Creating voiD Descriptions for Web-scale Data

Christoph Böhm[1], Johannes Lorey[1], Dandy Fenz[2], Eyk Kny[2], Matthias Pohl[2], Felix Naumann[1]

Hasso-Plattner-Institute, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
[1] firstname.lastname@hpi.uni-potsdam.de
[2] firstname.lastname@student.hpi.uni-potsdam.de

Abstract. When working with large amounts of automatically-crawled semantic data as presented in the Billion Triple Challenge (BTC), it is desirable to present the data in a manner best suited for end users. The Vocabulary of Interlinked Data (voiD) has been proposed as a means to annotate sets of RDF resources in order to facilitate human understanding. In this work, we introduce automatic generation of voiD descriptions which comprises different approaches for identifying (sub)datasets and annotating the derived subsets according to the voiD definition. Due to the complexity of the BTC dataset, the algorithms used for partitioning and augmenting the data are implemented in a Cloud environment utilizing the MapReduce paradigm.

1 Introduction

These days open data emerges from a variety of sources, e.g., government agencies, bio-science institutes, social networks, or community-driven knowledge bases. Often, such data is published as linked open data (LOD) – data that adheres to a set of guidelines to allow for easy reuse and semantic integration [5]. However, due to the wealth of information available, descriptive metadata is essential for every open dataset. Furthermore, we think that metainformation *between* distinct datasets represents valuable inter-domain knowledge and therefore provides additional insight.

Metadata is useful in a multitude of scenarios: The most obvious case is when a data engineer searches for information about a specific topic. How does she know what a dataset at hand is about and how can she quickly discover connections to other sources she already works with? A data source should provide this information in a standardized way. Crawling the LOD cloud is a second application. Here, raw statistics (e.g., the number of triples, resources, links, etc.) are of interest for scheduling crawling tasks and provisioning resources. Query answering for linked data is another scenario where statistics and metainformation can support decision making and help achieve better results. We believe that a wide availability of well-defined metadata will facilitate the causes of data interconnectivity and semantic integration. The Vocabulary of Interlinked Datasets (voiD) addresses this need.

VoiD. The Vocabulary of Interlinked Datasets is an RDF-based schema to describe linked datasets [4, 6]. By providing a standardized vocabulary, it aims at facilitating the discovery of linked datasets as well as their usage. VoiD offers two main classes: a `void:Dataset` describes collections of data published and maintained by a single provider. A `void:Linkset` on the other hand is a subclass of `void:Dataset` that describes entities linking to other sources. For linksets, interlinking predicates or link directions can be stored. Additionally, a number of properties defined in voiD are aimed at describing technical or statistical features of datasets.

VoiD has been introduced in 2009 and is already utilized by a number of projects. For instance, the authors of [9] plan to leverage voiD dataset statistics for query optimization. The voiD browser [3] allows to use URIs to search for datasets. These projects require voiD annotations which have been created for a number of sources, e.g., `data.gov.uk` [2] or the OECD Glossary [1]. Though it is considered to be fairly simple to produce voiD descriptions by hand, there are numerous sources in today's LOD cloud that do not provide them[1]. In our opinion, this is due to the manual effort required to create them. Also, we find that often that provided metadata is either incomplete or does not reflect the entire contents of a dataset. That's why we feature a set of algorithms and heuristics to create voiD descriptions automatically.

Structure of this paper. First, we introduce a basic clustering algorithm that outputs dataset information according to the voiD definition (Sec. 2.1). We then illustrate the computation of linksets, using both the original approach and a new *fuzzy* version (Sec. 2.2), followed by a description of dataset metadata generation (Sec. 2.3). Lastly, we propose three new types of datasets and the according clustering algorithms (Sec. 3) before concluding this work (Sec. 4) and discussing the evaluation criteria in the Appendix.

2 Generating voiD annotations

As mentioned above, voiD is centered around the concepts of data- and linksets. Therefore, we first provide a way to compute them in their original form before we discuss metadata generation and possible voiD extensions. Note that while in the next sections we will assume the semantic data at hand to be in a triple format (subject, predicate, object), our approaches also apply to quadruple format (with additional context information) as presented in the BTC 2010 dump.

2.1 Basic Dataset Clustering

The voiD standard associates a dataset with a single publisher, e.g., through a dereferencable HTTP URI or a SPARQL endpoint [6]. Thus, our basic dataset

[1] For example, at the time of writing the prominent DBpedia did not provide a voiD description.

clustering partitions a data corpus using the individual subject's uniform resource identifier if present. According to the W3C RDF format specification, a subject has to be either referenced by a URI or a blank node [8]. In the latter case, the associated triples are ignored for dataset clustering, as there is no publisher associated with them. Malformed URIs are disregarded as well. As the majority of subject URIs adhere to the HTTP scheme, all other schemes each form one individual cluster, such as the cluster of phone numbers (with its schema identifier `tel`). However, for non-HTTP schemes it is also possible to use domain-specific properties to differentiate subsets, e.g., the country code.

For subjects identified by a HTTP URI, two triples belong to the same dataset, iff the subjects' URL start with the same pattern. Here, the length of the pattern is determined by the longest common prefix of all subjects in a dataset ending in one of the characters `:`, `/`, or `#`. For example, the two subjects `http://dbpedia.org/resource/Category:Germany` and `http://dbpedia.org/resource/Tim_Berners-Lee` belong to the same dataset `http://dbpedia.org/resource/`. We will therefore identify a dataset by its URI endpoint, i.e., the `void:uriLookupEndpoint`. The generation of the remaining `void:Dataset` attributes is described in Subsec. 2.3.

Discovering datasets based on the longest common URI prefix requires two MapReduces runs. The first one determines all possible prefixes for every subject. For example, for `http://dbpedia.org/resource/Category:Germany` the prefixes `http://dbpedia.org/`, `http://dbpedia.org/resource/` and `http://dbpedia.org/resource/Category:` are discovered. In the final step, the most suitable prefix for a dataset is determined, i.e., the longest one common to all subjects in the dataset. In the example structure of Fig. 1, datasets identified by this basic clustering approach are identified by the color of the subjects that belong to them. In the BTC 2010 dump, we were able to detect 2,486 `void:Dataset` entities using this approach.

To avoid ambiguity, for the remainder of this paper the original, unpartitioned BTC 2010 dump will also be referred to as 'corpus', whereas a 'dataset' describes any logical subset of the corpus determined by one of the introduced partitioning approaches.

After clustering the data corpus, we calculate different voiD statistics of the newly discovered datasets, such as `void:numberOfPredicates`, `void:numberOfSubjects`, etc. Notice that these attributes each refer to unique entities within a dataset only. These voiD concepts already provide interesting insight into the structure of the datasets: For example, a low number for `void:numberOfPredicates` relative to `void:numberOfSubjects` could hint that the included entities all belong to the same type and thus share most of their attributes. For a heterogeneous dataset, like DBpedia, on the other hand, this relation is very different.

2.2 Linksets

Crisp Linksets. Besides datasets, the voiD standard also introduces the notion of linksets. A `void:Linkset` contains the links from one dataset to another

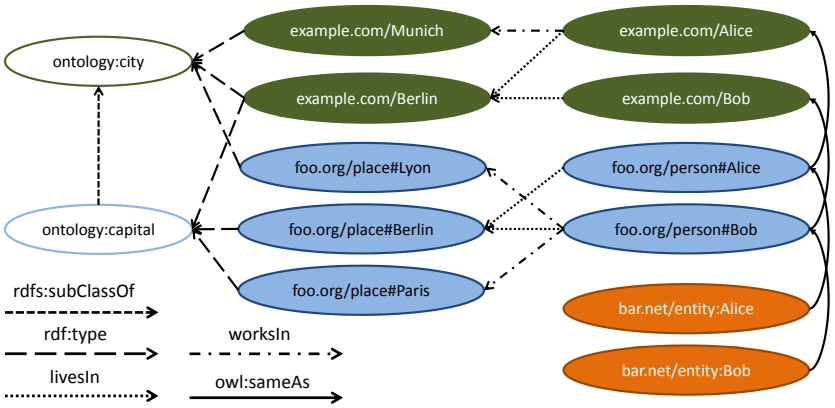

Fig. 1. Running example; filled ellipses indicate resources, arrows represent predicates.

basic	owl:sameAs connected	livesIn connected	transitive ($d = 1$)
example.com/Munich	example.com/Alice	example.com/Berlin	example.com/Munich
example.com/Berlin	foo.org/person#Alice	example.com/Alice	example.com/Berlin
example.com/Alice	bar.net/entity:Alice	example.com/Bob	foo.org/place#Lyon
example.com/Bob			
	example.com/Bob	foo.org/place#Berlin	example.com/Berlin
foo.org/place#Lyon	foo.org/person#Bob	foo.org/person#Alice	foo.org/place#Berlin
foo.org/place#Berlin	bar.net/entity:Bob	foo.org/person#Bob	foo.org/place#Paris
foo.org/place#Paris			
foo.org/person#Alice			
foo.org/person#Bob	remaining:	remaining:	remaining:
	example.com/Munich	example.com/Munich	example.com/Alice
bar.net/entity:Alice	example.com/Berlin	foo.org/place#Lyon	example.com/Bob
bar.net/entity:Bob	foo.org/place#Berlin	foo.org/place#Paris	foo.org/person#Alice
	foo.org/place#Lyon	bar.net/entity:Alice	foo.org/person#Bob
	foo.org/place#Lyon	bar.net/entity:Bob	bar.net/entity:Alice
			bar.net/entity:Bob

Table 1. Different datasets for running example in Fig. 1.

and is identified by triples in which the subject belongs to a different dataset than the object. In our implementation, a linkset may also be reflexive, i.e., it describes the connections within an individual dataset. However, linksets are not symmetric, but rather directed from one dataset to another. For example, we discovered 4,042 links from DBpedia to GeoNames, but 6,956 links in the other direction.

Once datasets have been determined, linksets among them can be obtained in one MapReduce run. In the Map phase, all quadruples in which both subject and object are members of previously identified datasets are extracted. The emitted

tuple then contains the subject and object dataset identifier as key. The Reduce phase, counts all tuples identified by the same key. Overall, we discovered 10,896 `void:Linkset` resources in the BTC 2010 corpus.

Fuzzy Linksets. In addition to these 'crisp' linksets, we also examine links between different datasets that are not explicitly stated. We introduce the notion of k-similarity, where two subjects are k-similar, iff k of their predicate/object combinations are exact matches. Def. 1 provides a formal definition. The intuition is that two subjects are to some degree similar if they share a common set of attributes, and might therefore be relatable, e.g., using one of the methods introduced in [7]. This in turn helps to identify 'fuzzy' linksets among datasets. These fuzzy linksets connect similar entities (and thereby datasets), which are not explicitly referenced by one another.

Definition 1. *For a fixed subject s_1, and a number of associated predicates $p_{1,i}$ with objects $o_{1,i}$, i.e., for the triples*

$$s_1 \ p_{1,1} \ o_{1,1}$$

$$s_1 \ p_{1,2} \ o_{1,2}$$

$$\dots$$

the set $\{(p_{1,1}, o_{1,1}), \dots, (p_{1,n}, o_{1,n})\}$ denotes all of the predicate/object combinations at hand. For $k > 0$, two subjects s_1 and s_2 are k-similar iff

$$|\{(p_{1,1}, o_{1,1}), \dots, (p_{1,n}, o_{1,n})\} \cap \{(p_{2,1}, o_{2,1}), \dots, (p_{2,m}, o_{2,m})\}| = k, \textit{with}$$

$$(p_{i,j}, o_{i,j}) = (p_{k,l}, o_{k,l}) \Leftrightarrow p_{i,j} = p_{k,l} \wedge o_{i,j} = o_{k,l}$$

A number of factors influence the 'interestingness' of fuzzy links. On the one hand, a higher value for k indicates that the two subjects have a high number of predicate/object combinations in common. Here, k (considered absolute or relative) can indicate similarity of subjects and thus a new relationship between the associated entities is disclosed. To illustrate this effect, we analyzed a 10% sample of the BTC 2010 corpus, containing 317 million quadruples. Of these, around 122 million had one predicate/object combination in common with at least one other quadruple. By setting k to 2, we were able to drastically decrease the number of associated quadruples to 25.

On the other hand, some predicate/object combinations appear very often, and are therefore not insightful. In the BTC 2010 corpus for example, the `rdf:type` predicate occurs quite frequently in conjunction with the `rdf:Resource` object, rendering a small value for k unsuitable for our approach and this specific combination. In general, more specific predicates, i.e., predicates that do not appear very often themselves, provided a better lead for detecting dataset similarity. Overall, k-similar linksets can be considered an extension to the `void:Linkset` class, revealing implicit, fuzzy connections between two datasets.

2.3 Dataset Metadata

The voiD standard introduces a number of properties that characterize a dataset. However, some of the properties are meant to be augmented manually by the data provider and cannot be derived automatically, e.g., the license of a dataset or the date of its creation. Hence, we limited ourselves to a subset of properties that can be deduced from the resources within the dataset, but still provide a data consumer with interesting insight.

The attribute `void:exampleResource` provides a link to a representative entity within the dataset. In our approach, we filtered the subject that provided the most predicate/object combinations as a sample resource. Presumably, this entity is described most thoroughly. The `dcterms:description` terms provide a textual, human-readable description of the dataset. We chose to base the textual description on the most common types of the subjects included in the dataset. More specifically, we filtered all resources described by a `rdf:type` predicate and ranked those according to the respective number of occurrences. For example, in DBpedia we discovered 12,151 subjects of type `http://dbpedia.org/ontology/Place`, 11,285 subject of type `http://dbpedia.org/ontology/Person`, etc. Thus, we can conclude that this dataset provides information about places, people, etc. We found that by limiting ourselves to the top ten types discovered, the generated description offers a good overview of the dataset contents.

3 Extending voiD Content

Detecting individual sub-datasets provides interesting insights into the contents of large data corpora. In its original voiD definition, a dataset identifies a set of data provided by a single publisher. We define *semantic datasets*, i.e., partitions of resources that share specific features. Specifically, we provide means to identify *connected* sets of resources or sets of *conceptually* similar resources. Given two such semantic datasets and respective linksets, one could, for instance, observe the connectivity among concepts.

Connected Datasets. Two resources reside within the same connected dataset, iff there is a link of a specific type between them. If the type of the link is undefined, any two connected resources share the same dataset and there are no linksets. In contrast, if the links among resources are fixed to a specific type, e.g., `owl:sameAs` or `livesIn` in Fig. 1, then one can derive meaningful linksets. Tab. 1 lists the datasets for `owl:sameAs` and the custom-defined `livesIn` link, respectively. In the first case, linksets contain `livesIn` and `worksAt` links whereas in the second case linksets contain `owl:sameAs` and `worksAt` links (for simplicity, we here disregard the other two link types). For computing connected datasets, we have implemented a two-phase MapReduce job that first assigns resources to individual clusters and then merges clusters if connected through specific relations.

Conceptual Datasets. Two resources are contained in the same conceptual dataset, iff they are of the same or of similar type. For this, we provide two approaches: The *hierarchical approach* assigns any resource to a dataset representing a concept and all its superconcepts (up to a certain level d). Consider Tab. 1 as an example: for $d = 1$, the first cluster contains all resources of type `ontology:city` whereas the second cluster comprises entities of type `ontology:capital`. By increasing d to 2, the transitive closure of `ontology:capital` is determined, and the aforementioned two clusters are merged into a single one. The number of MapReduce runs for this clustering approach is variable, depending on the number of iterations d allocated for detecting transitive links.

The *distinct approach* selects a single concept type per resource and assigns the resource to the respective dataset. For example, in Fig. 1 `example.com/Berlin` is both a `city` and a `capital`. The previous transitive approach assigns it to both respective clusters for $d = 1$, and forms a single cluster for $d > 1$. The distinct approach on the other hand selects exactly one specific dataset for the assignment. The dataset selection is driven by the concepts' levels of generality. We use the concepts' occurrence rates to find out that level of generality, i.e., we consider frequent concepts to be more general whereas infrequent concepts have a lower level of generality. This requires two MapReduce runs: the first one gathers statistical information about the concepts within a corpus, whereas the second determines the best-fitting dataset allocation based on the desired level of generality.

4 Conclusion

In this work, we have presented a scalable approach for segmenting, annotating, and enriching large corpora of linked data, as presented in the Billion Triple Challenge 2010 dataset. For this purpose, we have administered the current version of the Vocabulary of Interlinked Data as well as introduced a number of new ideas extending the current scope of voiD. We believe that the techniques introduced in this paper simplify the annotation process for Web-scale datasets and therefore encourages providers of such datasets to adopt voiD.

For detailed findings and an analysis of the complete BTC 2010 dataset, we kindly invite the reader to visit our website: `http://www.hpi.uni-potsdam.de/naumann/sites/btc2010/`. There, we also provide the source code for our approaches.

Acknowledgments

The authors thank Matthias Jacob, Martin Linkhorst, and Stefan Wehrmeyer for their contributions to this work.

References

[1] OECD Glossary of Statistical Terms. http://stats.oecd.org/glossary and http://oecd.dataincubator.org, access: 08/2010.

[2] Public Sector Information Catalogues Aggregator. http://bagatelles.ecs.soton.ac.uk/psi/federator/, access: 08/2010.

[3] K. Alexander. voiD Browser. http://kwijibo.talis.com/voiD/, access: 08/2010.

[4] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets: On the Design and Usage of voiD, the "Vocabulary Of Interlinked Datasets". In *WWW Workshop: Linked Data on the Web*, 2009.

[5] T. Berners-Lee. Linked Data Design Issues. http://www.w3.org/DesignIssues/LinkedData.html, 2006. access: 08/2010.

[6] R. Cyganiak, J. Zhao, K. Alexander, and M. Hausenblas. Vocabulary of interlinked datasets. http://semanticweb.org/wiki/VoiD, May 2010. access: 08/2010.

[7] H. Halpin and P. J. Hayes. When owl:sameas isn't the same: An analysis of identity links on the semantic web. In *Workshop on Linked Data on the Web*, April 2010.

[8] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, February 2004. access: 08/2010.

[9] J. Zemánek and S. Schenk. Optimizing sparql queries over disparate rdf data sources through distributed semi-joins. In *International Semantic Web Conference (Posters & Demos)*, 2008.

Appendix: Evaluation Criteria we have met

The techniques introduced in this paper allow for scalable profiling of the Billion Triple Challenge 2010 dataset. The derived information helps users in identifying the contents of the BTC corpus and also establishes interesting relationships between individual partitions of the original BTC dump, such as crisp and fuzzy linksets. One of the goals of our work was to use only resources included in the provided dataset to derive statistical attributes and other information. Hence, our approach does not rely on any external service or data, and can therefore be applied to (amongst other) all BTC datasets, past, present, or future.

We chose to leverage the requirements introduced in the voiD standard as a base for profiling the resources of the BTC corpus. We believe that the voiD concepts help consumers of linked open data in evaluating quality and suitability of such datasets for the respective application. Furthermore, we propose several extensions to the current voiD standard, all of them targeted at offering new perspectives on web-scale LOD.

Our implementation is based on the Apache Hadoop framework. As an execution environment, we chose Amazon's Elastic Compute Cloud (EC2) platform. This enabled us to parse the entire BTC 2010 corpus in relatively short time: Applying basic clustering, finding a suitable sample resource and textual description, and determining linksets for the entire BTC 2010 corpus required less than one hour on 20 High-CPU Extra Large (*c1.xlarge*) EC2 instances. The results and additional documentation can be found at http://www.hpi.uni-potsdam.de/naumann/sites/btc2010/.