Using Ontology-based Data Summarization to Develop Semantics-aware Recommender Systems

Tommaso Di Noia[1], Corrado Magarelli[2] Andrea Maurino[2], Matteo Palmonari[2], Anisa Rula[2]

[1] Polytechnic University of Bari, Via Orabona, 4, 70125 Bari, Italy
tommaso.dinoia@poliba.it
[2] University of Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1, 20126 Milano, Italy
firstname.lastname@unimib.it

Abstract. In the current information-centric era, recommender systems are gaining momentum as tools able to assist users in daily decision-making tasks. They may exploit users' past behavior combined with side/contextual information to suggest them new items or pieces of knowledge they might be interested in. Within the recommendation process, Linked Data have been already proposed as a valuable source of information to enhance the predictive power of recommender systems not only in terms of accuracy but also of diversity and novelty of results. In this direction, one of the main open issues in using Linked Data to feed a recommendation engine is related to feature selection: how to select only the most relevant subset of the original Linked Data thus avoiding both useless processing of data and the so called "curse of dimensionality" problem. In this paper, we show how ontology-based (linked) data summarization can drive the selection of properties/features useful to a recommender system. In particular, we compare a fully automated feature selection method based on ontology-based data summaries with more classical ones, and we evaluate the performance of these methods in terms of accuracy and aggregate diversity of a recommender system exploiting the top-k selected features. We set up an experimental testbed relying on datasets related to different knowledge domains. Results show the feasibility of a feature selection process driven by ontology-based data summaries for Linked Data-enabled recommender systems.

1 Introduction

Semantics-aware Recommender Systems (RSs) exploiting information held in knowledge graphs, as the ones available as Linked Data (LD), represent one of the most interesting and challenging application scenarios for LD. A high number of solutions and tools have been proposed in the last years showing the effectiveness of adopting LD (also referred to as knowledge graphs) as knowledge sources to feed a recommendation engine (see [19] and references therein for an overview). Nevertheless, how to automatically select the "best" subset of a LD dataset to feed a LD-based RS without affecting the performance of the recommendation algorithm is still an open issue. In other words, is there any valuable criterion to automatically perform a feature selection (FS) over semantic data available in the Web? Notice that the selection of the top-k features to use

in a RSs means 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 [18].

In most of the approaches proposed so far, usually, the FS process is performed by human experts that choose properties resulting more "suitable" for a given scenario. For instance, in the movie domain, properties as `dbo:starring` or `dbo:director` look more relevant than `dbo:releaseDate` or `dbo:runtime`. Analogously, for the book domain, properties as `dbo:literaryGenre` and `dbo:author` seem more representative than `dbo:numberOfPages` or `dbp:releaseNumber`. Unfortunately, a manual selection of features is strongly grounded in the knowledge domain and is not prone to be executed automatically. Over the years, many algorithms and techniques for feature selection , e.g., Information Gain, Information Gain Ratio, Chi Squared Test and Principal Component Analysis, have been proposed with reference to machine learning tasks. Yet, they mainly rely on statistical distribution of data in the dataset and they do not consider a characteristic which makes unique LD: they come with semantics attached. Ontologies give meaning to data through the modeling of classes, properties and their mutual relations. Information encoded in the ontological schema is often under-exploited when developing RSs based on LD; thus in a typical graph-based RS the exploration of the knowledge graph is driven exclusively by the data and it goes on by following the "fact" graph, without taking into account the knowledge lying in the ontology and then in its class hierarchy.

The main objective of this paper is thus to investigate how ontology-based data summarization [27, 14] can be used as a new and semantic-oriented feature selection technique for LD-based RSs thus improving results over other non semantic feature selection techniques. In particular, we define a feature selection method that automatically extracts the top-k properties that are deemed to be more important to evaluate similarity between instances of a given class on top of data summaries built with the help of an ontology. The method uses frequency and cardinality descriptors computed over schema patterns such as $\langle$`dbo:Film`, `dbo:starring`, `dbo:Actor`$\rangle$ extracted from the data.

We perform an experimental evaluation on three well-known datasets in the RS domain (*Movielens*, *LastFM*, *LibraryThing*) in order to analyze how the choice of a particular FS technique may influence the performance of recommendation algorithms in terms of quality measures such as Precision, Mean Reciprocal Rank (MRR), Catalog Coverage and Aggregate Entropy which are typical in the field of RSs [7]. Experimental results show that information provided in ontology-based data summaries selects features that achieve comparable, or, in most of the cases, better performance than state-of-the-art, semantic-agnostic analytical methods such as Information Gain [16].

We believe that these results are interesting also because of practical reasons. The use of statistical measures like the ones mentioned above for FS require that a user acquires an entire dataset beforehand and compute the measures on the whole dataset. Conversely, LD summaries are published on-line and summary-based FS can be performed even without acquiring the entire dataset and efficiently (on top of summary information). Thus, by using LD summaries for FS, a user could acquire and work with a subset of the dataset useful for him, without collecting and analyzing an entire

dataset. Finally, these results provide further evidence for the usefulness of these kinds of summaries and the informativeness of the information they encode.

Some intuitions behind this work were published in previous work [23], where we tested the use of frequency associated with schema patterns in a FS approach that included a manual preprocessing step. The approach was evaluated only in the movie domain, using, for the recommendation, a similarity measure based on graph kernels. In this paper, we provide a fully automatic FS method, which leads us to extend the ontology-based data summarization framework to compute cardinality descriptors. In addition, for the recommendation, we use a different and well-known similarity measure and conduct experiments in three different domains.

The paper is organized as follows: in Section 2, we introduce the ontology-based data summarization approach used in this work, while in Section 3, we describe the feature selection and recommendation methods. Section 4 is devoted to the explanation and discussion of the experimental results. Section 5 briefly reviews related literature for schema and data summarization as well as on recommender systems while Section 6 discuss conclusions and future work.

2 Ontology-driven Linked Data Summarization

While relevance-oriented data summarization approaches are aimed at finding subsets of a dataset or an ontology that are estimated to be more relevant for the users [28], vocabulary-oriented approaches are aimed at profiling a dataset, by describing the usage of vocabularies/ontologies used in the dataset. The summaries returned by these approaches are complete, i.e., they provide statistics about every element of the vocabulary/ontology used in the dataset [27]. Statistics captured by these summaries that can be useful for the feature selection process are the ones concerning the usage of properties for a certain class of items to recommend.

Patterns and frequency. In our approach we use pattern-based summaries extracted using the ABSTAT framework[3]. Pattern-based summaries describe the content of a dataset using schema patterns having the form $\langle C, P, D\rangle$, where C and D, are types (either classes or datatypes) and P is a property. For example, the pattern $\langle$`dbo:Film`, `dbo:starring`, `dbo:Actor`$\rangle$ tells that films exist in the dataset, in which star some actors. These patterns are extracted from relational and typing assertions found in RDF datasets. Differently from similar pattern-based summaries [14], ABSTAT uses the subclass relations in the data ontology, represented in a *Type Graph*, to extract only *minimal type patterns* from relational assertions, i.e, the patterns that are more type-wise specific according to the ontology[4]. A pattern $\langle C, P, D\rangle$ is a minimal type pattern for a relational assertion $\langle a, P, b\rangle$ according to a type graph $\mathcal{G}$ iff C and D are the types of a and b respectively, which are minimal in $\mathcal{G}$. In a pattern $\langle C, P, D\rangle$, C and D are referred to as *source* and *target* types respectively. A minimal type pattern $\langle$`dbo:Film`, `dbo:starring`, `dbo:Actor`$\rangle$ (simply referred to as *pattern* in

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

[4] If no ontology is specified, all types are minimal and patterns are extracted like in frameworks that do not adopt minimalization

the following) tells that there exist entities that have dbo:Film and dbo:Actor as minimal types which are connected through the property P. Non minimal patterns can be inferred from minimal patterns and the type graph. Therefore, they can be excluded as redundant without information loss, making summaries more compact [27]. Each pattern $\langle C, P, D\rangle$ is associated with a *frequency*, which reports the number of relational assertions $\langle a, P, b\rangle$ from which the pattern has been extracted.

Local cardinality descriptors. For this work, we have extended ABSTAT to extract *local cardinality descriptors*, i.e., cardinality descriptors of RDF properties, which are specific to the patterns in which the properties occur. To define these descriptors, we first introduce the concept of restricted property extensions. The *extension* of a property P restricted to a pattern $\langle C, P, D\rangle$ is the set of pairs $\langle x, y\rangle$ such that the relational assertion $\langle x, P, y\rangle$ is part of the dataset and $\langle C, P, D\rangle$ is a minimal-type pattern for $\langle x, P, y\rangle$. When referring to extensions, we keep the well-known terminology used in RDF triples, using subject and object to refer respectively to the first and the second element of each pair in the extension. Given a pattern π with a property P, we can define the functions:

- $minS(\pi)$, $maxS(\pi)$, $avgS(\pi)$, denoting respectively the minimum, maximum and average number of distinct subjects associated to unique objects in the extension of P restricted to π;
- $minO(\pi)$, $maxO(\pi)$, $avgO(\pi)$, denoting respectively the minimum, maximum and average number of distinct objects associated to unique subjects in the extension of P restricted to π.

All functions return integer values, and, in particular, $avgS$ and $avgO$ return the integer values closer to the real average values. ABSTAT can also compute global cardinality descriptors by adjusting the above mentioned definition so as to consider unrestricted property extensions. Local cardinality descriptors carry information about the semantics of properties as used with specific types of resources (in specific patterns) and can be helpful for selecting features used to compute the similarity between resources. For example, to compute similarity for movies, one would like to discard properties that occur in patterns π with dbo:Film as source type and $avgS(\pi) = 1$, i.e., properties where different objects are linked to different subjects (e.g., movies). We remark that the values of local cardinality descriptors for patterns with a property P may differ from values of global cardinality descriptors for P. As an example, for the property dbo:cinematography we find as global cardinality descriptors $minS = 1$, $maxS = 249$, $avgS = 5$, $minO = 1$, $maxO = 13$, $avgO = 1$. For the pattern $\langle$dbo:Film, dbo:cinematography, dbo:Person$\rangle$, we find as local cardinality descriptors $minS = 1, maxS = 249, avgS = 14, minO = 1, maxO = 7, avgO = 1$. More examples of local cardinality descriptors can be found in the faceted-search interface (ABSTATBrowse)[5]. Observe that our definition of extensions (restricted and unrestricted) are de facto based on the Unique Name Assumption, as they consider that two subjects or two objects denoted by different constants (URIs or literals) are distinct. We consider this acceptable in relation to the descriptive purpose of the cardinality descrip-

[5] http://abstat.disco.unimib.it/browse

tors. Finally, observe that all measures used in ABSTAT are intended to be expressive for end users, can be easily explained and are based on integer values.

In conclusion, ABSTAT takes a linked dataset and - if specified - one or more ontologies as input, and returns a summary that consists of: a type graph, a set of patterns, their frequency, local an global cardinality descriptors.

3 Semantics-aware Feature Selection

Feature selection is the process of selecting a subset of relevant attributes in a dataset, removing irrelevant or redundant attributes that can decrease the accuracy of the model at hand and increase the *overfitting* risk. Thanks to the feature selection process it is possible to improve the prediction performance, to provide faster and more cost-effective predictors and to give a better understanding of the process that generates the data [8]. A good feature selection technique should exclude features that give no, or little, information contribution, as low frequent features or, conversely, popular features assuming always different values. There are three typical measures of feature selection (i) "**filters**", statistical measures to assign a score to each feature (here the feature selection process is a preprocessing step and can be independent from learning[6]); (ii) "**wrapper**" where the learning system is used as a black box to score subsets of features [10]; (iii) **embedded methods** that perform the selection within the process of training [8]. In the following, we discuss two approaches used for the feature selection task: the first operates on the summarization of the datasets and the second operates on the instances of the datasets.

3.1 Feature Selection with Ontology-based Summaries

As described in Section 2, the ABSTAT framework provides two useful statistics: the pattern frequency and the cardinality descriptors that are used in the feature selection process as described in Figure 1. The process starts by considering all patterns $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ of a given class C occurring as a source type. The example in Figure 1 shows a subset of Π with `dbo:Film` as source type. The first step of our approach (*FILTERBY*) filters out properties based on the local cardinality descriptors. In particular, it filters only properties for which the average number of distinct subjects associated with unique objects is more than one ($avgS > 1$). The rational behind this step is to consider only those properties connecting one target type with many source types. In the example, patterns π_4 and π_8 with `dbo:wikiPageExternalLink` and `owl:sameAs` property respectively are removed because there exists on average only one subject of type `dbo:Film` associated with a distinct object. The second step of the process (*SELECTDISTINCTP*) selects all properties of the patterns in Π by applying the maximum of the pattern frequency (# in the Figure). Then, the properties are ranked (*ORDERBY*) in a descending order on pattern frequency and then k properties (*TOPK*) are selected (k=2).

In some datasets, such as DBpedia, properties may use redundant information by using same properties with different namespaces, e.g., `dbo:starring` and `dbp:starring`. For this reason, in such case, a pre-processing step for removing replicated properties to avoid redundant ones is requested (see Section 4.1).

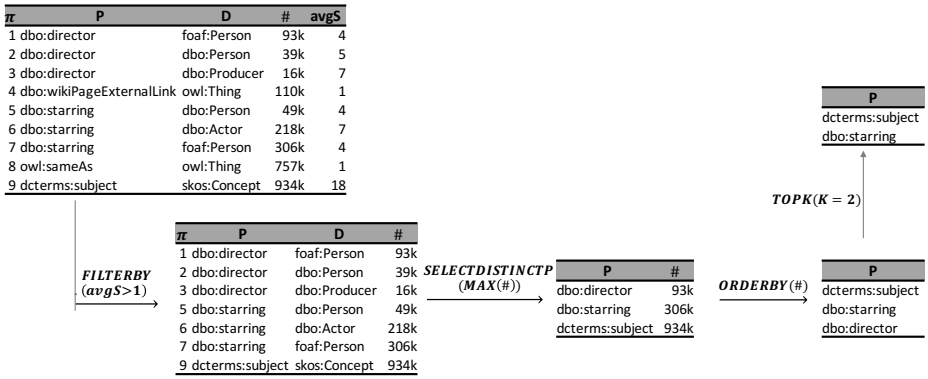

Fig. 1: Feature selection with ABSTAT
with source type `dbo:Film`

3.2 Feature Selection with State-of-the-art Techniques

In this work we consider RDF properties as features, so among the different feature selection techniques available in the literature, we initially selected *Information Gain*, *Information Gain Ratio*, *Chi-squared test* and *Principal Component Analysis* as their computation can be adapted to categorical features as LD and we then evaluated their effect over the recommendation results. The features selected from each technique have been used as an input of the recommendation algorithm that uses the Jaccard index as similarity measure. In order to identify the best technique among the one we selected, they have been evaluated by using Information Gain (IG), Gain Ration and Chi Squared Test. At the end, IG resulted as the best performing one[6]. In order to make the paper self-consistent we report hereafter the definition of Information Gain. It computes the expected reduction in entropy occurring when a feature is *present* versus when it is *absent*. For a feature f_i, IG is defined as [16]:

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

where $E(I)$ is the entropy of the data, I_v is the number of items in which the feature f_i (e.g. *director* for movies) has a value equal to v (e.g. *F.F.Coppola* 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 $E(I_v)$. Then features are ranked according to their IG value and the top-k ones are returned.

Feature pre-processing. LD datasets usually have a quite large feature set that can be, at the same time, very sparse depending on the knowledge domain. For instance, taking into account the movies available in *Movielens*, properties as `dbp:artDirection` or `dbp:precededBy` are very specific and have a lot of missing values. On the other

[6] For the sake of conciseness we do not report all the results here. Results obtained with other FS techniques can be found at `http://ow.ly/zAA530d0wu0`

hand, properties as `dbo:wikiPageExternalLink` or `owl:sameAs` always have different and unique values, so they are not informative 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-processing step has been done following [22]: we fixed a threshold $t_m = t_d = 97\%$ both for missing values and for distinct values and, then, we discarded features for which we had more than t_m of missing values and more than t_d of distinct values. We did such a pre-processing step for the three different recommendation datasets *Movielens*, *LastFM* and *LibraryThing*. Results of our analyses are depicted in Table 1. Please, note that we had to perform this pre-processing step only to the benefit of IG, as for ABSTAT the entire process has been done as explained in Section 3.1.

Dataset	# of features before pre-processing	# of features after pre-processing
Movielens	148	34
LastFM	271	25
LibraryThing	201	22

Table 1: Reduction on the number of features after the pre-processing step.

3.3 Recommendation Method

We implemented a content-based recommender system using an item-based nearest neighbors algorithm as in [18], where the similarity is computed by means of Jaccard's index. We use a Jaccard-based similarity because it is a straight and effective measure used in semantic recommendation for categorical features. Given two resources i and j in a LD dataset the metric calculates their similarity as:

$$jaccard(i,j) = \frac{|N(i) \bigcap N(j)|}{|N(i) \bigcup N(j)|} \tag{1}$$

where $N(i)$ and $N(j)$ are the neighbors of i and j in the RDF graph. In this work, the neighborhood of i (respectively j) includes all the nodes in the graph reachable starting from the resource i (respectively j) following the properties selected by the feature selection phase. The neighbors are thus one-hop features. Observe that considering values of frequent properties like `dbo:wikiPageExternalLink` or `owl#sameAs`, which have always different and unique values, would only bring noise when computing Jaccard similarity. In addition, computing Jaccard using the very large number of features that can be found in linked datasets (see Table 1) would not be efficient at runtime, considering that recommendations need to be computed almost at interactive time. These observations provide additional evidence for the need of a FS method before computing recommendations.

The similarity values are then used to recommend to each user the items which result most similar to the ones she has liked in the past. Given an item j and a user u, the following formula is used to predict the rating of items i which is unknown to the user:

$$r^*(u,i) = \frac{\sum_{j \in N(i) \cap r(u)} jaccard(i,j) \cdot r(u,j)}{\sum_{j \in N(i) \cap r(u)} jaccard(i,j)}$$

where $r(u)$ represents the items rated by the user u, and $r(u, j)$ the rating value given by the user u with respect to the item i. Therefore, the above equation takes into account the neighbors of i belonging to the user profile and computes an average of the user ratings to such neighbors weighted by the similarity values.

4 Experimental Evaluation

4.1 Experimental Setup

Datasets. The evaluation has been carried out on the three datasets belonging to three different domains, i.e. movies, books, and music. The first recommendation dataset we tested is based on the Movielens 1M. The original dataset contains 1,000,209 ratings (1-5 stars) given by 6,040 users to 3,883 movies. The second one, LibraryThing contains 7,112 users, 37,231 books and 626,000 ratings ranging from 1 to 10. The third dataset comes from recent initiatives on information heterogeneity and fusion in recommender systems[7] [3] and has been built on top of the `Last.fm` music system[8]. It originally contains 1,892 users, 17,632 artists and 92,834 relations between a user and a listened artist together with their corresponding listening counts.

Measures. While evaluating a recommendation algorithm we are interested in measuring its performances not just in terms of accuracy of the predicted results but also in terms of their diversity and novelty. Hence, depending on the adopted feature selection technique we are interested in evaluating the variations of different aspects (accuracy, diversity, novelty) in the final result. Indeed, some techniques may improve the accuracy of the recommendation, some improve diversity, others may provide a good trade-off between diversity and accuracy. Therefore, for evaluating the quality of our recommendation algorithm (given a particular feature selection technique) we used four different metrics. To evaluate recommendation **accuracy**, we used *Precision* (Precision@N) and *Mean Reciprocal Rank* (MRR). Precision@N is a metric denoting the fraction of relevant items in the Top-N recommendations. 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 to Top-10) [25]. **Aggregate diversity** is considered one of the other most important quality factors [1]. A good recommender system should provide a good balance between accuracy and diversity of results. For instance, recommendations not equally distributed among the items, even if accurate, indicate a low degree of personalization [1]. To evaluate aggregate diversity, we considered *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 they have a better distribution.

Implementation. Summaries are expected to be computed for entire datasets by data publishers or third parties and then accessed via web interface efficiently. Extracting the summary and the cardinality descriptors on the full DBpedia takes 6 to 8 hours

[7] `http://ir.ii.uam.es/hetrec2011/datasets.html`
[8] `http://www.lastfm.com`

on a single core (splitted in ≈ 1/4 for pattern stats and ≈ 3/4 for cardinality descriptors). Filtering and ranking using information in the summaries is then very efficient (a few milliseconds, on a base of more than 300K patterns extracted from DBpedia).

We tried different ranking and filtering functions by combining local cardinality descriptors and pattern frequency to study their effect on feature selection. For lack of space we include the best three combinations from those proposed in section 3.1:

- **AbsMaxS** considers as input of *SELECTDISTINCTP* the $maxS$ instead of the frequency of patterns.
- **AbsOccAvgS** considers as input of *FILTERBY* the $avgS$ and *SELECTDISTINCTP* the maximum of the pattern frequency.
- **AbsOcc*MaxS** considers as input of *SELECTDISTINCTP* the product of $maxS$ and pattern frequency.

Both for ABSTAT and IG we consider different configurations in the experimental settings:

- **noRep** Here, we consider the first N features selected and if there are both `dbo:` and `dbp:` feature (e.g. `dbo:starring` and `dbp:starring`) we delete the feature that appear later in the ranking.
- **withRep** Here, we take into account both `dbo:` and `dbp:` feature (e.g. if among the first N features there are either `dbo:starring` or `dbp:starring` we consider both features in the order in which they appear in the ranking).
- **Onlydbp** Here, if among the first N features selected there are both `dbo:` and `dbp:` feature we consider only the `dbp:` one.
- **Onlydbo** Conversely, here we take into account only the `dbo:` one. However, notice that in the experiments for *Movielens* and *Lastfm* datasets we never have this configuration as the features selected by such a configuration are exactly the same selected with the *noRep* configuration.

ABSTAT Baseline. As a baseline for ABSTAT-based feature selection we use TfIdf (short for term frequency–inverse document frequency). We use this as baseline for ABSTAT-based feature selection as TfIdf is a well-known measure to identify most relevant terms (properties in this case) for a document (a class in this case). We adopt *TfIdf* in our context where by document we refer to a set of patterns having the same subject-type and by term we refer to a property. *TfIdf* is based on the number of properties occurring in a document that corresponds to *Tf* and the logarithm of the ratio between the total number of documents and the number of documents containing the property that corresponds to *Idf*. While *TF* is proportional to the number of properties occurring in a document, *IDF* tries to penalize those properties that occur very frequently and those that rarely occur.

4.2 Results and Discussion

Tables 2, 3, 4 show the experimental results obtained on, respectively, *Movielens*, *Last.FM* and *LibraryThing* datasets in terms of **Precision**, **MRR**, **catalogCoverage**, and **aggrEntropy**. Results are computed over lists of top-10 items recommended by

	Precision@10		MRR@10		catalogCoverage@10		aggrEntropy@10	
Top K features	**5**	**20**	**5**	**20**	**5**	**20**	**5**	**20**
withrep.IG	.0658	.1078	.2192	.3417	.3829	.5280	7.56	8.50
withrep.AbsOccAvgS	**.1059**	**.1081**	**.3380**	.3477	.5398	.5253	8.70	8.53
withrep.AbsOcc*MaxS	.0967	.1074	.3274	**.3541**	.5962	.5247	8.87	8.54
withrep.AbsMaxS	.0919	.1030	.3065	.3400	**.6016**	**.5698**	**8.96**	**8.66**
withrep.TfIdf	.0565	.0851	.2267	.3326	.4347	.3360	8.36	7.80
norep.IG	.0841	.1076	.2961	.3390	.3372	.5226	7.94	8.44
norep.AbsOccAvgS	**.1066**	.1076	**.3388**	.3400	.5344	.5208	8.68	8.45
norep.AbsMaxS	.0885	.1063	.3075	**.3467**	**.6234**	**.5550**	**8.99**	**8.60**
norep.TfIdf	.0823	.0856	.2994	.3123	.3520	.3908	7.83	7.99
dbo.IG	.0841	.1076	.2961	.3390	.3372	.5226	7.94	8.44
dbo.AbsOccAvgS	**.1066**	.1067	**.3388**	.3402	.5344	.5208	8.68	8.51
dbo.AbsMaxS	.0885	.1059	.3075	**.3464**	**.6234**	**.5535**	**8.99**	**8.60**
dbo.TfIdf	.0823	.0856	.2994	.3123	.3520	.3908	7.83	7.99
dbp.IG	.0688	.1046	.2134	.3336	.2799	.5065	6.54	8.31
dbp.AbsOccAvgS	**.1065**	**.1059**	.3408	.3360	.5426	.5105	8.64	8.31
dbp.AbsMaxS	.0908	.1030	.3124	**.3396**	**.6219**	**.5395**	**8.98**	**8.52**
dbp.TfIdf	.0549	.0745	.1924	.2687	.2530	.3575	6.33	7.41

Table 2: Experimental results on the Movielens dataset. Bold values indicates that the difference with the other methods are statistical significant (T-test with p-value < 0.0001).

	Precision@10		MRR@10		catalogCoverage@10		aggrEntropy@10	
Top K features	**5**	**20**	**5**	**20**	**5**	**20**	**5**	**20**
withrep.IG	.0576	.0588	.2348	.2273	.3983	.4034	10.47	10.44
withrep.AbsOccAvgS	.0458	.0568	.2003	.2343	.3670	.4014	10.28	10.50
withrep.AbsOcc*MaxS	.0457	.0560	.2116	.2355	.3854	.3826	10.54	10.21
withrep.AbsMaxS	.0571	.0567	.2319	.2360	.3689	.4011	10.24	10.29
withrep.TfIdf	.0215	.0145	.1607	.1202	.1314	.2349	8.81	9.75
norep.IG	.0571	.0579	.2346	.2274	.3988	.4037	10.47	10.44
norep.AbsOccAvgS	.0561	.0593	.2328	.2329	.3982	.4030	10.54	10.48
norep.AbsOcc*MaxS	.0459	.0570	.2119	.2372	.3852	.3809	10.54	10.15
norep.AbsMaxS	.0541	.0567	.2301	.2365	.3653	.4008	10.24	10.29
norep.TfIdf	.0215	.0138	.1608	.1211	.1314	.2877	8.81	10.26
dbo.IG	.0571	.0579	.2346	.2274	.3988	.4037	10.47	10.44
dbo.AbsOccAvgS	.0561	.0593	.2328	.2329	.3982	.4030	10.54	10.48
dbo.AbsOcc*MaxS	.0459	.0570	.2119	.2372	.3852	.3809	10.54	10.15
dbo.AbsMaxS	.0541	.0567	.2301	.2365	.3653	.4008	10.24	10.29
dbo.TfIdf	.0579	.0605	.2374	.2477	.4086	.3991	10.55	10.20
dbp.IG	.0586	.0586	.2350	.2299	.4027	.4043	10.49	10.40
dbp.AbsOccAvgS	.0623	.0612	.2467	.2342	.3943	.4043	10.42	10.45
dbp.AbsOcc*MaxS	.0464	.0606	.2126	.2504	.3862	.3797	10.53	10.07
dbp.AbsMaxS	.0571	.0592	.2318	.2398	.3689	.4002	10.24	10.22
dbp.TfIdf	.0215	.0132	.1608	.1218	.1314	.2696	8.81	9.96

Table 3: Experimental results on the LastFM dataset.

the RS, which expresses the average number of items to be recommended in similar domains, using top-k features selected with different configurations. We conducted experiments using top-k selected features for different k, i.e., $k = 5, 10, 15, 20$, and all

	Precision@10		MRR@10		catalogCoverage@10		aggrEntropy@10	
Top K features	**5**	**20**	**5**	**20**	**5**	**20**	**5**	**20**
withrep.IG	.0501	.1325	.2283	.4102	.4290	.5051	11.00	11.18
withrep.AbsOccAvgS	**.1330**	.1320	**.4047**	.4105	.4812	.5036	11.10	11.18
withrep.AbsOcc*MaxS	.1102	.1227	.3649	.3749	**.5500**	.5332	11.40	11.36
withrep.AbsMaxS	.0371	.1156	.1249	.3691	.1680	**.5440**	9.79	11.39
withrep.TfIdf	.1017	.1158	.2960	.3584	.4210	.4602	10.86	10.97
norep.IG	.0501	.1311	.2283	.404	.4290	.5018	11.00	11.17
norep.AbsOccAvgS	**.1305**	**.1307**	**.3994**	**.4074**	.4890	.5019	11.11	11.15
norep.AbsOcc*MaxS	.1062	.1228	.3546	.3708	**.5362**	.5161	11.40	11.29
norep.AbsMaxS	.0392	.1227	.1952	.3715	.4520	**.5344**	11.09	11.34
norep.TfIdf	.1024	.1132	.3064	.3554	.4026	.4508	10.76	10.96
dbo.IG	.0411	.1319	.1989	.4083	.4425	.5053	11.06	11.20
dbo.AbsOccAvgS	.1283	.1292	.3986	.4063	.4915	.4949	11.14	11.14
dbo.AbsOcc*MaxS	.1062	.1214	.3546	.3710	.5362	.5109	11.40	11.27
dbo.AbsMaxS	.0381	.1211	.1927	.3727	.4291	.5222	10.97	11.31
dbo.TfIdf	.1024	.1132	.3064	.3554	.4026	.4508	10.76	10.96
dbp.IG	.0678	.1319	.2553	.4083	.4364	.5053	10.83	11.20
dbp.AbsOccAvgS	**.1319**	**.1316**	**.4026**	**.4113**	.4926	.5055	11.14	11.20
dbp.AbsOcc*MaxS	.1065	.1239	.3580	.3773	**.5444**	.5270	**11.42**	11.36
dbp.AbsMaxS	.0401	.1105	.1969	.3553	.4528	**.5447**	11.08	11.42
dbp.TfIdf	.0790	.1170	.2371	.3572	.3894	.4698	10.69	11.04

Table 4: Experimental results on the LibraryThing dataset. Bold values indicates that the difference with the other methods are statistical significant (T-test with p-value $<$ 0.0001).

configurations but we report only results for $k = 5, 20$ and best configurations for lack of space[9]. We highlight in bold only the values for which there is a statistical significant difference. For *Lastfm* dataset the differences are not statistical significant so the two methods are equivalent in selecting features.

Discussion. As an overall result, ABSTAT-based FS leads to the best results in terms of accuracy and diversity for both the movie and books domains while IG leads to better results (although not statistically significant) for music.

Specifically, considering the results on Movielens (Table 2), ABSTAT produces better accuracy with respect to IG in all the four configurations (noRep, withRep, Onlydbp, Onlydbo) both with 5 and 20 features. In terms of aggregate diversity, i.e. itemCoverage and aggrEntropy, ABSTAT is still the best choice, overcoming IG in almost all the situations. Interestingly, as for diversity we always obtain the best results with the AbsMaxS configuration. On Lastfm (Table 3) there are no particular differences, and hence the choice of the method seems irrelevant: both summarization-based and statistical methods are comparable. Eventually, on LibraryThing (Table 4), ABSTAT strongly beats IG in almost all the configurations. In particular, it gets more than twice of the precision and MRR respect to IG in top-5 features scenario. Also in this domain, ABSTAT is the best choice also in terms of catalog coverage on the AbsMaxS configuration, while the aggregate distribution is not particularly influenced by the two methods. Summing up,

[9] The interested reader can find results for all values of k and configurations on GitHub: `http://ow.ly/zAA530d0wu0`

Domain	Number of Minimal Patterns	Average Number of Triples	Variance
Movies	57757	74,015	549,313
Books	41684	44,966	169,478
Music	40481	80,502	981,509

Table 5: Ontological and data dimensions of the three datasets

ABSTAT beats IG in almost all the configurations on the two datasets Movielens and LibraryThing, while they act in the same way on the Lastfm dataset.

In order to investigate the reasons behind the different behaviors depending on the selected knowledge domain, we evaluated two orthogonal dimensions related to their corresponding sub-graphs. The two dimensions reflect the two different aspects related to the FS techniques. Indeed, while the ones based on ABSTAT are strongly grounded in the ontological nature of the knowledge graph, the others (IG, CHI and GR) mainly consider the triples representing data without taking into account the schema. Hence, for the three domains of movies, books and music we measured: (i) the number of minimal patterns and (ii) the average number of triples per resource and the corresponding variance. Regarding the former we may say that a higher number of minimal patterns means a richer and more diverse ontological representation of the knowledge domain. As for the latter, a high variance in the number of triples associated to resources is a clue of an unbalanced representation of the items to recommend. Hence, items with a higher number of triples associated result "more popular" in the knowledge graph compared to those with only a few. This may reflect in the rising of a stronger content popularity bias while computing the recommendation results. If we look at the values represented in Table 5 we see that while the music domain is the one with the lowest number of minimal patterns and the highest variance, books have the lowest values in terms of variance and, eventually movies show intermediate values in terms of variance and the highest number of minimal patterns. Based on these values we may assert that a higher sparsity in the knowledge graph data may give chance to statistical methods to beat ontological ones. In other words, it seems that the higher the sparsity of the knowledge graph at the data level, the lower the influence of the ontological schema in the selection of the most informative features to build a pure content-based recommendation engine.

5 Related Work

Summarization. Different approaches have been proposed for schema and data summarization. Here we compare our work to approaches that provide vocabulary/ontology-based summaries (or profiles) of linked data that describe, even if in an abstract way, the whole content of a dataset. We refer to [27] for a more detailed comparison also with summarization approaches for ontologies or aimed at representing only the most relevant content of a dataset. Several data profiling approaches have been proposed to describe linked data by reporting statistics about the usage of the vocabularies, types and properties. SchemeEx extracts interesting theoretic measures for large datasets, by considering the co-occurrence of types and properties [11]. Linked Open Vocabularies[10], RDFStats [12] and LODStats [2] provide such statistics. In contrast,

[10] `http://lov.okfn.org/`

ABSTAT represents connections between types using schema patterns, for which it also provides cardinality descriptors (a contribution of this paper). Loupe [14], a framework to summarize and inspect LD, extracts schema patterns that are similar to the ones extracted by ABSTAT, but without our minimalization approach, and their frequency. The additional information it provides (e.g., about provenance) does not include cardinality descriptors for properties or patterns. TermPicker extracts [24] patterns consisting in triples $\langle S, E, O \rangle$, where S and O are *sets* of types and E is a set of predicates. Instead, ABSTAT and Loupe extract patterns each consisting in a triple $\langle C, P, D \rangle$ where C and D are types and P a property. TermPicker summaries do not describe cardinality and are extracted from RDF data without considering relationships between types. Limited work has addressed the problem of extracting cardinality descriptors - related to mining cardinality constraints - as the ones introduced in this paper. According to a recent article [15], which proposes a method to define and discover classes of cardinality constraints with some preliminary results, current approaches focus only on mining keys or pseudo-keys (e.g., [26]). We discover richer statistics about property cardinality like the above mentioned work, but with a purely descriptive approach. In addition, we compute cardinality descriptors for properties occurring in specific schema patterns.

Recommender Systems. The world of recommender systems can be divided into two main classes: *Collaborative Filtering* and *Content-Based* systems. The former predicts users interests relying on the statistical information about the users ratings. The underlying assumption is that users sharing similar scores may have similar preferences, and similarly scored items may be of interest toassw the same users. Collaborative filtering recommender systems suffer from the data sparsity problem; while, content-based recommender systems exploit the descriptive content of the item (tags, textual description, etc.) in order to recommend items similar to the ones the user liked in the past. The latter do not suffer from data sparsity, as they do not rely on the ratings of different users[13]. Moreover, several times we miss descriptive content information about the items, by exploiting LD sources like `DBpedia`[4] we can overcome such a problem of missing content information. Several are the approaches proposed to exploit information extracted from LD in a recommendation scenario. One of the first approach for using LD in a recommender system was proposed by Heitmann and Hayes[9]. A system for recommending artists and music using DBpedia was presented in [21]. The task of cross-domain recommendation leveraging DBpedia as a knowledge-based framework was addressed in [5], while in [17] the authors present a content-based and context-aware method adopting a semantic representation based on a combination of distributional semantics and entity linking techniques. In [20] the authors use a a hybrid graph-based algorithm based on learning-to-rank method and path-based features extracted from networks built upon DBpedia and collaborative information [20]. To the best of our knowledge, the only approaches proposing an automatic selection of LD features are [16, 23]. In [16] seven different techniques for automatic selection of LD-based features are compared. Differently from [16], 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 for recommendation purposes. Differently from [23] here we have used a different recommendation algorithm, a new and full automatic approach to pre-processing and to rank

the features coming from the ABSTAT summarization tool and evaluated the approach on three different datasets. Finally, we observe that even approaches that do not perform automatic FS like [16, 4, 20] used (hand-crafted) FS to improve their performance.

6 Conclusions

In this work we investigated the role of ontology-based data summarization for feature selection in recommendation tasks. Having its roots in pure machine learning, feature selection techniques do not usually exploit the semantics associated to data while computing the importance of a set of features. Here we compare the results coming from ABSTAT, a schema summarization tool, with classical methods for feature selection and we show that the former are allowed to compute better predictions not just in terms of precision of the recommended items but also considering other dimensions such as diversity. Experiments have been carried out in three different knowledge domains namely movies, books and music thus showing the effectiveness of a feature selection based on schema summarization over classical techniques such as Information Gain.

In future work, we plan to use patterns and local cardinality descriptors provided by ABSTAT summaries to compute the most relevant paths to be used in multi-hop similarity measures used in RSs. As a result, more complex subgraphs that are estimated to be relevant for LD-based RSs could be extracted. In addition, based on the promising results obtained in the domain of LD-based RSs, we would like to extend our study to investigate the effectiveness of ontology-based schema summaries also in other application domains where semantic similarity measures are used, for example for semantic relatedness or entity co-resolution.

References

1. Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5), 2012.
2. Sören Auer, Jan Demter, Michael Martin, and Jens Lehmann. LODStats - An Extensible Framework for High-Performance Dataset Analytics. In *EKAW (2)*, 2012.
3. Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *RecSys*. ACM, 2011.
4. Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*. 2015.
5. Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. A generic semantic-based framework for cross-domain recommendation. In *2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems*, 2011.
6. Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In *SIGIR*. ACM, 2007.
7. Asela Gunawardana and Guy Shani. *Evaluating Recommender Systems*. Springer US, Boston, MA, 2015.
8. Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 2003.
9. Benjamin Heitmann and Conor Hayes. Using linked data to build open, collaborative recommender systems. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010.

10. Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2), 1997.
11. Mathias Konrath, Thomas Gottron, Steffen Staab, and Ansgar Scherp. Schemex - efficient construction of a data catalogue by stream-based indexing of linked data. *Web Semant.*, 16, 2012.
12. Andreas Langegger and Wolfram Wöß. RDFStats - an extensible RDF statistics generator and library. In *DEXA Workshops*, pages 79–83. IEEE, 2009.
13. Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*. Springer, 2011.
14. Nandana Mihindukulasooriya, Maria Poveda Villalon, Raul Garcia-Castro, and Asuncion Gomez-Perez. Loupe - An Online Tool for Inspecting Datasets in the Linked Data Cloud. In *ISWC Posters & Demonstrations*, 2015.
15. Emir Muñoz. On learnability of constraints from RDF data. In *ESWC*. Springer, 2016.
16. Cataldo Musto, Pasquale Lops, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. Semantics-aware graph-based recommender systems exploiting linked open data. In *UMAP*, 2016.
17. Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. Combining distributional semantics and entity linking for context-aware content-based recommendation. In *UMAP*, 2014.
18. Phuong Nguyen, Paolo Tomeo, Tommaso Di Noia, and Eugenio Di Sciascio. An evaluation of SimRank and Personalized PageRank to build a recommender system for the web of data. In *WWW '15 Companion*, pages 1477–1482. ACM, 2015.
19. Tommaso Di Noia. Knowledge-enabled recommender systems: Models, challenges, solutions. In *Proceedings of the 3rd International Workshop on Knowledge Discovery on the WEB, Cagliari, Italy, September 11-12, 2017.*, 2017.
20. Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Trans. Intell. Syst. Technol.*, 8(1), September 2016.
21. Alexandre Passant. Dbrec: Music Recommendations Using DBpedia. In *9th ISWC*, pages 209–224. Springer-Verlag, 2010.
22. Heiko Paulheim and Johannes Fümkranz. Unsupervised generation of data mining features from linked open data. In *WIMS*, 2012.
23. Azzurra Ragone, Paolo Tomeo, Corrado Magarelli, Tommaso Di Noia, Matteo Palmonari, Andrea Maurino, and Eugenio Di Sciascio. Schema-summarization in linked-data-based feature selection for recommender systems. In *SAC*. ACM, 2017.
24. Johann Schaible, Thomas Gottron, and Ansgar Scherp. Termpicker: Enabling the reuse of vocabulary terms by exploiting data from the linked open data cloud. In *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, pages 101–117, 2016.
25. Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys*. ACM, 2012.
26. Tommaso Soru, Edgard Marx, and Axel-Cyrille Ngonga Ngomo. ROCKER: A refinement operator for key discovery. In *WWW*. ACM, 2015.
27. Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino. ABSTAT: ontology-driven linked data summaries with pattern minimalization. In *2nd Workshop on Summarizing and Presenting Entities and Ontologies, co-located with ESWC.*, 2016.
28. Georgia Troullinou, Haridimos Kondylakis, Evangelia Daskalaki, and Dimitris Plexousakis. RDF Digest: Efficient Summarization of RDF/S KBs. In *ESWC*, 2015.