Unsupervised Generation of Data Mining Features from Linked Open Data

Heiko Paulheim
Technische Universität Darmstadt
Knowledge Engineering
Hochschulstrasse 10
64289 Darmstadt, Germany
paulheim@ke.tu-darmstadt.de

Johannes Fürnkranz
Technische Universität Darmstadt
Knowledge Engineering
Hochschulstrasse 10
64289 Darmstadt, Germany
fuernkranz@ke.tu-darmstadt.de

ABSTRACT

The quality of the results of a data mining process strongly depends on the quality of the data it processes. A good result is more likely to obtain the more useful background knowledge there is in a dataset. In this paper, we present a fully automatic approach for enriching data with features that are derived from Linked Open Data, a very large, openly available data collection. We identify six different types of feature generators, which are implemented in our open-source tool *FeGeLOD*. In four case studies, we show that our approach can be applied to different problems, ranging from classical data mining to ontology learning and ontology matching on the semantic web. The results show that features generated from publicly available information may allow data mining in problems where features are not available at all, as well as help improving the results for tasks where some features are already available.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based Services*; I.2.6 [**Artificial Intelligence**]: Learning—*Induction*

General Terms

Algorithms

Keywords

Machine Learning, Data Mining, Feature Generation, Linked Open Data, Semantic Web, Ontology Learning, Ontology Matching

1. INTRODUCTION

Data mining aims at finding regularities and patterns in data. For example, given a database of book sales, the book selling company can analyze which types of books sell better than others, which book stores have a larger turnover than others, and so on. In particular, such patterns can be used to make predictions about unknown entities – e.g., predicting whether a new book is going to sell well, or whether it is a good idea to open a new book store in a certain city. During the past decades, a large variety of algorithms for data mining have been developed [52]. Software packages like *Weka* [5] or *RapidMiner* [35] are industrial-strength toolkits that can be used for building highly sophisticated, intelligent applications.

However, the results produced by a data mining algorithm can only be as good as the data it gets as input. The data mining algorithm depends on the available features for formulating a pattern, thus a pattern can only be found if all the components that the pattern is composed of is present in the input data. For example, a valuable pattern in the book-selling domain could be *"Scientific books are sold well in cities with at least one large university"*. To find such a pattern, the book selling company needs to have access to up to date information on the cities where their books are sold, the categories of those books, the universities in the cities, etc.

Such general-purpose information, e.g., about cities, universities, and also books and their genres, is often available in the World Wide Web. With the Linked Open Data initiative [3], a large variety of such data has become available in a standardized, machine processable form. Built on W3C standards such as RDF [49] and SPARQL [51], general-purpose datasets as well as domain-specific datasets, e.g., from the medical domain are publicly available.[1]

In this paper, we introduce the open source toolkit *FeGeLOD*[2], which automatically creates useful data mining features from Linked Open Data. That extensible toolkit is based on *Weka* [18] and implements different methods for creating features from datasets in Linked Open Data. It takes a data file as input and automatically enhances it with further features in a fully automated, domain-agnostic process.

The rest of this paper is structured as follows. Section 2 gives an overview of related work. Section 3 discusses the approach and the implementation of our toolkit. In the following, we present two case studies in which we have used our approach for improving data mining results for existing data sets (Section 4), and mining datasets without any given features (Section 5). In Section 6, we present a deeper analysis of the quality of the features generated by our approach throughout the two first case studies. In Sections 7 and 8, we present two more case studies that show how our approach can be applied to ontology learning and ontology matching problems, respectively. We conclude with a summary and outlook on future work.

WIMS '12, June 13-15, 2012 Craiova, Romania
Copyright 2012 ACM 978-1-4503-0915-8/12/06 ...$10.00.

[1] See `http://lod-cloud.net/` for a an overview.

[2] `http://www.ke.tu-darmstadt.de/resources/fegelod`

2. RELATED WORK

Constructing features from Linked Open Data has sometimes been discussed in theory [47], acknowledging the use of the semantic web as background knowledge for data mining and machine learning tasks. There have also been some prototype implementations.

LiDDM [24] is an integrated system for data mining on the semantic web. It features different standard machine learning techniques, such as clustering and classification of semantic web instances, as well as discovery of association rules. However, the user has to declare the types of features to be extracted upfront by defining their own SPARQL query, which requires some knowledge about the ontology underlying the data set used.

A similar approach is taken by the *RapidMiner semweb plugin* [25], which preprocesses RDF data in a way that it can be further processed by a data mining tool, *RapidMiner* in that case. Like for *LiDDM*, the user has to specify a SPARQL query upfront for retrieving the data of interest.The authors propose different methods for turning set-valued data, such as several types of an instance, into simple nominal features in a way that the number of generated features is low and the approach scales well.

In [6], Cheng et al. have proposed a similar approach to feature generation as we have pursued with *FeGeLOD*. They perform some studies with traditional data mining tasks, such as text classification and product recommendation. Like the two approaches above, the approach foresees that the type of feature to be extracted has to be specified using specific SPARQL queries. Since the user has to specify the type of features to be generated by defining a SPARQL query, these approaches are *supervised* rather than *unsupervised*, and they require some domain knowledge from the user to define the query. In contrast, our approach works in an unsupervised way and does not require any domain knowledge from the user.

An approach using ontologies as background knowledge for data mining is *g-SEGS* [37]. Given an ontology, which provides a description of nominal attributes, the approach leverages the taxonomical knowledge in that ontology for improving the data mining process, e.g., for learning more general rules. Both the ontology and the mapping from the data set to that ontology has to be provided a priori in that approach.

Kämpgen and Harth suggest an approach for analyzing statistical linked data with online analytical processing (OLAP) tools. They discuss a common schema for such data and present various case studies. The two key differences is that our approach aims at running machine learning algorithms on the data, instead of OLAP methods, and that it uses general-purpose datasets (*DBpedia* at the moment) instead of special statistical datasets. Nevertheless, including statistical linked data sets in our approach may help increasing the quality of the prediction models created by machine learning algorithms in many of the examples discussed in this paper.

SPARQL-ML [26] is an approach that foresees the extension of the SPARQL query language [51] with a specialized statement to learn a model for a specific concept or numeric attribute in an RDF dataset. It therefore covers both classification and regression, other learning tasks – such as clustering or association rule mining – are thus not possible with SPARQL-ML. Furthermore, for applying the approach, it is required that the SPARQL endpoint of the data set that one wants to employ for data mining, e.g., *DBpedia*, supports SPARQL-ML, which is currently not very wide spread. In contrast, our approach can be used with *any* arbitrary SPARQL endpoint providing Linked Open Data.

There are also some works that use features from Linked Open Data for specific tasks, such as learning association rules [36], finding paths between entities [27], or ranking statements [16]. While those approaches often provide very good results for the specific task they are defined for, the work presented in this paper is rather designed as a broader framework with which such specific solutions can be built.

In summary, using linked open data and ontologies in data mining is an active field of research. Unlike most of the approaches discussed above, the work introduced in this paper follows a completely unsupervised approach, i.e., we aim at automatically generating features given a data set and a SPARQL endpoint, without the need of any prior knowledge about that particular endpoint and its ontology. Despite that property, the *FeGeLOD* prototype allows for creating various types of features due to the modular nature of the implementation, which provides extensibility with respect to new entity recognition, feature generation, and feature selection mechanisms.

3. APPROACH

The *FeGeLOD*approach for generating data mining features from Linked Open Data comprises three subsequent steps: entity recognition, the actual feature generation, and optional selection of a subset of the generated features. Figure 1 shows the process. Each of the steps allows for modular extensions with new algorithms.

After the initial data file is loaded (in CSV or ARFF format), a column is selected for which features are to be generated. In the first step, the entries in this column are mapped to URIs identifying entities in Linked Open Data. Then, features are generated for those entities, using different generation strategies, and a filtering algorithm ensures a result dataset of manageable size. The resulting file has additional columns (or features) which can then be used by a data mining tool. Each of the steps can be configured, with the tool running either in batch mode or controlled via a graphical user interface.

3.1 Entity Recognition

Entities in Linked Open Data are identified by URIs. For example, the Technical University of Darmstadt (TU Darmstadt) is identified by the URI `http://dbpedia.org/resource/Darmstadt_University_of_Technology` in the dataset *DBpedia*, a dataset automatically constructed from structured information contained in Wikipedia [4]. To obtain useful features from a dataset in Linked Open Data, the first necessary step is to identify the URIs that correspond to the entities in the dataset. In the book sales example introduced in section 1, the city name (or ZIP code) of the city would be resolved to a URI representing that city, which is inserted as a new feature, as shown in the first step of Fig. 1.

There are several possible ways for identifying such links. For *DBpedia*, as shown above, URIs always follow a common pattern, which most often has the name of the entity in question as one part. Therefore, we use a simple algorithm for "guessing" the correct URI: First, we turn the string (such as "darmstadt university of technology") into a corresponding string used for identification in *DBpedia*, i.e., capitalize all major words, replace blanks by underscores, etc. If this fails, we try to remove words in the end. For example, the URI for the "Auburn University at Montgomery" can be found as `http://dbpedia.org/resource/Auburn_University`. Since this algorithm is prone to false positives (e.g. it could fin the URI of the city `http://dbpedia.org/resource/Darmstadt` instead of the university URI), we allow for additional typechecking, either by letting the user select appropriate categories for the desired entities, or by using the majority of types extracted. For example, if most of the entities recognized are of type `http://dbpedia.org/ontology/University`, the entity `http://dbpedia`

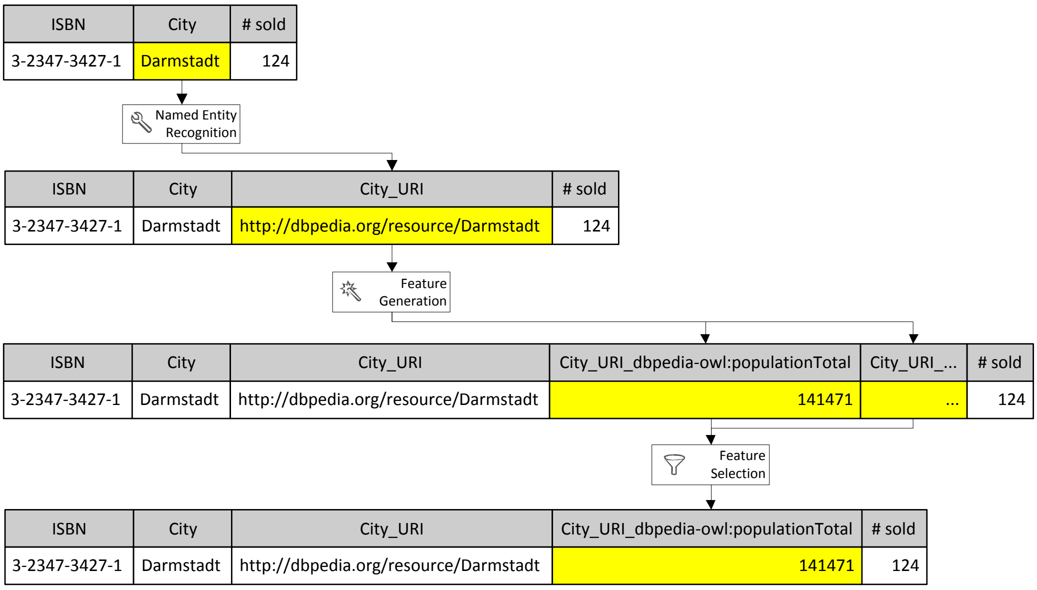

Figure 1: Our approach to feature generation from Linked Open Data. In a given data set, e.g., describing book sales in different cities, Linked Open Data URIs are identified using entity recognition. Then features are generated for those URIs, in this example providing more background information on those cities. A final feature selection step ensures a reasonable size of the resulting data set.

`.org/resource/Darmstadt` would be discarded since it is not of that type.

Although this approach to entity recognition is very simple, it provides reasonable results and recognition rates, as shown in the case studies below. The implementation and evaluation of more sophisticated entity recognition algorithms is an important item for future work, but beyond the scope of this paper.

3.2 Feature Generation

Once we have identified the corresponding URI for an entity, we can use it for generating features. Currently, *FeGeLOD* implements six different generation strategies, which are shown in Table 1 and explained in the remainder of this section. Two of them are only concerned with the entity itself, the other four take links to other entities into account.

The first generator creates one feature for each *data property* of an entity. Data properties are elementary values, such as the name and the population of a city. String-valued properties may often be ignored, because free text or names can typically not directly be used as features, or, in case of names or free-text descriptions (i.e., `rdfs:label` and `rdfs:comment`), do not provide any useful information for learning. However, sometimes, a limited set of string values, such as the name of the federal state a city is located in,[3] can be extracted into a useful nominal attribute. Numbers and dates are also turned into numeric and date features, and string attributes are turned into nominal features.

The second generator, which only works on an entity itself, examines all its *types*, i.e., all statements about the entity where the predicate is `rdf:type`. Since RDF allows an entity to be of several types, there are potentially different types and categories an entity may belong to. In *DBpedia*, for example, we can often find very detailed classification information, because *DBpedia* also contains lots of mappings to the more than 100,000 classes defined in *Yago* [46]. For example, the TU Darmstadt is of type *Organization*, *University*, and *Educational Institution* in the *DBpedia* ontology, as well as of type *Universities and Colleges in Germany* and *Technical Universities and Colleges* in *Yago*, amongst others.

The other four generators take the relation of entities to other entities into account. Two generators look at the *relations* in which an entity participates as a subject or object (i.e., incoming and outgoing edges in the RDF graph) and create a binary or a numeric attribute, respectively, specifying whether the relation exists, or how many relations exist from/to a particular number of other entities.[4]

Two more generators do not consider only relations, but also at the *related types*, i.e., they are concerned with *qualified relations*. For example, for the relation `locatedIn` relating entities with cities, the relation feature generator would only record the entities that are located in a city. The generators concerned with types of qualified relations, in contrast, would generate individual features for each type of entity (universities, companies, theatres, etc.) located in the city.

Table 1 sums up the SPARQL queries and the corresponding features generated. The table shows the basic forms of SPARQL queries; it is possible to impose further restrictions, e.g., only using relations and/or types from a set of given namespaces. Figure 2 depicts an excerpt of the *DBpedia* data about Darmstadt. Table 2 shows the features that would be generated for that example.

[3] Strictly speaking, this is the case where one of the central principles of Linked Open Data, i.e., *identify things with URIs* [3], is violated.

[4] We are aware that the semantics of these features deny two essential semantic fundamental principles of linked open data, the *open world assumption* and the *non-unique naming assumption*. For example, we would generate a feature stating that 42 scientists graduated from a particular university if we find 42 such relations, even though there are certainly more people that are not mentioned in the dataset. Moreover, some of the 42 may be duplicate identifiers for the same person. Despite its inaccuracy, the feature may have a value for data mining because, for example, the university may have a good reputation if it is the alma mater of 42 scientists that are well-known enough to be mentioned in a dataset like *DBpedia*.

Generator	Query	Feature	Values

Data properties	`SELECT ?p ?v WHERE {?x ?p ?v.`
	`FILTER(isLITERAL(?v)}`
		`?att_data_?p`	mixed

Types	`SELECT ?t WHERE {?x rdf:type ?t.}`	`?att_type_?t`	{true,false}

Relations (boolean)	`SELECT ?p WHERE {?x ?p ?y.}`
	`SELECT ?p WHERE {?y ?p ?x.}`
		`?att_in_boolean_?p`
		`?att_out_boolean_?p`
			{true,false}
			{true,false}

Relations (numeric)	`SELECT ?p ?y WHERE {?x ?p ?y.}`
	`SELECT ?p ?y WHERE {?y ?p ?x.}`
		`?att_in_numeric_?p`
		`?att_out_numeric_?p`
			numeric
			numeric

Qualified relations (boolean)	`SELECT ?p ?t WHERE {?x ?p ?y.`
	`?y rdf:type ?t.}`
	`SELECT ?p ?t WHERE {?y ?p ?x.`
	`?y rdf.type ?t.}`
		`?att_in_type_boolean_?p_?t`
		
		`?att_out_type_boolean_?p_?t`
			{true,false}
			
			{true,false}

Qualified relations (numeric)	`SELECT ?p ?y ?t WHERE {?x ?p ?y.`
	`?y rdf:type ?t.}`
	`SELECT ?p ?y ?t WHERE {?y ?p ?x.`
	`?y rdf.type ?t.}`
		`?att_in_type_numeric_?p_?t`
		
		`?att_out_type_numeric_?p_?t`
			numeric
			
			numeric

Table 1: SPARQL queries for the individual generators for a given resource ?x, and the corresponding attributes that are created. *?att* denotes the attribute name containing the original entity, such as a city's name.

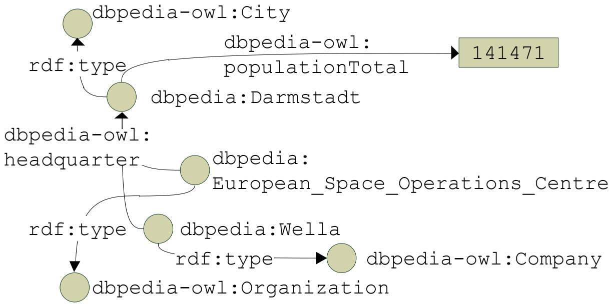

Figure 2: An excerpt from *DBpedia* showing some data about Darmstadt

3.3 Feature Selection

Some of the generators create a larger amount of features. Especially the feature sets generated by the qualified relation generators can become quite large. At the same time, some of the features may be extremely sparse. *Yago*, for example, defines many very specific categories, such as *"Art schools in Paris"*.

Especially for qualified relations, this can lead to a lot of features that are not too useful – the set of cities connected to an entity of type *"Art schools in Paris"* is probably limited to Paris alone, and all other instances will have a *zero* or *false* value for this feature. On the other hand, creating type features for `owl:Thing` will yield the value `true` for every instance. Such features will provide little value to a data mining tool, while their large number will impose certain scalability issues.

Therefore, it is useful to post-process the set of created features and discard features that have only little value. While there is a large body of work on feature selection [17], we have implemented only a rough heuristic so far: for a given threshold p, all features are discarded that have a ratio higher than p of missing, identical, or unique values (the latter is only applied to nominal features).

For our analysis, we have used values of 0.95 and 0.99 for p. That means that an attribute is discarded if it more than 95% (or 99%, respectively) of its values are missing, identical, or are different nominal values.

4. CASE STUDY 1: DATA MINING PROBLEMS WITH EXISTING FEATURES

To evaluate the value of our feature generation approach, we have tested it in several settings. The first case study is concerned with *existing* data mining problems, i.e., data mining problems that already come with a set of features. We have conducted tests with two datasets: the zoo data set from the UCI[5] [15], which poses a classification problem (i.e., classifying animals), and the AAUP[6] faculty salary data set[7], which poses a regression problem (i.e., predicting the average salary and compensation of different staff groups at American universities).

The zoo dataset has a string-valued attribute, containing the animal's name, 16 numeric attributes (15 of which only have 0 and 1 values), a class attribute with seven different values, and 101 instances. The goal is to predict the class of animal from the observed attributes.

The AAUP dataset has five numeric and two nominal attributes, plus a string-valued attribute containing the university's name, as input variables. It consists of 1161 instances. There are eight target variables, describing the average salary and the average compensation of different staff at the universities. From the latter, we have picked two target variables: the average salary and the average compensation of full professors.

For the zoo dataset, we have recognized 54.5% of the entities in *DBpedia* while for the AAUP data set, we have recognized 73.0% of all entities. We have run all six feature generators on both data sets. Furthermore, we have analyzed the performance of data mining algorithms on the unfiltered data, as well as with threshold values of 0.95 and 0.99 for p (see section 3.3).

Table 3 shows the classification accuracy for the zoo dataset using three common machine learning algorithms, i.e., Naïve Bayes [23], the Support Vector Machine algorithm *SMO*[8] [39], and the rule learning algorithm *Ripper* [7], all in their respective standard configurations in the publicly available *Weka* data mining library [18]. A higher accuracy corresponds to a better classification result.

[5] University of California, Irvine
[6] American Association of University Professors
[7] `http://www.amstat.org/publications/jse/jse_data_archive.htm`
[8] Sequential Minimal Optimization

Generator	Feature Name	Feature Value
Data properties	`dbpedia-owl:populationTotal`	141471
Types	`type_dbpedia-owl:City`	true
Relations boolean	`dbpedia-owl:headquarter_in_boolean`	true
Relations boolean	`dbpedia-owl:headquarter_in_numeric`	2
Qualified relations boolean	`dbpedia-owl:headquarter_in_type _dbpedia-owl:Organization_boolean`	true
	`dbpedia-owl:headquarter_in_type _dbpedia-owl:Company_boolean`	true
Qualified relations numeric	`dbpedia-owl:headquarter_in_type _dbpedia-owl:Organization_numeric`	1
	`dbpedia-owl:headquarter_in_type _dbpedia-owl:Company_numeric`	1

Table 2: Features generated for the example shown in Fig. 2

Feature Set	Algorithm	FS Threshold		
		1.00	0.99	0.95
plain	NB	97.03		
	Ripper	89.11		
	SMO	93.07		
plain + data properties	NB	95.05	95.05	–
	Ripper	**90.10**	89.11	–
	SMO	92.08	92.08	–
plain + types	NB	97.03	97.03	97.03
	Ripper	**94.06**	**91.09**	**91.09**
	SMO	**97.03**	**97.03**	**96.04**
plain + relations (boolean)	NB	90.10	91.09	92.08
	Ripper	**91.09**	**90.10**	89.11
	SMO	**95.05**	**95.05**	**95.05**
plain + relations (numeric)	NB	79.21	79.21	79.21
	Ripper	87.13	87.13	87.13
	SMO	**95.05**	**95.05**	**95.05**
plain + qualified relations (boolean)	NB	68.32	68.32	68.32
	Ripper	**92.08**	**90.10**	**94.06**
	SMO	**95.05**	**95.05**	**95.05**
plain + qualified relations (numeric)	NB	82.18	92.08	82.18
	Ripper	**91.09**	**96.04**	**91.09**
	SMO	**94.06**	**96.04**	**94.06**

Table 3: Accuracy results for the zoo dataset, using Naïve Bayes (NB), Ripper, and Support Vector Machines (SMO) as classificators, 10-fold cross validation, and using different feature selection (FS) threshold values p for feature selection. Results improving over the plain dataset without any generated features are marked in bold. The data properties feature generator did not generate any features that were kept when performing feature selection with a threshold of $p = 0.95$. From the joint feature set of all generated attributes with $p = 1.0$, we had to remove three attributes of type date for processing with Naïve Bayes and SMO.

The results demonstrate that, at least for SMO and Ripper, better classification can be achieved in many cases (especially using the generators for types and qualified relations), while Naïve Bayes often performs worse when adding a large amount of generated features. The latter is not surprising because the Naïve Bayes classifier assumes that the features are independent given the target class. If many of the extracted features have a correlation (e.g., type features for matching classes in *DBpedia* and *Yago*), this assumption is clearly violated, which may cancel out the positive effect of the additional information. SVMs and rule learning algorithms are not sensitive to this issue.

Table 4 shows the results of the two regression tasks (average salary and compensation of full professors) of the AAUP dataset. We have again used three common regression algorithms, i.e., Linear Regression, the regression variant of the Support Vector Machine implementation SMO [44], and M5 Rules [21], all in their respective standard configurations in *Weka*. The tables depict the root relative squared error of the prediction. A lower error corresponds to a better regression result.

It is obvious that the second task (predicting the average compensation) is harder than the first one (predicting the average salary); therefore, it has the larger potential for being improved by additional attributes. However, for both tasks, improvements are possible. While the Linear Regression results do not improve when adding features, there is room for improvement when using one of the other two regression algorithms. Especially the results produced with SMO are strongly improving, as well as SMO is the most robust when adding a larger amount of (possibly irrelevant) features. On the other hand, it can also be observed that the number of features can become too large to be handled by some approaches, especially by linear regression.

Overall, the results show that adding features from Linked Open Data can improve the result quality of some data mining algorithms. However, not every algorithm benefits from the same type of features in the same way, thus, it is necessary to choose a good combination of a data mining and a feature generation algorithm.

For both M5 and linear regression, the results are worse when no post selection of attributes is done, i.e., the filtering is done with a threshold of 1.00. Support vector machines, on the other hand, perform well with such large attribute sets, as in the classification case.

5. CASE STUDY 2: DATA MINING PROBLEMS WITHOUT EXISTING FEATURES

In our second case study, we have analyzed the performance of our approach for *new* data mining problems, i.e., problems where no features except for an entity and a target attribute are given. We have again used two different data sets for the evaluation. The first data set is the 1999 edition of the Mercer quality of living survey,[9] which is based on surveys on the quality of life in 218 cities worldwide. The results are mapped to a numeric scale, which is normalized so that 100 is the value of New York City resulting in a

[9] Data available at `http://across.co.nz/qualityofliving.htm`

Feature Set	Algorithm	*Average salary*			*Average compensation*		
		FS threshold			FS threshold		
		1.00	0.99	0.95	1.00	0.99	0.95
plain	LR	6.54			71.92		
	M5	6.54			59.88		
	SMO	7.08			74.12		
plain + data properties	LR	T	T	14.00	T	T	519.39
	M5	T	6.79	35.14	T	65.03	70.89
	SMO	14.03	7.45	17.59	297.39	777.07	1130.26
plain + types	LR	55.53	9.46	9.46	104.93	72.33	71.17
	M5	99.46	**6.47**	**6.46**	98.90	**53.96**	**53.96**
	SMO	10.65	8.21	8.20	**61.97**	**72.51**	**72.34**
plain + relations (boolean)	LR	59.58	32.44	27.24	118.42	88.56	92.48
	M5	98.58	6.64	6.64	98.60	**51.35**	**51.28**
	SMO	12.00	8.91	8.81	**65.08**	82.23	79.73
plain + relations (numeric)	LR	59.79	189.90	189.90	125.33	162.02	162.02
	M5	92.09	6.86	6.86	99.06	**52.78**	**52.78**
	SMO	12.04	9.58	9.58	**63.03**	79.06	79.06
plain + qualified relations (boolean)	LR	M	M	M	M	M	M
	M5	97.28	97.28	97.28	98.92	98.92	98.92
	SMO	13.48	13.49	13.49	**65.20**	**65.21**	**65.21**
plain + qualified relations (numeric)	LR	M	M	M	M	M	M
	M5	94.41	94.41	94.41	98.99	98.99	98.99
	SMO	13.23	13.23	13.23	**69.13**	**69.13**	**69.13**

Table 4: Results (relative root squared error) for the AAUP dataset, using Linear Regression (LR), M5 rules (M5) and Support Vector Machines (SMO) as regression algorithms, 10-fold cross validation, and using different feature selection (FS) threshold values p for feature selection. For running the full data properties data set with SMO, four date attributes have been removed. Results improving over the plain dataset without any generated features are marked in bold. For the relations numeric and the qualified relations boolean generator, the feature sets for $p = 0.95$ and $p = 0.99$ are identical, for the qualified relations generator, the feature sets for all three thresholds are identical. Tasks marked with T have timed out (i.e., not finished after one week), tasks marked with M have run out of memory (i.e., consumed more than 8GB).

range between 23 and 106. The second dataset is the 2010 edition of the Corruption Perceptions Index by Transparency International,[10] which assigns a numeric corruption index between 0 and 10 to 178 countries of the world [28]. Both datasets consist only of two columns: one with the city's or country's name, the other with the respective index.

For the Mercer dataset, we have created three equally sized classes with *Weka*'s default discretization algorithm [14]. Thus, the task is to classify the quality of living in a city as low, medium, or high quality. Like in the first case study, we have used three established classification algorithms. The classification accuracy results are depicted in table 5. As a baseline, we have run a classifier that always predicts the largest class. The results show that each generator produces useful features, since the baseline is outperformed by every combination of a feature generator and a classifier.

Some further interesting aspects can be observed. First, it is noteworthy that there is no feature generation strategy which is the best choice for all classification algorithms. While each classifier reaches its minimum accuracy using the type features, the maxima are found with different feature sets.

For the transparency international dataset, we have again used, like in the first case study, three common regression algorithms, i.e., Linear Regression, the regression variant of SMO, and M5 rules. The results are depicted in table 6. The baseline (predicting the average) can be beaten in most of the cases, but unlike in the first case study, one of the algorithms (M5 rules) performs rather badly.

When dealing with data mining problems without any given features, the motivation is not always to obtain predictions for new instances. Instead, it is sometimes more interesting to produce an *explanation* or an *interpretation* for the given phenomenon. In the datasets analyzed above, it may be interesting to find reasons what the typical attributes of a corrupt country are, or what makes the quality of life in a city high or low. Of the approaches used, rule learning approaches are particularly well suited to produce an interpretable model of such a phenomenon.

For the Mercer data set, the Ripper rule learner finds some useful information about the characteristics that separate low quality from high quality cities, using the different feature generators. For example, low quality cities are characterized as

- hot and very large cities (highest temperature in June larger than 27 C, area larger than $334km^2$)
- cold cities (highest temperature in January lower than 16 C) where no music has been recorded (no incoming relations of type `recorded- In`)
- cities in Africa (latitude less than 24, longitude less than 47, ignoring the sign)

We have performed a similar analysis with the Transparency International dataset, transforming the numerical class attribute into two discrete classes and running Ripper with the different generators. Some example rules found for states that have a low perceived corruption are

- OECD member states (entities from the *Yago*category `OECD-MemberEconomy`)

[10] Data available at `http://www.transparency.org/policy_research/surveys_indices/cpi/2010/results`

Feature Set	Algorithm	FS Threshold p		
		1.00	0.99	0.95
	Baseline	32.41		
data properties	NB	61.57	62.04	56.48
	Ripper	56.94	57.41	58.80
	SMO	68.06	68.06	67.59
types	NB	63.43	55.09	45.83
	Ripper	42.59	47.22	45.37
	SMO	57.41	53.24	46.30
relations (boolean)	NB	53.24	52.31	56.94
	Ripper	54.17	56.94	**61.11**
	SMO	**68.98**	68.06	64.81
relations (numeric)	NB	57.87	57.87	52.78
	Ripper	56.94	56.02	50.93
	SMO	61.11	61.11	63.43
qualified relations (boolean)	NB	50.93	61.11	54.63
	Ripper	46.30	50.46	49.54
	SMO	67.13	65.74	58.80
qualified relations (numeric)	NB	**67.59**	65.74	54.63
	Ripper	48.15	50.46	47.69
	SMO	66.67	68.06	**68.98**

Table 5: Mercer Quality of Living Dataset: Accuracy of Naïve Bayes (NB), Ripper, and Support Vector Machines (SMO) estimated by 10-fold cross-validation with three feature selection (FS) threshold values. Higher values indicate a better classification result. The best result for each classifier algorithm is marked in bold.

Feature Set	Algorithm	FS Threshold		
		1.00	0.99	0.95
	Baseline	100.00		
data properties	LR			
	M5	101.64	101.65	101.69
	SMO	82.29	82.35	82.31
types	LR	98.08	97.25	**73.67**
	M5	103.45	103.45	**76.41**
	SMO	74.94	78.03	76.01
relations (boolean)	LR	94.23	95.39	95.79
	M5	102.82	102.67	103.16
	SMO	67.43	**67.38**	72.16
relations (numeric)	LR	96.16	96.85	98.85
	M5	101.37	103.26	104.98
	SMO	97.37	114.13	147.02
qualified relations (boolean)	LR	M	M	90.26
	M5	101.91	101.91	102.34
	SMO	74.70	68.41	72.69
qualified relations (numeric)	LR	M	M	90.59
	M5	101.91	101.91	102.34
	SMO	74.70	68.41	72.69

Table 6: Transparency International Dataset: Relative root squared eror (RMSE) of Linear Regression (LR), M5 rules (M5) and Support Vector Machines (SMO) estimated by 10-fold cross-validation with three feature selection (FS) threshold values. Lower values indicate a better prediction. The best result for each classifier algorithm is marked in bold. Results marked with *M* could not be computed with 8GB of memory.

- Countries with a Human Development Index (HDI) larger than 78%. The HDI is calculated from life expectancy, education level, and economic performance of a country[11]
- Countries that are the headquarter of more than ten companies, but less than two cargo airlines[12]
- Countries that have more than ten mountains

These rules show that even if the relative error in the regression task for the Transparency International dataset is comparatively high, many of the rules that can be found indeed make sense. However, the last example shows that this does not apply to all the rules found. In particular, they may not be interpreted as cause-and-effect rules in a straight forward way. Rules may reflect very different circumstances, ranging from accidental correlations to actual cause-and-effect relations. In some cases, it may also not be trivial to decide which part of the rule is actually a cause and which is an effect – for example, companies may prefer less corrupt countries as their headquarters, but it is also possible that a larger number of companies hints at a more developed economy and thus lower perceived corruption.

Encouraged by these findings, we have developed the tool *Explain-a-LOD*, which uses the features generated by *FeGeLOD*together with rule learning algorithms for creating explanations on a dataset such as the ones discussed in this case study. The tool imports a statistical dataset, like the Mercer quality of living dataset, extracts features with *FeGeLOD*, runs a rule learning algorithm, and verbalizes the rules found as explanation. In an initial user study, users rated the explanations generated by the tool as useful, non-trivial, and surprising [38].

The results from the second case study strengthen the observations from the first one: features from Linked Open Data can be beneficially used for data mining, also for tasks where no other attributes are given. However, a careful evaluation is needed, as some algorithms seem to be sensitive to this additional information, which not always has high quality.

6. ASSESSMENT OF FEATURE QUALITY

In the case studies above, we have shown that Linked Open Data can serve as a source for deriving useful features for data mining tasks. So far, we have assessed the quality of the generated features only *indirectly* by showing their influence on the performance of different data mining algorithms. In this section, we are going to discuss a *direct* assessment of the quality of the generated features.

The case studies could have been performed with *any* type of features. Therefore, we have analyzed the features generated in these tasks with respect to two common feature quality measures: information gain ratio [40] and χ^2 value [31]. The number of features generated with each generator are depicted in table 7.

Figure 3 shows the gain ratio and the χ^2 value of the features generated for the zoo dataset in section 4. The figure depicts the measures for the top 10% of the features, the next best 10% of the features, etc. Although those curves look very differently for the individual data sets[13], there are a few recurring patterns:

[11] http://hdr.undp.org/en/statistics/

[12] As discussed above, the interpretation of this rule is not accurate when considering the open world assumption underlying Linked Open Data. A more concise interpretation would be "Countries that are the headquarter of more than ten companies, but less than two cargo airlines, which are important enough to have a dbPedia entry".

[13] A complete set of evaluations for all datasets examined in this paper can be found at http://www.ke.tu-darmstadt.de/resources/fegelod/feature-quality-estimation

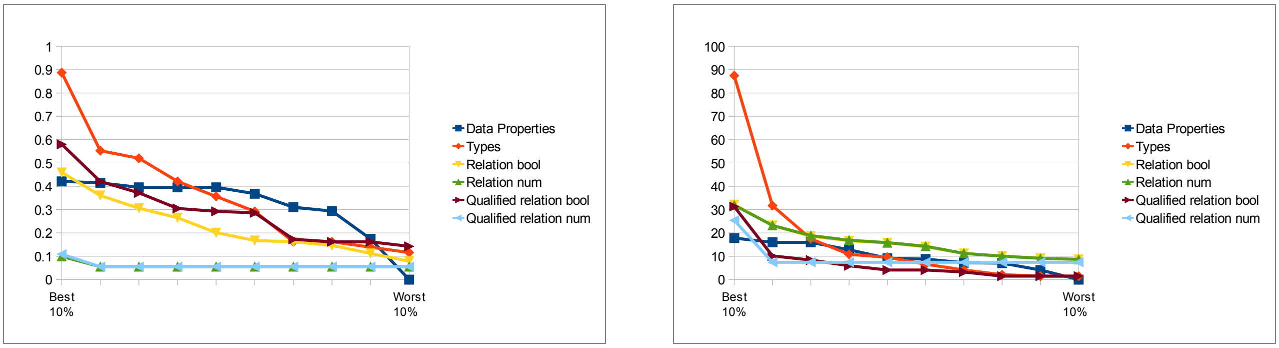

Figure 3: Gain ratio (left) and χ^2 value (right) of the features generated for the zoo dataset

- Data properties are most often rated high with both heuristics. Furthermore, they often have a higher average value than most of the other feature generators, which produce some high-valued, but also a lot of low-valued features.
- Type features are often rated high in terms of information gain ratio (but not necessarily in terms of χ^2).
- χ^2 often rates unqualified relation features very high.

In addition to those evaluations, we have done a user study with the tool *Explain-a-LOD* introduced above, which produces hypotheses for interpreting statistical datasets. To examine the subjective value of the different feature types, we have asked the participants to rate hypotheses generated based on the different feature types (see [38] for details). In that user study, we have observed that type features lead to the best-rated hypotheses, while other feature types produce mixed findings with respect to the individual data set and hypothesis generation strategy. We have also observed that the perceived plausibility of a hypothesis does not reflect the algorithmic rating, e.g., correlation score or confidence.

These findings, as well as the results from the case studies, show that there is not a best practice for producing highly useful features which works for each data set and each data mining algorithm. Instead, some experiments are required to find an optimal feature generation mechanism for each data set and algorithm. We have accounted for that finding by designing our prototype of *FeGeLOD* in a way that allows for combining different generators in a modular fashion.

7. CASE STUDY 3: ONTOLOGY LEARNING

In the third case study, we have addressed a different problem: the automatic induction of schema and ontology information from the instances of a dataset [11, 19, 33]. For datasets that do not have a richly axiomatized ontology, such ontology learning is useful for allowing more powerful reasoning on the dataset [45]. Ontology learning may also be used as a means to refine existing, weakly axiomatized ontologies by adding new formal axioms [32].

	Zoo	AAUP	Mercer	Transparency International
Data	36	452	1,205	614
Types	37	472	622	237
Relations	226	646	2,414	1,523
Qualified Relations	930	33,253	48,441	34,302

Table 7: Number of features generated for the different data sets. This table shows the numbers without any post-processing feature selection. The boolean and numerical variants of relations and qualified relations produce an equal number of features.

For employing our feature generation approach to the ontology learning problem, we follow the approach suggested in [8] and [48], where the Apriori association rule learner [1] is used for learning information about an ontology that underlies a given data set.

To show that our approach is applicable to the problem of ontology learning, we have used two datasets. The first dataset is a snapshot of FOAF (friend of a friend) documents on the semantic web. The FOAF ontology is one of the most prevalent ontologies [20], and it is used for describing persons, their affiliations and relations to other people. For our evaluation, we have used a dataset from 2005 [9] comprising 7118 documents, each describing one person. From that dataset, we tried to infer the FOAF ontology[14].

The second dataset consists of an ontology defining persons, groups of persons, and places occurring in the New Testament, and a corresponding instance set of 724 of such persons, groups, and places[15]. The two datasets have been selected since they provide both a set of instances which can be used for learning, as well as an ontology (which is the target to be learned) that can be used as a gold standard for the evaluation.

In our case study, we have concentrated on learning only very basic ontological features, i.e., subclass relations and the domain and range of relations. To that end, we have used the generators for types and for boolean relation features, and run an Apriori association rule miner on the generated dataset. The assocation rules found can be transferred back into ontology axioms, as depicted in table 8. If more than one rule of type 2 or 3 is found for the same predicate, they should be translated into an axiom stating that `?p` has the domain or range of the *union* of all the types (or an automatically generated superclass of the classes found), rather than letting all the axioms stand by themselves (which would implicitly mean that `?p` has the *intersection* of all types as its domain or range).

An example for a concept that can be learned using our approach is the `foaf:knows` relation holding between two persons, indicating that those persons know each other. When discovering the association rule

```
p_in_boolean_foaf:knows = true →
        p_type_foaf:Person,
```

[14] http://xmlns.com/foaf/spec/

[15] http://www.semanticbible.com/ntn/ntn-overview.html

	Association Rule	Ontology Axiom
1	`?att_type_?t1=true → ?att_type_?t2=true`	`?t1 rdfs:subClassOf ?t2.`
2	`?att_in_boolean_?p=true → ?att_type_t=true`	`?p rdfs:range ?t.`
3	`?att_out_boolean_?p=true → ?att_type_t=true`	`?p rdfs:domain ?t.`

Table 8: Association rules and their transformation to ontology axioms. The feature names follow the naming convention introduced in table 1.

Ontology	Axiom	Precision	Recall	F-Measure	Precision*	Recall*	F-Measure*
FOAF	class	1.00	1.00	1.00	1.00	1.00	1.00
	subClassOf	0.00	0.00	0.00	0.00	0.00	0.00
	property	0.79	0.54	0.64	0.79	0.54	0.64
	domain	0.31	0.31	0.31	0.84	0.84	0.84
	range	1.00	0.50	0.67	1.00	0.50	0.67
New Testament Names	class	1.00	0.43	0.60	1.00	0.43	0.60
	subClassOf	0.00	0.00	0.00	0.00	0.00	0.00
	property	1.00	0.76	0.87	1.00	0.76	0.87
	domain	0.07	0.14	0.10	0.43	0.86	0.57
	range	0.05	0.06	0.06	0.45	0.58	0.51

Table 9: Ontology learning results, depicting the recall, precision, and F-measure for the different types of axioms. The measures marked with an asterisk depict the relaxed figures for those axioms that were actually discoverable from the data set, ignoring those concepts that were not used in the respective data set.

we can infer that the relation `foaf:knows` has the range `Person` (which implicitly tells us that there is a relation called `foaf:knows` and a class called `Person`). Likewise, we can infer the domain of the relation, which completes the axioms defining that concept.

Table 9 shows the results of learning ontologies on those datasets, depicting the recall, precision, and F-measure for the different types of axioms. For each type of axiom analyzed, we have compared the axioms learned by our approach to the axioms in the corresponding gold standard ontology. We have analyzed both the absolute recall, precision, and F-measure, as well as a *relaxed* set of measures, which involves only classes and relations that are explicitly used in the instance sets. For example, the class `foaf:document`, which is defined in the FOAF ontology, is not used in our instance set, therefore, we have ignored it when computing the relaxed recall.

The classes and properties have been learned implicitly by applying simple RDFS entailment rules [50]: `?c1 rdfs:subClass ?c2` entails that `c1` and `c2` are classes. With the same type of rules, properties can be identified.

The figures show that the results are mixed. While the retrieval of classes and properties already works quite well, the results for properties, ranges, and subclasses are not always satisfying. A large problem with the above datasets is that they have no fully materialized type information. For example, in the New Testament Names dataset, there are quite a few entities of type `Man` and `Woman`, respectively, while the common superclass `Human` is not used at all in the instance set. Thus, the latter class cannot be discovered, neither can be the corresponding subclass relations. Since in our data set, no entities were described using a class and its superclass at the same time, our approach was not able to discover any subclass relations.

Furthermore, when trying to learn domain and range restrictions for such classes, it is learned that a lot of relations have the union of `Man` and `Woman` as their range, while the actual range (the class `Human`) is not discovered. However, such issues cannot be solved fully automatically. A possible option would be to involve a user in an interactive setting, asking them to define a common superclass for classes that co-occur frequently.

Overall, the results show that our approach may not be applied only to classical data mining problems, but also to ontology learning. While our first results are encouraging, one can try to further improve these results in various ways, such as, e.g., learning restrictions for individual classes, or exploiting the numerical feature generators for learning cardinality restrictions.

8. CASE STUDY 4: ONTOLOGY MATCHING

The fourth case study addresses as a similar problem as ontology learning. While ontology learning is always concerned with creating or improving a *single* ontology, ontology matching aims at creating relations between *multiple* ontologies [2, 12, 13]. When dealing with simple mappings, i.e., correspondences between single classes in different ontologies, the problem is very close to the problem of ontology learning: it is necessary to find `subClassOf` relations between classes defined in different ontologies. An exact match of two classes can be predicted from two symmetric `subClassOf` statements.

Ontology matching is an essential building block for the Semantic Web and Linked Open Data. Since many ontologies are used in parallel for different datasets [20], an intelligent agent may only make sense from those different datasets if the mappings between the used ontologies are known. While Linked Open Data provides ample information about instance-level correspondences, mappings on the terminological level are currently rare [22].

Since some datasets in Linked Open Data use multiple ontologies in parallel, they allow for applying an instance-based matching approach [10], i.e., exploiting information about instances belonging to ontology classes for constructing a mapping between such classes. As discussed above, the *DBpedia* dataset uses both its own ontology as well as the *Yago* classification system [4]. Thus, it can be used to predict a mapping between those two ontologies.

For our fourth case study, we used a partial gold standard mapping between the *DBpedia* and the *Yago* ontology[16]. To construct

[16] `http://www.netestate.de/De/Loesungen/DBpedia-YAGO-Ontology-Matching`

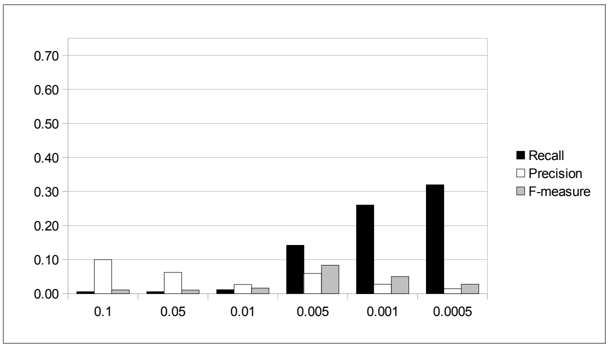

Figure 4: Matching results with different values for the minimum rule support, without applying any post processing

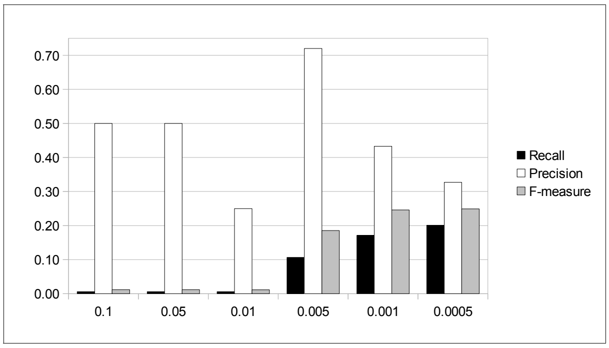

Figure 5: Matching results with different values for the minimum rule support, using rule confidence for conflict resolution

the dataset, we first gathered a set of instances that are an `rdf:type` of at least one of the types defined in the partial gold standard. On this set of instances, we have run the generator creating boolean features. The resulting data set has 231,635 instances and 98,414 attributes. Since a boolean attribute is created for each possible class, but each instance only belongs to a few classes, the data set is extremely sparse.

Like in the third case study, we have ran an Apriori association rule miner on the data set, using different values for the minimum support for each rule. We have learned association rules with one condition in the head and one in the body. For each learned pair of rules $X \rightarrow Y$ and $X \rightarrow Y$, where X and Y depict classes from different ontologies, we have assumed a mapping between those classes. We have evaluated that mapping against the a partial reference alignment between *DBpedia* and *Yago*, using the recall, precision, and F-measure formulas for partial reference alignments defined in [42].

We have used two different approaches for creating the mapping: in the first approach, all mappings found for which association rules in both directions exist have been taken into account. In the second approach, we have enforced a 1:1 mapping by resolving ties based on the confidence scores of the rules that have lead to creating the mapping. The results are shown in figures 4 and 5, respectively, using different values for a rule's minimum support (i.e., fraction of instances that a particular rule has to cover) for constructing the rules. It can be observed that with a decreasing minimum support, the recall value increases at the cost of precision. Filtering the resulting mappings based on the rules' confidence scores can clearly improve precision while not sacrificing too much recall, which shows that there is a clear benefit in using not only the learned rules themselves, but also the meta information provided by the learning algorithm. However, it is not clear whether the rules' confidence score is the best criterion, since precision takes its maximum at a medium value for a rule's minimum support, i.e., when a lower number of rules is generated. This hints at difficulties in selecting the correct mapping from a larger set of candidates.

In total, an F-measure of 25% can be achieved. While this is not too much compared to powerful state-of-the-art ontology matching approaches, our instance-based approach is capable of discovering mappings that are hard to find with merely syntactical, schema-level tools, e.g., the correspondence between `dbpedia:ProtectedArea` and `yago:Park`. The possibility to find those difficult mappings makes our approach an ideal candidate for using it as an addendum to the currently prevalent schema-level approaches for increasing the result quality.

While we have looked only at simple class mappings so far, it is possible to use further features for learning more complex mappings [41], such as "A `YAGO:FilmDirector` is a `dbPedia:Person` in who is the `dbpedia:director_of` of at least one `dbPedia:Film`". Those could be easily discovered by using the generators for qualified relation features as well and including them in the rule learning dataset. However, gold standards for evaluating such complex mappings are currently not available.

9. CONCLUSION AND OUTLOOK

In this paper, we have introduced an approach for unsupervised generation of data mining features from Linked Open Data. Our approach comprises three steps: entity recognition, feature generation, and feature selection mechanism. We have implemented our approach in a modular architecture which allows for using different algorithms for each of those steps. The focus of this paper has been on the actual feature generation step. We have introduced six different generation algorithms and applied them to various scenarios. The approach described in this paper works completely unsupervised, i.e., it can process a data set without the need of any further information given by the user, and without any domain knowledge. However, some basic information on objects' types may improve the precision in the entity recognition step.

In a number of case studies, we have shown how our approach can be applied to a number of different scenarios. Even with the simple feature generation technique that we used in this paper, the generated features may help improve the results in data mining tasks where features already exist, as well as for dealing with data mining problems without any features. Furthermore, we have shown how our approach can be used in the fields of ontology learning and ontology matching.

The results of the case studies and the assessments of the generated features show that there is no feature generation strategy that performs optimally for every given data set, data mining problem, and machine learning algorithm. At the moment, selecting suitable generators requires some experimentation. In the future, we aim at further assisting the user by predicting the benefit of generators in a given scenario and provide suggestions for generators.

As the focus of this paper has been on the actual generation step, the entity recognition as well as the feature selection step have been implemented in a very straight-forward way in order to provide a proof of concept. However, both steps deserve further attention. For entity recognition, existing approaches such as *DBpedia* Spotlight [34] or the *DBpedia* lookup service may be employed (see [43] for a survey).

Some entity recognition algorithms do not only deliver the identified entity, but also a confidence score. Entities which have been identified with a lower confidence score could be assigned a lower weight, so their influence on the learned model would be reduced. Such an adjustment could help improving the overall result.

As some of the generators create a very large number of features, the employment of more sophisticated feature selection mechanisms [17] would be beneficial. Here, it is especially important to find mechanisms that are capable of dealing with the very sparse nature of Linked Open Data. Furthermore, with respect to runtime optimization, it would be beneficial to be able to discard features on the fly, i.e., in the moment in which they are generated, so that unnecessary queries are avoided. We are currently investigating whether the logic-based framework for feature relevancy, which has been introduced in [29], could be useful for this task.

At the moment, all of our generators are restricted to the entity and its immediate neighbors. This imposes a limitation on the sort of features that can be found. For example, for the perceived corruption of a country, features such as the turnover of companies located in that country might be useful. Likewise, relations to individuals might also provide useful features (e.g., the relation of a country to its government type, which is a relation between individuals,[17] would be highly useful for predicting the perceived corruption). However, generating such features leads to an unmanageable explosion of the feature space. Our approach, like other state-of-the-art approaches [30], foresees fully calculating the metric of each attribute while it is being examined. A more scalable approach would have to rely on quicker approximate estimates of a feature's prospective value instead, performing an on-the-fly selection of features and interesting paths to follow.

So far, we have only used data gathered from *DBpedia*. As many data sets in Linked Open Data are linked to *DBpedia* [4], this is a good starting point. However, extending the feature generation to other data sets may also provide useful information. Finding good heuristics for identifying useful data sets while avoiding an explosion of the feature space would lead to a significant improvement of the feature generation mechanism. It will also be interesting to compare the result quality of predictive models generated from different data sets.

In summary, our case studies have shown that our approach to feature generation provides a useful framework to build solutions for different problems. All the algorithms described in this paper are available in an open-source implementation, which is based on the wide-spread machine learning framework *Weka*. With this, we hope to provide a valuable and useful toolkit to the community.

Acknowledgments

This work was supported by the German Science Foundation (DFG) project FU 580/2 "Towards a Synthesis of Local and Global Pattern Induction (GLocSyn)".

10. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *20th International Conference on Very Large Data Bases*, pages 478–499. Morgan Kaufmann, Los Altos, CA, 1994.

[2] Z. Bellahsene, A. Bonifati, and E. Rahm, editors. *Schema Matching and mapping*. Springer, 2011.

[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics - Science Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[5] R. R. Bouckaert, E. Frank, M. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. WEKA — Experiences with a Java open-source project. *Journal of Machine Learning Research*, 11:2533–2541, Sept. 2010.

[6] W. Cheng, G. Kasneci, T. Graepel, D. Stern, and R. Herbrich. Automated feature generation from structured knowledge. In *20th ACM Conference on Information and Knowledge Management (CIKM 2011)*, 2011.

[7] W. W. Cohen. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[8] C. d'Amato, N. Fanizzi, and F. Esposito. Inductive learning for the semantic web: What does it buy? *Semantic Web Journal*, (9), 2009.

[9] L. Ding, L. Zhou, T. Finin, and A. Joshi. How the semantic web is being used: An analysis of foaf. In *Proceedings of the 38th International Conference on System Sciences*, 2005.

[10] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th international conference on World Wide Web*, pages 662–673, New York, NY, USA, 2002. ACM.

[11] L. Drumond and R. Girardi. A survey of ontology learning procedures. In F. L. G. de Freitas, H. Stuckenschmidt, H. S. Pinto, A. Malucelli, and Ó. Corcho, editors, *3rd Workshop on Ontologies and their Applications*, volume 427 of *CEUR Workshop Proceedings*, 2008.

[12] J. Euzenat, A. Ferrara, C. Meilicke, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Sváb-Zamazal, V. Svátek, and C. Trojahn. Results of the Ontology Alignment Evaluation Initiative 2010. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010)*. 2010.

[13] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer, 2007.

[14] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuousvalued attributes for classification learning. In *Thirteenth International Joint Conference on Articial Intelligence*, volume 2, pages 1022–1027. Morgan Kaufmann Publishers, 1993.

[15] A. Frank and A. Asuncion. UCI Machine Learning Repository, 2010.

[16] T. Franz, A. Schultz, S. Sizov, and S. Staab. Triplerank: Ranking semantic web data by tensor decomposition. In *International Semantic Web Conference (ISWC)*, 2009.

[17] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, editors. *Feature Extraction – Foundations and Applications*. Springer, 2006.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[19] M. Hazman, S. R. El-Beltagy, and A. Rafea. Article: A survey of ontology learning approaches. *International Journal of Computer Applications*, 22(8):36–43, May 2011.

[20] M. Hepp. Possible ontologies: How reality constraints building relevant ontologies. *IEEE Internet Computing*, 11(1):90–96, January/February 2007.

[17] `dbpedia:Germany` and `dbpedia:Federal_State`

[21] G. Holmes, M. Hall, and E. Frank. Generating rule sets from model trees. In *Twelfth Australian Joint Conference on Artificial Intelligence*, pages 1–12. Springer, 1999.

[22] P. Jain, P. Hitzler, A. P. Sheth, K. Verma, and P. Z. Yeh. Ontology alignment for linked open data. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Z. 0007, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *9th International Semantic Web Conference, ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 402–417. Springer, 2010.

[23] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.

[24] V. N. P. Kappara, R. Ichise, and O. Vyas. Liddm: A data mining system for linked data. In *Workshop on Linked Data on the Web (LDOW2011)*, 2011.

[25] M. A. Khan, G. A. Grimnes, and A. Dengel. Two pre-processing operators for improved learning from semanticweb data. In *First RapidMiner Community Meeting And Conference (RCOMM 2010)*, 2010.

[26] C. Kiefer, A. Bernstein, , and A. Locher. Adding data mining support to sparql via statistical relational learning methods. In *5th European Semantic Web Conference (ESWC 2008)*, pages 478–492, 2008.

[27] K. J. Kochut and M. Janik. Sparqler: Extended sparql for semantic association discovery. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, pages 145–159. Springer, 2007.

[28] J. G. Lambsdorff. *The Institutional Economics of Corruption and Reform*. Cambrige Press, 2007.

[29] N. Lavrac, J. Fürnkranz, and D. Gamberger. Explicit feature construction and manipulation for covering rule learning algorithms. In J. Koronacki, Z. W. Ras, S. T. Wierzchon, and J. Kacprzyk, editors, *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski*, volume 262 of *Studies in Computational Intelligence*, pages 121–146. Springer, 2010.

[30] H. T. Lin, N. Koul, and V. Honavar. Learning relational bayesian classifiers from rdf data. In *10th International Semantic Web Conference (ISWC2011)*, 2011.

[31] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *7th International Conference on Tools with Artificial Intelligence*, pages 388–391, 1995.

[32] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72 – 79, 2001.

[33] A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 173–190. Springer, 2004.

[34] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, 2011.

[35] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, editors, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, pages 935–940, New York, NY, USA, August 2006. ACM.

[36] V. Nebot and R. Berlanga. Mining association rules from semantic web data. In *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems - Volume Part II*, pages 504–513, Berlin, Heidelberg, 2010. Springer.

[37] P. K. Novak, A. Vavpetič, I. Trajkovski, and N. Lavrač. Towards semantic data mining with g-segs. In *Proceedings of the 11th International Multiconference Information Society (IS 2009)*, 2009.

[38] H. Paulheim. Generating Possible Explanations for Statistics from Linked Open Data. In *9th Extended Semantic Web Conference (ESWC)*, 2012.

[39] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[40] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[41] D. Ritze, C. Meilicke, O. Svab-Zamazal, and H. Stuckenschmidt. A pattern-based ontology matching approach for detecting complex correspondences. In *Fourth International Workshop on Ontology Matching*, 2009.

[42] D. Ritze and H. Paulheim. Towards an automatic parameterization of ontology matching tools based on example mappings. In *Sixth International Workshop on Ontology Matching*, 2011.

[43] G. Rizzoa and R. Troncy. Nerd: evaluating named entity recognition tools in the web of data. In *Workshop on Web Scale Knowledge Extraction (WEKEX'11)*, 2011.

[44] S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy. Improvements to the smo algorithm for svm regression. In *IEEE Transactions on Neural Networks*, 1999.

[45] G. Stumme, A. Hotho, and B. Berendt. Semantic web mining - state of the art and future directions. *Journal of Web Semantics*, 4(2):124–143, 2006.

[46] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706. ACM, 2007.

[47] V. Tresp, M. Bundschus, A. Rettinger, and Y. Huang. Towards machine learning on the semantic web. In P. C. Costa, C. D'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web I*, pages 282–314. Springer-Verlag, Berlin, Heidelberg, 2008.

[48] J. Völker and M. Niepert. Statistical schema induction. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Part I*, pages 124–138, Berlin, Heidelberg, 2011. Springer-Verlag.

[49] W3C. RDF. `http://www.w3.org/TR/2004/REC-rdf-concepts`, 2004.

[50] W3C. RDF Semantics. `http://www.w3.org/TR/rdf-mt/`, 2004.

[51] W3C. SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`, 2008.

[52] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition, 2005.