DEQA: Deep Web Extraction for Question Answering*

Jens Lehmann[1,2], Tim Furche[1], Giovanni Grasso[1], Axel-Cyrille Ngonga Ngomo[2], Christian Schallhart[1], Andrew Sellers[1], Christina Unger[3], Lorenz Bühmann[2], Daniel Gerber[2], Konrad Höffner[2], David Liu[1], and Sören Auer[2]

[1] Department of Computer Science, Oxford University, Wolfson Building, Parks Road, Oxford OX1 3QD
firstname.lastname@cs.ox.ac.uk
[2] Institute of Computer Science, University of Leipzig, 04103 Leipzig
lastname@informatik.uni-leipzig.de
[3] Bielefeld University, CITEC, Universitätsstraße 21–23, 33615 Bielefeld
cunger@cit-ec.uni-bielefeld.de

Abstract. Despite decades of effort, intelligent object search remains elusive. Neither search engine nor semantic web technologies alone have managed to provide usable systems for simple questions such as "find me a flat with a garden and more than two bedrooms near a supermarket."

We introduce DEQA, a conceptual framework that achieves this elusive goal through combining state-of-the-art semantic technologies with effective data extraction. To that end, we apply DEQA to the UK real estate domain and show that it can answer a significant percentage of such questions correctly. DEQA achieves this by mapping natural language questions to SPARQL patterns. These patterns are then evaluated on an RDF database of current real estate offers. The offers are obtained using OXPath, a state-of-the-art data extraction system, on the major agencies in the Oxford area and linked through LIMES to background knowledge such as the location of supermarkets.

1 Introduction

Answering questions such as "find me a flat to rent close to Oxford University with a garden" is one of the challenges that has haunted the semantic web vision since its inception [3]. Question answering has also been the holy grail of search engines, as recently illustrated by both Google and Bing touting "structured data" search and "query answering". Though both of these efforts have made

* The research leading to these results has received funding under the European Commission's Seventh Framework Programme (FP7/2007–2013) from ERC grant agreement DIADEM, no. 246858, IP grant agreement LOD2, no. 257943 and Eurostars E!4604 SCMS.

P. Cudré-Mauroux et al. (Eds.): ISWC 2012, Part II, LNCS 7650, pp. 131–147, 2012.
© Springer-Verlag Berlin Heidelberg 2012

great strides in answering questions about general, factual knowledge, they have fallen short for more transient information such as real estate, tickets, or other product offerings. Vertical search engines and aggregators also fail to address such questions, mostly due to a lack of natural language understanding and limited background knowledge.

This is true even though data extraction and semantic technologies aim to address this challenge from quite opposite directions: On the one hand, the aim of web extraction is to obtain structured knowledge by analyzing web pages. This does not require publishers to make any changes to existing websites, but requires re-engineering the original data used to generate a website. On the other hand, semantic technologies establish the means for publishers to directly provide and process structured information, avoiding errors in extracting ambiguously presented data, but placing a considerable burden on publishers. Despite this chasm in how they approach question answering, neither has succeeded in producing successful solutions for transient, "deep" web data (in contrast to general, Wikipedia-like knowledge and web sites).

In this paper, we show that in this very dichotomy lies the solution to addressing deep web question answering: We present DEQA, a system that allows the easy combination of semantic technologies, data extraction, and natural language processing and demonstrate its ability to answer questions on Oxford's real estate market. The data is extracted from the majority of Oxford's real estate agencies, despite the fact that none publishes semantic (or other structured) representations of their data, and combined with background knowledge, e.g., to correlate real estate offers with points of interest such as the "Ashmolean Museum" or close-by supermarkets.

DEQA is the first comprehensive framework for *deep web question answering* approaching the problem as a combination of three research areas: **(1)** *Web data extraction* – to obtain offers from real estate websites, where no structured interface for the data is available (which happens to be the case for all Oxford real estate agencies). **(2)** *Data integration* – to interlink the extracted data with background knowledge, such as geo-spatial information on relevant points of interest. **(3)** *Question answering* – to supply the user with a natural language interface, capable of understanding even complex queries. For example a query like "find me a flat to rent close to Oxford University with a garden" can be answered by DEQA. However, this cannot be achieved without adaptation to the specific domain. The unique strength of DEQA is that it is based not only on best-of-breed data extraction, linking, and question answering technology, but also comes with a clear methodology specifying how to adapt DEQA to a specific domain. In Section 3, we discuss in detail what is required to adapt DEQA to a new domain and how much effort that is likely to be.

DEQA *Components.* We developed DEQA as a conceptual framework for enhancing classic information retrieval and search techniques using recent advances in three technologies for the above problems, developed by the three groups involved in DEQA: DIADEM at Oxford, AKSW at Leipzig, and CITEC at Bielefeld.

(1) OXPath is a light-weight data extraction system particularly tailored to quick wrapper generation on modern, scripted web sites. As demonstrated in [9], OXPath is able to solve most data extraction tasks with just four extensions to XPATH, the W3C's standard query language for HTML or XML data. Furthermore, through a sophisticated garbage collection algorithm combined with tight control of the language complexity, OXPath wrappers outperforms existing data extraction systems by a wide margin [9]. For the purpose of integration into DEQA, we extended OXPath with the ability to direct extract RDF data, including type information for both entities and relations.

(2) The real estate offers extracted with OXPath contain no or little contextual knowledge, e.g., about general interest locations or typical ranges for the extracted attributes. To that end, we link these extracted offers with external knowledge. This is essential to answer common-sense parts of queries such as "close to Oxford University". Specifically, we employ the LIMES [23,22] framework, which implements time-efficient algorithms for the discovery of domain-specific links to external knowledge bases such as DBpedia [1].

(3) To apply question answering in a straightforward fashion on the supplied, extracted, and enriched knowledge, we employ the *TBSL* approach [28] for translating natural language questions into SPARQL queries. TBSL disambiguates entities in the queries and then maps them to templates which capture the semantic structure of the natural language question. This enables the understanding of even complex natural language containing, e.g., quantifiers such as the most and more than, comparatives such as higher than and superlatives like the highest – in contrast to most other question answering systems that map natural language input to purely triple-based representations.

Using the combination of these three technologies allows us to adjust to a new domain in a short amount of time (see Section 3), yet to answer a significant percentage of questions about real estate offers asked by users (see Section 4).

Contributions. These results validate our hypothesis, that the combination of these technologies can (and may be necessary to) yield accurate question answering for a broad set of queries in a specific domain. This is achieved without requiring publishers to provide structured data and at a fairly low effort for domain adaptation. Specifically,

(1) DEQA is the first comprehensive *deep web question answering system for entire domains* that can answer the majority of natural language questions about objects only available in form of plain, old HTML websites (Section 2).
(2) These websites are turned into structured RDF data through an *extension of OXPath for RDF output,* providing a concise syntax to extract object and data properties (Section 2.1).
(3) By extracting this data into RDF and *linking* it with background knowledge, it can answer not only queries for specific attributes ("in postcode OX1"), but also queries using *common-sense criteria* ("close to Oxford University"), see Section 2.2.
(4) With TBSL, we are able to map such queries to SPARQL expressions even if they include *complex natural language expressions* such as "higher than".

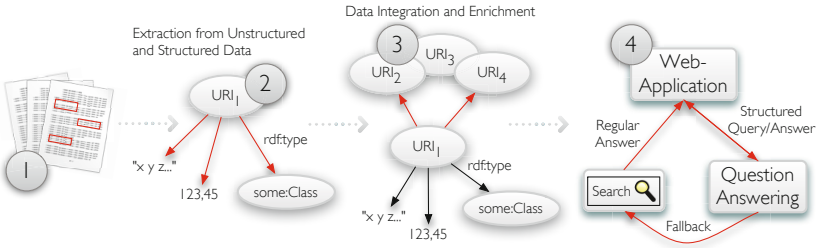

Fig. 1. Overview of the DEQA conceptual framework.

(5) DEQA provides a *methodology and framework* that can be rapidly instantiated for new domains, as discussed in Section 3.
(6) As a *case study*, we instantiate DEQA to Oxford's entire real estate market, involving the 20 largest real estate agents and all of their properties on sale, and illustrate the necessary effort.
(7) A user-centric *evaluation* demonstrates that DEQA is able to answer many of the natural language questions asked by users (Section 4).

With these contributions, DEQA is the first comprehensive framework for deep web query answering, covering the extraction and data collection process as well as the actual query answering, as elaborated in Section 5.

2 Approach

The overall approach of DEQA is illustrated in Figure 1: Given a particular domain, such as real estate, the first step consists of identifying relevant websites and extracting data from those. This previously tedious task can now be reduced to the rapid creation of OXPath wrappers as described in Section 2.1. In DEQA, data integration is performed through a triple store using a common base ontology. Hence, the first phase may be a combination of the extraction of unstructured and structured data. For instance, websites may already expose data as *RDFa*, which can then be transformed to the target schema, e.g. using *R2R* [4], if necessary. This basic RDF data is enriched, e.g. via linking, schema enrichment [16,6], geo-coding or post-processing steps on the extracted data. This is particularly interesting, since the LOD cloud contains a wealth of information across different domains which allows users to formulate queries in a more natural way (e.g., using landmarks rather than postcodes or coordinates). For instance, in our analysis of the real estate domain, over $100k$ triples for $2,400$ properties were extracted and enriched by over $100k$ links to the LOD cloud. Finally, question answering or semantic search systems can be deployed on top of the created knowledge. One of the most promising research areas in question answering in the past years is the conversion of natural language to SPARQL queries [28,18,27], which allows a direct deployment of such systems on top of a triple store. Finally, DEQA first attempts to convert a natural language query to SPARQL, yet can fall back to standard information retrieval, where this fails.

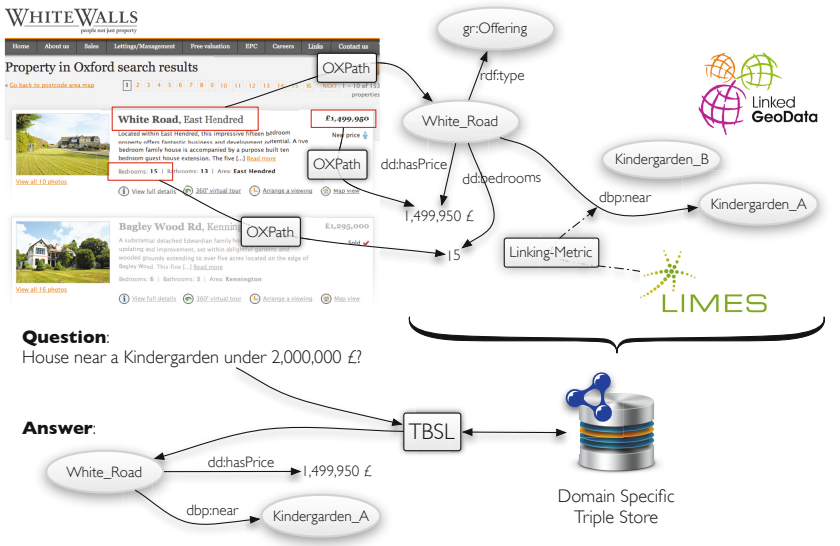

Fig. 2. Implementation of DEQA for the real-estate domain

The domain-specific implementation of the conceptual framework, which we used for the real estate domain, is depicted in Figure 2. It covers the above described steps by employing state-of-the-art tools in the respective areas, OXPath for data extraction to RDF, LIMES for linking to the linked data cloud, and TBSL for translating natural language questions to SPARQL queries. In the following, we briefly discuss how each of these challenges are addressed in DEQA.

2.1 OXPath for RDF Extraction

OXPath is a recently introduced [9] modern wrapper language that combines ease-of-use (through a very small extension of standard XPATH and a suite of visual tools [15]) with highly efficient data extraction. Here, we illustrate OXPath through a sample wrapper shown in Figure 3.

This wrapper directly produces RDF triples, for which we extended OXPath with *RDF extraction markers* that generate both data and object properties including proper type information and object identities. For example the extraction markers `<:(gr:Offering>` and `<gr:includes(dd:House)>` in Figure 3 produce – given a suitable page – a set of matches typed as `gr:Offering`, each with a set of `dd:House` children. When this expression is evaluated for RDF output, each pair of such matches generates two RDF instances related by `gr:includes` and typed as above (i.e., three RDF triples).

To give a more detailed impression of an OXPath RDF wrapper assuming some familiarity with XPATH, we discuss the main features of Figure 3:

(Line 1) We first load the web site `wwagency.co.uk`, a real estate agency serving the Oxford area, and click on their search button without restricting the

```
doc("http://wwagency.co.uk/")//input[@name='search']/{click/}/
(descendant::a[@class='pagenum']
    [text()='NEXT'][1]/{click[wait=1]/})*
 /descendant::div.proplist_wrap:<(gr:Offering)>
 [?.//span.prop_price:<dd:hasPrice(xsd:double)=
   substring-after(.,'£')>]
 [?.//a[@class='link_fulldetails']:<foaf:page=string(@href)>]
 [?.:<gr:includes(dd:House)>
   [?.//h2:<gr:name=string(.)>]
   [?.//h2:<vcard:street_address=string(.)>]
   [?.//div.prop_maininfo//strong[1]:<dd:bedrooms=string(.)>]
   [? .//img:<foaf:depiction=string(@src)>]
```

Fig. 3. OXPath RDF wrapper

search results. Therein, `{click/}` is an *action* which clicks on all elements in the current context set, in this case, containing only the search button. This action is *absolute,* i.e., after executing the action, OXPath continues its evaluation at the root of the newly loaded document.

(Lines 2–3) Next, we iterate through the next links connecting the paginated results. To this end, we repeat within a *Kleene star* expression the following steps: we select the first link which is of class `'pagenum'` and contains the text `'NEXT'`. The expression then clicks on the link in an absolute action `{click[wait=1]/}` and waits for a second after the `onload` event to ensure that the heavily scripted page finishes its initialization.

(Line 4) On each result page, we select all `div` nodes of class `proplist_wrap` and extract an `gr:Offering` instance for each such node. Aside from the CSS-like shorthand for classes (analogously, we provide the `#` notation for ids), this subexpression uses the first *RDF extraction marker:* This extraction marker `:<gr:Offering>` produces an *object instance,* because it does not extract a value necessary to produce a *data property.* The remainder of the expression adds object and data properties to this instance, detailing the offering specifics.

(Lines 5–6) We extract the price of the offering by selecting and extracting the `span` of class `prop_price` within the offering `div`. In particular, the marker `:<dd:hasPrice(xsd:double)=substring-after(.,'£')>` specifies the extraction of a `dd:hasPrice` data property of type `xsd:double` with the value stated after the `'£'` character. The nesting of RDF properties follows the predicate nesting structure, and thus, as the price is extracted inside a predicate following the extracted offering, this price is associated with the offering. We use an *optional predicate,* `[?`ϕ`]`, to ensure that the evaluation continues, even if an offering does not name a price and the predicate extraction fails.

(Line 7) Links to details pages are extracted as `foaf:page` data properties.

(Lines 8–12) Aside having a price, an offering also needs to refer to a property, extracted next. In Line 8, with `:<gr:includes(dd:House)>`, we extract an instance of the `dd:House` class as *object property* of the previous offering (because of the predicate nesting), related via `gr:include`. The remaining four lines extract the name,

address, the number of bedrooms, and the property images as data properties belonging to the `dd:House` instance, as all those properties are extracted within nested predicates.

This wrapper produces RDF triples as below, describing two instances, the first one `dd:g31g111` representing a house with 4 bedrooms in Marston, and the second one `dd:g31g109` representing an offer on this house at GBP 475000.

```
dd:g31g111
    a dd:House ;            dd:bedrooms 4 ;
    gr:name "William Street,  Marston OX3" ;
    vcard:street-address "William Street,  Marston OX3" ;
    foaf:depiction "http://www.wwagency.com/i_up/111_1299510028.jpg" .
dd:g31g109
    a gr:offering ;       dd:hasPrice "475000"^^xsd:double ;
    gr:includes dd:g31g111 .
```

For more details on OXPath, please refer to [9]. We also provide the full set of wrappers on the project home page.

2.2 LIMES

We discuss the LIMES specification used to link and integrate the RDF data extracted by OXPath with *LinkedGeoData* – a vast knowledge base extracted from OpenStreetMaps containing spatial data including points-of-interest such as schools. The following listing shows an excerpt of the specification that links houses extracted by OXPath with nearby schools. Every link discovery process requires a set S of source and T target instances that are to be linked. In LIMES, these can be defined by specifying the *restrictions* on the instances as well as the set of *properties* that these instances must possess to be linked. In our example, the set S (specified by the tag `<SOURCE>`) consists of `oxford:House` which possess a longitude and a latitude. Similarly, the set T (which is omitted in the listing for brevity) was defined as all the schools whose latitude lies between 50 and 52 degrees and whose longitude lies between -2 and -1 degrees. For instances $a \in S$ and $b \in T$, the similarity is set to

$$\frac{1}{1 + \sqrt{(a.\texttt{wgs:lat} - b.\texttt{wgs:lat})^2 + (a.\texttt{wgs:long} - b.\texttt{wgs:long})^2)}}. \quad (1)$$

Two instances are then considered close to each other (described by the predicate `dbp:near`) if their similarity was at least 0.95.

```
<SOURCE> <ID>oxford</ID>
...
        <VAR>?a</VAR>
        <RESTRICTION>?a a oxford:House</RESTRICTION>
        <PROPERTY>wgs:lat AS number</PROPERTY>
        <PROPERTY>wgs:long AS number</PROPERTY> </SOURCE>
...
<METRIC>euclidean(a.wgs:lat|wgs:long, b.wgs:lat|wgs:long)</METRIC>
```

```
  <ACCEPTANCE> <THRESHOLD>0.9975</THRESHOLD>
10              <FILE>allNear.ttl</FILE>
                <RELATION>dbp:near</RELATION> </ACCEPTANCE>
12 ...
```

The property values of all schools from LinkedGeoData that were found to be close to houses extracted by OXPath were subsequently retrieved by LIMES and loaded into the DEQA triple store.

2.3 TBSL Question Answering

Figure 4 gives an overview of our template-based question answering approach TBSL [28]. The system takes a natural language question as input and returns a SPARQL query and the corresponding answer(s) as output. First, the natural language question is parsed on the basis of its part-of-speech tags and a domain-independent grammar comprising for example wh-words, determiners, and numerals. The result is a semantic representation of the natural language query, which is then converted into a SPARQL query template. This template fixes the overall structure of the target query, including aggregation functions such as filters and counts, but leaves open slots that still need to be filled with URIs corresponding to the natural language expressions in the input question. For example, the question "Give me all flats near Oxford University" yields the following template query, which contains a class slot for some URI corresponding to "flats", a resource slot for some URI corresponding to "Oxford University", and a property slot that expresses the "near" relation:

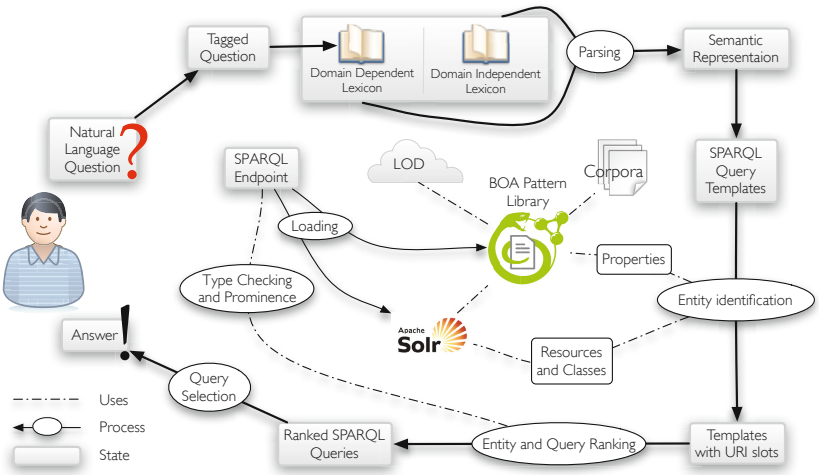

Fig. 4. Overview of the TBSL question answering engine (source: [28])

```
SELECT ?y WHERE {
     ?y ?p1 ?y0.
     ?y rdf:type ?p0.
}
```

- `y0`: "Oxford University" (resource)
- `p0`: "flats" (class)
- `p1`: "near" (object property)

In order to fill these slots, entity identification approaches are used to obtain appropriate URIs, relying both on string similarity and natural language patterns compiled from existing structured data in the Linked Data cloud and text documents (cf. [10]). This yields a range of query candidates as potential translations of the input question. Those candidates are ranked on the basis of string similarity values, prominence values, and schema conformance checks. The highest ranked queries are then tested against the underlying triple store and the best answer is returned to the user.

3 Domain Adaption Costs

DEQA requires instantiation for a specific domain, however, through advances in semantic and web extraction technologies this adaptation involves far less efforts than in the past and is now feasible even with limited resources. We substantiate this claim by discussing the resources required for our case study on Oxford's real estate for **(1)** system setup and domain adaptation and for and **(2)** maintaining the wrappers and links to background knowledge.

The first step in adapting DEQA to a new domain is the *creation or adaption of a suitable domain ontology* in RDFS. In our case, the ontology consists of 5 object properties, 7 data properties, 9 classes, and 10 individuals, all specified in less than 150 lines of turtle code. We were cautious to capture all relevant cases. Hence we build the ontology iteratively while fitting a dozen representative offers from 4 different agencies into the ontology – reaching already a saturation. The entire process of ontology creation took four domain experts a couple of hours.

Web Extraction. Having the ontology, we need to develop wrappers to extract the data from the relevant sites. The process consists of identifying the relevant DOM features to frame the data to be extracted, and running sufficiently many tests to check the wrapper's behavior on other pages from the same site. The wrappers we employ in our case study took on average 10 minutes each to create, such that it took an OXPath expert less than a day to identify the 20 most relevant web sites and write appropriate wrappers. To ease OXPath wrapper generation, we relied on Visual OXPath [15], a supervised wrapper induction system that generates highly robust wrappers from few examples: The system embeds a real browser, and records user interaction on the page (e.g., navigation, click, form filling). Once, the relevant data has been reached, the user marks the data to extract (e.g., price), and checks whether Visual OXPath generalizes the selection correctly, in case refining the selection. In our user study [15], we show

that even users without prior knowledge of OXPath can create a wrapper in less than three minutes (not counting testing and verification) on average.

Linking. Creating LIMES link specifications can be carried out in manifold ways. For example, LIMES provides active learning algorithms for the semi-automatic detection of link specifications that have been shown to require only a small number of annotations (i.e., $10 - 40$ depending on the data quality) to detect high-quality link specifications [24,21]. Given that we had clear definition of the two predicates `near` (for distances up to 2km) and `atWalkingDistance` (for distances up to 500m) to be computed for the domain at hand, we chose to create link specifications manually for each of these predicates.

Question Answering. The component for parsing a user question and constructing query templates requires only little domain adaptation. The core part of the lexicon that is used for parsing comprises domain-independent expressions that can be re-used, all other entries are built on the fly. The only part that was added for DEQA were lexical entries covering some typical tokens with fixed mappings to URIs in the given domain, e.g. "near". This has been done for six mappings, resulting in 20 domain-specific entries. The required manual effort amounts to less than an hour.

System Maintenance

The frequency to which a wrapper needs to be updated is directly correlated to its robustness. Robustness measures the degree of a wrapper to still select the same nodes after changes on the page. Both [15,12] show that wrappers without robustness consideration have limited lifespan, but Visual OXPath implements a number of techniques to prolong the fitness of its wrappers. In particular, given only a single example, Visual OXPath suggests a list of expressions ranked by robustness of the generated wrapper. We have evaluated the top-ranked suggested wrappers over 26 weeks, showing that they fail only after 26 weeks in contrast to average wrappers that fail in $9 - 12$ weeks. In Oxford real estate, we estimate that wrapper maintenance will involve about one failing wrapper per week.

Linking and Question Answering. The system maintenance for the LIMES link specifications is minimal. If the schema is not altered, the specifications created can simply be reran when the data endpoints are updated. In case of an alteration of the schema, the `PROPERTY` and `METRIC` tags of the specification need to be altered. This is yet a matter of minutes if the schema of both endpoints is known. If the new schema is not known, then the link specification has to be learned anew. Previous work [21] has shown that even on large data sets, learning such a specification requires only about 5 min. For question answering, no regular maintenance effort is usually required. An exception, both for linking and question answering, are schema changes. Such changes can in rare cases invalidate specifications, in which case they have to be altered manually. TBSL is flexible in terms of schema changes as long as entities use appropriate labels or URIs. For instance, in [28] was applied to the DBpedia ontology with hundreds of classes and properties without requiring manual configuration for adapting it to

Table 1. Evaluation results and failures

(a) Evaluation results

number of questions	100
—SPARQL queries created	71
—SPARQL queries returning results	63
—SPARQL queries with correct results	49
—exactly intended SPARQL query	30
—SPARQL queries with incorrect results	14

(b) Failure reasons

failures	
—data coverage	9
—linguistic coverage	18
—POS tagging	2
—other reasons	6

this schema. However, previously manually added domain-specific configuration entries for improving the performance of TBSL may require updates in case of schema changes.

4 Evaluation

The components comprising the DEQA platform have been evaluated in the respective reference articles, in particular [9] for OXPath, [23] for LIMES, and [28] for TBSL. Hence, we are mostly interested in an evaluation of the overall system, as well as specific observation for the Oxford real estate case study. The main benefit of DEQA is to enhance existing search functionality with question answering. Therefore, we evaluate the overall system for the real-estate domain by letting users ask queries and then verifying the results.

First DEQA was instantiated for Oxford real-estate as described in Section 3. The OXPath wrappers, the LIMES specs and the TBSL configuration are all publicly available at `http://aksw.org/projects/DEQA`. Our dataset consists of more than 2400 offers on houses in Oxfordshire, extracted from the 20 most popular real estate agencies in the area. The wrappers extract spatial information from 50% of the agencies, typically extracted from map links. For all properties in our dataset, we extract street address and locality. The full postcode (e.g., OX27PS) is available in 20% of the cases (otherwise only the postcode area, e.g., OX1 for Oxford central is available). 96% of all offers expose directly the price, the remaining 4% are "price on inquiry". Images and textual descriptions are available for all properties, but not all agencies publish the number of bathrooms, bedrooms and reception rooms. These offers are enriched by LIMES with $93,500$ links to near (within 2 kilometres) and $7,500$ links to very near (within 500 metres) spatial objects. The data is also enriched by loading $52,500$ triples from LinkedGeoData describing the linked objects. Domain specific spatial mappings were added to TBSL, e.g. "walking distance" is mapped to "very near".

We asked 5 Oxford residents to provide 20 questions each. They were told to enter questions, which would typically arise when searching for a new flat or house in Oxford. We then checked, whether the questions could be parsed by TBSL, whether they could successfully be converted to a SPARQL query on the underlying data and whether those SPARQL queries are correct.

4.1 Results and Discussion

It turned out that most questions would be impossible to answer by only employing information retrieval on the descriptions of properties in Oxford. Many questions would also not be possible to answer via search forms on the respective real-estate websites, as they only provide basic attributes (price, bedroom numbers), but neither more advanced ones (such as "Edwardian", with garden) nor have a concept of close-by information (such as close to a supermarket). Even if they can be answered there, the coverage would be low as we extracted data using over 20 wrappers. While some questions had similar structures, there is little overlap in general.

The results of our experiment are shown in Tables 1a and 1b. Most questions can be converted successfully to SPARQL queries and many of those are the SPARQL queries intended by users of the system. Hence, DEQA provides significant added value in the real estate domain in Oxford despite the relatively small effort necessary for setting up the system. For the questions, which were not correctly answered, we analysed the reasons for failure and summarise them in Table 1b. If questions were not correctly phrased, such as "house with immediately available", they lead to part-of-speech tagging problems and parse failure. Such issues will be dealt with by integration query cleaning approaches into DEQA. In some cases TBSL could not answer the question because it lacks certain features, e.g. negation such as "not in Marston" or aggregates such as average prices in some area. But since TBSL uses a first order logical representation of the input query internally, those features can be added to the QA engine in the future. Support for some aggregates such as COUNT already exists. In some cases, on the other hand, data was insufficient, e.g. users asking for data that was neither extracted by OXPath nor available through the links to LinkedGeoData, e.g. "house in a corner or end-of-terrace plot". Moreover, some questions contain vague, subjective criteria such as "cheap", "recently" or even "representative", the exact meaning of which heavily depends on the user's reference values. In principle, such predicates could be incorporated in TBSL by mapping them to specific restrictions, e.g. cheap could be mapped to costs for flats less than 800 pounds per month. The extended version of DEQA will be compared with classical retrieval engines to quantify the added value of our approach.

An example of a successful query is "all houses in Abingdon with more than 2 bedrooms":

```
SELECT ?y WHERE {
  ?y a <http://diadem.cs.ox.ac.uk/ontologies/real-estate#House> .
  ?y <http://diadem.cs.ox.ac.uk/ontologies/real-estate#bedrooms> ?y0 .
  ?y <http://www.w3.org/2006/vcard/ns#street-address> ?y1 .
  FILTER(?y0 > 2) .
  FILTER(regex(?y1,'Abingdon','i')) .
}
```

In that case, TBSL first performs a restriction by class ("House"), then it finds the town name "Abingdon" from the street address and it performs a filter on

the number of rooms. Note that many QA systems over structered data rely on purely triple-based representations (e.g. PowerAqua [19]) and therefore fail to include such filters.

Another example is "Edwardian houses close to supermarket for less than 1,000,000 in Oxfordshire", which was translated to the following query:

```
SELECT ?x0 WHERE {
  ?x0 <http://dbpedia.org/property/near> ?y2 .
  ?x0 a <http://diadem.cs.ox.ac.uk/ontologies/real-estate#House> .
  ?v <http://purl.org/goodrelations/v1#includes> ?x0 .
  ?x0 <http://www.w3.org/2006/vcard/ns#street-address> ?y0 .
  ?v <http://diadem.cs.ox.ac.uk/ontologies/real-estate#hasPrice> ?y1 .
  ?y2 a <http://linkedgeodata.org/ontology/Supermarket> .
  ?x0 <http://purl.org/goodrelations/v1#description> ?y .
  FILTER(regex(?y0,'Oxfordshire','i')) .
  FILTER(regex(?y,'Edwardian ','i')) .
  FILTER(?y1 < 1000000) .
}
```

In that case, the links to LinkedGeoData were used by selecting the "near" property as well as by finding the correct class from the LinkedGeoData ontology.

4.2 Performance Evaluation

We conclude this evaluation with a brief look at the system performance, focusing on the resource intensive background extraction and linking, which require several hours compared to seconds for the actual query evaluation. For the real-estate scenario, the TBSL algorithm requires 7 seconds on average for answering a natural language query using a remote triple store as backend. The performance is quite stable even for complex queries, which required at most 10 seconds. So far, the TBSL system has not been heavily optimised in terms of performance, since the research focus was clearly to have a very flexible, robust and accurate algorithm. Performance could be improved, e.g., by using fulltext indices for speeding up NLP tasks and queries.

Extraction. In [9] we show that OXPath 's memory requirements are independent of the number of pages visited: For DEQA, the average execution time of our wrappers amounts to approximately 30 pages/min. As we do not want to overtax the agencies' websites, this rate is high enough to crawl an entire website in few minutes. For OXPath this rate is quite slow, but is rooted in inherent characteristics of the domain: **(1)** Many real estate websites are *unable to serve requests at higher rates,* and **(2)** supply *heavily scripted pages,* containing many images or fancy features like flash galleries. Indeed, the evaluation of OXPath is dominated by the browser initialisation and rendering time [9], amounting to over 80% in the real estate case.

Linking. The runtime of the link discovery depends largely on the amount of data to link. In our use case, fetching all data items for linking from the endpoints

required less than 3 minutes while the link discovery process itself was carried out in 0.6 seconds for discovering the near-by entities and 0.3 seconds for the entities at walking distance.

In summary, the data extraction and linking can be easily done in a few minutes per agency and can be run in parallel for multiple agencies. This allows us to refresh the data at least once per day, without overtaxing the resources of the agencies.

5 Related Work

DEQA is, to the best of our knowledge, the first *comprehensive deep web question answering* system addressing the whole process from data extraction to question answering. In contrast, previous approaches have been limited either with respect to their access to deep web data behind scripted forms [20] by targeting only common-sense, surface web data, or by requiring user action for form navigation (MORPHEUS, [11]). Though "federated" approaches that integrate data from different forms have been considered [17], none has integrated the extracted data with existing background knowledge, limiting the types of questions that can be answered. In the following, we briefly discuss related work for each of DEQA's components to illustrate why we believe this is the right combination.

Web Extraction. To extract the relevant data from the real estate agencies, we can resort essentially to three alternatives in web data information extraction [7], namely traditional information extraction, unsupervised data extraction, or supervised data extraction, with OXPath falling into the last category. *Information extraction* systems, such as [8,2], focus on extraction from plain text which is not suitable for deep web data extraction of product offers, where most of the data is published with rich visual and HTML structure, yielding much higher accuracy than IE systems. *Unsupervised data extraction* [30,13] approaches can use that structure, but remain limited in accuracy mostly due to their inability to distinguish relevant data from noise reliably. Thus, the only choice is a supervised approach. In [9] OXPath and related supervised approaches are discussed at length. In summary, OXPath presents a novel trade-off as a simpler, easier language with extremely high scalability at the cost of more sophisticated data analysis or processing capabilities. As shown in DEQA, such abilities are better suited for post-processing (e.g., through LIMES for linking).

Linking. LIMES [24] offers a complex grammar for link specifications, and relies on a hybrid approach for computing complex link specifications. In contrast to LIMES, which employs lossless approaches, [26] uses a candidate selection approach based on discriminative properties to compute links very efficiently but potentially loses links while doing so. Link Discovery is closely related with record linkage and deduplication [5]. Here, the database community has developed different blocking techniques to address the complexity of brute force comparison [14] and very time-efficient techniques to compute string similarities for record linkage (see e.g., [29]). In recent work, machine learning approaches

have been proposed to discover link specifications. For example [21] combine genetic programming and active learning while [25] learns link specifications in an unsupervised manner.

Question Answering. There is a range of approaches to QA over structured data, for an overview see [18]. Here we discuss TBSL in contrast to two prominent systems to exemplify two opposite key aspects: *PowerAqua* [19], a purely data-driven approach, and *Pythia* [27], which heavily relies on linguistic knowledge. TBSL specifically aims at combining the benefits of a deep linguistic analysis with the flexibility and scalability of approaches focusing on matching natural language questions to RDF triples. This contrasts with *PowerAqua* [19], an open-domain QA system that uses no linguistic knowledge and thus fails on questions containing quantifiers and comparisons, such as the most and more than. *Pythia* [27], on the other hand, is a system that relies on a deep linguistic analysis, yet requires an extensive, manually created domain-specific lexicon.

6 Conclusion

DEQA is a comprehensive framework for *deep web question answering*, which improves existing search functionality by combining web extraction, data integration and enrichment as well as question answering. We argue that recent advances allow the successful implementation of the DEQA framework and consider this to be one of the prime examples for benefits of semantic web and artificial intelligence methods. We instantiate DEQA for the real estate domain in Oxford and show in an evaluation on 100 user queries that DEQA is able to answer a significant percentage correctly. In addition, we provided a cost analysis which describes the setup and maintenance effort for implementing DEQA in a particular domain. All used software components as well as the actual queries and used configuration files are freely available (`http://aksw.org/projects/DEQA`).

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC/ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
2. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: IJCAI, pp. 2670–2676 (2007)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
4. Bizer, C., Schultz, A.: The R2R framework: Publishing and discovering mappings on the web. In: COLD (2010)
5. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. 41(1), 1–41 (2008)
6. Bühmann, L., Lehmann, J.: Universal OWL Axiom Enrichment for Large Knowledge Bases. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 57–71. Springer, Heidelberg (2012)

7. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. IEEE TKDE 18(10), 1411–1428 (2006)
8. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: an experimental study. Artif. Intell. 165, 91–134 (2005)
9. Furche, T., Gottlob, G., Grasso, G., Schallhart, C., Sellers, A.: OXPath: A language for scalable, memory-efficient data extraction from web applications. In: VLDB, pp. 1016–1027 (2011)
10. Gerber, D., Ngonga Ngomo, A.-C.: Bootstrapping the Linked Data Web. In: Proc. of WekEx at ISWC (2011)
11. Grant, C., George, C.P., Gumbs, J.D., Wilson, J.N., Dobbins, P.J.: Morpheus: a deep web question answering system. In: iiWAS, pp. 841–844 (2010)
12. Gulhane, P., Madaan, A., Mehta, R., Ramamirtham, J., Rastogi, R., Satpal, S., Sengamedu, S.H., Tengli, A., Tiwari, C.: Web-scale information extraction with vertex. In: ICDE, pp. 1209–1220 (2011)
13. Kayed, M., Chang, C.H.: FiVaTech: Page-level web data extraction from template pages. IEEE TKDE 22(2), 249–263 (2010)
14. Köpcke, H., Thor, A., Rahm, E.: Comparative evaluation of entity resolution approaches with fever. In: VLDB, pp. 1574–1577 (2009)
15. Kranzdorf, J., Sellers, A., Grasso, G., Schallhart, C., Furche, T.: Spotting the tracks on the OXPath. In: WWW (2012)
16. Lehmann, J., Auer, S., Bhmann, L., Tramp, S.: Class expression learning for ontology engineering. J. of Web Semantics 9, 71–81 (2011)
17. Lin, J.: The Web as a resource for question answering: Perspectives and challenges. In: LREC 2002 (2002)
18. Lopez, V., Uren, V., Sabou, M., Motta, E.: Is question answering fit for the semantic web? A survey. Semantic Web J. 2, 125–155 (2011)
19. Lopez, V., Fernández, M., Motta, E., Stieler, N.: PowerAqua: Supporting users in querying and exploring the Semantic Web content. Semantic Web Journal (2012), http://iospress.metapress.com/content/f2346411255409u5/
20. Mollá, D., Vicedo, J.L.: Question answering in restricted domains: An overview. Comput. Linguist. 33(1), 41–61 (2007)
21. Ngonga Ngomo, A.-C., Lyko, K.: EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 149–163. Springer, Heidelberg (2012)
22. Ngonga Ngomo, A.-C.: A time-efficient hybrid approach to link discovery. In: OM@ISWC (2011)
23. Ngonga Ngomo, A.-C., Auer, S.: A time-efficient approach for large-scale link discovery on the web of data. In: IJCAI (2011)
24. Ngonga Ngomo, A.-C., Lehmann, J., Auer, S., Höffner, K.: Raven – active learning of link specifications. In: OM@ISWC (2011)
25. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised Learning of Link Discovery Configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012)
26. Song, D., Heflin, J.: Automatically Generating Data Linkages Using a Domain-Independent Candidate Selection Approach. 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. 649–664. Springer, Heidelberg (2011)

27. Unger, C., Cimiano, P.: Pythia: Compositional meaning construction for ontology-based question answering on the Semantic Web. In: NLDB (2011)
28. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.C.N., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: WWW, pp. 639–648 (2012)
29. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)
30. Zhai, Y., Liu, B.: Structured Data Extraction from the Web Based on Partial Tree Alignment. IEEE TKDE 18(12), 1614–1628 (2006)