Web Semantics: Science, Services and Agents on the World Wide Web 24 (2014) 33–39

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Exploring Linked Data with contextual tag clouds

Xingjian Zhang *, Dezhao Song, Sambhawa Priya, Zachary Daniels, Kelly Reynolds, Jeff Heflin

Department of Computer Science and Engineering, Lehigh University, USA

ARTICLE INFO

Article history:
Received 31 May 2013
Received in revised form
1 November 2013
Accepted 16 December 2013
Available online 30 January 2014

Keywords:
Linked Data
Tag cloud
Semantic data exploration
Scalability

ABSTRACT

In this paper we present the contextual tag cloud system: a novel application that helps users explore a large scale RDF dataset. Unlike folksonomy tags used in most traditional tag clouds, the tags in our system are ontological terms (classes and properties), and a user can construct a context with a set of tags that defines a subset of instances. Then in the contextual tag cloud, the font size of each tag depends on the number of instances that are associated with that tag and all tags in the context. Each contextual tag cloud serves as a summary of the distribution of relevant data, and by changing the context, the user can quickly gain an understanding of patterns in the data. Furthermore, the user can choose to include RDFS taxonomic and/or domain/range entailment in the calculations of tag sizes, thereby understanding the impact of semantics on the data. In this paper, we describe how the system can be used as a query building assistant, a data explorer for casual users, or a diagnosis tool for data providers. To resolve the key challenge of how to scale to Linked Data, we combine a scalable preprocessing approach with a specially-constructed inverted index, use three approaches to prune unnecessary counts for faster online computations, and design a paging and streaming interface. Together, these techniques enable a responsive system that in particular holds a dataset with more than 1.4 billion triples and over 380,000 tags. Via experimentation, we show how much our design choices benefit the responsiveness of our system.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

We present the contextual tag cloud system[1] as an attempt to address the following questions: How can we help casual users explore the Linked Open Data (LOD) cloud? Can we provide a more detailed summary of linkages beyond the LOD cloud diagram?[2] Can we help data providers find potential errors or missing links in a multi-source dataset of mixed quality? When a user wants to design a SPARQL query for an unfamiliar dataset, they must resolve three basic questions: (1) Syntactic Correctness: "What classes are available?" (2) Semantic Correctness: "Does this class refer to the concept I expect?" (3) Meaningful Results: "Does the dataset hold enough knowledge coded with the vocabulary I choose?" Since there are two aspects of a dataset: the ontological terms (classes and properties) and the instances, the questions cannot be answered by only viewing the ontology axioms or only inspecting a small sample of instances. A combined view of both aspects is necessary. Furthermore, there are two types of linkages: ontological alignment and `owl:sameAs` links between instances. The usability of multi-source RDF dataset is largely affected by the erroneous or missing links of both kinds in the dataset. If we can emphasize the unlikely facts, then data providers will have a tool to help them uncover such problems in the dataset.

Our solution is to use tag clouds to display statistical information about the distribution of instances among various ontological terms. A key feature is that each tag cloud is relative to a type consisting of ontological terms that is dynamically defined by the user. In analogy to traditional Web 2.0 tag cloud systems, an instance is like a web document or photo, but is "tagged" with formal ontological classes, as opposed to folksonomies. Tags are then another name for the categories of instances. We extend the expressiveness and treat classes, properties and inverse properties as tags that are assigned to any instances that use these ontological terms in their triples. The font sizes in the tag cloud reflect the number of matching instances for each tag. We allow the user to change their focus on a specific subset of instances in the dataset by specifying a combination of ontological terms as the context on the fly, and then the resulting contextual tag cloud will resize tags to indicate intersection with this context.

With any uncurated dataset, one must maintain a healthy skepticism towards all axioms. Although materialization can lead to many interesting facts, a single erroneous axiom could generate thousands of errors. Rather than attempting to guess which axioms are worthwhile, our system supports multiple levels of inference; and at any time a user can view tag clouds with the same context

* Corresponding author. Tel.: +1 610 758 4235.
E-mail address: xiz307@lehigh.edu (X. Zhang).

1 http://gimli.cse.lehigh.edu:8080/btc/.

2 http://lod-cloud.net/.

1570-8268/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.websem.2013.12.004

under different entailment regimes, which helps users understand the dataset better and helps data providers investigate possible errors in the dataset.

Starting from our initial version of the system [1] that used DBPedia data, we add features and load the entire BTC2012 dataset. This complex dataset contains 1.4 billion triples, from which we extract 198.6M unique instances, and assign more than 380K tags to these instances. This multi-source, large-scale dataset brings us challenges in achieving acceptable runtime performance, affordable preprocessing, and user-interface design. The rest of the paper is organized as follows: we first formally define the concepts and computation problems, and then showcase some use scenarios along with introduction to system functionalities; then we discuss the preprocessing steps, online computation and multi-level inference; after that we provide some experimental results; then we compare with related works; and lastly we conclude.

2. Basic concepts

Given an RDF dataset, an entailment regime R defines what kind of entailment rules will be applied to the explicit triples. In our implementation, we have two specific sets of rules: R_{Sub} for sub/equivalent class/property entailment (rdfs5, rdfs7, rdfs9, rdfs11[3]); and R_{DR} for property domain/range entailment (rdfs2, rdfs3). We also support the combination of these two sets, leading to four distinct entailment regimes $\mathcal{R} = \{\emptyset, R_{Sub}, R_{DR}, R_{Sub} \cup R_{DR}\}$.

Let $\mathcal{I}$ be the set of all the instances, and $\mathcal{T}$ be the set of all possible tags assigned to instances in the dataset. Given R, the function $\text{Tags}_R : \mathcal{I} \to 2^{\mathcal{T}}$ returns all the tags that are assigned to the given instance under R-inference closure. For $i \in \mathcal{I}$ we assign three types of tags: (1) *Class* C, if $\langle i, \texttt{rdf:type}, C\rangle$ is entailed under R. (2) *Property* p, if $\exists j \in \mathcal{I}$, $\langle i, p, j\rangle$ is entailed. (3) *Inverse Property* $p-$, if $\exists j \in \mathcal{I}$, $\langle j, p, i\rangle$ is entailed. Note under monotonic logic, $R_1 \subseteq R_2 \Rightarrow \text{Tags}_{R_1}(i) \subseteq \text{Tags}_{R_2}(i)$. The function $\text{Inst}_R : 2^{\mathcal{T}} \to 2^{\mathcal{I}}$ returns the set of instances that have been assigned the given set of tags. For convenience, we define the frequency of a set of tags T as $f_R(T) = |\text{Inst}_R(T)|$.

Given that we are substituting tags for triples, we can generalize various entailment rules into tag subsumptions. Tag t_1 is a sub tag of tag t_2 if and only if for all sets of assertional triples $\text{Inst}_R(\{t_1\}) \subseteq \text{Inst}_R(\{t_2\})$. Then the domain/range entailment can be turned into sub tag relations. If $\langle p, \texttt{rdfs:domain}, C\rangle$ and $\langle p, \texttt{rdfs:range}, D\rangle$, then p is a sub tag of C and $p-$ is a sub tag of D.

A context is an expression of tags dynamically constructed by a user. In our implementation, we allow intersections of any number of tags or the negation of tags. A *Negation Tag* $\sim t$ is virtually assigned to an instance i, if $t \notin \text{Tags}_R(i)$. Note that the semantics are based on negation-as-failure. We argue that this is the correct semantics for a system where what is not said is sometimes as important as what is said. Thus a context with $\{t_1, \ldots, t_n, \sim s_1, \ldots, \sim s_m\}$ actually defines a subset of instances: $\text{Inst}_R(\{t_1, \ldots, t_n\}) - \bigcup_{x=1,\ldots,m} \text{Inst}_R(\{s_x\})$. For a given context and entailment regime R, the system shows all the tags used by any instance in the subset specified by the context, and the size of each tag reflects the number of instances having this tag within the subset.

For convenience, we omit the subtle details required to process negation tags for the remainder of this paper. This allows us to present a simplified exposition where a context $T \subset \mathcal{T}$ is a set of tags, and the instances specified by the context is $\text{Inst}_R(T)$.

We define a contextual tag cloud, given context $T \in \mathcal{T}$ and entailment regime R, as a list of tags $[t_1, \ldots, t_n]$ with various font sizes $[fs_1, \ldots, fs_n]$ that reflects the instance sizes $[f_R(T \cup \{t_1\}), \ldots, f_R(T \cup \{t_n\})]$. We always map the total number of instances to the max font size, map 1 to the min font size, and for any given tag frequency, we use log functions on it to calculate the font sizes so that the tag cloud shows differences of tags in orders of magnitude.

3. System features and use cases

The initial tag cloud has context $T = \emptyset$ or semantically $T = \texttt{owl:Thing}$, and the tags in the cloud reflect the absolute sizes of instances related to each tag. We put classes and properties into two separate views, so that users will not treat a property called "author" (which may have domain Publication) as a class name by mistake. To emphasize that difference, we also add an icon with "C" or "P" in front of each tag. If a tag is clicked, it will be added to the current context, and then a new tag cloud will be shown for the updated context. A user can add/remove any tags to/from the context, and explore any dynamically defined types of instances. A user can also switch to Instance View to investigate the detailed triples of instances specified by the context.

A user can also change the inference regime, which by default is R_{Sub}, the subsumption inference. Usually we can expect tags to become larger when more inference is introduced. If R entails that a set of tags are equivalent, we choose a canonical tag to group them under. We display a $\equiv$ after the canonical tag to indicate this; clicking it will display the equivalent tags. Also for any tag cloud, we can turn on the negation mode, and then the tag sizes indicate how many instances do not have this tag under the current context and inference level. A negation tag can be also added to the context, which mathematically means the relative complement. For example in Fig. 1, the property tag cloud with context `foaf:Group` and $\sim$`schema:MusicGroup` shows us the common property usages of instances of `foaf:Group` that are not instances of `schema:MusicGroup`.

With the BTC dataset, a challenging problem for UI design is how we can show so many tags in the tag cloud. A straightforward idea is to show tag clouds in pages. To help users locate specific tags in the tag cloud, we initially sort the tags alphabetically by their local names. When the system receives a request (context T and inference R), it will process tags in the same alphabetic order, and then stream out whatever is available for the requested page. If the user chooses to browse tags alphabetically, then the streaming of results is generally able to stay ahead of the user by pre-fetching results for tags on subsequent pages. Instead of browsing, a user can also search for tags by keywords. We index the local name, `rdfs:label` and `rdfs:comment` (if it exists) for each tag to support such keyword search. The retrieved tags will then be shown in the tag cloud sorted by their relevance to the keyword with their frequencies under the current context and inference regime. In addition, we provide sorting by tag frequency as another option, so that users can easily see the most popular tags under the current context and inference. However, we have to wait until all the frequencies are computed to enable this sort option. For some contexts, it can take a few minutes for the overall computation of thousands of pages of results. We show a progress bar of the computation and the estimated time left; and while waiting for frequency sorting to be available, users can still browse by alphabetical order or search with keywords.

We believe our system can be used for multiple purposes. Here we shall briefly describe four scenarios of a user interacting with the BTC dataset.

Choose the right terms for SPARQL. A user wants to build a SPARQL query on lakes, but does not know what classes about lakes are available. Then by starting with a keyword search "lake", the user is presented with a tag cloud with all tags that match the keyword,

[3] http://www.w3.org/TR/rdf-mt/#RDFSRules.

Download English Version:

https://daneshyari.com/en/article/557817

Download Persian Version:

https://daneshyari.com/article/557817

Daneshyari.com