Deriving human-readable labels from SPARQL queries

Conference Paper · September 2011

DOI: 10.1145/2063518.2063535 · Source: DBLP

3 authors:

Basil Ell
Bielefeld University & Oslo University
21 PUBLICATIONS **98** CITATIONS
SEE PROFILE

Denny Vrandečić
Google Inc.
85 PUBLICATIONS **2,365** CITATIONS
SEE PROFILE

Elena Simperl
University of Southampton
280 PUBLICATIONS **2,424** CITATIONS
SEE PROFILE

Deriving Human-readable Labels from SPARQL Queries

Basil Ell, Denny Vrandečić, Elena Simperl
Institute AIFB - Karlsruhe Institute of Technology
Karlsruhe, Germany
basil.ell@kit.edu, denny.vrandecic@kit.edu, elena.simperl@kit.edu

ABSTRACT

Over 80% of entities on the Semantic Web lack a human-readable label. This hampers the ability of any tool that uses linked data to offer a meaningful interface to human users. We argue that methods for deriving human-readable labels are essential in order to allow the usage of the Web of Data. In this paper we explore, implement, and evaluate a method for deriving human-readable labels based on the variable names used in a large corpus of SPARQL queries that we built from a set of log files. We analyze the structure of the SPARQL graph patterns and offer a classification scheme for graph patterns. Based on this classification, we identify graph patterns that allow us to derive useful labels. We also provide an overview over the current usage of SPARQL in the newly built corpus.

Categories and Subject Descriptors

E.2 [**Data Storage Representations**]: Linked representations

General Terms

Theory, Measurement

Keywords

Semantic Web, Labels, SPARQL

1. INTRODUCTION

The Semantic Web is built on the concept of identifying entities and their relations. Entities are identified by Uniform Resource Identifiers (URIs) that enable the identification of both web documents and real-world objects [3]. Whenever a user interacts with an application performing queries on linked data, such as Linked Data browsers [7] (e.g. Sig.ma [15] or Tabulator [4]), the retrieved data needs to be presented in a user-friendly way. This allows the application to be usable also by persons not proficient in Semantic Web technologies. Therefore the properties `rdfs:label`[1] and `rdfs:comment` from the RDF vocabulary may be used to provide a human-readable version of a resource's name besides its URI [5]. However, even though their usage is recommended, for many URIs no human-readable documentation is provided (as shown in Section 2).

Identifiers in programming languages and software systems can be arbitrarily chosen by developers (besides the lexical constraints given by the respective programming language). However, using meaningful identifiers and following naming conventions increases the productivity and quality of the software during software maintenance [6, 12], evolution [8] and program comprehension [6]. In order to compensate for missing labels of URIs we derive labels that are meaningful to users by analyzing how linked data is used. SPARQL is a query language for RDF data that allows specifying queries on linked data. Issuers of these queries may chose meaningful identifiers for the same reasons as in programming languages. Therefore a SPARQL query may contain meaningful identifiers of variables in the context of URIs and by analyzing SPARQL query logs we can observe how users, be it human or non-human agents, interact with linked data. We extract SPARQL queries from the web server log files of two prominent data sources in the LOD cloud, namely *dbpedia.org* and *Semantic Web Dog Food (SWDF)*, and show to which extent meaningful identifiers are used and how labels for URIs can be derived in order to compensate the situation of missing labels.

We begin with a description of the current state of human-readability of linked data available on the Web in Section 2, describe our analysis and findings in Section 3, carry out an evaluation in Section 4, present related work in Section 5, and conclude in Section 6.

2. HUMAN-READABILITY OF THE LOD CLOUD

The Billion Triple Challenge (BTC) 2010 corpus[2] is a dataset consisting of linked data crawled from the web which is stored as *ntriples*. Here, each of the 3,167,799,445 ntriples is a quad constituted by a subject, a predicate, an object, and a context, where the context is the URI of the resource

[1]Throughout this paper we omit prefix definitions for the sake of readability and brevity but use common prefixes where their expansion is known by the service provided at `http://prefix.cc`.

[2]Available at `http://km.aifb.kit.edu/projects/btc-2010/`, (accessed May 2011)

I-SEMANTICS 2011, 7th Int. Conf. on Semantic Systems, Sept. 7-9, 2011, Graz, Austria

property	#occurrences	share
rdfs:label	183,986,380	5.81%
rdfs:comment	174,573,990	5.51%
dc:title	16,919,520	0.53%
foaf:name	6,021,670	0.19%

Table 1: Number of triples with the given URI in the property position in the BTC corpus.

the triple has been crawled from.[3] When ignoring the context, thus reducing the quads to triples, the dataset contains 1,441,499,718[4] distinct triples. We discovered that in this dataset, over 10% of the triples were instantiating labeling information as shown in Table 1. Besides `rdf:label`, the properties `dc:title` and `foaf:name` are used to provide a human-readable label for an entity.

In the corpus, there are 159,177,123 distinct subjects. 7,774,615 distinct URIs have an `rdfs:label`, 3,956,892 distinct URIs have an `rdfs:comment`, 2,938,683 distinct URIs have a `dc:title`, and 1,258,979 distinct URIs have a `foaf:name`. 22,020,910 (13.83%) resources in the subject position have either an `rdfs:label` (6.33%), an `rdfs:comment` (2.49%), a `dc: title` (4.92%), or a `foaf:name` (2.46%). This means that 137,156,213 (86.17%) entities have no value for any of these four properties defined in the corpus, leaving a huge number of entities without a human-readable name.

Applications usually use one of the following options when dealing with the problem of missing human-readable labels:

1. The URI itself is displayed to the user. The URI can be meaningful for some users that do not regard it as noise and that are capable of deriving the meaning from some readable strings in the URI. However, this requires URIs that have been created by following a convention to use meaningful names for URIs.[5] Displaying the URI also often leads to an overly technical feel of the interface.

2. The last part of the URI is used, i.e. the local name of the qualified name. For example for the URI `http://www.example.com/about#bob` the fragment identifier `bob` is used, and for the URI `http://www.example.com/people/alice` the last part of the path is used, i.e. `alice`.

3. A more complex mechanism, as e.g. used in Protégé [14]. Here a user may specify which properties are to be used for the task of labeling.

For example when displaying data available in the linked data cloud for the artist Sidney Bechet using the linked data browser *Sig.ma*, the list of information items for his affiliation contains, amongst other items, the following three items:

- `http://rdf.freebase.com/ns/m.049jnng`
- `http://rdf.freebase.com/ns/m.043j22x`
- `Sidney Bechet and His Orchestra`

For the first two items no human-readable labels are available to Sig.ma, therefore the URI is displayed which does not represent anything meaningful to the user besides the information that Freebase contains information about Sidney Bechet.

To the best of our knowledge no approach exists to systematically guess labels for resources.

3. ANALYSIS

3.1 USEWOD2011 dataset

The USEWOD2011 corpus[6] contains server log files from DBpedia [2] and SWDF [11]. In total this dataset contains 19,770,157 log items for DBpedia and 7,992,850 log items for SWDF. The number of SPARQL queries is 5,159,387 (26.10%) for DBpedia and 2,033,021 (25.44%) for SWDF.

Semantic Web Dog Food is a continuously growing dataset of publications, people and organisations in the Web and Semantic Web area, covering several of the major conferences and workshops, including WWW, ISWC, and ESWC. The logs contain two years of request to the server from about 12/2008 until 12/2010.[7] The dataset consist of SWDF log files from 2008-11-01 to 2010-12-14.

The DBpedia knowledge base has been created by extracting information from Wikipedia, thus covering many domains, and contains 672 million RDF triples. The USEWOD2011 corpus contains the log files from 27 days between 2009 and 2010.

The log files conform to the Apache Combined Log Format[8] with small modifications: for the purpose of anonymization the IP in the IP address field is replaced with 0.0.0.0. To allow location-based analyses, a field with the country code of the original IP is appended to log entry. Moreover a hash of the original IP is appended to allow to distinguish users. The following log entry consists of the blank IP address, request date, abbreviated request string, response code, response size, country code, and the hash of the original IP.

```
0.0.0.0 - - [01/Jul/2009:10:15:44 +0100]
"GET /sparql?query=SELECT... HTTP/1.1"
200 1183 "-" "Java/1.6.0_13" "FR"
"5ee07b08fad8d44d388c6aff91651f1db70e2c23"
```

Some log entries represent received SPARQL SELECT queries [13], such as the following query, taken from the DBpedia logs:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT DISTINCT ?x ?abstract
```

[3]Thus representing 12,57% of the estimated size of the Web of Data, 25,200,042,407 triples according to `http://www4.wiwiss.fu-berlin.de/lodcloud/` (accessed May 2011)

[4]`http://gromgull.net/blog/2010/09/redundancy-in-the-btc2010-data-its-only-1-1b-triples/` (accessed 2011-06-29)

[5]However, `http://www.w3.org/Provider/Style/URI` recommends not to use topic names in a URI since thereby an URI's creator binds herself to some classification that can be subject to change and would therefore require a renaming of the URI which is considered as undesired.

[6]`http://data.semanticweb.org/usewod/2011/`

[7]`http://data.semanticweb.org/usewod/2011/challenge.html`

[8]`http://httpd.apache.org/docs/current/logs.html`

```
WHERE {
?x rdfs:label "Fanfare_Ciocarlia".
  OPTIONAL {
     ?x dbpprop:abstract ?abstract
  }.
}
```

This query request entities that have an `rdfs:label` `Fanfare_Ciocarlia` and, if available, `dbpprop:abstract`'s value for this entity. This query contains two triple patterns. A triple pattern is a triple consisting of subject, predicate, and object where each element can be a variable. In the example the first triple pattern

`?x rdfs:label "Fanfare_Ciocarlia"`

consists of a variable, a URI, and a literal. The second triple pattern

`?x dbpprop:abstract ?abstract`

consists of a variable, a URI, and a variable.

3.2 Preprocessing

From the 19,770,157 (DBpedia) and 7,988,587 (SWDF) lines in the log files, where each log event, such as a SPARQL SELECT event or other HTTP requests, is represented by one line, we were able to extract 5,147,626 (DBpedia) and 2,037,238 (SWDF) SPARQL SELECT queries. Furthermore, for each set of identical queries only one instance is selected. The remaining sets contained 1,212,932 (DBpedia) and 195,641 (SWDF) SPARQL SELECT queries. The queries were parsed using the Perl module RDF::Query::Parser::SPARQL which is available at CPAN.[9] When querying the SPARQL endpoint of DBpedia, providing namespace definitions for common prefixes is unnecessary. However the parser could not successfully process a query in the absence of namespace definitions for prefixes used in the query. For each of the prefixes[10] we added the usual prefix definition where necessary in order to parse these queries successfully.

The result of this phase is a list of 1,212,932 SPARQL SELECT queries for DBpedia and 195,641 queries for SWDF consisting of 2,242,800 and 213,029 triple patterns respectively. In the case of DBpedia 3,933,989 queries (76.44%) were ignored and 705 queries could not be parsed, whereas in the case of SWDF 1,841,472 queries (90.39%) were ignored and from the not-ignored 125 queries could not be parsed.

3.3 Query patterns

Figure 1 (note the logarithmic scale) presents the number of triple patterns per query. Most queries are rather simple with only 1 to 3 triple patterns.

We classify triple patterns into the set of triple pattern classes P where $P := C \times C \times (C \cup L)$, $C = \{R, V, B\}$, R denotes a resource, V denotes a variable, B denotes a blank node, and L denotes the set of literals. For example the triple pattern

`dbpedia:Karlsruhe dbo:populationTotal ?population`

belongs to class RRV. Given this classification, triple pattern classes exist that do not contain a variable, such as RRR.

[9]Version 2.903, `http://search.cpan.org/~gwilliams/RDF-Query-2.903/lib/RDF/Query/Parser/SPARQL.pm`

[10]annotation, cc, cohere, conf, dbo, dbpedia, dbpedia-owl, dbpprop, dc, dcterms, foaf, geo, georss, gr, ical, kuaba, lsdis, mo, nao, nco, nfo, nid3, nie, nmo, opo, owl, rdf, rdfs, rss, scot, sioc, sioct, skos, and vs.

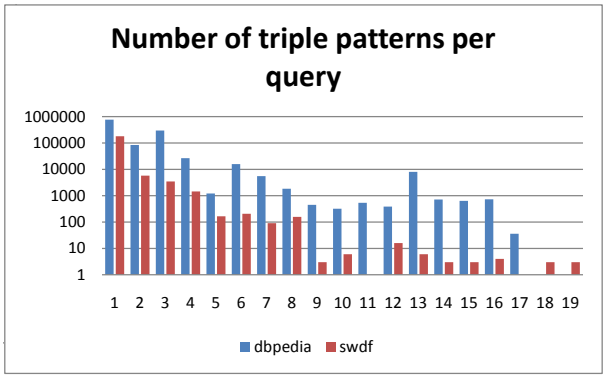

Figure 1: Number of triple patterns per query in dbpedia and SWDF dataset (logarithmic scale).

These are ignored since within the scope of this work we focus on variables.

We analyzed which triple pattern classes constitute SPARQL queries. Figures 2 and 3 present basic graph structures as hypergraphs. For example in the DBpedia dataset we found 11,282 SPARQL SELECT queries that consist of two triple patterns of class RRV and three triple patterns of class VRV. Circular nodes represent hyperedges and rectangular nodes represent triple pattern classes. Each hyperedge is a group of triple pattern classes and contains all triple pattern classes that it connects to via multi-edges. The number of occurrences of each graph structure is denoted by the number in the circular node. In order to focus on the most common patterns in the figures, we pruned the graph to show only those patterns that occur more than 5000 times in case of DBpedia and 1000 times in case of SWDF. Before pruning, the graphs depicted 329 (112) graph patterns for DBpedia (SWDF). It can be seen that most queries (766,662; 63.29%) in DBpedia consist of only one single triple pattern of class RVV, whereas in the SWDF dataset most queries (180,670; 94.11%) consist of one single triple pattern of class RRV. From the graphs it can be derived how often a certain triple pattern occurs at least in the respective dataset. For example the triple pattern RRV occurs at least $4*19102+528582+3*557540+3*22548$ times in the DBpedia dataset. Due to the pruning this number can be smaller than the actual number. The hypergraphs can also be used to predict for a given triple pattern class how likely an instance from another class will co-occur in a SPARQL query. For example, a triple pattern of class VRV more likely occurs with a VRR triple pattern than with a VRL triple pattern.

3.4 Analyzing variable names

We classify variable names as follows:

short A variable name is considered *short* if it has a string length up to 2 chars. Variable names of that type that frequently occur are `s`, `p`, `o`, and `x`.

stop A variable name is considered as a *stop* word if it is not short and does not add information that could be

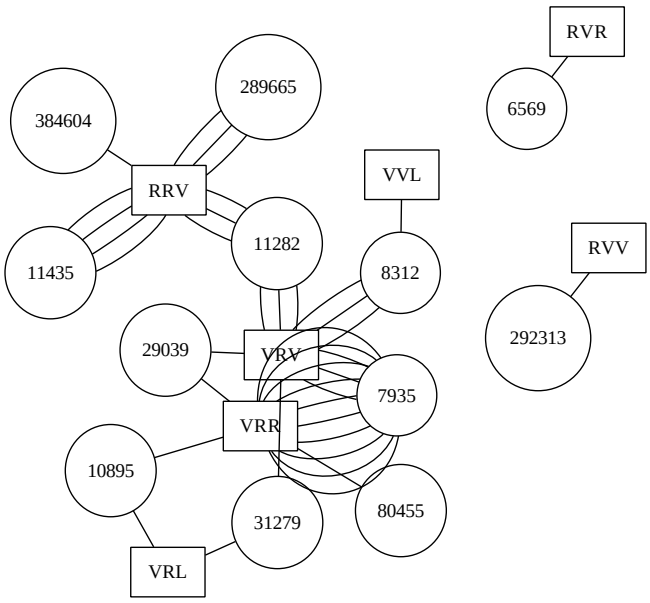

Figure 2: Most frequent query patterns in DBpedia as hypergraph (with more than 5000 occurrences).

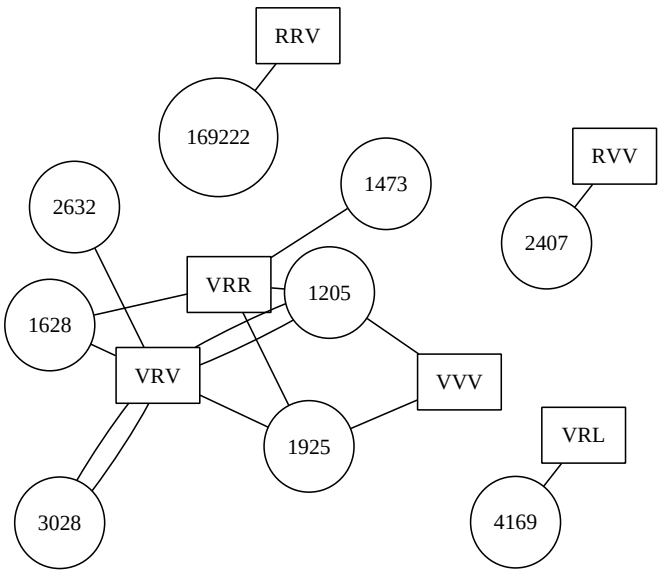

Figure 3: Most frequent query patterns in SWDF as hypergraph (with more than 1000 occurrences).

Subject pos.		Predicate pos.		Object pos.	
resource	158244	property	7592	category	12358
res	61803	pred	134	uri	11775
category	12639	left	56	url	8375
subject	8224	predicate	39	value	6452
instance	3013	prop	27	hasValue	457

Table 2: Most frequent variable names that are stop words in DBpedia logs.

used to describe an URI, such as a variable `subject` in subject position. For each position (subject, predicate, object) we created the lists of stop words manually while exploring the data resulting in three lists containing 31, 25, and 28 words respectively. Table 2 shows the top stop words for each of the three possible positions in a triple pattern and how often they occurred in the data. The complete list of stop words is available online.[11]

lang If a variable name is neither short nor a stop word and if it belongs to a natural language, such as the word `artist`, which belongs (non-exclusively) to the English language, then it is classified as *lang*. Strings that contain the separator char "_" or that are in camelCase are split into its constituents. Each constituent of length ≥ 4 needs to be classified as *lang* in order for the string to be classified as *lang*. For checking which language a word most likely belongs to we use the Corpex[12] webservice. The Corpex dataset consists of all words and their frequencies as extracted and counted from instances of Wikipedia in multiple languages [16]. For each language $l \in \{de, en, es, fr, it\}$ we request the number of exact occurrences in the dataset derived from the respective Wikipedia corpus and normalize this value by dividing it by the total number of words in this corpus for language l. We ignore words for which this results in a value less than $5 * 10^{-7}$, which corresponds to words that occur less than 1000 times in the English Wikipedia. The language for which this score is highest is assumed as the language to which this word belongs to.

nolang Variable names that are not *short*, not *stop*, and not *lang*, are classified as *nolang*.

Figure 4 and Figure 5 show the distribution of classes of variable names for each triple pattern class for the DBpedia and the SWDF datasets.

3.5 Query pattern classes

We distinguish between query patterns and triple patterns, where a query pattern is a set of triple patterns. We ignore anything but triple patterns inside the SPARQL query, e.g. `UNION`, `OPTIONAL`, etc. For each pattern that we describe here we provide an example from the actual data and formulate an assumption about how this triple pattern class can be used to derive labels. An evaluation of these assumptions is presented in Section 4.

In the following, we discuss six of the query pattern classes. The subsequent evaluation will cover all classes, but due to

[11] `http://people.aifb.kit.edu/bel/label/sparql_logs/`
[12] `http://km.aifb.kit.edu/sites/corpex/`

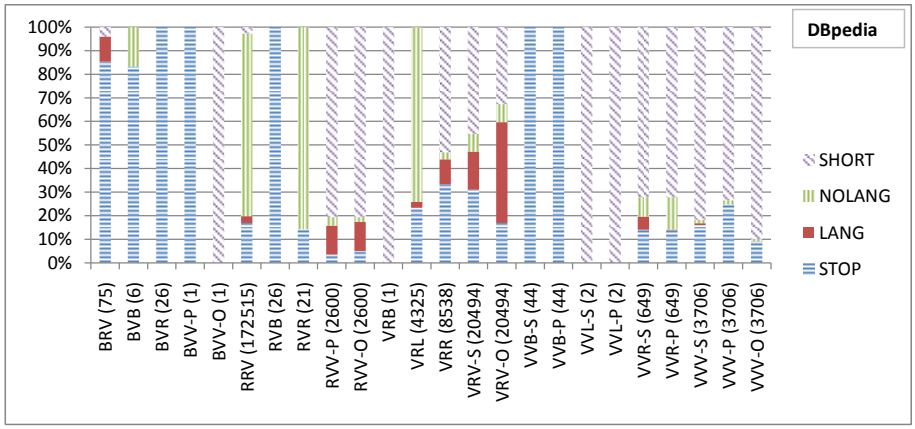

Figure 4: Distribution of classes of variable names for each triple pattern class for DBpedia dataset.

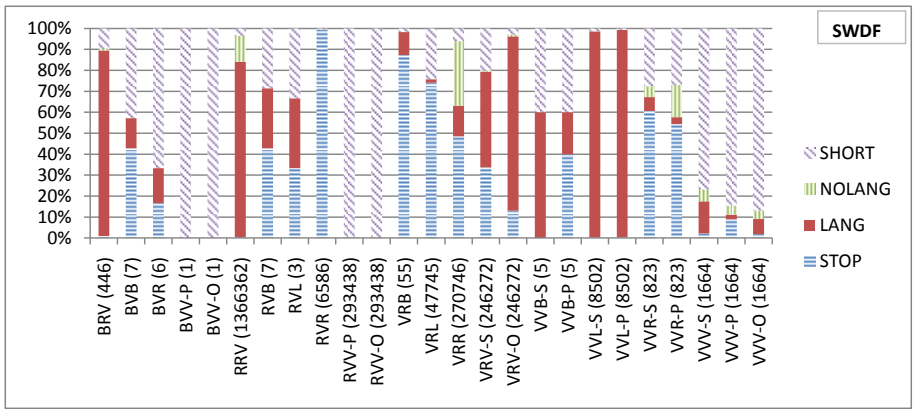

Figure 5: Distribution of classes of variable names for each triple pattern class for SWDF dataset.

space constraints we offer here only the most interesting and frequently occurring classes.

3.5.1 1 × RVV

A query pattern of class 1 × RVV consists of one triple pattern of class RVV which consists of a resource as the subject, a variable as the predicate, and a variable as the object. 292,313 queries (24.13 % of all DBpedia queries) of this class were extracted from the DBpedia dataset, 2,407 queries (1.25 % of all SWDF queries) from the SWDF dataset. For example:

```
dbpedia:Claude_Debussy ?p ?place .
```

Most variable names are either short (99.47% in predicate position, 99.41% in object position) or stop words (0.25% in predicate and object position). Therefore this query pattern is not a fruitful source for deriving labels. However, we found queries containing RVV patterns where additional information about the property is encoded in a filter expression as in the following query:

```
SELECT ?label WHERE {
  <http://data.semanticweb.org/person/daniel-herzig>
    ?label_prop ?label .
    FILTER(REGEX(str(?label_prop),
      "(label|summary|name)$", "i"))
} LIMIT 1
```

A user issuing this query seems to be striving to find something that is expected to be human-readable. The query returns a list of all values for properties that end with `label`, `summary`, or `name`, such as the properties `rdfs:label` or `foaf:name`. In the current approach we ignore this additional information.

3.5.2 1 × RRV

A query pattern of class 1 × RRV consists of one triple pattern of class RRV which consists of a resource as the subject, a resource as the predicate, and a variable as the object. 384,604 queries (31.75% of all DBpedia queries) of this class were extracted from the DBpedia dataset, 169,222 (88.15% of all SWDF queries) from the SWDF dataset. For example:

```
<http://dbpedia.org/page/NASA>
  <http://www.w3.org/2003/01/geo/wgs84_pos#lat>
    ?lat
```

In this example the name of the variable ("lat") in the triple R_1R_2V is equal to the local name (the part behind the # or respectively the last slash) of R_2, as in the following example where the local name is the part behind the last slash:

```
<http://dbpedia.org/page/NASA>
  <http://dbpedia.org/property/agencyName>
    ?agencyName
```

Our assumption is that V' is a meaningful label for R_2 in R_1R_2V iff $eq_{ln}(R_2, V) \wedge lang(V)$ where $eq_{ln}(R_2, V)$ stands for the string equality of the local name of the URI R_2 and the name of the variable V, and $lang(V)$ evaluates to true if the name of the variable V is considered as *lang* thus being a word from a natural language as checked using the Corpex webservice. V' can be derived from V by substitution of separators "_", "+", and "-" to spaces and splitting camelcased words into its constituents.

3.5.3 3 × RRV

A query pattern of class 3 × RRV consists of three triple patterns where each triple pattern is of class RRV and consists of a resource as the subject, a resource as the predicate, and a variable as the object. 289,665 queries (23.91 %) of this class were extracted from the DBpedia dataset. The homogeneity of this graph pattern is high. When ignoring the subjects, which are the same in every triple pattern, then the number of distinct queries is 3. While the structure of each query is the same, the 3 queries differ in the variable's names and properties used. For example:

```
select ?image ?abstract ?redirect where  {
  {
    dbpedia:Tiger_Rag dbo:thumbnail ?image
  } UNION {
    dbpedia:Tiger_Rag rdfs:comment ?abstract
    FILTER ( lang(?abstract) = "en" )
  } UNION {
    dbpedia:Tiger_Rag dbpedia2:redirect ?redirect
  }
}
```

Due to the relatively large number of queries of that class and due to the small size of distinct queries, these queries are expected to be application-specific and not repeatedly issued by an human agent. Due to the small number of unique queries, queries from this class are not promising to be analyzed in order to harvest labels.

3.5.4 1 × VRR, 1 × VRV

A query pattern of class 1 × VRR, 1 × VRV consists of one triple pattern of class VRR and one triple pattern of class VRV. 29,039 queries (2.40 %) of this class were extracted from the DBpedia dataset. For example:

```
?artist
  skos:subject
    dbpedia:Category:People_from_Karagandy_Province .
?artist rdfs:label ?label .
FILTER (lang(?label)='en') .
FILTER (?artist != dbpedia:Aslan_Maskhadov) .
```

87.85% of all queries of class *1 × VRR, 1 × VRV* have a similar structure: the variables in the subject position are equal in both triple patterns, the two variables in the VRV triple pattern are different, the properties in both triple patterns are different, the property in the VRR triple pattern is not `rdfs:type`, the property in the VRV triple pattern is `rdfs:label`, and the variables are `artist` and `label`. An agent seems to be iterating over a set of entities and creating a query for each entity. Thereby the names of the variables are the same for each such query. Due to the homogeneity of queries from this class and due to the small number of distinct variable names used, this class is not promising to be analyzed in order to derive labels.

3.5.5 2 × VRV

A query pattern of class 2 × VRV consists of two triple patterns of class VRV. 3,028 queries (1.58 %) of this class were extracted from the SWDF dataset. For example:

```
?person foaf:name ?name .
OPTIONAL { ?person foaf:mbox ?email }
```

This pattern seems promising at first glance: it seems that we can derive labels for the properties from the variables in the object position. But in 98.22% of such queries, the variable names are either short or stop words.

3.5.6 Any graph with VRR

Finally, we discuss graph patterns that contain at least one VRR triple pattern. For example:

```
?Player dbo:Athlete dbo:SoccerPlayer .
```

This pattern occurs 270,746 times (22.32%) in the DBpedia dataset. The assumption is, that V' is a human-readable label for R_2 in VR_1R_2 iff $lang(V) \wedge R_1 = rdf : type$, as in the following example:

```
?person rdf:type foaf:Person .
```

If the property is not `rdf:type`, then the name of the variable in the subject position can not be used to derive a label. That means that in this example `Person` can be derived as a label for the class `foaf:Person`. The following examples demonstrate that in each case the variable's name could be used for a different purpose:

Case 1
```
?ground skos:subject
        dbpedia:Category:Parks_in_Indiana .
```

Case 2
```
?singer foaf:page
        <http://en.wikipedia.org/wiki/Bob_Dylan> .
```

Case 3
```
?airline dbo:headquarter dbpedia:Germany .
```

Case 4
```
?maceo owl:sameAs dbpedia:Maceo_Parker .
```

The variable name in Case 1 could be used as a label for the resource in object position, the variable name in Case 2 could be used as a label for class of the resource in object position, the variable in Case 3 could be used as a label for a class in the domain of the property, and the variable in Case 4 could be used as the name of the instance in object position.

4. EVALUATION

For the assumptions presented in the previous section we performed the following evaluation: For every tuple consisting of a URI and a label that we derived from analyzing the query log of a data source (the *guessed label*), we query the data source in order to see if a label already exists (via either the `rdfs:label`, `rdfs:comment`, `dc:title`, or `foaf:name` properties). If no given label exists in the data sources, we analyze whether the BTC2010 corpus provides a label. If this also fails, we dereference the URI in order to access their RDF description and see if this RDF description provides a label. We regard this as the *given label*. We then compare the given labels with the guessed labels.

We consider a guessed label as being correct if it is equal to the given label or if the given label is a substring of length ≥ 4 of the guessed label or if the guessed label is a substring of length ≥ 4 of the given label or if the Levenshtein edit distance between given and guessed label is ≤ 4. If a guessed label does not meet these criteria we manually evaluated for a random subset of all those cases whether or not this label is meaningful and appropriate for the resource. Two patterns have been selected for evaluation that were the most interesting and frequently occurring classes.

4.1 1 × RRV

In the DBpedia dataset for all 1,366,362 triples of class RRV we found 549,093 triples where the condition $eq_{ln}(R_2, V)$ holds, and 916,673 triples where the condition $\neg eq_{ln}(R_2, V)$ holds. From these triples we extracted 226 pairs of URIs and guessed labels. We started with an automatic evaluation by comparing the guessed labels with the given labels. This resulted in 54.5% guessed labels being correct. Then we evaluated the remaining guessed labels manually. The overall evaluation resulted in 68.5% being correct, 9.1% being correct within a given context and 22.4% as being wrong. Examples for guessed labels that are considered as being correct in a certain context only are `actor` for the property `dbo:starring` or `location` for the property `dbo:residence`. Examples for guessed labels that are considered as being wrong either because they are stop words that haven't been added to the list of stop words or considered wrong elsewise such as the label `contained in` for the property `dbpprop:creator` for which the `rdfs:label creator` is known.

4.2 Any graph with VRR

80,455 queries in the DBpedia dataset contain a triple pattern of class VRR. For 60 distinct URIs we derived 36 labels for classes – some labels were derived for multiple URIs, such as `river` for both `dbo:River` and `dbpedia:Category:Rivers_of_Spain`. We started with checking the labels automatically. 25% of the guessed labels where automatically confirmed as being correct. The remaining guessed labels were checked manually. Of the labels derived from this class, 53.3% are correct, 46.7% were considered as being wrong. An example for a guessed label considered as being correct is `station` for the class `dbo:RadioStation`. An example for a guessed label considered as being wrong is the label `scientist` for the class `dbo:SoccerPlayer`.

5. RELATED WORK

Until now, query logs have been analyzed with different intentions in the context of the Semantic Web:

- The authors of [1] aim at finding the most frequently used language elements focusing on the most expensive SPARQL operations. These results may assist designers of indices, stores, optimizers, and benchmarks in making reasonable assumptions and taking plausible decisions. They also perform their analysis on the USEWOD2011 corpus.
- While [9] also analyses query logs, the authors are focusing on the *semantic gap* – the gap between the content provided in the Semantic Web, and the information needs as expressed by web users submitting keyword queries to search engines.
- The authors of [10] analyze a similar dataset consisting of access logs and find out whether a human user or a machine agent is consuming linked data. They perform an analysis of the SPARQL queries found in the log files and expect these results to be beneficial when choosing which indexes to pre-compute and store for serving LOD data.

To the best of our knowledge SPARQL queries have not yet been analyzed with a focus on exploiting variable names. Moreover no approach exists to systematically derive labels for resources in the Web of Data.

6. CONCLUSIONS

We presented an approach for automatically deriving labels for entities based on the extensive analysis of a big corpus of SPARQL queries. The methods that we have developed have achieved an acceptable precision as evaluated on the DBpedia and SWDF datasets. That means, most of the labels that we guessed based on the SPARQL queries matched the already given labels.

Regarding the quality of variable names used in SPARQL queries for labels we found out that guessed labels can be less specific than useful, since in the context of a query a variable name may be only as specific as necessary for the purpose of disambiguation. For example in the following query it is sufficient to name the variable `artist` instead of e.g. `MusicalArtist`.

```
SELECT ?artist ?x WHERE {
  ?artist rdf:type dbpedia-owl:MusicalArtist .
  ?x dbprop:influencedBy ?artist .
}
```

We noticed that in most cases the labels were useful for terminological entities, i.e. classes and properties. We have, for now, only regarded the queries by themselves. In future work we expect to analyze them in parallel with the data, especially with the given answers. Thereby we expect to gain further interesting information for deriving labels and to achieve a higher accuracy.

We have published the derived labels on the Web, so that the results can be inspected and actually also used by Web of Data applications.[13]

The regarded datasets (DBpedia and SWDF) provide through the process they are created an almost full coverage with labels for all their entities, which makes them highly atypical as we have seen in our analysis of BTC2010 in Section 2. At the same time, this enables us to thoroughly evaluate our approach since we can use the already given labels, allowing us to automatically evaluate our approach. We expect that the same methods can be applied to other datasets with available SPARQL query logs and where the labels are missing. With the methods presented here we can automatically derive labels for properties and classes in these datasets, thus making an important step towards a higher reusability of the knowledge published on the Web of Data.

7. ACKNOWLEDGMENTS

Part of this work has been carried out in the framework of the German Research Foundation (DFG) project entitled: "Entwicklung einer Virtuellen Forschungsumgebung für die Historische Bildungsforschung mit Semantischer Wiki-Technologie – Semantic MediaWiki for Collaborative Corpora Analysis" (INST 5580/1-1), in the domain of "Scientific Library Services and Information Systems" (LIS).

8. REFERENCES

[1] M. Arias, J. D. Fernández, M. A. Martínez-Prieto, and P. de la Fuente. An Empirical Study of Real-World SPARQL Queries. *CoRR*, abs/1103.5043, 2011. informal publication.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2008.

[3] D. Ayers and M. Völkel. Cool URIs for the Semantic Web. World Wide Web Consortium, Note NOTE-cooluris-20081203, December 2008.

[4] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[5] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 2004.

[6] F. Deissenboeck and M. Pizka. Concise and Consistent Naming. *Software Quality Journal*, 14(3):261–282, Sept. 2006.

[7] T. Heath. How Will We Interact with the Web of Data? *IEEE Internet Computing*, 12:88–91, September 2008.

[8] M. Lehman. Software evolution threat and challenge. Professorial and Jubilee Lecture. 9th international Stevens Award, hosted by ICSM 2003.

[9] P. Mika, E. Meij, and H. Zaragoza. Investigating the Semantic Gap through Query Log Analysis. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 441–455, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] K. Möller, M. Hausenblas, R. Cyganiak, and G. A. Grimnes. Learning from Linked Open Data Usage: Patterns & Metrics. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*, 2010.

[11] K. Möller, T. Heath, S. Handschuh, and J. Domingue. Recipes for Semantic Web Dog Food - The ESWC and ISWC Metadata Projects. pages 802–815. 2008.

[12] T. M. Pigoski. *Practical Software Maintenance: Best Practices for Managing Your Software Investment.* John Wiley & Sons, Inc., New York, NY, USA, 1996.

[13] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C recommentation, W3C, January 2008.

[14] The Protégé Project. Protégé. `http://protege.stanford.edu`, 2010.

[15] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: live views on the web of data. In *WWW*, pages 1301–1304, 2010.

[16] D. Vrandečić, P. Sorg, and R. Studer. Language Resources extracted from Wikipedia. In *Proceedings of International Conference on Knowledge Capture (K-Cap)*. Sheridan Printing, June 2011.

[13] `http://people.aifb.kit.edu/bel/label/sparql_logs/`