How Linked Data can aid Machine Learning-based Tasks

Michalis Mountantonakis and Yannis Tzitzikas

Institute of Computer Science, FORTH-ICS, Greece, and
Computer Science Department, University of Crete, Greece
`{mountant,tzitzik}@ics.forth.gr`

Abstract. The discovery of useful data for a given problem is of primary importance since data scientists usually spend a lot of time for discovering, collecting and preparing data before using them for various reasons, e.g., for applying or testing machine learning algorithms. In this paper we propose a general method for discovering, creating and selecting, in an easy way, valuable features describing a set of entities for leveraging them in a machine learning context. We demonstrate the feasibility of this approach by introducing a tool (research prototype), called `LODsyndesis`$_{\mathcal{ML}}$, which is based on Linked Data technologies, that a) discovers automatically datasets where the entities of interest occur, b) shows to the user a big number of useful features for these entities, and c) creates automatically the selected features by sending SPARQL queries. We evaluate this approach by exploiting data from several sources, including *British National Library*, for creating datasets in order to predict whether a book or a movie is popular or non-popular. Our evaluation contains a 5-fold cross validation and we introduce comparative results for a number of different features and models. The evaluation showed that the additional features did improve the accuracy of prediction.

Keywords: Linked Data, Machine Learning, Feature Discovery & Selection, Automatic Classification, Prediction

1 Introduction

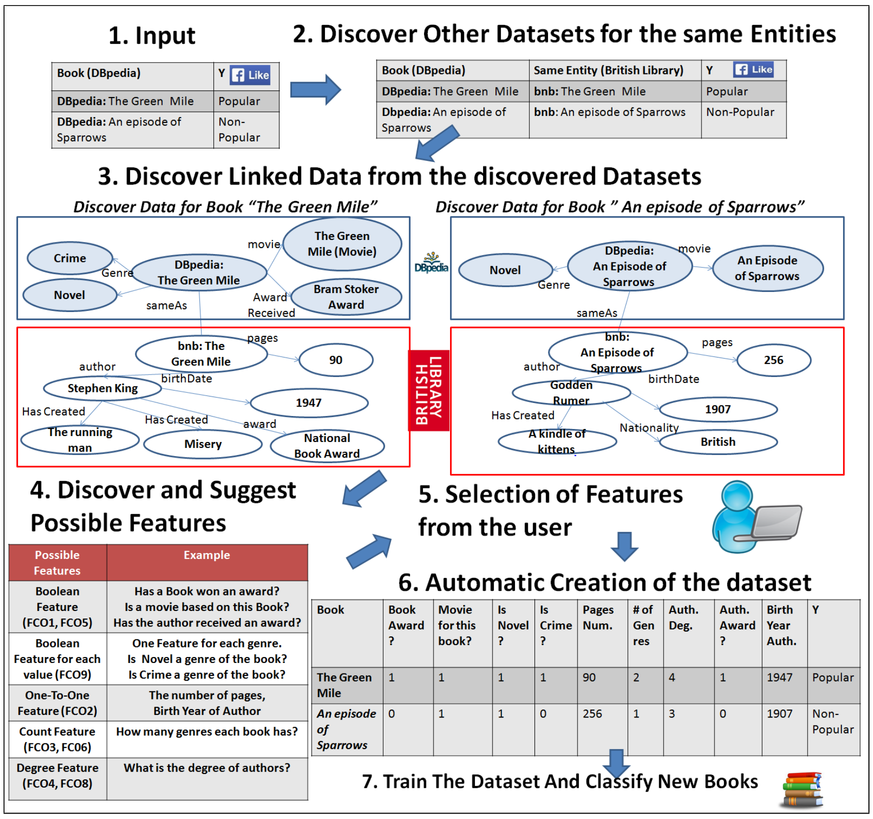
It has been written that "Data scientists spend 50%-80% of their time in collecting and preparing unruly digital data, before it can be explored for useful nuggets"[1], thereby, it is beneficial to investigate novel methods for reducing the aforementioned cost. The objective of this paper is to propose a method, that is based on Linked Data, for discovering, creating and selecting, in an easy way, valuable features describing a set of entities for being used in any *Machine Learning* (*ML*) problem.

Linked Data [4] refers to a method of publishing structured data while its ultimate objective is linking and integration. It is based on Semantic Web technologies, such as HTTP, URI and RDF, which enables the information to be read automatically by computers and data from different sources to be connected and queried. It differs from other traditional data formats predominantly due to the following reasons: Firstly, data linking facilitates the discovery of datasets containing information about a specific entity (or a set of entities). Secondly, datasets can be integrated more easily through the existence of common entities and common schema elements, which is desirable for exploiting the complementarity of information. For instance, one dataset can contain information about the authors of a book and another about user reviews for that book, thereby, the integration of such datasets offer more features for the entities. Thirdly, complex features can be derived by exploiting SPARQL [17] queries (e.g., "number of awards for each book") and graph metrics (e.g., average degree of an entity). A lot of datasets are published in RDF format, i.e., LODStats [6] provides statistics about approximately ten thousand discovered linked datasets.

In this work we show how the wealth of Linked Data and the ML machinery can be jointly exploited for improving the quality of automated methods for various time consuming and/or tedious tasks, which are important also in the area of digital libraries, like automatic semantic annotation or classification, completion of missing values, clustering, or computing recommendations. Specifically, we focus on exploiting Linked Data for discovering and creating features for a set of entities. We introduce a process where (i) we discover datasets and URIs containing information for a set of entities by exploiting *LODsyndesis* [12], (ii) we provide the user with a large number of possible features that can be created for these entities (including features for direct and indirect related entities of any path) and (iii) we produce automatically a dataset for the features selected by the user. For testing whether this enriched dataset can improve ML tasks, we report experimental results over two datasets (from [19]) for predicting the popularity of a set of movies and books. Figure 1 illustrates the running example, where we create features for classifying whether a book is *Popular* or *Non-Popular*, containing data discovered from *DBpedia* [10] and *British National Library* [16]. We evaluate this approach by performing a 5-fold cross validation for estimating the performance of different models for the produced datasets. The evaluation showed that the additional features did improve the accuracy of prediction.

The rest of this paper is organized as follows: Section 2 discusses background and related approaches, Section 3 states the problem and describes the functionality of the proposed tool (research prototype), Section 4 discusses the steps

[1] http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html


1. Input
2. Discover Other Datasets for the same Entities
3. Discover Linked Data from the discovered Datasets
Discover Data for Book "The Green Mile"
Discover Data for Book " An episode of Sparrows"
Crime
Genre
Novel
DBpedia: The Green Mile
movie
The Green Mile (Movie)
Award Received
Bram Stoker Award
sameAs
bnb: The Green Mile
pages
90
author
birthDate
Stephen King
1947
Has Created
The running man
Misery
award
National Book Award
DBpedia
BRITISH LIBRARY
Novel
Genre
DBpedia: An Episode of Sparrows
movie
An Episode of Sparrows
sameAs
bnb: An Episode of Sparrows
pages
256
author
Godden Rumer
birthDate
1907
Has Created
A kindle of kittens
Nationality
British
4. Discover and Suggest Possible Features
5. Selection of Features from the user
6. Automatic Creation of the dataset
7. Train The Dataset And Classify New Books

Book (DBpedia)	Y Like
DBpedia: The Green Mile	Popular
DBpedia: An episode of Sparrows	Non-Popular

Book (DBpedia)	Same Entity (British Library)	Y Like
DBpedia: The Green Mile	**bnb:** The Green Mile	Popular
Dbpedia: An episode of Sparrows	**bnb:** An episode of Sparrows	Non-Popular

Possible Features	Example
Boolean Feature (FCO1, FCO5)	Has a Book won an award? Is a movie based on this Book? Has the author received an award?
Boolean Feature for each value (FCO9)	One Feature for each genre. Is Novel a genre of the book? Is Crime a genre of the book?
One-To-One Feature (FCO2)	The number of pages, Birth Year of Author
Count Feature (FCO3, FC06)	How many genres each book has?
Degree Feature (FCO4, FCO8)	What is the degree of authors?

Book	Book Award ?	Movie for this book?	Is Novel ?	Is Crime ?	Pages Num.	# of Genres	Auth. Deg.	Auth. Award ?	Birth Year Auth.	Y
The Green Mile	1	1	1	1	90	2	4	1	1947	Popular
An episode of Sparrows	0	1	1	0	256	1	3	0	1907	Non-Popular

Fig. 1: Running Example

of the process, Section 5 reports the results of the evaluation and discusses the effectiveness of the proposed features, and finally, Section 6 concludes the paper.

2 Background & Related Work

Background. The Resource Description Framework (`RDF`) [2] is a graph-based data model. RDF uses `Triples` in order to relate Uniform Resource Identifiers (`URIs`) or anonymous resources (`blank nodes`) where both of them denote a `Resource`, with other `URIs`, `blank nodes` or constants (`Literals`). Let $\mathcal{U}$ be the set of all URIs, $\mathcal{B}$ the set of all `blank nodes`, and $\mathcal{L}$ the set of all Literals. In Linked Data each statement (or triple) is of the form subject-predicate-object where a subject corresponds to an entity (e.g, a book, a person, etc.), a predicate (or property) to a characteristic of an entity (e.g., genre of a book) and an object to the value of the predicate for a specific subject, e.g., in the following triple $\langle$*The Green Mile*, *hasAuthor*, *Stephen King*$\rangle$, *The Green Mile* is the subject, *hasAuthor* the predicate and *Stephen King* the object. Let S be the set of all `subjects`, P the set of all `properties`, and O the set of all `objects`. Formally, a *triple* is any element of $\mathcal{T} = S \times P \times O$, where $S = \mathcal{U} \cup \mathcal{B}$, $P = \mathcal{U}$ and $O = \mathcal{U} \cup \mathcal{L} \cup \mathcal{B}$, while an *RDF graph* (or dataset) is any finite subset of $\mathcal{T}$. The linking of datasets is realized by the existence of common URIs, referring to schema elements (defined through RDF Schema and OWL [2]), instances, as well as by equivalence relationships expressed via the `owl:sameAs` predicate.

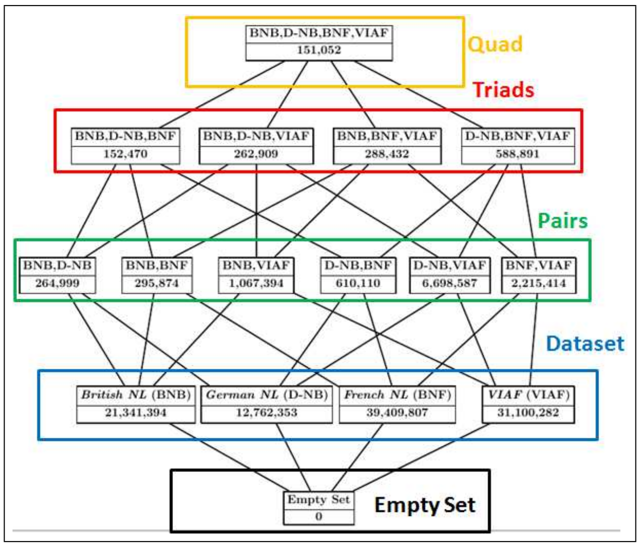


Fig. 2: Lattice of four Digital Library datasets (common Real world Objects)

LODsyndesis provides query services and measurements that are useful for several important tasks like (a) object co-reference, (b) dataset discovery, (c) visualization, and (d) connectivity assessment and monitoring [12]. Its public website also provides measurements that concern the commonalities of Linked Datasets, i.e., it provides the number of common real world objects between any set of datasets, that is the number of classes of equivalence of URIs after having computed the symmetric and transitive closure of the set of `owl:sameAs` relationships from all datasets. Such measurements can be visualized as lattices and Figure 2 shows a lattice for four digital libraries datasets (i.e., *British National Library*, *German National Library*, *French National Library* and *VIAF*). It is evident that these four datasets share 151,052 real world objects. All these equivalent URIs (e.g., among these four datasets) can be found by using the object co-reference service offered by *LODsyndesis*. For instance, one can find all the equivalent URIs among any set of sources, e.g., give me all the equivalent URIs among *British*, *German* and *French National Libraries*, or all the equivalent URIs for Jules Verne (e.g. `http://bnb.data.bl.uk/id/person/VerneJules1828-1905`).

Related Work. There are several proposals for using Linked Data for generating features. LiDDM [14] is a tool that retrieves data from Linked Data cloud by sending queries. For finding possible features the users can either construct their own queries or use an automatic SPARQL query builder that shows to the users all the possible predicates that can be used (from a specific SPARQL endpoint). It offers also operators for integrating and filtering data from two or more sources. The authors in [5] presented a modular framework for constructing semantic features from Linked Data, where the user specifies the SPARQL queries that should be used for generating the features. Another work that uses SPARQL queries is described in [13], where the user can submit queries which are combined with SPARQL aggregates (e.g., count). Comparing to our approach, the previous tools presuppose that the user is familiar with SPARQL, and they do not assist the user in discovering automatically datasets containing information for the same entities. The closest tool to our approach is FeGeLOD [15] which combines data from several datasets by traversing `owl:sameAs` paths and generates automatically six different categories of features. *RapidMiner Semantic Web Extension* tool [18] (which is the extension of FeGeLOD) supports the same features while it integrates the data that are derived from multiple sources. Instead we show the provenance of the data without integrating them, i.e., if a feature is provided by two or more sources, the user can decide which source to select for creating this feature. Moreover, we also discover datasets containing the same entities by exploiting *LODsyndesis* [12], where the class of equivalence for each entity has already been pre-computed for more than 300 datasets, whereas the aforementioned tool finds relevant data by traversing links on-the-fly. Finally, we also provide other kinds of features, such as degree of an entity, boolean features for each value of a predicate, as well as features for "sub-entities", i.e., entities correlated with the entities that one wants to classify (e.g., actors of a movie).

3 Linked Data-based Feature Creation Operators

Let E be the set of entities for which we want to generate features. Below we will show how we can derive a set of features $(f_1, ..., f_k)$ where each f_i is a feature and $f_i(e)$ denotes the value of that feature for an entity $e \in E$. Each $f_i(e)$ is actually derived by the data that are related to e. Specifically we have identified the following nine (9) frequently occurring *Linked Data-based Feature Creation Operators*, for short *FCO*s. In their definition, shown in Table 1, P denotes the set of properties, p, p_1 and p_2 are properties and hereafter $\mathcal{T}$ denotes the triples for the entities that are indexed by *LODsyndesis*.

Table 1: Feature Creation Operators

id	Operator defining f_i	Type	$f_i(e)$
1	p.exists	boolean	$f_i(e) = 1$ if (e, p, o) or $(o, p, e) \in \mathcal{T}$, otherwise $f_i(e) = 0$
2	p.value	num/categ	$f_i(e) = \{ v \mid (e, p, v) \in \mathcal{T}\}$
3	p.valuesCard	int	$f_i(e) = \|\{ v \mid (e, p, v) \in \mathcal{T}\}\|$
4	degree	double	$f_i(e) = \|\{(s, p, o) \in \mathcal{T} \mid s = e \text{ or } o = e\}\|$
5	p1.p2.exists	boolean	$f_i(e) = 1$ if $\exists$ $o2$ s.t. $\{(e, p1, o1), (o1, p2, o2)\} \subseteq \mathcal{T}$
6	p1.p2.count	int	$f_i(e) = \|\{ o2 \mid (e, p1, o1), (o1, p2, o2) \in \mathcal{T}\}\|$
7	p1.p2.value.maxFreq	num/categ	$f_i(e)$ = most frequent $o2$ in $\{ o2 \mid (e, p1, o1), (o1, p2, o2) \in \mathcal{T}\}$
8	average degree	double	$f_i(e) = \frac{\|triples(C)\|}{\|C\|}$ s.t. $C = \{ c \mid (e, p, c) \in \mathcal{T}\}$ and $triples(C) = \{(s, p, o) \in \mathcal{T} \mid s \in C \text{ or } o \in C\}$
9	p.values.AsFeatures	boolean	for each $v \in \{ v \mid (e, p, v) \in \mathcal{T}\}$ we get the feature $f_{iv}(e) = 1$ if (e, p, v) or $(v, p, e) \in \mathcal{T}$, otherwise $f_{iv}(e) = 0$

In our running example of Figure 1, FCO1 can be used for representing whether a book has been nominated for winning an award or not. FCO2 suits to properties that are functional (one-to-one), e.g. person's birth country, number of pages of the book, and its value can be numerical or categorical. FCO3 counts the values of a property, e.g. the number of genres of a book. FCO4 measures the number of distinct triples that involve e, in our running example the degree of the author of "The Green Mile" book is 3, while the degree of the author of "An episode of Sparrows" is 2. FCO5-FCO9 correspond to features related to "sub-entities" or "related" entities to e. Specifically, FCO5 corresponds to one characteristic of a "sub(related)-entity" of e, e.g. whether at least one actor of a movie has won an award in the past or not. FCO6 counts the distinct values of one characteristic of the "sub-entities", e.g. the total number of movies where the actors of a movie have played. FCO7 finds the most frequently occurring characteristic of these entities, e.g. the country where most of the actors of a movie were born. FCO8 measures the average number of distinct triples for a set of "sub-entities", e.g., the average number of triples for the actors of a movie. The last one, FCO9, does not create one feature but a set of features, e.g. one boolean feature for each genre that a book can possibly belong to. In our running example, we take all genres of both books and for each genre (e.g., novel) we create a distinct boolean feature (both books belong to the genre Novel, but only "The Green Mile" book belongs to the genre Crime). Generally, the operators FCO1-FCO4 and FCO9 concern a single entity (e.g., a book, a person, a country, etc.) while operators FCO5-FCO8 a set of entities (e.g., all actors of a movie). Consequently, for the "sub-entities" that are connected through a functional property (one-to-one) with the entities that we want to classify, operators FCO1-FCO4 and FCO9 are used instead of operators FCO5-FCO8. The user can explore direct or indirect "sub-entities", e.g., authors of a book, countries of authors of a book and so forth, for any formulated path, while the list of operators can be easily extended by adding more operators.

Additional Functionality of $\texttt{LODsyndesis}_{\mathcal{ML}}$.

Here we introduce some useful (for the user) metadata and restrictions for feature selection .

"Completeness" of a Property for a given set of entities. We compute the percentage of instances for which a given property exists, e.g., the percentage of books for which we have information about the number of their pages. If E'_p is the set of entities being subject or object of triples with predicate p, i.e. $E'_p = \{e \in E \mid (e\ p\ o) \text{ or } (o\ p\ e) \in \mathcal{T}\}$, then the percentage is given by $\frac{|E'_p|}{|E|}$.

Multiplicity & Range of a Property. Here we find the multiplicity of a specific property, i.e., whether it is a one-to-one or one-to-many relation. We define the set of one-to-one properties as $P_{1-1} = \{p \mid (e\ p\ o_i) \in \mathcal{T}$ and $\nexists\ (e\ p\ o_{ii}) \in \mathcal{T},\ o_i \neq o_{ii}, \forall\ e \in E\ \}$. The rest properties, i.e., one-to-many, are defined as $P_{Many} = P \setminus P_{1-1}$, while we denote as $range(p) \in \{String, Numeric, \mathcal{U}\}$ a property's range, i.e., whether it is a set of Strings, Numeric Values or URIs.

Restrictions derived from metadata. Table 2 shows the restrictions which are derived by taking into account the "completeness", the multiplicity and the range of a property. It is worth mentioning that the "completeness" of a property can also be exploited for discovering missing values for the entities. In addition, the users can define their own restrictions, e.g., they can exclude properties that belong to popular ontologies such as rdf, $rdfs$, $foaf$ and owl.

Table 2: Restrictions of features with respect to the characteristics of a property

Feature Operators	**Can be Applied for**
Boolean (FCO1, FCO5)	All properties having $\frac{\lvert E'_p \rvert}{\lvert E \rvert} < 1$
Boolean for each Value (FCO9)	All properties $p \in P_{Many}$, $range(p) \neq Numeric$
One-to-one Relationship (FCO2)	All properties $p \in P_{1-1}$
Count (FCO3, FCO6)	All properties $p \in P_{Many}$
Degree (FCO4, FCO8)	All properties having $range(p) = \mathcal{U}$

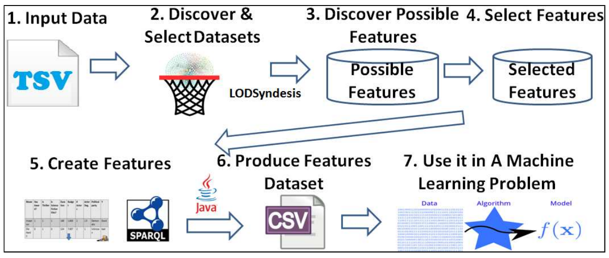


Fig. 3: Process of `LODsyndesis`$_{\mathcal{ML}}$

4 The Steps of the Proposed Approach

Here we describe the tool (research prototype) `LODsyndesis`$_{\mathcal{ML}}$ that we have designed and implemented. It is worth noting that `LODsyndesis`$_{\mathcal{ML}}$ discovers and creates features by exploiting Linked Data for any domain. Even a user that is not familiar with Semantic Web technologies and SPARQL can use it for creating features for feeding a Machine Learning problem. The process is shown in Figure 3 and is described in brief below. First, it takes as input a file containing a set of URIs that refer to particular entities, i.e., movies, books and so forth. In case of knowing the entities but not their URIs, one can exploit an entity identification tool like DBpedia Spotlight [11] and XLink [7] for detecting automatically a URI for a specific entity. Then, it connects to *LODsyndesis* for discovering automatically datasets containing information for the same entities and shows to the user the available datasets. Afterwards, it discovers and shows to the user possible features that can characterize the entities (or related "sub-entities") of the dataset and the user selects which features to create. The next step is to create the features and to produce the output dataset to be used in any ML problem. Below, we describe in more detail the whole process, while additional information and a demo can be found in `http://www.ics.forth.gr/isl/LODsyndesis`.

1. Input: The input of `LODsyndesis`$_{\mathcal{ML}}$ is a file in tab separated value (tsv) format containing URIs describing entities and possibly their class, e.g., URIs for a book and if each book is Popular or Non-Popular.

2. Discover Data by using LODsyndesis: `LODsyndesis`$_{\mathcal{ML}}$ reads the tsv file and connects to *LODsyndesis* [12] in order to discover (a) datasets containing information for the same entities and (b) the URIs for these entities for each dataset (the indexes of LODsyndesis have already pre-computed the closure of *owl:sameAs* relationships for 300 datasets). Then, the user selects the desired datasets. Concerning the running example of Figure 1, we observe that we found two different datasets containing information for the books of that example .

3. Discover Possible Features: `LODsyndesis`$_{\mathcal{ML}}$ sends SPARQL [17] queries for a sample of the aforementioned entities to the SPARQL endpoints of the selected datasets. Afterwards, a number of possible features and their provenance are discovered and returned to the user. Therefore, in this step we do not create any feature, we just discover possible features and we apply the restric-

tions described in Section 3. The result is a table where each row corresponds to a possible feature derived from a specific source while each column consists of a checkbox for a specific feature category. The order that the features appear in the rows is descending with respect to the "completeness" of each property. Particularly, when a property occurs for all the entities, it is placed first in the list, while those with the smallest number of occurrences are placed at the end of the list. Moreover, the user can view the metadata described in Section 3. Afterwards, the user can select the desired features (by taking into account their provenance) and can also explore features for (direct or indirect) "sub-entities" of any formulated path and create more features.
4. Feature Selection and 5. Feature Creation: The user selects the desired features and clicks on a button for initiating the dataset creation. Then, the tool sends SPARQL queries for creating the features. For each feature operators category, it sends $|E|$ in number SPARQL queries (one query per entity e for each operator). It is worth noting that for values that are neither numeric nor boolean, it performs a mapping for converting them to numeric. Concerning missing values, we just put a unique constant value. However, for improving datasets' quality, several transformations could be applied after this step, like those proposed in [3] for removing erroneous and inconsistent data or filling missing values. In this paper we do not focus on this task and the data used in the experiments have not been transformed or cleaned by using such techniques.
6. Production of Features' Dataset and 7. Exploitation of the Produced Dataset in a ML problem: The user is informed that the process is completed and that two csv files have been produced: one for the categorical and one for the continuous features. Then, the produced datasets can be given as an input for a ML problem (e.g., classification of books).

5 Evaluation

The datasets, which are used in our experiments (derived from [19]), contain the URIs of movies and books from *DBpedia* [10] and the corresponding classification value, i.e., *Popular* or *Non-Popular* according to the number of Facebook users' likes. We use 1,570 entities for Movies Dataset and 1,076 entities for Books Dataset. The initial datasets are loaded and then more data are discovered by using $\texttt{LODsyndesis}_{\mathcal{ML}}$ from the following sources: *British National Library* [16], *Wikidata* [20] and *DBpedia* [10]. In particular, we exploit $\texttt{LODsyndesis}_{\mathcal{ML}}$ for discovering, selecting and creating a number of different features for predicting the class of these entities. Afterwards, *MATLAB* [1] is used for performing a) a 5-fold cross validation for model selection and b) a comparison of a number of different models for measuring accuracy, which is defined as: $accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$ [21]. For each dataset, we repeat the 5-fold cross validation process 15 times for different sizes of the test set, i.e. 10%, 20%, & 30%. Each time a chi-square test of independence [23] is performed (for excluding variables that are independent of the class variable) for 4 different values of significance level (or threshold) a: 0.01, 0.05, 0.1, 1. For each value of threshold *a* we test 10 different models: (a) 2 *Naive Bayes* models (Empirical & Uniform), (b) 3 *Random Forest* models with 50 trees and different min leaf sizes: 1, 3 & 5, (c) 3 *K-Nearest Neighbours* models with K: 3, 5 &15, (d) a *linear SVM model* and (e) the *trivial* model. In each iteration the best model is obtained for the training set (by using cross validation). Finally, the accuracy of the best model is estimated on the test set.

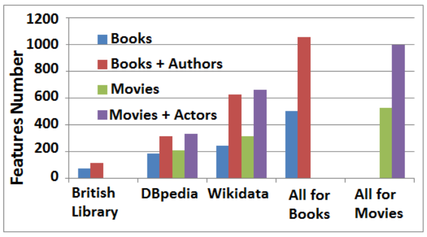


Fig. 4: Features Number Per Dataset For Books & Movies

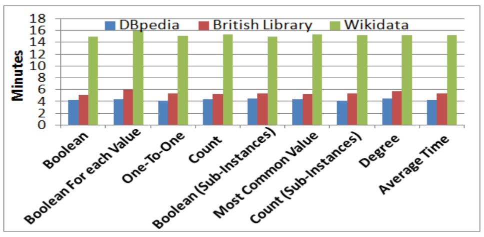
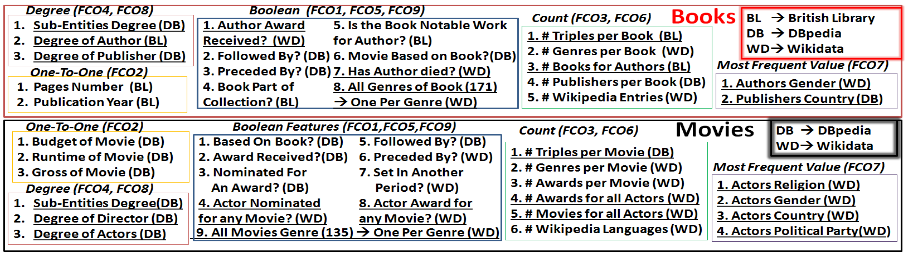


Fig. 5: Generation Time for each Feature Operators Category and Dataset


Degree (FCO4, FCO8)
1. Sub-Entities Degree (DB)
2. Degree of Author (BL)
3. Degree of Publisher (DB)

One-To-One (FCO2)
1. Pages Number (BL)
2. Publication Year (BL)

Boolean (FCO1, FCO5, FCO9)
1. Author Award Received? (WD)
2. Followed By? (DB)
3. Preceded By? (DB)
4. Book Part of Collection? (BL)
5. Is the Book Notable Work for Author? (BL)
6. Movie Based on Book?(DB)
7. Has Author died? (WD)
8. All Genres of Book (171) → One Per Genre (WD)

Count (FCO3, FCO6)
1. # Triples per Book (BL)
2. # Genres per Book (WD)
3. # Books for Authors (BL)
4. # Publishers per Book (DB)
5. # Wikipedia Entries (WD)

Books

BL → British Library
DB → DBpedia
WD→ Wikidata

Most Frequent Value (FCO7)
1. Authors Gender (WD)
2. Publishers Country (DB)

One-To-One (FCO2)
1. Budget of Movie (DB)
2. Runtime of Movie (DB)
3. Gross of Movie (DB)

Degree (FCO4, FCO8)
1. Sub-Entities Degree(DB)
2. Degree of Director (DB)
3. Degree of Actors (DB)

Boolean Features (FCO1,FCO5,FCO9)
1. Based On Book? (DB)
2. Award Received?(DB)
3. Nominated For An Award? (DB)
4. Actor Nominated for any Movie? (WD)
5. Followed By? (DB)
6. Preceded By? (WD)
7. Set In Another Period? (WD)
8. Actor Award for any Movie? (WD)
9. All Movies Genre (135) → One Per Genre (WD)

Count (FCO3, FCO6)
1. # Triples per Movie (DB)
2. # Genres per Movie (WD)
3. # Awards per Movie (WD)
4. # Awards for all Actors (WD)
5. # Movies for all Actors (WD)
6. # Wikipedia Languages (WD)

Movies

DB → DBpedia
WD→ Wikidata

Most Frequent Value (FCO7)
1. Actors Religion (WD)
2. Actors Gender (WD)
3. Actors Country (WD)
4. Actors Political Party(WD)

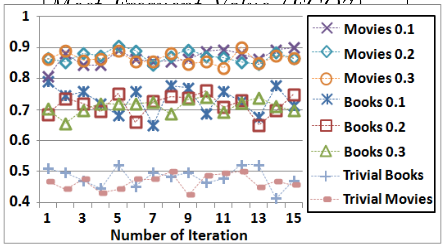
Fig. 6: Selected Features for Books & Movies with their Provenance

Creation of Features. In Figure 4 we can observe how the number of possible features increases when a) more datasets are added and b) features of "sub-entities" are created, i.e., approximately the possible features are doubled when we explore a "sub-entity" (e.g. the authors of a book). Moreover, Figure 5 shows the time for generating a feature for each different category (and each dataset) for 1,076 books. As we can see, the generation time depends highly on the dataset to which we send SPARQL queries, e.g., DBpedia's response time is much shorter than Wikidata's. Concerning the generation time of a specific feature operators category, the degree operators (FCO4, FCO8) and the boolean for each value of a predicate (FCO9) need more time to be generated while the remaining ones need approximately the same time on average for being generated. Finally, the execution time for retrieving the similar entities from *LODsyndesis* was 105 seconds.

Results for Movies Dataset. Figure 6 shows the selected features (and their category) for the dataset of movies (features belonging in the additional categories that we propose in this paper are underlined). In total we sent 39,250 queries and we created 159 features (147 categorical and 12 continuous). In Figure 7 we can see a plot with the accuracy of each test size (using the best model

Table 3: Accuracy for each Feature Operators Category (Movies & Books test size 0.2)

Feature Operators Category	**Average (Movies)**	**Max (Movies)**	**Average (Books)**	**Max (Books)**
All Features (FCO1-FCO9)	0.871	0.906	0.730	0.762
Continuous Features	0.861	0.896	0.709	0.739
New Features (FCO4-FCO9)	0.835	0.865	0.650	0.675
Existing Features (FCO1-FCO3)	0.827	0.855	0.694	0.716
Count (FCO3,FCO6)	0.830	0.862	0.706	0.709
1-1 Relationship (FCO2)	0.791	0.808	0.570	0.607
Categorical Features	0.760	0.818	0.673	0.694
Boolean (FCO1, FCO5, FCO9)	0.750	0.774	0.634	0.656
Degree (FCO4,FCO8)	0.741	0.780	0.608	0.627
Most Frequent Value (FCO7)	0.698	0.758	0.569	0.585



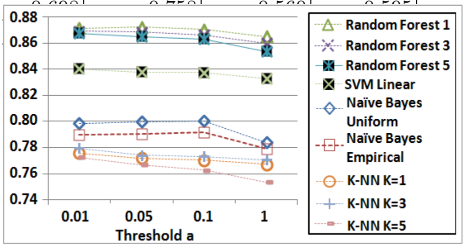


Fig. 7: Accuracy in Each Iteration & Test Size for Dataset Books and Movies

Fig. 8: Average Accuracy of Models in Cross Validation for Movies with Test Size 0.2

selected by the cross validation process) and we can observe that the accuracy is much higher comparing to a trivial case while the highest variation occurred for test size equal to 0.1. In all iterations, the best model was a *Random Forest* (with different parameters in many cases). Figure 8 shows the average accuracy for each model (and each threshold a) in the cross validation process for test size 0.2. We observe that *Random Forest* models achieved higher accuracy (mainly when min leaf size equals 1, i.e., Random Forest 1) comparing to the other models. The next ones with the highest accuracy is the *linear SVM*, followed by the two *Naive Bayes* models and finally the *K-NN* ones. However, all these models are better comparing to the trivial one whose accuracy is approximately 0.5. Table 3 shows the average and maximum accuracy for each different features' category in descending order with respect to their average accuracy. The continuous ones (mainly the count features, i.e., FCO3 and FCO6) seem to be the most predictive while all the categories achieved high accuracy comparing to a trivial case. Moreover, the average accuracy of features that other approaches also support (i.e., FCO1-FCO3) was 0.827 while for the additional features that we propose (e, FCO4-FCO9 in Table 3) the average accuracy was 0.835. By combining all the categories of features, the average accuracy was 0.871, which means that the additional features improved the accuracy in this particular problem.

Results for Books Dataset. Figure 6 shows the selected features (and their categories) for the books dataset. In total we sent 21,520 queries and we created 190 features (180 categorical and 10 continuous). In Figure 7 we can see a plot with the accuracy of each test size and we observe that the accuracy is much higher comparing to the trivial case while the highest variation occurred for test size equal to 0.1. In 42 iterations, the best model was a *Random Forest* (with different parameters in many cases) while in 3 cases the best model was a *linear*

SVM while the variations of *K-NN* algorithms were more effective than the *Naive Bayes* ones. As we can observe in Table 3, the combination of all features gave the maximum accuracy, while the continuous features, and especially the count features (FCO3, FCO6), were more predictive comparing to the remaining ones. Moreover, for the feature operators FCO1-FCO3 the average accuracy was 0.694 while for the feature operators FCO4-FCO9 was 0.65. However, by combining both types of features the average accuracy improved, i.e., 0.73, therefore the additional features improved the accuracy for books' dataset, too.

6 Concluding Remarks

We have shown how we can exploit the wealth of Linked Data and the ML machinery for improving the quality of automatic classification. We presented a tool, called `LODsyndesis`$_{\mathcal{ML}}$, which exploits Linked Data (and the related technologies) for discovering automatically features for any set of entities. We categorized the features and we detailed the process for producing them. For evaluating the benefits of our approach, we used two datasets and the results showed that the additional features did improve the accuracy of predictions, while the most effective model for both datasets was a *Random Forest* one. As future work, we plan to extend our tool for supporting more operators and transformations for improving the quality of the produced dataset. Furthermore, we plan to evaluate our tool in other tasks, e.g., completion of missing values, to support users' SPARQL endpoints and to connect our tool with SPARQL-LD [8] for incorporating also data stored in files (i.e. not hosted by SPARQL endpoints). Finally, it would be interesting to investigate techniques for automatic feature selection [22], such as those described in [9], where a novel machine learning-based feature selection method is used for predicting candidates features usefulness.

Acknowledgements. This work has received funding from the European Union's Horizon 2020 Research and Innovation programme under the BlueBRIDGE project (Grant agreement No: 675680).

References

1. MATLAB - MathWorks. `https://www.mathworks.com/products/matlab.html`.
2. G. Antoniou and F. Van Harmelen. *A semantic web primer.* MIT press, 2004.
3. S. Bischof, C. Martin, A. Polleres, and P. Schneider. Collecting, integrating, enriching and republishing open city data as linked data. In *International Semantic Web Conference*, pages 57–75. Springer, 2015.
4. C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.
5. W. Cheng, G. Kasneci, T. Graepel, D. Stern, and R. Herbrich. Automated feature generation from structured knowledge. In *CIKM*, pages 1395–1404. ACM, 2011.
6. I. Ermilov, J. Lehmann, M. Martin, and S. Auer. Lodstats: The data web census dataset. In *International Semantic Web Conference*, pages 38–46. Springer, 2016.

7. P. Fafalios, M. Baritakis, and Y. Tzitzikas. Configuring named entity extraction through real-time exploitation of linked data. In *WIMS14*, page 10. ACM, 2014.
8. P. Fafalios, T. Yannakis, and Y. Tzitzikas. Querying the web of data with sparql-ld. In *International Conference on TPDL*, pages 175–187. Springer, 2016.
9. G. Katz, E. C. R. Shin, and D. Song. Explorekit: Automatic feature generation and selection. In *ICDM, 2016*, pages 979–984. IEEE, 2016.
10. J. Lehmann, R. Isele, Jakob, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.
11. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *I-SEMANTICS*, pages 1–8. ACM, 2011.
12. M. Mountantonakis and Y. Tzitzikas. On measuring the lattice of commonalities among several linked datasets. *Proceedings of the VLDB Endowment*, 9(12), 2016.
13. J. Mynarz and V. Svátek. Towards a benchmark for lod-enhanced knowledge discovery from structured data. In *KNOW@ LOD*, pages 41–48, 2013.
14. V. Narasimha, P. Kappara, R. Ichise, and O. Vyas. Liddm: A data mining system for linked data. In *Workshop on LDOW*, volume 813, 2011.
15. H. Paulheim and J. Fümkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of WIMS 2012*, page 31. ACM, 2012.
16. M. Pennock and M. Day. Managing and preserving digital collections at the british library. *Managing Digital Cultural Objects: Analysis, discovery and retrieval*, page 111, 2016.
17. E. Prud' Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.
18. P. Ristoski, C. Bizer, and H. Paulheim. Mining the web of linked data with rapidminer. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:142–151, 2015.
19. P. Ristoski, G. K. D. de Vries, and H. Paulheim. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International Semantic Web Conference*, pages 186–194. Springer, 2016.
20. D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
21. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2016.
22. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.
23. M. F. Zibran. Chi-squared test of independence. *Department of Computer Science, University of Calgary, Alberta, Canada*, 2007.