Mapping Keywords to Linked Data Resources for Automatic Query Expansion

Isabelle Augenstein[1], Anna Lisa Gentile[1], Barry Norton[2], Ziqi Zhang[1], and Fabio Ciravegna[1]

[1] Department of Computer Science, University of Sheffield, UK
[2] Ontotext, UK
{i.augenstein,a.l.gentile,z.zhang,f.ciravegna}@dcs.shef.ac.uk, barry.norton@ontotext.com

Abstract. Linked Data is a gigantic, constantly growing and extremely valuable resource, but its usage is still heavily dependent on (i) the familiarity of end users with RDF's graph data model and its query language, SPARQL, and (ii) knowledge about available datasets and their contents. Intelligent keyword search over Linked Data is currently being investigated as a means to overcome these barriers to entry in a number of different approaches, including semantic search engines and the automatic conversion of natural language questions into structured queries. Our work addresses the specific challenge of mapping keywords to Linked Data resources, and proposes a novel method for this task. By exploiting the graph structure within Linked Data we determine which properties between resources are useful to discover, or directly express, semantic similarity. We also propose a novel scoring function to rank results. Experiments on a publicly available dataset show a 17% improvement in Mean Reciprocal Rank over the state of the art.

1 Introduction

Linked Open Data grows at an astonishing rate, and is becoming a gigantic data source that can be harnessed to power various applications. One of the current challenges for accessing and consuming Linked Data on the Web is dealing with heterogeneity [8], i.e. that data can be distributed and duplicated in different data stores, and described by different vocabularies. As a result, accessing and consuming Linked Data on the Web requires end users to be aware of where to find the data (which datasets) and how it is represented (which vocabularies). Additionally, users need to have working knowledge of formal query languages such as SPARQL to access the datasets. This creates high barriers of entry to creating applications that use Linked Data.

A simple and convenient way for users to express their information needs [8] is the usage of natural language queries (*NLQs*). Therefore, we assume that enabling a user to use natural language to query Linked Data would offer a conducive alternative to formal query languages. The first step in translating a NLQ to SPARQL is to map the keywords in the query to Linked Data resources. Most current semantic search approaches offer limited abstraction from the keywords contained in the query, i.e. the mappings are based merely on keywords and their lexical derivatives ([14], also [18], [17]).

Approaches that just consider variations on a lexicographic level often fail entirely to map keywords to Linked Data resources [17]. More advanced approaches therefore enrich their query with semantically similar words, for example by looking up synonyms in a dictionary, often WordNet ([18], [17], [12]), but dictionaries in general contain a very limited vocabulary and even WordNet has very few named entities.

Freitas et al. [7] argue that matching natural language terms to resources in Linked datasets can easily cross taxonomic boundaries and suggest that expanding the query with a more general semantically similar class of terms produces better results than either matching keywords on the mere base of lexicographic similarity or by WordNet-based expansion. Semantic similarity describes the strength of semantic association between two words. It includes the classical relations hypernymy, hyponymy, meronymy and synonymy. Semantic relatedness is more general than similarity and also covers antonymy and other semantic associations [2].

This idea is further supported by recent studies on exploratory query expansion, where users are presented with not only semantically similar, but also generally related keywords [9]. However, the information about related keywords is usually presented to the user but not directly integrated into the ranking of results [4,16]. Furthermore, existing approaches are typically based on a single knowledge source such as WordNet [16], or Wikipedia [7], and are bound to the specific structure of the knowledge source assumed to be known a-priori. We argue that such methods are highly restricted by the scope of the knowledge source and difficult to adapt across domains.

The major contribution of this paper is the basis for a highly generic, adaptable *automatic query expansion* approach that (i) automatically finds and ranks URIs of related resources for keywords, and (ii) that does not impose a-priori knowledge about data sources or knowledge bases. Compared to the foregoing work of Freitas et al. our contribution is to (i) offer a means to benefit from the multitude of properties between resources in Linked Data, not just the use wikiPageWikiLink property, representing the internal Wikipedia links as used in that foregoing work; (ii) offer means to automatically rank the effectiveness of such properties in expressing a useful notion of semantic similarity; (iii) to move beyond simply considering Wikipedia/DBpedia and demonstrate effectiveness across large datasets.

Our approach hinges on the ability to take a keyword from the query, w and expand this into a larger set of keywords, E_w, based on the labellings of 'neighbours' in whole graph of Linked Open Data. In the current work this expanded keyword set is then used to find candidates within a fixed target vocabulary (i.e. we use the whole Linking Open Data Cloud to help find alternative keywords via similar resources but evaluate these by using them to find terms in the DBpedia ontology for the purposes of cross-evaluation). The training phase of our approach is intended to apply supervised learning to identify which relationships (properties) among those available in Linked Open Data express useful semantic similarities.

Compared to the state of the art, this approach is completely independent of the structure of underlying datasets and vocabularies. It is therefore highly generic and adaptable across different, and growing, domains covered by Linked Open Data. It is also highly flexible; as we show that it can be used to expand keywords based on any generalised or specialised semantic relations given suitable training data. We report

experiments on the gold standard used in [10]. Results show a 17% improvement in Mean Reciprocal Rank with respect to figures reported in [10].

The paper is structured as follows: Section 2 discusses relevant related work, Section 3 describes our query expansion method and Section 4 presents an evaluation of our method on the DBpedia ontology dataset. We discuss future work directions and conclude in Section 5.

2 Related Work

Freitas et al. [8] discuss the challenges and limitations of querying Linked Data. They identify the main streams of works addressing them, including those based on Information Retrieval (*IR*) methods and those based on Natural Language Processing (*NLP*).

IR based approaches provide, at different levels, an exploration of the underlying datasets in a Web search style, i.e. one which treats documents as *bags of words*, where each word is a mere token. The basic idea behind IR methods is to apply keyword search directly over Linked Data triples. To achieve this goal the triples are tokenised and indexed according to one of several possible strategies. *Watson* [6] is among the class of Semantic Search engines that crawl, analyse and (token) index semantic data and ontologies. *Sindice* [14] also provides a token index for textual search over Semantic Web resources, as well as a SPARQL endpoint over the merged graph-indexed data it has crawled, and allows (cached) retrieval of resources whether found by textual or graph (SPARQL) matching. The graph processor extracts and indexes all textual property values — including, but not limited to labels — for each resource in the graph and tokenises these. In the case that the property attaching a resource to a textual value is through an inverse functional property, this allows searching for blank nodes (which do not have an explicit identifier). *SearchWebDB* [15,16] allows users to formulate queries in terms of keywords, which are translated to structured queries representing possible interpretations of the user's information needs. Returned results are the triples satisfying those queries, ranked with decreasing syntactic similarity from the original query keywords. The keyword matching process is limited to the literals associated with each objects, but neighbour resources are also retrieved to help the user select the best answer. In a subsequent work, semantically similar entries from WordNet are also used for the keyword matching process [16]. *Falcons Object Search* [4] performs the keyword matching process against labels, literals and textual descriptions of each object and immediately linked objects. A drawback of the search is the assumption that keywords in the query directly match labels for resources, although users can iteratively refine the query with additional keywords. Ranking of results is performed according to text similarity (tf-idf), weighted by the popularity of each object. The popularity is a score calculated at index time, as a logarithmic function of the number of documents where the object is mentioned.

NLP based methods attempt to derive a SPARQL query from a natural language question (NLQ), returning a single answer or the set of results compatible with the user question. These approaches mainly rely on linguistic resources to expand and disambiguate keywords, with the drawbacks of coverage and lack of generality in the relationships considered mentioned earlier. *FREyA* [5] is a natural language interface for

querying ontologies. It performs syntactic parsing of the NLQ and involves the user in the translation process if necessary. The key feature of the approach is that the user's choices are used for training the system in order to improve its performance over time. *Gingseng* [1] and *NLP reduce* [11] are two natural language interface question answering systems. *NLP reduce* allows the user to enter either full natural language questions, sentence fragments, or just keywords. The question is reduced to a set of query terms, which are matched against a precomputed index of the ontology lexicon, where for all lexicon terms stemming and synonym expansion have been performed. *Gingseng* does not provide full natural language interface, but rather guides user input with respect to the considered ontologies, suggesting term and relations present in the underlying ontologies. *PowerAqua* [12] implements a strategy for (i) combining and (ii) ranking answers from distributed heterogeneous data sources in the context of a multi-ontology question answering task. The result list integrates triples coming from different data stores, ranked using a multi-level ranking algorithm, which combines three different criteria (popularity, confidence and semantic interpretation). Two preprocessing steps are performed to translate the natural language question. A linguistic component extracts relevant terms from the question, then a mapping component identifies potentially suitable ontologies for the terms searching for occurrences of the terms, but also synonyms, hypernyms and hyponyms.

Freitas et al. [7] argue that combining NLP and IR techniques with a richer exploitation of the graph structure beyond involved entities, would significantly improve performance of the overall process. They propose a method which allows end users to query Linked Data using natural language queries. The search is performed in three steps. First, they recognise key entities in the query (Named Entities and lexicon terms); then they retrieve potential DBpedia entries matching each entity The evaluation is performed against the DBpedia portion of the QALD evaluation query set [3]. As a general approach, we are inclined to follow directions in [7], as we will discuss in Section 5, but the aim of this paper is a focused evaluation of the preprocessing step which translates and expands terms in the NLQ in a set of Linked Data resources. Although both NLP-centric works and IR based methods point out the importance of such a preprocessing step, they do not provide an in vitro evaluation of such task. Keyword expansion in queries has been extensively addressed in the traditional IR settings [3], but it is still mostly untapped in the context of Linked Data, as mentioned in [9]. We follow the evaluation proposed by Freitas et al. [9], who created a test set for this specific task on Linked Data and elaborate on their proposed method, by providing a richer exploitation of the graph structure around involved concepts.

[3] `http://www.sc.cit-ec.uni-bielefeld.de/sites/www.sc.cit-ec.uni-bielefeld.de/files/sharedtask.pdf`

Table 1. *Labelling* properties

rdfs:label
foaf:name
dc:title
skos:prefLabel
skos:altLabel
fb:type.object.name

Table 2. *Labelling*$^+$ properties

foaf:name	0.26666668
rdfs:label	0.18666667
dc:title	0.16867469
sindice:label	0.16842106
skos:prefLabel	0.15730338
commontag:label	0.13333334
skos:altLabel	–
fb:type.object.name	–

3 Method

At the start of our approach we have a keyword w for which the user wants to find representative resources in Linked Data. The dataset[4] we operate on consists of a set of resources, R, including a set of properties used to denote specific relationships between resources, $P \subset R$. Literals are a further subset of resources $L \subset R$ with a lexical representation, which are disjoint from P (i.e. $P \cap L = \emptyset$). As usual, for RDF, the data consists of a set of statements, $S \subset (R \setminus L) \times P \times R$, each being a 3-tuple (triple) consisting of a 'subject', 'predicate' and an 'object'. In the current application, and the method described here, this is applied in a more focussed setting, where the intention is to find concepts (classes and properties) in a fixed vocabulary; we call the target classes $TC \subset R$, where for each $c \in TC$ we have $(c, \mathsf{rdf{:}type}, \mathsf{rdfs{:}Class}) \in S$ and target properties $TP \subset P$, where for each $p \in TP$ we have $(p, \mathsf{rdf{:}type}, \mathsf{rdf{:}Property}) \in S$.

We have chosen a set of labelling properties, i.e. properties whose values are expected to be literals which might be worthwhile in identifying distinct concepts, $Labelling \subset P$, shown in Table 1. We make claims neither that this is a complete nor, in each part, useful; indeed Table 2 shows that the method itself reveals two further properties we had not considered (in bold) and that, according to the precision metric we explain below, two that we did include were not useful in the evaluation.

In order to find representative concepts we construct from w an expanded set of keywords, E_w, that improve the chance of finding the most fitting concept in the target vocabulary according to its labelling (under the properties *Labelling*). These expanded sets are more general than 'synsets' (sets of synonyms within dictionary-oriented terms) in both scope, including a huge potential range of named entities, and in the flexibility of the semantic relationships covered. A distinctive feature of our method is that the data is not presumed or intended to be governed by the vocabulary targetted but rather the results improve as the dataset considered increases beyond the target (as we shall show in the next section, we cross-evaluate against a method choosing terms from the DBpedia ontology, but use a much larger portion of the Sindice cache to find the expanded set).

As well as exploiting the labels of resources across the whole dataset, our algorithm learns a rank over all properties, not just those used for labelling, according to how

[4] N.B. in terms of the Linking Open Data Cloud this merged graph can, and does in our evaluation, span several individual datasets in the sense they are registered at the Data Hub.

useful they are in expressing useful notions of semantic similarity. This is achieved by a supervised training period, described in Section 3.2, in which the first stages of the algorithm, forming the expanded set and then using this to find candidates in the target vocabulary, is followed by a manual choice between candidates. As a result of this training, we learn a precision for the properties and rank then subset these to create an ordered list $\vec{Rel} \subset P$ that the final part of the algorithm can then use to automatically distinguish between the fitness of candidates, as described in Section 3.3.

3.1 Candidate Identification

In the first stage of the algorithm, i.e. lines 1-7 in Algorithm 1, triples in the dataset where the keyword w is used as the label of a resource, r, are identified, via any labelling property $labelling1 \in Labelling$. Next the neighbours of r, who are the objects of *any* relationship $p \in Q$ (where $Q \subseteq P$ is a parameter) from this subject resource, are found. These neighbours fall into two groups: the first are themseves literals; the second are not literals, but may have literals attached via a labelling property $labelling2 \in Labelling$. The resulting set of labels is called an 'expanded set', E_w, relative the original w.

Algorithm 1 identifyCandidates(w, Q)

1: **for all** r such that $(r, labelling1, w) \in S$ **and** $labelling1 \in Labelling$ **do**
2: **for all** s such that $(r, p, s) \in S$ **and** $p \in Q$ **do**
3: **if** $s \in L$ **then**
4: $E_w \leftarrow s$
5: **else**
6: $E_w \leftarrow \{lit \mid (s, labelling2, lit) \in S$ **and** $labelling2 \in Labelling\}$
7: **end if**
8: $Cand_w \leftarrow Cand_w \cup \{(p, \text{findTargetsFromExpandedLabels}(E_w))\}$
9: **end for**
10: **end for**

This expanded set is used, for each examined property p, to find a list of terms from the target ontology by Algorithm 2.

Algorithm 2 findTargetsFromExpandedLabels(Labels)

for all $label \in Labels$ **do**
$Tokens = \{label\} \cup \text{tokenise}(label)$
end for
for all $token \in Tokens$ **do**
for all t such that $(t, labelling3, token) \in S$ **and** $labelling3 \in Labelling$ **and** $t \in TC \cup TP$ **do**
$Targets \leftarrow Targets \cup \{t\}$
end for
end for
return $Targets$

Algorithm 2 relies on a function *tokenise* which splits multi-word labels into a set of words, i.e. tokenise("programming language") = {"programming", "language"}.

Finally, having obtained the targets according to the given property, Algorithm 1 pairs that property and the target resources found in the set $Cand_w \subset P \times \mathcal{P}(R)$, or propertly $Q \times \mathcal{P}(R)$. At this stage the method branches according to whether we are training, or running automatically with respect to an existing training period, which has defined a ranked short-list $\vec{Rel} \subset P$ that express a useful notion of semantic similarity.

3.2 Training

In order to train, we select a list of keywords to stand for w, with P (i.e. the whole set of properties in the dataset) standing for Q, in several iterations of the algorithm presented in Section 3.1. This training set, $\overrightarrow{W^{train}} = \{w_1^{train}, w_2^{train}, ..., w_n^{train}\}$, consists of nouns and compound nouns in general use; our only constraint in the evaluation was that this set is disjoint from the keywords used in the test set, as described in Section 4.

The aim of training is to produce a precision measure for every property in P^{train}, that is every property exercised in finding candidates across $\overrightarrow{W^{train}}$, i.e.:

$$P^{train} = \{p \mid (p, T) \in Cand_w \text{ where } w \in \overrightarrow{W^{train}})\}$$

The first stage is to manually choose the best (not necessarily unique) among the candidates under each p that occurs in $Cand_w$, for each $w \in \overrightarrow{W^{match}}$. We therefore first generalise $Cand_w$ to $\overrightarrow{Cand^{train}}$, that is an ordered set matching $\overrightarrow{W^{train}}$ such each $Cand_i^{train} = Cand_{w_i^{train}}$.

Based on the manual selection we then also have an ordered set $\overrightarrow{Best}$ such that $Best_{w,p} \subset T_{w,p}$ where $(p, T_{w,p}) \in Cand_w$ for each $w \in \overrightarrow{W^{train}}$.

In order to compute precision for each p, we now define two functions $hits(p, w)$ and $candidates(p, w)$, both defined for each $p \in P^{match}$ and $w \in \overrightarrow{W^{train}}$, as follows:

$$\text{hits}(p, w_i^{train}) = \{\ t \mid t \in T \text{ where } (p, T) \in Cand_i^{train} \text{ and } t \in Best_{w_i^{train}, p}\ \}$$
$$\text{candidates}(p, w_i^{train}) = \{\ t \mid t \in T \text{ where } (p, T) \in Cand_i^{train}\ \}$$

We can then apply the following simple calculation:

$$\text{prec}(p) = \frac{\sum_{w \in \overrightarrow{W^{train}}} \text{hits}(p, w)}{\sum_{w \in \overrightarrow{W^{train}}} \text{candidates}(p, w)}$$

We then define some treshold $0 \leq \theta \leq 1$, and use $prec$ to define an ordered (ranked) subset of properties shown to encode semantic relatedness, $\vec{Rel}$, as the greatest set with the following definition:

$$rel_i \in \vec{Rel} \text{ iff } rel_i \in P^{match} \wedge prec(rel_i) > \theta \wedge \nexists j \text{ such that } j > i \wedge prec(j) < prec(i)$$

3.3 Judging Candidate Fitness

Having obtained the ranked list of properties for semantic relatedness by training, we can now take any set of candidates over a test set of keywords, $\overrightarrow{W^{test}}$, and apply Algorithm 1 to obtain candidates. In this case however we take only properties from $\vec{Rel}$ to form the other parameter, Q. As a result we assume $\overrightarrow{Cand^{test}}$ matching the definition in Section 3.2, over properties P^{test}, again assuming a matching definition (note, by design: $P^{test} \subseteq \vec{Rel}$).

In order to judge fitness of each candidate resource r in $Cand_i^{test}$ (i.e. where $r \in R$ and $(p, R) \in Cand_i^{test}$ for some p) to the corresponding keyword, w_i^{test}, we combine, as a numerical product, the precision of the property, p, used to find that candidate and a tf-idf score for r, computed as follows:

$$\mathrm{tf}(r, w_i^{test}) = \frac{\sum_{p \in \vec{Rel}} \begin{smallmatrix}1 \text{ if } r \in R \\ 0 \text{ if } r \notin R\end{smallmatrix} \text{ where } (p, R) \in Cand_i^{test}}{1 + \sum_{v \in \overrightarrow{W^{train}}} \sum_{p \in \vec{Rel}} \begin{smallmatrix}1 \text{ if } r \in R \\ 0 \text{ if } r \notin R\end{smallmatrix} \text{ where } (p, R) \in \overrightarrow{Cand^{train}}}$$

And the 'inverse' (i.e., logarithmic) document frequency as:

$$\mathrm{idf}(r, \overrightarrow{W^{train}}) = log \frac{|\overrightarrow{W^{train}}|}{1 + \sum_{v \in \overrightarrow{W^{train}}} \begin{smallmatrix}1 \text{ if } \exists p,R \text{ such that } r \in R \text{ and } (p,R) \in \overrightarrow{Cand^{train}} \\ 0 \text{ otherwise}\end{smallmatrix}}$$

4 Evaluation

For evaluation purposes we used the DBpedia ontology as the target vocabulary (i.e. *TC* were DBpedia ontology classes and *TP* were DBpedia ontology properties), and a dataset of the Sindice cache (therefore S consists of many statements from datasets registered at the Data Hub[5], as shown in the Linking Open Data Cloud, together with further semantic data found by crawling).

4.1 Gold Standard and Evaluation Metric

We derived a gold standard, containing 178 keywords of which 134 have at least one representation in the DBpedia ontology, from the experiment described in [9]. When examining this paper's set up and results, we detected some minor errors which we have corrected in our cross-evaluation. In re-evaluating the approach used by Freitas et al. [9] the actual figures for their approach did not change significantly with respect to our improvements (0.6 vs. 0.64 in the original evaluation).

The first class of correction actually rates Freitas et al.'s results higher than their own evaluation as the keyword *beatles* was marked as not available in the DBpedia ontology (*N/A*), although their results list contained the DBpedia class dbpedia-owl:Artist, which we score as a hit and credit them for this in all such cases. Another group of errors concern keywords that are marked as *N/A*, where the correct result is not contained in the results set but actually available in DBpedia (e.g. dbpedia-owl:manufacturer and dbpedia-owl:Automobile for the keyword *honda*).

[5] `http://datahub.io/`

Table 3. Top 23 properties used and their precision

foaf:name	0.267	owl:sameAs	0.121
fb-common:topic	0.189	wn20schema:gloss	0.1
rdfs:label	0.187	opencyc:seeAlsoURI	0.093
dc:subject	0.182	rdfs:seeAlso	0.082
dc:title	0.169	dbpedia-owl:abstract	0.0774
sindice:label	0.168	rdfs:comment	0.0676
rdfs:suBClassOf	0.168	rdfs:range	0.0667
skos:prefLabel	0.157	rdfs:subClassOf	0.0656
fb-type:object	0.143	fb:documented_object	0.0619
wn20schema:derivationallyRelated	0.143	dbpedia-owl:wikiPageWikiLink	0.0487
wn20schema:containsWordSense	0.138	dc:description	0.0471
commontag:label	0.133		

Additionally, some results were marked as correct matches although on closer inspection there are better candidates; for example, for the keyword *feline*, the match dbpedia-owl:genus was accepted, although a less general result is dbpedia-owl:Mammal.

Finally, there were some inconsistencies between Freitas et al.'s self evaluation when comparing the result table and their downloaded gold standard (i.e., the correct answer was marked in the gold standard, but marked as *N/A* in the result table).

The updated dataset and the results of all evaluations are available for download[6].

Freitas et al. [7] evaluate query expansion as an information retrieval task, where expansion candidates are ranked according to their relevance. The measure used for evaluation is *Mean Reciprocal Rank* (MRR) which measures the quality of the ranking by calculating the inverse rank of the best result. In this study, we follow the same approach. That is, we apply MRR to the resulting ranked list of candidates given by our algorithm.

4.2 Approach

Since our approach requires a supervised training phase, we manually created a training set following the same principles outlined in [7]. The training set contains 40 keywords. After applying the training phase, 194 candidate relations are learnt. We used a precision threshold of 0.045 to cut off the candidate relation list. This resulted in 23 relations which can be found in Table 3.

As well as producing, when projected over labelling properties (those with plain literal values), an interesting comparison (Table 2) with our original set of labelling properties (Table 1), as observed earlier, this list also brings to light a common but useful error in the Sindice-crawled data, the misspelling suBClassOf, which turns out to be higher precision (0.168) than the correct predicate subClassOf (0.0656).

[6] http://oak.dcs.shef.ac.uk/LODIE/know-a-lod-2013

Table 4. Set of example results

Query: [spacecraft]	Query: [engine]	Query: [factory]
dbpedia-owl:Spacecraft	**dbpedia-owl:engine**	dbpedia-owl:manufacturer
dbpedia-owl:spacecraft	**dbpedia-owl:gameEngine**	**dbpedia-owl:plant**
dbpedia-owl:satellite	dbpedia-owl:Artwork	dbpedia-owl:Canal
dbpedia-owl:missions	**dbpedia-owl:AutomobileEngine**	dbpedia-owl:engine
dbpedia-owl:launches	dbpedia-owl:Locomotive	dbpedia-owl:class
dbpedia-owl:closed	dbpedia-owl:fuel	dbpedia-owl:Album
dbpedia-owl:vehicle	dbpedia-owl:added	dbpedia-owl:product
dbpedia-owl:Rocket	dbpedia-owl:Musical	**dbpedia-owl:assembly**
Query: [bass]	Query: [wife]	Query: [honda]
dbpedia-owl:Fish	**dbpedia-owl:spouse**	**dbpedia-owl:manufacturer**
dbpedia-owl:Instrument	dbpedia-owl:Criminal	dbpedia-owl:discovered
dbpedia-owl:instrument	**dbpedia-owl:person**	dbpedia-owl:Asteroid
dbpedia-owl:voice	dbpedia-owl:status	dbpedia-owl:engine
dbpedia-owl:partner	dbpedia-owl:education	dbpedia-owl:season
dbpedia-owl:note	dbpedia-owl:Language	**dbpedia-owl:vehicle**
dbpedia-owl:Musical	dbpedia-owl:sex	**dbpedia-owl:Automobile**
dbpedia-owl:lowest	**dbpedia-owl:family**	dbpedia-owl:participant

4.3 Results

In order to illustrate the output of our method, Table 4 lists keyword queries and their ranked lists. The correct results the user chose in the evaluation are marked, as in the gold standard, in bold. These example queries illustrate that our approach is able to find resources of extended keywords based on different kinds of semantic relationships. These include identity (e.g. spacecraft - dbpedia-owl:spacecraft), hyponymy (e.g. engine - dbpedia-owl:AutomobileEngine), hypernymy (e.g. wife - dbpedia-owl:spouse, dbpedia-owl:person, dbpedia-owl:family), meronymy (e.g. honda - dbpedia-owl:engine), and synonymy (e.g. factory - dbpedia-owl:plant). Our approach is also able to different senses for ambiguous keywords (e.g. bass - dbpedia-owl:Fish, dbpedia-owl:Instrument, dbpedia-owl:instrument. The authors leave it to the reader's imagination to explain why the class Criminal is ranked so highly in the matches for the keyword "wife".

The results of our evaluation are shown in Tables 5 and 6. Our MRR (*LOD Keyword Expansion*) is 0.77, which means most of the best results are located on the first position. We can further show 17% improvement in MRR in comparison with the Wikipedia based approach (*ESA*) Freitas et al. [9] followed. Overall, *LOD Keyword Expansion* is able to answer 90% of the keyword queries (only taking into account keywords that are represented in DBpedia), which is 3% more than *ESA*. We also compare our approach to a simple *String match* baseline and a String match baseline enriched with WordNet synonyms (*String match + WordNet*) [9]. Our approach is able to answer twice as many keyword queries as *String match* and 38% more than *String match + WordNet*.

Table 5. Mean reciprocal rank (MRR)

Model	MRR
ESA	0.6
LOD Keyword Expansion	0.77

Table 6. Percentage queries answered (Recall)

Model	Recall
Strich match	0.45
String match + WordNet	0.52
ESA	0.87
LOD Keyword Expansion	0.90

5 Conclusion and Outlook

In this paper, we have described a method for automatic query expansion for Linked Data resources which is based on using properties between resources within the Linking Open Data cloud. In our evaluation, we examined how useful these different properties are for finding semantic similarities and thereby finding alternative (expanded) keywords, and applied our method to the DBpedia ontology as a target. Compared to the state of the art [9], we show an improvement of 17% in Mean Reciprocal Rank.

The approach described currently bases query expansion on semantic similarity. Related work discussed above [17] follows a multi-strategy approach; first using string similarity then, if the desired result is not achieved, lexical expansion with WordNet, and finally, where these simpler methods have failed, using ESA (as in [9]). They show that the best result is achieved by combining all these approaches. We would speculate, therefore, that our results could be achieved with such a hybrid approach, bringing to bear our general means for semantic similarity only after the lower-hanging fruit has been picked off.

Future work will also re-apply some feedback based on the approach documented here. Firstly, as shown in Tables 1 versus 2, our set of labelling properties could be adjusted, extending the reach of the method. Secondly, once this set is fixed we could fine-tune the algorithm by learning over the set of properties that express semantic similarity separately from the labelling properties (which are currently discarded from Algorithm 2).

While this work focused on the evaluation of the keyword expansion process per se, we intend to perform an in vivo evaluation of the task, by performing a semantic search evaluation. One way we foresee for privileging one concept rather than the other in the simultaneous matching is exploiting Encyclopaedic Knowledge patterns (EKPs) [13], to exploit not only available relations between candidate concepts, but also statistical evidence in the data about connections which have been actually used in instance data.

References

1. Bernstein, A., Kaufmann, E., Kaiser, C.: Querying the semantic web with ginseng: A guided input natural language search engine. In: 15th Workshop on Information Technologies and Systems, Las Vegas, NV. pp. 112–126 (2005)
2. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Journal of Computational Linguistics 32(1), 13–47 (2006)
3. Carpineto, C., Romano, G.: A Survey of Automatic Query Expansion in Information Retrieval. ACM Comput. Surv. 44(1), 1 (2012)
4. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: Proceedings of the 17th international conference on World Wide Web. pp. 1101–1102. ACM (2008)
5. Damljanovic, D., Agatonovic, M., Cunningham, H.: FREyA: An interactive way of querying Linked Data using natural language. In: The Semantic Web: ESWC 2011 Workshops. pp. 125–138. Springer (2012)
6. d'Aquin, M., Baldassarre, C., Gridinoc, L., Sabou, M., Angeletou, S., Motta., E.: Watson: Supporting next generation semantic web applications. In: Proceedings of the WWW/Internet conference, Vila real, Spain (2007)
7. Freitas, A., Curry, E., Oliveira, J.G., O'Riain, S.: A Distributional Structured Semantic Space for Querying RDF Graph Data. International Journal of Semantic Computing 5(04), 433–462 (2011)
8. Freitas, A., Curry, E., Oliveira, J.G., O'Riain, S.: Querying heterogeneous datasets on the linked data web: Challenges, approaches, and trends. Internet Computing, IEEE 16(1), 24–33 (2012)
9. Freitas, A., Curry, E., O'Riain, S.: A distributional approach for terminological semantic search on the linked data web. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 384–391. ACM (2012)
10. Freitas, A., Oliveira, J., O'Riain, S., Curry, E., Pereira da Silva, J.: Querying Linked Data using semantic relatedness: A vocabulary independent approach. Natural Language Processing and Information Systems pp. 40–51 (2011)
11. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: A "naıve" but Domain-independent Natural Language Interface for Querying Ontologies. In: 4th European Semantic Web Conference (2007)
12. Lopez, V., Nikolov, A., Fernandez, M., Sabou, M., Uren, V., Motta, E.: Merging and ranking answers in the semantic web: The wisdom of crowds. The semantic web pp. 135–152 (2009)
13. Nuzzolese, A., Gangemi, A., Presutti, V., Ciancarini, P.: Encyclopedic knowledge patterns from wikipedia links. The Semantic Web–ISWC 2011 pp. 520–536 (2011)
14. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice. com: a document-oriented lookup index for open linked data. International Journal of Metadata, Semantics and Ontologies 3(1), 37–52 (2008)
15. Tran, T., Wang, H., Haase, P.: SearchWebDB: Data web search on a pay-as-you-go integration infrastructure. Tech. rep., Technical report, University of Karlsruhe (2008)
16. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. pp. 405–416. IEEE (2009)
17. Walter, S., Unger, C., Cimiano, P., Bär, D.: Evaluation of a layered approach to question answering over linked data. In: The Semantic Web–ISWC 2012, pp. 362–374. Springer (2012)
18. Wang, H., Tran, T., Haase, P., Penin, T., Liu, Q., Fu, L., Yu, Y.: Searchwebdb: Searching the billion triples. In: Billion Triple Challenge at the International Semantic Web Conference (2008)