, by accessing summaries online at limited cost. Once the features are selected, a user may download a subset of an entire data set that is known to be useful for recommendation.

In this paper we use summaries produced by a vocabulary-oriented summarization framework named ABSTAT[‡]. ABSTAT takes a linked data set and - when available - one or more ontologies used in this data set as input, and returns a summary. The summary consists in a set of *patterns* having the form $\langle C, P, D\rangle$, with C and D being types, i.e., concepts or datatypes, and P being an RDF property. We refer to C and D as source and target types, respectively. Each pattern $\langle C, P, D\rangle$ tells that there exist some instance of type C linked to some instance of type D through the property P. For example, a pattern $\langle$`dbo:Film`, `dbo:starring`, `dbo:Actor`$\rangle$ tells that there are instances of `dbo:Film` linked to instances of type `dbo:Actor` through the property `dbo:starring` in the data set. The summary is complete for relational assertions in an RDF data set, i.e., assertions about individuals: for every relational assertion $< x, p, y >$ that exist in the data set, at least one pattern is generated, i.e., every such assertion is represented by at least one pattern. The generation of these patterns is based on explicit typing assertions, e.g., $\langle$`dbr:Tom_Cruise`, `rdf:type`, `dbo:Actor`$\rangle$ or on implicit typing assertions (for literals), e.g., `1962-01-01`$^{\texttt{xsd:date}}$ extracted from the dataset. Observe that since one resource can be subject of several typing assertions, more than one pattern can be generated from one relational assertion.

Differently from other approaches that also extract vocabulary-based patterns from linked data sets [14, 7], ABSTAT applies a pattern minimalization technique leveraging the relations between types defined in the ontologies (when the ontologies are used in the summarization process). Instead of generating all patterns that can be generated from one relational assertion according to the definition above, only patterns based on minimal types of resources are generated. For example, if `dbo:Actor` is defined as subclass of `dbo:Person` in the ontology, from a relational assertion such as $\langle$`dbr:Rain_Man`, `dbo:starring`, `dbr:Tom_Cruise`$\rangle$ and two typing assertions $\langle$`dbr:Tom_Cruise`, `rdf:type`, `dbo:Actor`$\rangle$ and $\langle$`dbr:Tom_Cruise`, `rdf:type`, `dbo:Person`$\rangle$, ABSTAT generates a unique pattern $\langle$`dbo:Film`, `dbo:starring`, `dbo:Actor`$\rangle$, discarding the pattern $\langle$`dbo:Film`, `dbo:starring`, `dbo:Person`$\rangle$ as redundant. Of course, there may be films that star entities for which `dbo:Actor` does not occur as minimal types, because, e.g., these entities are more generically defined as instances of `dbo:Person`. Thus, a pattern like the one represented by $\langle$`dbo:Film`, `dbo:starring`, `dbo:Person`$\rangle$ may still occur in the data set, telling that there are films starring entities that are not specifically defined as `dbo:Actor`, but generically as `dbo:Person`. Additional information provided in summaries and of major importance for feature selection is pattern *frequency*, which counts the occurrences of patterns in the data set. For example, $\langle$`dbo:Film`, `dbo:starring`, `dbo:Actor`$\rangle$[10662] tells that 10662 instances of `dbo:Film` are linked to instances of type `dbo:Actor` through the property `dbo:starring` in the data set. For more details about the summarization process, the impact of minimalization on the size of extracted summaries, the use of ABSTAT summaries to support data set understanding, and the services through which summaries are accessible via web interfaces we refer to a previous paper [22]. For the work in this paper we extended ABSTAT with APIs that ease the interaction with third party applications. In particular, APIs support look up of patterns using filters on source types, properties and target types, plus ranking based on pattern frequency. Using these APIs we have provided a way to look up top frequent properties for a source type, ranked by occurrence. Using this functionality, we have defined a **top-k property feature selection with ABSTAT** as follows: given the input type $C =$ `dbo:Film`, we select all patterns with C as source type and order them by frequency; then we extract the top-k **distinct** properties from the ordered list of patterns (discarding *data type properties* as they are not present in the ranked list returned by IG). In this approach we used summaries extracted from DBpedia dumps that focus on the relational content in the knowledge base and do not include annotation properties used to link entities to external resources, e.g., wikilinks, images and wikipedia subjects.

[‡]ABSTAT summaries for several datasets can be explored at http://abstat.disco.unimib.it:8880/

5. GRAPH KERNELS

In [18] two **graph-based kernel methods** for LOD-enabled recommender systems have been proposed. The first is based on an *entity-based item neighborhood mapping*, while the second on *path-based item neighborhood mapping*. They rely on graph-based Item Representation on knowledge graph defined as $G = \{t | t \in E \times R \times E\}$ where E is the set of entities, R is the set of properties and $I \subseteq E$ the items (e.g. movies), considered as a particular type of entities. In the two kernels, the non-directed version of the original RDF graph is considered. For a generic item i it is possible to define its h-hop neighborhood graph G_i^h as the subgraph of a knowledge graph G induced by the set of triples involving entities in E_i^h. Where E_i^h is the set of entities reachable in *at most* h hops form i according to the shortest path in G. Starting from this graph-based representation it is possible to define two feature mappings as in the follow.

Entity-based item neighborhood mapping. In this approach, a feature in a mapping represents an entity in E with a score that indicates the weight associated to that entity in G_i^h. The resulting feature vector $\phi_E(G_i^h)$ is:

$$\phi_E(G_i^h) = (w_{i,e_1}, w_{i,e_2}, ...w_{i,e_m}, ..., w_{i,e_t})$$

where the weight associated to the entity e_m is computed as

$$w_{i,e_m} = \sum_{l=1}^{h} \alpha_l \cdot c_{l,e_m} \text{ with } \alpha_l = \frac{1}{1+\log(l)} \text{ and}$$

$$c_{l,e_m} = |\{(e_n, p, e_m) \mid e_n \in \widehat{E}_i^{l-1} \ \wedge \ e_m \in \widehat{E}_i^{l}\} \bigcup \{(e_m, p, e_n) \mid e_m \in \widehat{E}_i^{l} \ \wedge \ e_n \in \widehat{E}_i^{l-1}\}| \quad (1)$$

where $\widehat{E}_i^l = E_i^l \setminus E_i^{l-1}$ is the set of entities *exactly* l hops far from i. While c_{l,e_m} denotes the number of triples connecting e_m to entities in the previous hop $(l-1)$, whether e_m could be the subject or object of the triple. In other words, c_{l,e_m} indicates the *occurrence* of the entity e_m in the item neighborhood at distance l. The more the entity e_m is connected to neighboring entities of i, the more it is descriptive of i. α_l is a decay factor depending on the distance l from the item i, penalizing farther entities from the item. It allows one to take into account the *locality* of those entities in the graph neighborhood.

Path-based item neighborhood mapping. Differently from the previous mapping, here a feature is represented as a sequence of nodes (path) in G. Given two entities e_1 and e_n, we call *path* the sequence of nodes $e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$ met while traversing the graph to go from e_1 to e_n. Therefore, each feature refers to several variants of paths rooted in the item node i. We first collect all the paths rooted in i which can be indicated as sequence of entities $i \cdot e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$. Then, from those paths, other features are defined considering every sub-paths. Specifically sub-paths are composed by only those entities progressively farther from the item. Considering the path given above we build the following features: $e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$, $e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$, ..., $e_{n-1} \cdot e_n$, e_n. This choice allows to explicitly represent substructures shared between items with no overlapping in their immediate neighborhoods but somehow connected at farther distance. Items connected to the same entities have same common structures because both closer and farther entities are shared. Items connected to different entities which are linked directly or at a farther distance to same entities share less or none sub-paths depending on how much far the common entities are, if any.

In formula, let P_i be the set of paths rooted in i and P_i^* be the list of all possible sub-paths extracted from them. Let $p_m(i)$ and $p_m^*(i)$ be the $m-th$ elements in P_i and P_i^*, respectively. Then, the feature vector is: $\phi_P(G_i^h) = (w_{i,p_1^*}, w_{i,p_2^*}, ...w_{i,p_m^*}, ..., w_{i,p_t^*})$ where each weight w_{i,p_m^*} is computed as:

$$w_{i,p_m^*} = \frac{\#p_m^*(i)}{|p_m| - |p_m^*|}$$

with $|p_m|$ indicating the length of path p_m to which $p_m^*(i)$ refers and $\#p_m^*(i)$ the occurrence of $p_m^*(i)$ in P_i^*. The denominator is a discounting factor taking into account the difference between the original path p_m and its sub-path p_m^*. The shorter the sub-path the more the discount as the sub-path contains entities farther from the item.

Results presented in [18] about the two graph-based kernel methods show significant improvements with respect to state of the art collaborative algorithms on two different datasets: one of sounds coming from Freesound.org, the other one of songs gathered from Last.fm and Songfacts.com. They also show that semantic enrichment of the initial knowledge graph performed by means of entity linking techniques is a good choice to boost the performances of a recommendation system in terms of novelty and aggregate diversity.

5.1 Metrics

For evaluating the quality of our recommendation algorithm (given a particular feature selection technique) we use four metrics, as each one of them measures a different dimension. To evaluate recommendation **accuracy**, we use Precision and Mean Reciprocal Rank (MRR). $Precision@N$ is a metric denoting the fraction of relevant items in the top-N recommendations. Let $rel(u,i)$ be a boolean function that represents the relevance of an item i for the user u, with value 1 for relevant and 0 for non-relevant items, then $Precision@N$ is calculated as follows:

$$Precision@N = \frac{\sum_{i=1}^{N} rel(u,i)}{N} \quad (2)$$

MRR computes the average reciprocal rank of the first relevant recommended item, and hence results particularly meaningful when users are provided with few but valuable recommendations (i.e., Top-1 or Top-3)[21]. MRR is computed as:

$$MRR = \frac{1}{|U|} \sum_{i=1}^{|U|} \frac{1}{rank_i} \quad (3)$$

where U is the set of users and $rank_i$ is the rank position of the first relevant recommendation for the i-th user. To evaluate **aggregate diversity**, we consider *catalog coverage* (the percentage of items in the catalog recommended at least once) and *aggregate entropy* [1]. The former is used to assess the ability of a system to cover the item catalog, namely to recommend as many items as possible. While the latter measures the distribution of the recommendations across all the items, showing whether the recommendations are concentrated on a few items or are better distributed. The coverage is a valuable measure for systems that recommend lists of items. The metric is computed as follow:

$$Coverage \quad = \quad \frac{| \bigcup_{u \in U} I_L(u) |}{| I |} \quad (4)$$

where $I_L(u)$ is the set of items contained in the list L for user u and I is the set of all available items (i.e. the catalog). Aggregate entropy can be computed as:

$$AggrEntropy = -\sum_{i \in I} \left(\frac{rec(i)}{total}\right) ln \left(\frac{rec(i)}{total}\right) \quad (5)$$

where $rec(i)$ is the number of users who received the item i as recommendation, and *total* the overall number of recommendations across all users. In a good recommender system there should be a trade-off between accuracy and diversity, diversity should be improved while maintaining adequate accuracy.

6. EXPERIMENTAL EVALUATION

The evaluation of the two feature selection methods has been done via the well-know **Movielens** 1M dataset. In

Entity-based Graph kernel	Top-K features	Precision@10	MRR@10	itemCov@10	aggrEntropy@10
IG	5	0.02327	0.15578	0.54262	8.96
	10	0.01734	0.13599	0.90658	10.24
	15	0.02055	0.14685	0.91989	10.19
ABSTAT	5	0.02035	0.14694	0.54953	9.12
	10	0.01651	0.13705	0.64346	9.42
	15	**0.02062***	0.13757	0.67417	9.42

Path-based Graph kernel	Top-K features	Precision@10	MRR@10	itemCov@10	aggrEntropy@10
IG	5	0.02266	0.16248	0.58971	9.12
	10	0.01518	0.13221	0.88252	10.26
	15	0.01387	0.13069	0.89762	10.25
ABSTAT	5	0.02026	0.15310	0.54825	9.13
	10	0.01519	0.13331	0.57461	9.33
	15	**0.01726***	**0.13510***	0.62606	9.46

Table 1: Experimental results using the entity-based and the path-based Graph kernel recommendation algorithms.

order to enrich it with information from Linked Data, we started from a dump of the DBpedia dataset [§] and we limited it to the movie domain by linking movies in Movielens dataset with their corresponding DBpedia entries. Since not all the movies in the dataset have a corresponding resource in DBpedia, the final reduced dataset is composed by **3689** movies, **4297** users and **942590** ratings.

Feature pre-processing. Our dataset includes a large amount of features and, generally speaking, LOD dataset may have a quite large feature set that could be, at the same time, very sparse. For example, in our dataset the properties "`dbp:artDirection`" or `dbp:precededBy` are very specific with lots of missing values. On the other hand, properties as `dbo:wikiPageExternalLink` or `owl#sameAs` have always different and unique values, so they are not informative at all for a recommendation task. For this reason, before starting the feature selection process with IG, we performed a preliminary step to reduce *redundant* or *irrelevant* features that bring little value to the recommendation task, but, at the same time, pose scalability issues. The pre-pocessing step has been done following [20], we fixed two thresholds: one for the missing values $p_m = 99\%$ and one for the distinct values $p_d = 98.9\%$ and, then, we discarded features for which we have more than p_m of missing values and more than p_d of distinct values. From **247** initial properties, after such a pre-processing step we obtained **41** properties. Notice that we had to perform this pre-processing step only to the benefit of IG, as for ABSTAT this step was unnecessary. Indeed, in ABSTAT properties with lots of missing values go automatically at the end of the ranking, so they are never selected if we are interested in the top-N ones.

Recommendation algorithm The two graph-based kernels proposed in Section 5 were used to compute the similarity between pairs of items in the dataset. Indeed, given the graph-based nature of the underlying LOD dataset they result an ideal candidate to evaluate how similar two items are by comparing the resources which can be reached by random walks on the graph [18]. Indeed, such similarity values are used to recommend to each users the items which result most similar to the ones she has liked in the past. Specifically, we implemented a content-based recommender system based on an item-based k-Nearest Neighbors algorithm. More formally, the formula used to compute the relevance of an item j for a user u, takes into account the neighbors of i belonging to the user profile $profile(u)$ and the rating $r(u, j)$ assigned by the user u.

$$P(u,i) = \frac{\sum_{j \in neighbors(i) \cap profile(u)} sim(i,j) \cdot r(u,j)}{\sum_{j \in neighbors(i) \cap profile(u)} sim(i,j)}$$

The graph-based kernels are used as similarity function - sim in the previous equation. The set $neighbors$ contains the k items most similar to i. In this work we fixed k equals to 20 which is a quite typical value in a recommendation task [10].

ABSTAT	IG
dbo:starring	dbp:director
dbo:director	dbp:starring
dbo:writer	dbp:producer
dbo:producer	dbp:cinematography
dbo:musicComposer	dbp:editing
dbp:music	dbp:music
dbo:distributor	dbo:writer
dbo:language	dbo:musicComposer
dbo:cinematography	dcterms:subject
dbo:country	dbp:distributor
dbo:editing	dbp:studio
dbp:studio	dbp:screenplay
dbp:extra	dbp:title
dbp:screenplay	dbp:country
dbp:genre	dbp:language

Table 2: Top 15 features selected by SMZ and IG

Results. Table 1 shows the results for entity-based and path-based graph kernel algorithms, respectively. When selecting only the first 5 features, the two feature selection methods, IG and ABSTAT, show good values of accuracy, but lower values of aggregate diversity, especially in term of coverage. This is not really surprising as with a lower number of features, the system does not have enough diversified information to select more items and the effect of the popularity bias is stronger. Increasing the number of features the value of diversity increases at the expense of the accuracy. However, a good balance remains between accuracy and diversity thus showing a good trade-off between the two [2]. Table 2 shows the top 15 features selected by ABSTAT and IG respectively. It is worth noticing that although there is an overlap between the two lists, this not complete and that, differently from IG, ABSTAT is able to retrieve `dbp:genre` in the top 15 features which carries out meaningful information. Moreover, IG tends to select properties from the `dbp` prefix while ABSTAT from `dbo` (as an example,

[§] http://downloads.dbpedia.org/2015-10/

see `dbp:starring` in IG and `dbo:starring` in ABSTAT). It is interesting to point out that having a higher number of features is also a plus if we consider other facilities that can be exposed by a recommendation engine such as explanation [23]. Given a recommended item, a richer set of features allows the computation of a better and more exhaustive explanation on the reason why that particular item has been recommended to the user. Explanation services are a gaining momentum also after the so called "right to explanation" introduced by the European Union in its General Data Protection Regulation[¶][6]. In Table 1 we have highlighted in bold the configurations where ABSTAT outperforms IG (the * symbol indicates that the differences between ABSTAT and the IG baseline are statistically significant with $p - value < 0.001$ according to the paired t-test.) The implementation of the recommendation algorithm presented in this work and all the experimental results are available on GitHub[‖].

7. DISCUSSION AND FUTURE WORK

In this work we start our investigation on the role of the ontological schema available in LD datasets when used for data- and knowledge-intensive tasks as that of recommendation. In previous works, many approaches and techniques coming from machine learning have been applied to develop recommendation engines that exploits LD data but they never take into the proper account the semantics encoded in the class hierarchies which plays a fundamental role to provide a clear and explicit meaning to data. Here, we try to shed some light in understanding if the ontological information encoded into the LD dataset can be summarized and exploited as a feature selection techniques in recommendation tasks. For this purpose, we used a schema summarization tool (ABSTAT) to automatically select a set of features as input for the recommendation algorithm and compared it with a classical technique as Information Gain. We then implemented an item k-NN recommender system which uses two graph kernels to compute recommendations. We have seen how ABSTAT outperforms IG, in terms of trade-off between accuracy and aggregate diversity of results when the number of selected features grows. As future work we want to further investigate the role of semantics in feature selection for recommender systems, evaluating the performance of the algorithm with different configurations of the graph kernel (e.g. increasing the length of the paths explored in the knowledge graph). We are also in the process of evaluating the approach presented here with other datasets such as Wikidata and LinkedMDB.

8. REFERENCES

[1] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE TKDE*, 24(5), May 2012.

[2] P. Castells, N. J. Hurley, and S. Vargas. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*. Springer 2015.

[3] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*. 2015.

[4] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci. A generic semantic-based framework for cross-domain recommendation. In *Proc. of 2nd HetRec Workshop*, 2011.

[5] X. Geng, T.-Y. Liu, T. Qin, and H. Li. Feature selection for ranking. In *Proceedings of the 30th ACM SIGIR*, 2007.

[6] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *ArXiv e-prints*, June 2016.

[7] T. Gottron, M. Knauf, A. Scherp, and J. Schaible. ELLIS: interactive exploration of linked data on the level of induced schema patterns. In *Proc. of 2nd SumPre Workshop*, 2016.

[8] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. of Machine Learning Research*, 3, 2003.

[9] B. Heitmann and C. Hayes. C.: Using linked data to build open, collaborative recommender systems. In *In: AAAI Spring Symposium: Linked Data Meets Artificial IntelligenceâĂŹ. (2010*, 2010.

[10] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*, 2010.

[11] M. Jarrar and M. Dikaiakos. A Query Formulation Language for the Data Web. *IEEE TKDE*, 24(5), 2012.

[12] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2), 1997.

[13] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems handbook*, 2011.

[14] N. Mihindukulasooriya, M. Poveda-Villalón, R. García-Castro, and A. Gómez-Pérez. Loupe - an online tool for inspecting datasets in the linked data cloud. In *Proc. of ISWC 2015 Posters & Demonstrations Track*, 2015.

[15] C. Musto, P. Lops, P. Basile, M. de Gemmis, and G. Semeraro. Semantics-aware graph-based recommender systems exploiting linked open data. In *Proc. of 24th UMAP*, 2016.

[16] C. Musto, G. Semeraro, P. Lops, and M. de Gemmis. Combining distributional semantics and entity linking for context-aware content-based recommendation. In *Proc. of 22nd UMAP*, 2014.

[17] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM TIST*, 8(1), 2016.

[18] V. C. Ostuni, S. Oramas, T. Di Noia, X. Serra, and E. Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM TIST*, 2016.

[19] A. Passant. *dbrec — Music Recommendations Using DBpedia*. 2010.

[20] H. Paulheim and J. Fümkranz. Unsupervised generation of data mining features from linked open data. In *Proc. of 2nd WIMS*, 2012.

[21] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proc. of 6th ACM RecSys*, 2012.

[22] B. Spahiu, R. Porrini, M. Palmonari, A. Rula, and A. Maurino. ABSTAT: ontology-driven linked data summaries with pattern minimalization. In *Proceedings of 2nd SumPre Workshop*, 2016.

[23] N. Tintarev and J. Masthoff. *Explaining Recommendations: Design and Evaluation*. 2015.

[24] G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. RDF Digest: Efficient Summarization of RDF/S KBs. In *Proc. of ESWC*, 2015.

[¶]Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) [2016] OJ L119/1

[‖]https://github.com/sisinflab/SAC2017