Statistical Knowledge Patterns: Identifying Synonymous Relations in Large Linked Datasets

Ziqi Zhang[1], Anna Lisa Gentile[1], Eva Blomqvist[2], Isabelle Augenstein[1], and Fabio Ciravegna[1]

[1] Department of Computer Science, University of Sheffield, UK
[2] Department of Computer and Information Science, Linköping University, Sweden
{z.zhang,a.l.gentile,i.augenstein,f.ciravegna}@dcs.shef.ac.uk, eva.blomqvist@liu.se

Abstract. The Web of Data is a rich common resource with billions of triples available in thousands of datasets and individual Web documents created by both expert and non-expert ontologists. A common problem is the imprecision in the use of vocabularies: annotators can misunderstand the semantics of a class or property or may not be able to find the right objects to annotate with. This decreases the quality of data and may eventually hamper its usability over large scale. This paper describes Statistical Knowledge Patterns (SKP) as a means to address this issue. SKPs encapsulate key information about ontology classes, including synonymous properties in (and across) datasets, and are automatically generated based on statistical data analysis. SKPs can be effectively used to automatically normalise data, and hence increase recall in querying. Both pattern extraction and pattern usage are completely automated. The main benefits of SKPs are that: (1) their structure allows for both accurate query expansion and restriction; (2) they are context dependent, hence they describe the usage and meaning of properties in the context of a particular class; and (3) they can be generated offline, hence the equivalence among relations can be used efficiently at run time.

1 Introduction

The Web of Data is a rich common resource with billions of triples available in thousands of datasets and Web documents (including RDFa and microdata annotations) created by a growing number of people, including non expert ontologists (e.g. Web managers generating schema.org microdata annotations). This, however, brings the risk of imprecision in the use of vocabularies (schemas and ontologies), including their systematic misuse. Annotators can misunderstand the semantics of a concept or relation or may not be able to find the right classes and properties to annotate with[1]. This decreases the quality of data and may eventually hamper its usability over large scale. The problem gets even more complex when using several datasets together, which may refer to the same classes, e.g. DBpedia types, but use different sets of properties.

[1] Throughout this paper we use the terms "concept" and "class" interchangeably, to mean a concept defined in a vocabulary (i.e., ontology). Similarly, we use the term "relation" as a synonym for "property".

In this paper we focus on a problem found in numerous datasets: the use of classes and properties which are alien to the reference vocabulary provided for the dataset, but that may be equivalent to existing classes or properties in the reference vocabulary, or may be used to extend that vocabulary at data creation time. This issue generates low recall when querying datasets, as the user must guess which properties are actually used in the dataset as opposed to the ones formally defined in the vocabulary. A similar issue also arises when attempting to query interlinked datasets, but only being aware of the vocabulary used in one of them. The linked datasets may use different properties, but still contain overlapping and complementary data. Without exploring property usage in all those datasets, queries may miss relevant parts of the data.

We propose the definition and use of Statistical Knowledge Patterns (SKP) as a mean to address these issues. An SKP is class-specific and encapsulates key information about an ontology class, including synonymous properties used within (and potentially between) datasets. SKPs aim at reducing the complexity of understanding and querying data, by reducing the variety of properties to only include the core properties of the main SKP class, and their characteristics. We propose an unsupervised approach to generate SKPs based on statistical data analysis, and introduce a measure of "synonymity" of two properties of a class, which is used to cluster synonymous properties. Effectively, an SKP addresses the vocabulary heterogeneity of classes based on their usage data, within datasets, or between datasets that are linked through that class. One possible usage of an SKP is query expansion when querying the data underlying the SKP (shown in Sect. 5).

Both pattern extraction and pattern usage are completely automated. The main benefits of SKPs are: that (1) their structure allows for both accurate query expansion and restriction; (2) they are context dependent, hence they can describe the usage and meaning of properties in the context of a particular class and even within a specific dataset (or group of datasets), hence also accounting for synonymity that hold only in specific repositories, domains or communities; and (3) they can be generated offline, hence the synonymity among properties can be used efficiently at run time.

The paper is organised as follows: Sect. 2 describes related work; Sect. 3 introduces the SKP generation method, and Sect. 4 presents the synonymity measure and property clustering in detail; Sect. 5 describes our experiments and discusses the evaluation of our approach; Sect. 6 concludes the paper and discusses future work.

2 Related Work

Knowledge Patterns (KP) have been defined as general templates or structures used to organise knowledge [8]. In the Semantic Web scenario they are used both for constructing ontologies [3,7,14] and for using and exploring them [2,10,11,13].

In the area of ontology engineering several kinds of patterns [7] have been used. On such type is the *Content Ontology Design Patterns* (CODPs), which are small, reusable pieces of ontologies that consist of just a few classes. They represent core concepts in an ontology and are either extracted or re-engineered from ontologies or other data structures. CODPs are similar to SKPs in the way that they also represent concepts with their most distinguishing characteristics. Unlike SKPs however, they have to be created manually or semi-automatically, and since they are abstract patterns intended for being used as "templates" in ontology engineering they usually lack any direct connection to data and cannot directly (without manual specialisation) be used for querying Linked Data. Since CODPs represent an abstract top-down view, they additionally do not consider aspects such as diversity and synonymy among properties, which is one of the things we focus on in this paper.

The approach closest to ours is the generation of *Encyclopedic Knowledge Patterns* (EKPs) [11], which have been built mainly for usage in exploratory search [10]. The EKP generation process exploits statistics of links from Wikipedia to select which classes are the most representative for describing each concept. The assumption is that if entities of a class A frequently link to entities of class B, then class B is an important descriptor for class A. This information is formalised as small OWL ontologies (the EKPs), each having one main class and relations to other (significantly frequent) classes. The main purpose of EKPs is to filter out irrelevant data when presenting DBpedia entities, while the ability to query for data is not a primary concern. Hence, EKPs mainly contain abstractions of properties, such as "linksToClassB", which expresses the fact that instances of class A commonly link to instances of class B (links which could in many cases in turn be represented by DBpedia properties, but not necessarily). This is however not sufficient for our case, since our main goal is to use SKPs to query actual data. Hence, we propose an extension of EKPs, which also include a sufficient coverage of actual properties of the datasets. Basse et al. [2] also exploit statistics from a specific dataset to produce topic frames of that dataset. In contrast to Nuzzolese et al. [11] they don't produce a pattern for each class but rather generate clusters of classes (up to 15 classes each) that reflect main topics of the dataset, which is again not sufficient for our goal. Presutti et al. [13] explore the challenges of capturing KPs in a scenario where explicit knowledge of datasets is neither sufficient nor straight-forward. They propose a dataset analysis approach to capture KPs and support datasets querying. Our SKPs expand on this work as not only do we capture direct statistical information from the underlying datasets, but also further characterise relevant properties with additional features (e.g. synonymous properties and range axioms), which we show to be beneficial for querying datasets. As well as exploratory purposes, another common usage of KPs is within Query Expansion (QE), and Question Answering (QA) in general. Typical approaches [5] use the lexicalizations of concepts to map natural language to URIs (with NLP techniques) but they may fail to capture synonymous relations with completely different lexicalizations.

A core component of our method of creating SKPs is measuring synonymity between ontology properties. This is related to the work on linking ontological resources in general [6,9,15]. A large amount of work in this area addresses linking ontology classes and data instances, linking properties, however is insufficiently addressed. Typical approaches employ similarity metrics such as string edit distance and semantic similarity measures. However, string similarity fails to identify equivalent relations if their lexicalisations are wholly distinct, which is very common in Linked Data. Semantic similarity often depends on taxonomic structures in existing ontologies [4]. Unfortunately, many relations which are used in Linked datasets are invented arbitrarily or originated from rudimentary ontologies [12]. Our previous research [1] shows that a bottom-up approach that uses Linked Data statistics offers effective solution to measuring similarity. Therefore in this work we introduce a data-driven synonymity measure for properties on Linked Data and we use it in the construction of SKPs.

3 SKP Construction Overview

A Statistical Knowledge Pattern (SKP) is an ontological view over a class (defined in a reference ontology), and captures and summarises the usage of that class (hereafter called the *main class* of the SKP) in data. An SKP is represented and stored as an OWL ontology. The term "statistical" refers to that the pattern is constructed based on statistical measures on data. Each SKP contains: (1) properties and axioms involving the main class derived from a reference ontology; (2) properties and axioms involving the main class that are not expressed in the reference ontology, but which can be induced from statistical measures on statements published as Linked Data.

The generation of SKPs is mainly characterized by the identification (based on data triples) and selection of (1) synonymous (i.e. interchangeable) properties; (2) ranges for properties that have no prior range in the reference ontology. Not all properties (or clusters of synonymous properties) are stored in the final SKP. To decide which ones are representative of the SKP main class, their *relevance* is measured based on the frequency of usage in available data. The information encoded in the SKP is specific to the main class, i.e., it does not show a general interpretation of the involved properties but rather the specific way they are used with the main class. For example, the same property may be present in several SKPs, but with distinct range axioms and as part of separate property clusters, depending on how it is used with the respective main class of each SKP. A concrete example is the property *dbp:lakeName*[2] which is synonymous to *foaf:name*, but only for the class *dbo:Lake*.

At the moment we only consider the properties that are used with instances of the main class in the subject position, as part of the characterization of that class, which in our experience is usually the case for Linked Data, e.g., the way the DBpedia Ontology is structured. One may also consider properties with the opposite "direction", i.e., instances of the main class as objects (for example, *isLocationOf* instead of *hasLocation*), however, we do not include this at the moment (acknowledging the risk of loosing some fraction of the data) since we have no method to determine what the main focus of a triple is, we therefore make the simple assumption that being a subject of a triple means that this is the entity the triple is describing. The SKP generation is fully automated, whereby SKPs can be re-generated as soon as data change, without manual effort. At the same time SKPs are used as stored resources hence increasing usage efficiency.

Relation to EKP Extraction. Since SKPs are an extension of EKPs [11], if an EKP already exists it can be used as an abstract frame for the concrete properties and axioms

[2] Prefixes used are $dbo :< http : //dbpedia.org/ontology/ >$,
$dbp :< http : //dbpedia.org/property/ >$,
$skos :< http : //www.w3.org/2004/02/skos/core\# >$,
$foaf :< http : //xmlns.com/foaf/0.1/ >$,
$rdfs :< http : //www.w3.org/2000/01/rdf - schema\# >$

that are added through our SKP generation method. In particular, the abstract properties introduced by EKPs (i.e., "links to class X") can be used to group properties with overlapping range axioms, to give the SKP a more intuitive structure and improve human understandability of the pattern. However, we do not restrict an SKP to being an EKP, i.e., a set of "paths" in DBpedia (c.f. Def. 2 in [11]), but rather the SKP notion is both independent of what reference ontology and datasets are used, and the resulting SKPs may include any OWL axioms that can be statistically induced from data. Our method for extracting the patterns also differs significantly from the EKP extraction method in [11], i.e., we use triple data from the dataset in focus (DBpedia is used as an example for the experiments) while the EKPs are extracted from a wikilink dataset, which means that we operate on triples using distinct named properties rather than just "links". With respect to comparing the methods, we are not using the EKP notion of "path", c.f. [11] where a path is defined as a triple with the first element being the subject type, the second being the property *dbo:wikiPageWikiLink* and the third being the object type, i.e., the first and last element of a path being classes. In our case we are not initially interested in the object types, but rather the object instances (resources) themselves, hence, the path abstraction is replaced by actual RDF triples, with concrete instances (resources) as subjects and objects, but selected based on the fact that the subject type is the main class of the SKP. This means that, for instance, we also include datatype properties, and triples where the object type is missing, as opposed to the EKP extraction method. Similarly to [11], however, we only use single triples, not chains of triples. Paths are selected for EKP inclusion based on their so called "path popularity", i.e., a measure on how large fraction of the individuals of the main class have links to an individual of a certain type (c.f. indicators in [11]), putting the focus on the object types (classes). SKPs, on the contrary, put the main focus on the properties, and apply a frequency measure on property usage, i.e. subject-object pairs of RDF triples using that property (c.f. experiments on different such measures and thresholds in Sect. 5.1), where subjects are all instances of the SKP main class, for setting an inclusion threshold.

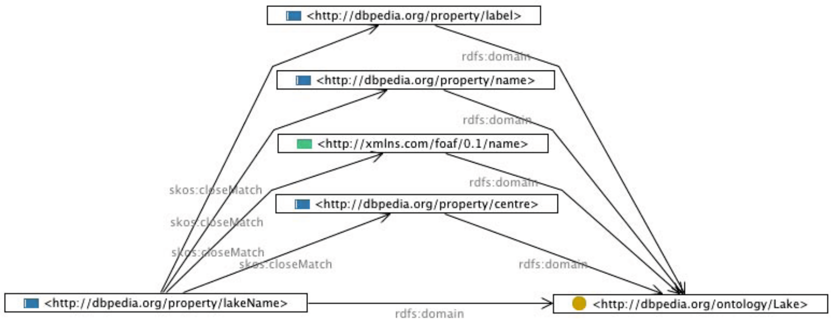
<http://dbpedia.org/property/label>
rdfs:domain
<http://dbpedia.org/property/name>
rdfs:domain
<http://xmlns.com/foaf/0.1/name>
rdfs:domain
skos:closeMatch
<http://dbpedia.org/property/centre>
skos:closeMatch
skos:closeMatch
skos:closeMatch
rdfs:domain
<http://dbpedia.org/property/lakeName>
<http://dbpedia.org/ontology/Lake>
rdfs:domain

Fig. 1. Extract from the SKP for the DBpedia Ontology class Lake (in TopBraid Composer's RDF graph notation)

SKP Example. As an example, we take the SKP generated for the DBpedia Ontology class *dbo:Lake*[3]. A property from the reference ontology, which is sufficiently frequent in actual data to be included in the SKP, is *dbo:shoreLength*, hence, it is included in the SKP. Additionally, it has been found to be synonymous to the property *dbp:shore*, hence a *skos:closeMatch* assertion is added for the two properties. In this case r_i = *dbo:shoreLength* is a reference property having a set of "synonymous properties" SR_i induced from data (in this case consisting of only one member). Figure 1 illustrates an extract from the same SKP showing another example; the property *dbp:lakeName* has been clustered with the properties *dbp:name*, *dbp:label*, *foaf:name*, and *dbp:centre*, which is expressed via the *skos:closeMatch* assertion. The selection of synonymous properties is obtained via a *synonymity* measure, described in Section 4, but before taking a deeper look at the measure we provide an overview of the steps of the two main phases of the overall method; detecting synonymity, and selecting properties.

Synonymity of Properties. To create an SKP we identify the properties used for the SKP main class based on data and measure their synonymity. We propose a novel synonymity measure of properties or relations to be detailed in Section 4. The overall process is:

1. Query the dataset for all the instances (IND) of the main class; query the dataset for all triples having any $i \in IND$ in subject position (IND_{subj}) and collect the types (through *rdf:type* or a datatype) of the objects of all those triples.
2. For each property used in IND_{subj}, collect the subset of IND having the property as predicate, IND_{prop} - the *subject-object* pairs of this set represents the characteristics of that property, given the main class at hand.
3. Do a pairwise comparison of all subject-object pairs in IND_{prop} for all the properties and calculate a *synonymity* score for each pair.
4. Use the *synonymity* scores (representing evidence of properties being interchangeable) to cluster properties representing the same semantic relation.

Selection of Properties. The aim of the above process is to discover for each specific main class clusters of properties with the same meaning. In practice, certain number of properties are found to be noise or non-representative. Thus we further refine the set of selected properties for each SKP as follows:

1. Calculate the frequencies of properties used in data, i.e. counting distinct objects in IND_{prop}. For clusters, treat the cluster as if it was a single property hence add the frequency counts of the constituent properties.
2. Use a cutoff threshold T to filter out unfrequent properties (or clusters). Add those above the threshold to the SKP, including information about their appropriate property type (e.g. *owl:DatatypeProperty* or *owl:ObjectProperty*), with their original namespace intact. We experiment with several approaches for setting the threshold T, and report on this in Section 5.
3. For each member of a property cluster that is added to the SKP, add a *skos:closeMatch* relation between the cluster members.

[3] `http://ontologydesignpatterns.org/skp/Lake.owl`

4. For each property, create the set of possible ranges. Construct a range axiom including each range class that is given to the property in the reference ontology (if present), and if no range axiom is present, construct a range axiom as the union of each range class that can be identified in data (frequent object types of the triples).
5. Add *rdfs:subPropertyOf* axioms for those properties where the ranges match some abstract EKP property (i.e., the "links to class X" abstract properties).
6. Store the SKP as an OWL2 file.

4 Synonymity Measure and Property Clustering

We consider synonymity to be symmetric and argue that the synonymity for each distinct pair of properties or relations depends on three components: triple overlap, cardinality ratio and clustering.

Triple Overlap. evaluates the degree of overlap in terms of the usage of properties in triples. Let p be a property and r_p be the set of triples containing p as predicate, and let $SO(p)$ be the collection of subject-object pairs from r_p and SO_{int} the intersection

$$SO_{int}(p, p') = SO(r_p) \cap SO(r_{p'}) \tag{1}$$

then the triple overlap $TO(p, p')$ is calculated as

$$MAX\{\frac{|SO_{int}(r_p, r_{p'})|}{|r_p|}, \frac{|SO_{int}(r_p, r_{p'})|}{|r_{p'}|}\} \tag{2}$$

Intuitively, if two properties p and p' have a large overlap of subject-object pairs in their data instances, they are likely to have identical meaning. The MAX function minimises the impact of infrequently used, but still synonymous relations (i.e., where the overlap covers most triples of an infrequently used relation but only a very small proportion of a much more frequently used).

Subject Agreement. While triple overlap looks at the data in general, subject agreement looks at the overlap of subjects of two relations, and the degree to which these subjects have overlapping objects. Let $S(p)$ return the set of subjects of relation p, and $O(p|s)$ returns the set of objects of relation p whose subjects are s, i.e.:

$$O(p|s) = O(r_p|s) = \{t_o | t_p = p, t_s = s\} \tag{3}$$

we define:

$$S_{int}(p, p') = S(r_p) \cap S(r_{p'}) \tag{4}$$

$$\alpha = \frac{\sum_{S \in S_{int}(p,p')} \begin{cases} 1 & \text{if } |O(p|s) \cap O(p'|s)| > 0 \\ 0 & \text{otherwise} \end{cases}}{|S_{int(p,p')}|} \tag{5}$$

$$\beta = \sqrt{|S_{int}(p, p')| / |S(p) \cup S(p')|} \tag{6}$$

then the agreement $AG(p, p')$ is

$$AG(p, p') = \alpha \cdot \beta \tag{7}$$

In Equation 7, α counts the number of overlapping subjects whose objects have at least one overlap. The higher the value of α, the more the two relations agree in terms of their shared subjects. We do not consider the absolute value of overlap because both p and p' can be 1:many relations and a low overlap value could mean that one is densely populated while the other is not, which does not necessarily mean they do not agree. β evaluates the degree to which two relations share the same set of subjects. The agreement $AG(p, p')$ balances the two factors by taking their product. As a result, relations that have high level of agreement will have more subjects in common (β), and a large proportion of shared subjects who also have shared objects (α).

Cardinality Ratio is a ratio between cardinality of the two relations. Cardinality of a relation $CD(p)$ is calculated based on data:

$$CD(p) = \frac{|r_p|}{|S(r_p)|} \tag{8}$$

and the cardinality ratio is calculated as

$$CDR(p, p') = \frac{MIN\{CD(p), CD(p')\}}{MAX\{CD(p), CD(p')\}} \tag{9}$$

On a sufficiently large sample, the derived cardinality value should be close to the conceptually true value and two equivalent relations should also have the same cardinality. The final synonymity measure integrates all the three components to return a value in [0, 2]:

$$E(p, p') = \frac{TO(p, p') + AG(p, p')}{CDR(p, p')} \tag{10}$$

Clustering. We apply the measure to every pair of relations of a concept of interest, and keep those with a non-zero synonymity score. The goal of clustering is to create groups of synonymous relations based on the pair-wise synonymity scores. We use a simple rule-based agglomerative clustering algorithm that uses a number of thresholds. First, we build initial clusters based on all property pairs. We rank all property pairs by their synonymity score, then we keep a pair as an initial cluster if (i) its score and (ii) the number of triples covered by each property are above a certain threshold, T_{minSyn} and T_{minTP} respectively (i.e., we create a cluster containing p and p' if $E(p, p') > T_{minSyn}$ and $|r_p| > T_{minTP}$ and $|r_{p'}| > T_{minTP}$). Next, to merge clusters, given an existing cluster C and a new pair (p, p') where either $p \in C$ or $p' \in C$, the other property is added to C if $E(p, p')$ is close to the average of all equivalence scores of connected pairs in C. The closeness is determined by another threshold $T_{minSynRel}$, which is a fractional number. Thus if the average of the equivalence scores of all connected pairs in C is 0.7, the new pair (p, p') is merged with C if $E(p, p') > 0.7 * T_{minSynRel}$. This preserves the strong connectivity in a cluster. This is repeated until no further merge action is taken.

5 Evaluation

To evaluate the use of SKPs we have chosen to focus on two main aspects: (1) the extent to which the SKPs describe and characterise the underlying data, and give access to that data; (2) the extent to which the identification of "synonymous" properties improve on retrieval coverage, without introducing erroneous data in the result. Thus we perform two sets of experiments for different purposes.

We used DBpedia as the underlying semantic data repository in this experiment and we query the live DBpedia SPARQL endpoint to retrieve data, and use the DBpedia Ontology as reference ontology[4]. In this evaluation, we created SKPs for 34 DBpedia classes[5], which are exactly the classes that can be extracted from the queries of the QALD1 question answering dataset[6]. These classes were selected for the experiments since we intend to in the future apply the SKPs to query expansion and make use of the QALD1 dataset as a benchmark. The SKP construction method is not restricted to any particular ontology or datasets, but can be applied to any reference ontology-dataset pair for which they are needed. In the experiments we use $T_{minSyn} = 0.1$, $T_{minTP} = 0.01\%$ and $T_{minSynRel} = 0.6$ as thresholds.

5.1 SKP Observation

SKPs aim at reducing the variety of properties to only include the core properties of the main SKP class, however, to be useful in practice, such a reduced representation should still allow for accessing as large part of the underlying data as possible. We have measured two aspects of each SKP, (1) the absolute number of properties included in the SKP and the fraction of the total number of distinct properties of the main class that this set represents, and (2) the fraction of the total number of triples (where the subject is an instance of the main class) that the properties included in the SKP allows to cover. The ideal situation would be that a low absolute number (1) of properties would still render an almost perfect coverage of triples (2).

However, first we need to generate a set of SKPs to assess, and for this we need to set an appropriate property selection threshold. Three sets of experiments have therefore been performed, each applying a different method for setting the threshold on what properties to include in the SKP. For each such threshold, the parameters of the threshold have been varied, so as to evaluate (a) the amount of properties included, and (b) the amount of triples covered, in each case. This leads us to conclude both which method for setting the threshold that seems to perform best over the SKP test set, and also gives us an evaluation of how well the SKPs using that threshold perform on criteria (1) and (2).

A naive approach would be to set an *absolute threshold* on the count of triples using a certain property, i.e. including all properties with more than a certain number of

[4] `http://dbpedia.org/sparql`, ontology: `http://dbpedia.org/ontology`

[5] The preliminary SKPs used in the evaluation, including skos:closeMatch statements, can be found at `http://ontologydesignpatterns.org/skp/SKPs130510.zip`

[6] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald1/evaluation/dbpedia-test.xml`

triples (subject type being the SKP main class). Figure 2 shows the performance of this approach, illustrating both the best (maximum triple coverage and minimum fraction of included properties), worst (minimum triple coverage and maximum fraction of included properties) and the average performance of each criteria (average triple coverage and average fraction of included properties), over the SKP set. At an absolute threshold of 20 triples we have a triple coverage between 79 and 98%, at an included property fraction of between 14 and 45%.

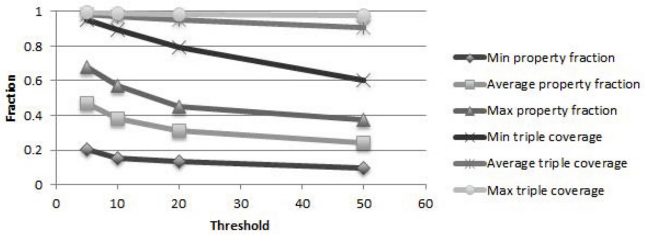

Fig. 2. Performance of the absolute thresholds (when set to 5, 10, 20 and 50 triples)

A more elaborate threshold would consider the properties that represent at least *a certain fraction* of the total number of triples (subject type being the SKP main class). Although this may sound reasonable at first glance, Figure 3 shows that this threshold actually performs worse than the absolute threshold. There is actually no value of the fraction that we could find which both guarantees us to get any properties at all, for all the SKPs in our set, but with no SKP on the other hand including the complete set of properties (even those with very low frequency). As in the previous figure, Figure 3 shows the best, worst, and average performance over the SKP set. While we can get the worst case property fraction included to drop to about 75% (at the 1% threshold) then the coverage of triples has already dropped to below 70%.

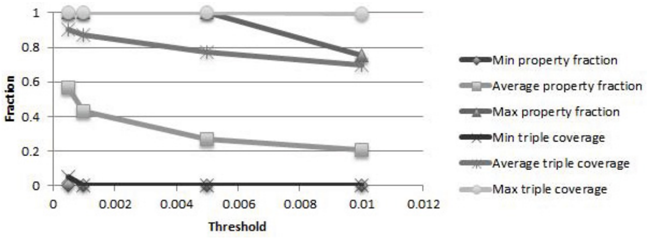

Fig. 3. Performance of the triple fraction threshold (when set to between 0.05% and 1% of triples)

Finally, we explore a *normalised threshold* that turns out to perform best. We first calculate the average number of triples per property (where the property set still constitute all triples where the subject type is the main class), and then set a threshold as a fraction of that average. With this threshold we, hence, both take into account the size of the triple set (as in the previous method), but also the number of properties used for instances of the main class. In Figure 4 the performance of this threshold is shown. Note that up until around 0.6 (meaning that properties are included if the size of their triple set constitute at least 60% of the average size of a triple set for any property used for the main class) the triple coverage stays really high, i.e. in the worst case still above 88%, with a worst case fraction of included triples of 38%. Hence, this indicates that using this threshold, we can probably discard at least 62% of all properties (usually more), for *any* class in the DBpedia ontology, without reducing the amount of triples we can still access to below 88% (on average we can even access 94% of the triples, or in the best case 97%).

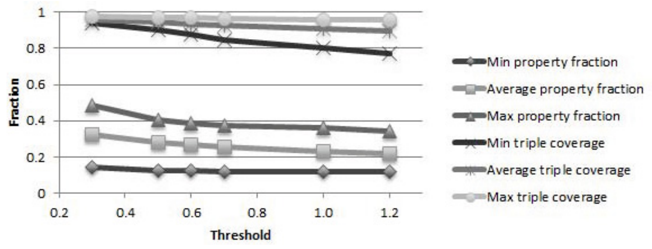

Fig. 4. Performance of the final threshold (when set to between 0.3% and 1.2% of the average number of triples per property)

Although we expect this to be true also for other datasets, and for generating SKPs over several interlinked datasets, we can of course not guarantee that this is the best threshold in all cases. However, this experiment also shows how one can (completely automatically) test a threshold calculation method, to select the best one. Hence, when generating SKPs for other datasets, it is recommended to rerun these experiments, to find the “optimal” threshold for those datasets.

Nevertheless, the 0.6 threshold has been used for the SKPs generated for the rest of our experiments, and Table 1 shows the characteristics of the SKPs generated using this threshold. The name of the SKP is equal to the name of the main class, as defined in the DBpedia Ontology (our reference ontology). First we present some statistics on the main class itself, i.e. the number of instances in DBpedia version 3.8, the total number of triples with those instances in the subject position, and the total number of distinct properties that can be found in that set of triples. Next, we present some statistics on the SKP generated for that main class, i.e. the total number of properties included in the SKP, the fraction of the total number of properties the select ones represent (criteria (1) mentioned previously), and the fraction of the included properties that are not defined in the reference ontology. The latter aspect gives a first indication of how much added information about the main class our SKPs contain, compared to the DBpedia Ontology itself. On average, 78% of the properties in our SKPs are not defined in the DBpedia ontology. Finally, we present the ability of the SKP to cover the actual triples in the DBpedia dataset, for the main class, through the number of triples covered and the fraction of the total number of triples that the selected ones represent (criteria (2)).

5.2 Using SKPs for Query Expansion

To evaluate the benefit of synonymous properties provided by the SKP we create a query expansion experiment to study (i) the increase in recall (added data) with (ii) the decrease in precision (introduced errors) by using the added properties. For each SKP we consider the set R_{ont} of all properties defined by a reference ontology - DBPedia in this case, and we generate a query for each $r_i \in R_{ont}$ such as "SELECT DISTINCT ?s ?o WHERE {?s a Main_Concept_of_SKP . ?s r_i ?o .}". The set of values of $?o$ returned from each query is considered the *baseline value set* for the property r_i, denoted by V_{r_i}. Then let SR_i be the set of synonymous properties to r_i. For each $r_i \in R_{ont}$ we generate a query for each $sr_i^j \in SR_i$ in the same form as before to retrieve the set of values $V_{sr_i^j}$. Then the total *expanded value set* (denoted by EV_{r_i}) is the union of all $V_{sr_i^j}$ for each sr_i^j. To determine the accuracy of values in EV_{r_i}, we manually analyse each property in each SR_i and annotate the property as either a *correct synonymous property* for r_i or not. Annotating property synonymity involved four computer scientists, and the Inter-

Table 1. Characteristics of the generated SKPs

SKP name (main class)	no. of instances	no. of triples	no. of properties	no. of properties in SKP	fract. of properties included	fract. of non-ontology properties	no. of triples covered	fract. of triples covered
Actor	2912	19833	246	88	0.36	0.77	18847	0.95
AdministrativeRegion	28229	20721	1235	436	0.35	0.90	18432	0.89
Agent	956476	18395	877	232	0.26	0.69	16197	0.88
Architect	1348	19540	169	38	0.22	0.74	18786	0.96
Bridge	2775	23446	351	94	0.27	0.68	21711	0.93
BritishRoyalty	6563	17918	250	66	0.26	0.79	17037	0.95
Cave	238	6100	105	31	0.30	0.87	5677	0.93
City	24423	27064	876	238	0.32	0.89	24210	0.89
Company	44516	17010	352	102	0.29	0.63	16037	0.94
Country	2710	22530	666	194	0.29	0.87	21286	0.94
Currency	333	6807	173	64	0.37	0.97	6072	0.89
Eukaryote	199085	15983	255	78	0.31	0.85	14853	0.93
FictionalCharacter	9878	19441	377	120	0.32	0.85	18352	0.94
Film	88503	17674	153	48	0.32	0.65	16847	0.95
Lake	10294	19915	412	51	0.12	0.63	19140	0.96
Language	6860	19357	149	35	0.23	0.89	17942	0.93
Magazine	3388	18986	252	41	0.16	0.78	18427	0.97
MilitaryConflict	10691	20904	187	32	0.17	0.66	20209	0.97
Model	1416	18576	257	63	0.25	0.84	17770	0.96
Mountain	13259	19895	359	65	0.18	0.82	18799	0.94
MountainRange	1691	26717	322	106	0.33	0.79	25089	0.94
Museum	3148	17631	290	44	0.15	0.80	16831	0.95
OfficeHolder	32373	21706	476	103	0.22	0.65	20039	0.92
Organisation	200789	23777	840	194	0.23	0.69	21434	0.90
Person	763644	17479	580	168	0.29	0.67	15649	0.90
Place	638879	25527	1126	280	0.25	0.86	22751	0.89
PoliticalParty	3311	20822	267	84	0.31	0.75	19903	0.96
Protein	12042	16513	178	48	0.27	0.75	16037	0.97
River	24267	18759	333	128	0.38	0.74	17382	0.93
Royalty	6563	17283	237	63	0.27	0.78	16315	0.94
SoccerClub	15727	30454	192	47	0.24	0.81	29485	0.97
Species	202339	16097	265	75	0.28	0.84	14933	0.93
TelevisionShow	23480	24595	292	95	0.33	0.65	23681	0.96
Website	2388	15555	295	48	0.16	0.85	14773	0.95

Annotator-Agreement (IAA) based on a sample is 0.72. Then, all the corresponding values returned by the respective query that uses this property are marked either correct or wrong. The rationale is that we assume each individual triple on the Linked Data to be semantically correct, or in other words, that data publishers have respected the semantic meanings of predicates when releasing triple datasets. Although this is not always true on the Linked Data, we believe this gives a reasonable approximation of accuracy.

Using these annotations, we study the accuracy of the synonymity measure and also the increase ratio in the retrievable data due to the inclusion of each synonymous property. We create two sets of statistics for this purpose. First, given a reference property r_i and each of its synonymous property sr_i^j, we compute an *increase ratio* as $IR = \sqrt{\frac{|V_{sr_i^j}|}{|V_{r_i}|}}$. We rank all pairs of $\langle r_i, sr_i^j \rangle$ by descending order of their synonymity scores, and plot IR for each pair (Figure 5). In total there are 561 pairs of relations for all SKPs.

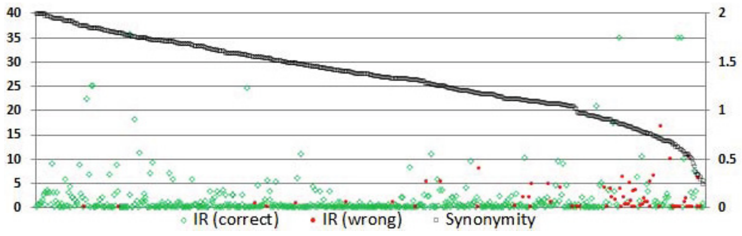

Fig. 5. Increase Ratio (IR) for each $\langle r_i, sr_i^j \rangle$ (marked as green $\diamond$ where the synonymous property is correct and red $\times$ where it is wrong), ranked by synonymity scores ($\square$) in descending order. IR is aligned to the left y-axis and synonymity scores are aligned to the right.

As shown in Figure 5, the synonymity measure correctly predicts synonymous properties in most cases. In many cases, the increase ratio is significant, suggesting that data retrieval can considerably boost recall by including SKP synonymous properties in the query process. There appears to be an inverse correlation between the synonymity score and the correctness of prediction. With high synonymity scores (e.g., > 1.4) the vast majority of synonymous properties discovered for the reference properties are found correct; however, when errors are made, the increase ratio is very low, meaning little noise is added. This is a useful feature as, when necessary, we can apply higher threshold in order to ensure high precision in retrieval. There is no correlation between the increase ratio and the synonymity score. This is expected as on the one hand, synonymity describes the extent to which two properties are “interchangeable” while the increase ratio addresses what they have “in difference”. Ideally, for the purpose of data retrieval, we would like to have a re-ranking process to combine both synonymity and increase ratio in order to promote properties that are highly synonymous, and can also potentially add a lot information to each other. We will explore this in future.

While Figure 5 looks at individual pairs of properties independently, from the SKP construction point of view we are more interested in the incremental performance as the SKP is expanded by progressively adding synonymous properties and data instances.

For this purpose, we simulate an ontology engineering process, where an engineer expands the reference ontology by adding synonymous properties and also new data instances retrieved by such properties. The engineer may rank each pair of reference property with a synonymous property by their score, and progress by incrementally add one property at a time. At each iteration it_k, we study (i) the incremental increase ratio (IIR) due to new data instances retrievable by correct synonymous properties added up at it_k and (ii) the incremental noise ratio (INR) due to new data instances retrievable by incorrect synonymous properties added up at it_k. Let $V^+_{sr_i^j} \subset V_{sr_i^j}$ be the values added by correct synonymous properties and $V^-_{sr_i^j} \subset V_{sr_i^j}$ be the values added by incorrect synonymous properties, the calculation of IIR and INR at each iteration k are calculated as

$$IIR_k = \frac{|\bigcup_{i=1}^{k} V^+_{sr_i^j}|}{|\bigcup_{i=1}^{k}(V_{sr_i^j} \cup V_{r_i^j})|} \quad INR_k = \frac{|\bigcup_{i=1}^{k} V^-_{sr_i^j}|}{|\bigcup_{i=1}^{k}(V_{sr_i^j} \cup V_{r_i^j}|}$$

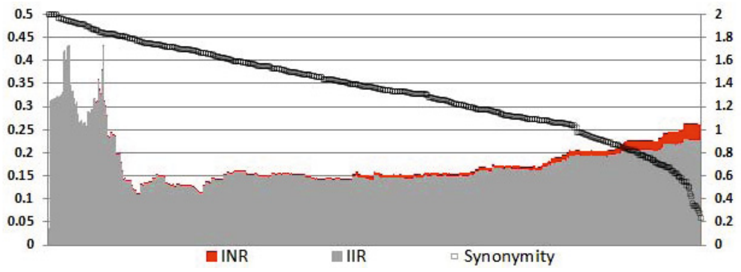

Fig. 6. Incremental Increase Ratio (IIR) and Incremental Noise Ratio (INR) at decreasing synonymity score

Figure 6 shows generally consistent patterns with Figure 5. A high synonymity score rarely introduces errors and when it does, the noise added to the ontology is trivial. As the score drops, noise becomes notable and possibly harms ontology construction.

Error Analysis. To understand the limitations of our method we manually analysed incorrect predictions given by the synonymity measure. We categorise three main sources of errors: (1) highly semantically related properties; (2) property range ambiguity and (3) arguable human annotations.

For many *highly semantically related properties*, we found that often there is a high degree of overlap in their object values. As a result, the synonymity measure makes incorrect predictions based on the data. For example, cities may have the same average temperatures across several months. As a result, our method may predict properties such as “averageTemperatureJune” and “averageTemperatureJuly” to be synonymous. For countries, the property that describes the largest city is considered synonymous with property that describes the capital of the country.

Another source of errors is *property range ambiguity*. We noticed that for some incorrect synonymous property pairs, a common characteristic is that the ranges of the properties derived from data (i.e., the types of the objects of the properties) are ambiguous. Using the synonymous pair *dbo:country* and *dbo:location* for the concept *dbo:MountainRange* as an example, we retrieve the most specific type (ignoring data types) of the objects for each property respectively. We noticed that, *dbo:country* have two distinct ranges in data and the most frequently used is *dbo:Country*, covering 96% of data; while *dbo:location* has 5 distinct ranges and the most frequently used is *dbo:Place*, covering 46% of data, which suggests that the objects of this property is highly inconsistent in terms of their types. Intuitively, if a property's range is ambiguous it would be difficult to assess its synonymity with other properties. We will incorporate this information in our measure in future work.

We also noticed some examples of *highly arguable human annotations*. As an example, *dbo:successor* and *dbpp:after* is predicted synonymous for the class *dbo:Royalty*. However, our annotators considered this example to be incorrect. We manually checked the data and discovered that among all object values (only those that have object types) of *dbpp:after*, 92% belong to the type *dbo:Royalty* and describes a successor of a royalty; and 98% (including *dbo:Royalty*) belong to a class representing a position or person, in which case it describes a successor of certain kind. Thus arguably, although the two properties appear insufficiently synonymous, the data provides additional strong evidence for us to consider them as synonymous for the specific class *dbo:Royalty*.

6 Conclusion and Outlook

In this paper we have introduced the notion of Statistical Knowledge Patterns (SKPs), for capturing the properties used with a certain concept in a bottom-up data-oriented way. We have presented an unsupervised method for generating SKPs, and evaluated it on the DBpedia dataset using the DBpedia ontology as a reference vocabulary. Our evaluation shows that SKPs are able to significantly reduce the number of properties we need to consider, while still maintaining a high coverage of the dataset, compared to considering the complete set of properties present in data. We believe that SKPs are an efficient way to avoid noise in the data, since this is most often present in the "long tail" of property usage. Additionally, the evaluation shows that the clustering of properties, into sets of synonymous properties, allows us to perform query expansion, using the SKP, with high accuracy. Since the methods are completely automated, it is easy to maintain an up-to-date set of SKPs for your dataset, or even across datasets, in order to be able to efficiently query data with sufficiently high recall at any point in time. The main benefits of SKPs include: (1) allowing for both accurate query expansion (through the synonymy of properties) and restriction (through the range axioms); (2) their context dependent nature, describing the usage and meaning of properties in the context of a particular concept and even within a specific dataset; and (3) allowing SKPs to be generated offline (but continuously updated), so that they can be used efficiently at run time.

On an abstract level SKPs can be compared to context-sensitive linguistic resources, such as linguistic frames. Previously, top-down approaches have been used to reengineer linguistic frames, e.g. FrameNet, to KPs. An interesting line of future research is to integrate them with bottom-up approaches such as EKP and SKP generation. As future work we also plan to focus on the second major feature of the SKPs, namely the new range axioms that were introduced based on observations of data. Just as for the properties themselves, the ranges are also concept-specific. We intend to experimentally validate the range extraction method, and to evaluate the potential of using the range axioms for restricting property selection in query formulation. Other improvements of the method could include taking into account more features of the reference ontology when performing the SKP extraction, e.g. the class hierarchy and additional axioms. We will also generate SKPs for the complete DBpedia ontology[7], and other major sources of Linked Data. In particular, we intend to explore the construction of "cross-dataset" SKPs that include synonymous properties from multiple linked datasets. We then intend to use them in an Information Extraction scenario, for extracting seed data corresponding to the natural language questions of a human user. In such a scenario, SKPs will be an essential component, since they represent the actual properties that are used in data, and since they help to break down the query formulation problem into manageable pieces.

Acknowledgements. Part of this research has been sponsored by the EPSRC funded project LODIE: Linked Open Data for Information Extraction, EP/J019488/1.

References

1. Augenstein, I., Gentile, A.L., Norton, B., Zhang, Z., Ciravegna, F.: Mapping Keywords to Linked Data Resources for Automatic Query Expansion. In: Proc. of the 2nd International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data (2013)
2. Basse, A., Gandon, F., Mirbel, I., Lo, M.: DFS-based frequent graph pattern extraction to characterize the content of RDF Triple Stores. In: Proceedings of the WebSci 2010: Extending the Frontiers of Society On-Line, Raleigh, NC, US, April 26-27 (2010)
3. Blomqvist, E.: Ontocase-automatic ontology enrichment based on ontology design patterns. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 65–80. Springer, Heidelberg (2009)
4. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Comput. Linguist. 32(1), 13–47 (2006)
5. Cabrio, E., Aprosio, A.P., Cojan, J., Magnini, B., Gandon, F., Lavelli, A.: QAKiS @ QALD-2. In: Proceedings of the ESWC 2012 Workshop Interacting with Linked Data, Heraklion, Greece (2012)
6. Duan, S., Fokoue, A., Hassanzadeh, O., Kementsietsidis, A., Srinivas, K., Ward, M.J.: Instance-Based Matching of Large Ontologies Using Locality-Sensitive Hashing. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 49–64. Springer, Heidelberg (2012)
7. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook of Ontologies. International Handbooks on Information Systems, vol. 2, Springer (2009)
8. Gangemi, A., Presutti, V.: Towards a pattern science for the Semantic Web. Semantic Web 1(1-2), 61–68 (2010)

[7] The catalogue will be published at `http://ontologydesignpatterns.org/skp/`

9. Le, N.T., Ichise, R., Le, H.B.: Detecting hidden relations in geographic data. In: Proceedings of the 4th International Conference on Advances in Semantic Processing, pp. 61–68 (2010)
10. Musetti, A., Nuzzolese, A., Draicchio, F., Presutti, V., Blomqvist, E., Gangemi, A., Ciancarini, P.: Aemoo: Exploratory Search based on Knowledge Patterns over the Semantic Web. In: Finalist of the Semantic Web Challenge 2011 (2011)
11. Nuzzolese, A.G., Gangemi, A., Presutti, V., Ciancarini, P.: Encyclopedic knowledge patterns from wikipedia links. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 520–536. Springer, Heidelberg (2011)
12. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Discovering concept coverings in ontologies of linked data sources. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 427–443. Springer, Heidelberg (2012)
13. Presutti, V., Aroyo, L., Adamou, A., Schopman, B.A.C., Gangemi, A., Schreiber, G.: Extracting Core Knowledge from Linked Data. In: Proc. of the 2nd Intl. Workshop on Consuming Linked Data (COLD 2011), Bonn, Germany, vol. 782, CEUR-WS.org (2011)
14. Presutti, V., Blomqvist, E., Daga, E., Gangemi, A.: Pattern-based ontology design. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 35–64. Springer, Heidelberg (2012)
15. Schopman, B., Wang, S., Isaac, A., Schlobach, S.: Instance-Based Ontology Matching by Instance Enrichment. Journal on Data Semantics 1(4), 219–236 (2012)