Advanced Search, Visualization and Tagging of Sensor Metadata

Ioannis Paparrizos, Hoyoung Jeung, Karl Aberer

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
{firstname.lastname}@epfl.ch

***Abstract*— As sensors continue to proliferate, the capabilities of effectively querying not only sensor data but also its metadata becomes important in a wide range of applications. This paper demonstrates a search system that utilizes various techniques and tools for querying sensor metadata and visualizing the results. Our system provides an easy-to-use query interface, built upon semantic technologies where users can freely store and query their metadata. Going beyond basic keyword search, the system provides a variety of advanced functionalities tailored for sensor metadata search; ordering search results according to our ranking mechanism based on the PageRank algorithm, recommending pages that contain relevant metadata information to given search conditions, presenting search results using various visualization tools, and offering dynamic hypergraphs and tag clouds of metadata. The system has been running as a real application and its effectiveness has been proved by a number of users.**

I. Introduction

As sensors become increasingly widespread in the modern world, they produce huge volumes of data. Such data includes not only measurement readings but also associated metadata, such as sensor specification, invalid data history, deployment location, current status of sensor etc. Since sensor data is generally processed with its associated matadata, querying sensor metadata becomes important. In addition, users often search particular metadata in order to understand, analyze, and validate associated sensor data.

Unfortunately, most sensor metadata management schemes have neglected the importance of the metadata search [1], [2], and support merely basic search functionalities, e.g., keyword search. Thus, they are unable to effectively capture various attributes of sensor metadata.

In this paper, we present an advanced search system for sensor metadata[1], running over a large-scale real application, i.e., the *Swiss Experiment Platform*[2] [3], [4], where various research institutes share metadata as well as real-time environmental observation data. The key features of the system are briefly highlighted as follows:

- It employs the *Semantic MediaWiki* [5] with an underlying relational database system for maintaining sensor metadata, which provides a user-friendly environment. Our system retrieves search results not only from the underlying relational database, but also from the semantically related web pages (stored as RDF graphs) with respect to metadata and query inputs.
- Search results are sorted according to our ranking mechanism built upon the PageRank algorithm [6]. Furthermore, a recommendation system proposes metadata pages containing relevant information to the query inputs based on semantic properties that are scored high by the PageRank algorithm.
- In addition to basic search options (e.g., keyword, sort by, order by), our system offers a rich body of advanced options for taking search inputs and visualizing the results. These include map-based browsing of metadata pages, real-time bar and pie diagrams, graph representations showing semantic relations and similarities among metadata attributes, and presenting maps when search results are associated with locations.
- User-browsable hypergraphs are dynamically generated based on the linking structure of the metadata pages. These form real-time tag clouds in our search system, which shows the trends of metadata based on linear normalization of terms and graph's cliques.

The demonstration presents an overview of our sensor metadata search system, focusing on describing the above key features.

II. The Metadata Search System

We maintain sensor metadata in the *Sensor Metadata Repository* (SMR) [3], which is established upon *Semantic MediaWiki* [5]. It offers a technique of annotating wiki pages with semantics in the form of (attribute, value)-pairs, modeling any process by meaningfully annotating the entities, and connecting them semantically to each other. Then, queries in the system are processed using a combination of SQL and SPARQL [7] query languages since the sensor metadata information is stored in both a relational database and RDF graphs. Furthermore, we then enrich this search capability by employing several advanced mechanisms and algorithms.

Fig. 1 illustrates the search mechanisms in our system. The *Query Interface* module takes user's inputs for queries within their privileges, since a user may not have a full access to the whole metadata. It then demonstrates the results with various visualization tools. The *Query Management* module is responsible for processing the queries, while taking into

[1] http://www.swiss-experiment.ch/index.php/Special:RunQuery/AdvancedSearchMetadata

[2] http://www.swiss-experiment.ch

account the mapping of RDF schema to database schema. It also connects with several other modules such as Google Maps API, the GraphViz library, the Google Pie and Bar APIs, the HyperGraph API etc. for dynamically visualizing search results in effective formats.

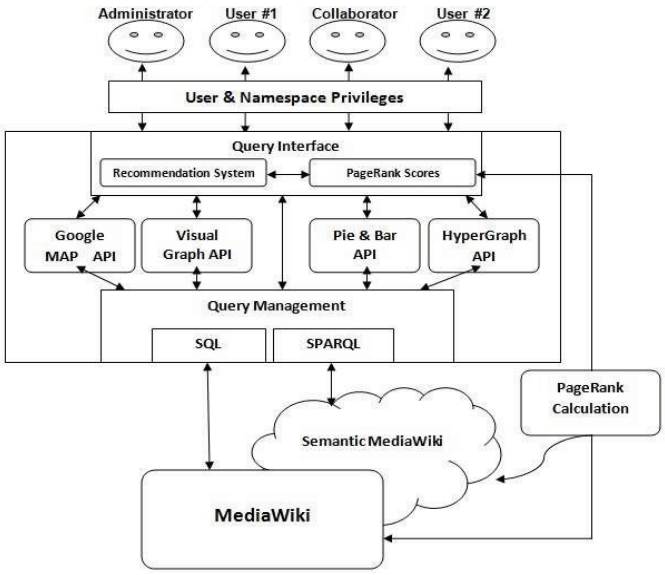


Fig. 1. System architecture.

The system presents search results in various manners, according to the types of query results. Fig. 2 demonstrates some snapshots of the visualizations. In addition to plain tabular formats and bar/pie diagrams, search results that contain positional information can be presented over maps while using different colors for describing the degree of matching of each result with respect to given join predicates. Graph visualization represents the associations (with directed arcs) of sensor metadata in the results as each metadata page may have references in several properties that are different or identical (classification of pages based on similarities of their metadata). Dynamic HyperGraphs allow users to browse pages according to their linking structure and help them identify popular (clustered) pages. Our search system also offers tag clouds that show the trends of either semantic properties or user-generated tags. Details for the dyanamic tagging system are described in Section IV.

The results matched to queries are ranked by our ranking metric described in the following section. In addition, a recommendation mechanism is embedded to our system. This presents relevant pages based on the combination of query inputs and properties that are high-scored by the PageRank algorithm.

III. Ranking Search Results

Every metadata page in our system has two kinds of linking structures: one is the links provided by the RDF graphs and metadata properties, and the other is the normal web-page links from one page to another. We extend the original PageRank algorithm [6] to consider these two links simultaneously

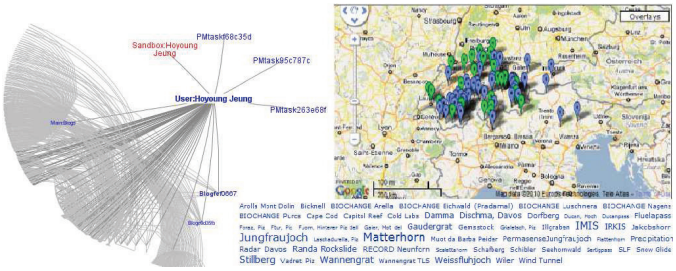


Fig. 2. Snapshots of visualized search results.

for scoring the metadata pages. This is a non-trivial problem as not all of the metadata pages have semantic attributes and thus both linking structures become important to provide an objective ranking.

Pagerank scores need to be updated regularly as new metadata pages are continuously created. Thus, it is necessary to evaluate the convergence and calculation time of several methods that solve the Pagerank algorithm in order to identify which one is most appropriate for our double linking structure.

Pagerank algorithm can be solved either as an *Eigen System* or as a *Linear System*. A web graph adjacency matrix A with elements A_{ij} equal to 1 if there is a link from i to j or equal to 0 otherwise. This can be normalized by setting $P_{ij} = \frac{A_{ij}}{deg(i)}$ where $deg(i)$ is the number of out-links. However, some metadata pages may not have out-links, called dangling nodes, which makes the calculation of Pagerank problematic. One way to overcome this difficulty is to slightly change the transition matrix P to a row-stochastic matrix P':

$$P' = P + du^T \tag{1}$$

where d is the dangling page indicator, and u is some probability distribution over pages (normally $u_i = \frac{1}{n}$). Due to the strong connectivity in the Web graph, a small degree of teleportation has to be added in every page. Thus P' is rewritten as P'':

$$P'' = cP' + (1-c)eu^T, e = (1, ..., 1) \tag{2}$$

where c is a teleportation coefficient. In practice $0.85 \leq c < 1$. After these modifications, matrix P'' becomes row-stochastic and irreducible. Therefore, simple power iterations

$$x(k+1) = (P'')^T x(k) \tag{3}$$

for the eigensystem $(P'')^T x = x$ converge to its principal eigenvector. Now, combining Eq. 1 - 3 we get

$$[cP^T + c(ud^T) + (1-c)(ue^T)]x = x. \tag{4}$$

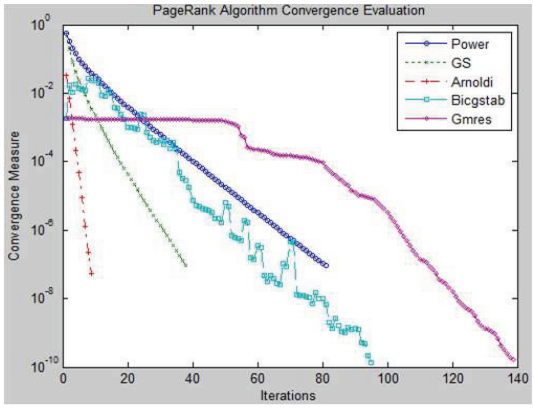
Eq. 4 can be written as a linear system

$$(I - cP^T)x = kv \tag{5}$$

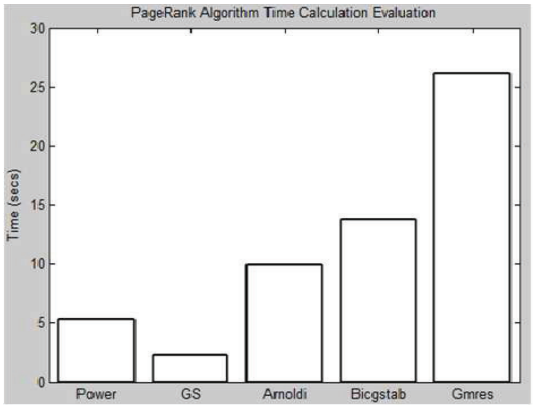
where $k = k(x) = ||x|| - c||P^T x|| = (1-c)||x|| + (d^T x)$. All assumptions taken and full proofs can be found at [8].

To solve the pagerank algorithm we consider several iterative methods [9], [10] - as a direct method incurs high

computational cost, due to the large-size, asymmetric property of the matrix. Specifically, we take a set of Krylov subspace methods to solve the linear system of eq. 5, such as *Generalize Minimum Residual (GMRES)*, *Gauss-Siedel iterations (GS)*, *Arnoldi iterations*, *Biconjugate Gradient Stabilized (BiCGSTAB)* etc. Fig. 3 exhibits the evaluation of these methods. The Gauss-Siedel method outperforms the others with respect to the convergence iterations and computational efficiency. Thus, we use that for the *Pagerank Calculation* module of our advanced search system.



(a) Convergence Evaluation



(b) Time Evaluation

Fig. 3. PageRank Evaluation

IV. Dynamic Tagging

Tag clouds are important in sensor metadata search as they can offer an easy, conceivable way to show the current trends of metadata (e.g. which institutions or universities participate mostly, which is the most popular project, and which deployments are widely used etc.). Our tagging system takes into consideration not only the importance (frequency) of a term but also its semantic meaning.

A. Architecture of Dynamic Tagging System

The tagging system consists of several components illustrated in Fig. 4. The *Interface* module provides the neccesary commands in order to create tags and to accept users' inputs for visualizing tag clouds. The *Parser* module is responsible for connecting to the SMR, exchanging data, fetching and storing tags. A *Cache* mechanism is also implemented to decrease the number of computations and data exchanges. Users are able to create tags in each webpage, describing the topic of it or the metadata. As tags can also be considered the values of metadata properties of the page. The stored tags are given as input to the *Matrix Transformation* module. This module then computes tag matrices based on using the cosine similarity measure (two tags considered similar for a threshold above 50%). Each matrix is considered as a graph in which 1 denotes a link from one tag to another and 0 denotes no linking between tags. This transformation is done in the *Graph* module. The transformed graph is then used as input to the *Max Clique Algorithm* module which calculates all the cliques of the graph and stores the information of clique-tag relationships. Last, the *Font Size Calculation* module computes the font size of each tag depending on a mathematical formula that is described in the following section. The result is then sent to the SMR for visualization. The modularized implementation of the tagging system allows easy modifications on the ranking mechanism (e.g. by replacing the *Max Clique Algorithm* module we can focus on other graph properties or by modifying the *Font Size Calculation* module we can promote tags differently depending on the application).

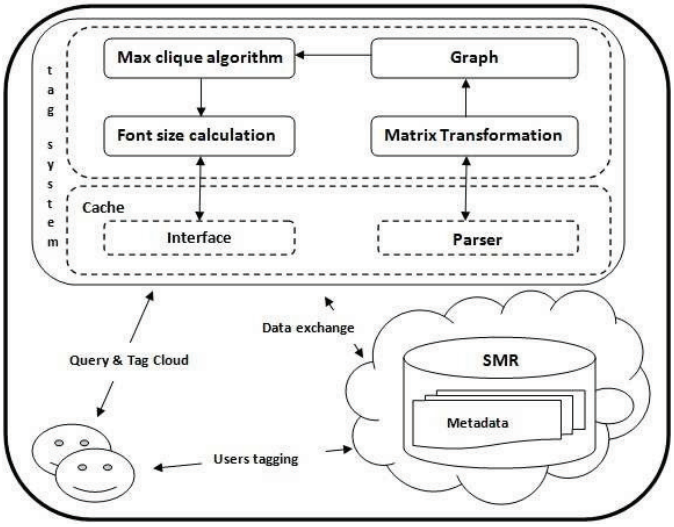


Fig. 4. Architecture of Dynamic Tagging System.

B. Tagging Algorithm

Once all the tags to be shown are selected, the next step is to calculate the font size of each tag based on its incidence (frequency). The frequency of each tag corresponds to the number of entries that are assigned to each page. Specifically, the font size of tag_i is mapped to a size scale of 1 through f, where t_{min} and t_{max} are specifying the range of available font sizes.

We also consider the maximum clique problem. By computing the cliques of a graph with tags, we can promote in

the tag cloud the tags that belong to cliques as well as we can identify the semantics behind the tags. Fig. 5 illustrates the semantics of tag "Apple" which belongs to two cliques. Different colors indicate different cliques.

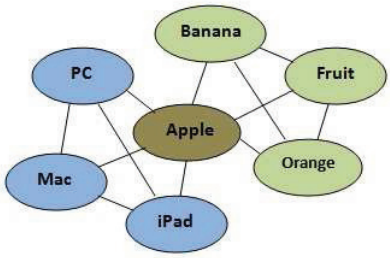


Fig. 5. Semantically important the cliques in tag graphs.

In our system we used the Bron-Kerbosch algorithm[3] for finding maximal cliques in an undirected graph [11] which is frequently reported as being more effiecient than alternatives which in theory are better for inputs with few maximal independent sets [12].

The formula with which the font size of each tag is computed as:

$$s_i = \lceil \frac{c_i * \omega(maxclique_i)}{C} + \frac{f_{max}(t_i - t_{min})}{t_{max} - t_{min}} \rceil \qquad (6)$$

for $t_i > t_{min}$; else $s_i = 1$. In this formula s_i is the fontsize, f_{max} is the maximum fontsize, t_i is the count of tag, c_i is the number of cliques the tag belongs, C is the number of cliques (always ≥ 1), $maxclique_i$ is the maximum clique a tag_i belongs, $\omega(maxclique_i)$ is the order of clique (number of nodes), t_{min} is the minimum frequency and t_{max} the maximum frequency of tags.

V. Demonstration

In the demonstration we first present our *Bulk-loading Interface*[4] where users can upload huge volume of metadata to the SMR[5] that stores all metadata from deployments all over the world (Fig. 6). We explain the metadata schema and we highlight how easily users can register and edit their metadata in the system without any programming.

Then, the easy-to-use advanced search interface (Fig. 7) is presented, covering autocomplete features, drop-down menus that change dynamically based on the chosen properties of schema and map-based browsing of metadata pages. The audience in the demonstration will be able to participate in giving inputs for query, retrieve the corresponding metadata from the SMR, visualize the search results using a variety of tools, such as (clustered) maps, hypergraphs, pie/bar diagrams etc., and browse them. Last, the tagging system will be presented and tag clouds showing the trends in our system will be generated in real time.

[3]The code is based on Katharina Wäschle's implementation which was extended to optimize candidate tag selection and minimize recursion steps

[4]http://www.swiss-experiment.ch/bulkload/bulkload.html

[5]http://www.swiss-experiment.ch/index.php/Fieldsite:Home

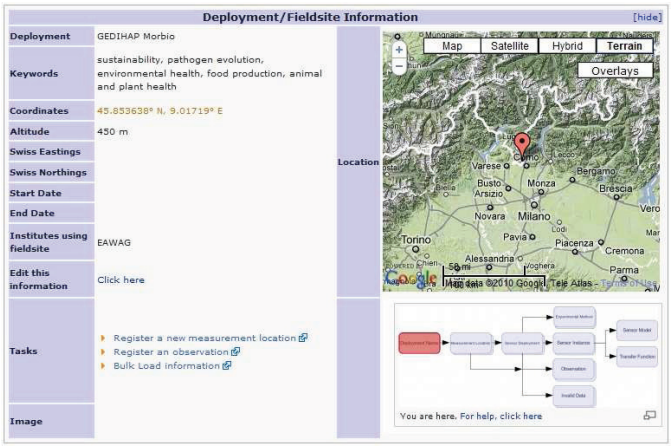


Fig. 6. Snapshots of Sensor Metadata Repository

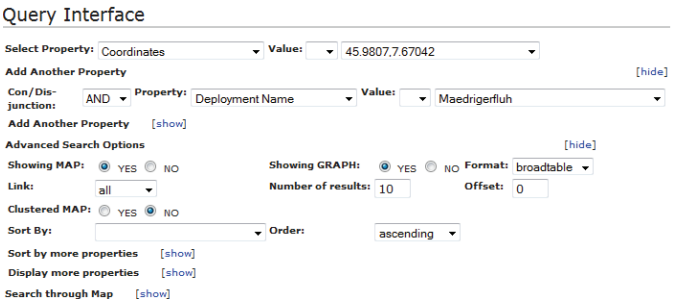


Fig. 7. Snapshots of Query Interface

References

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, M. Cherniack, J. hyon Hwang, W. Lindner, A. S. Maskey, E. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik, "The design of the borealis stream processing engine," in *CIDR*, 2005, pp. 277–289.

[2] M. Balazinska, H. Balakrishnan, and M. Stonebraker, "Load management and high availability in the medusa distributed stream processing system," in *SIGMOD*, 2004, pp. 929–930.

[3] N. Dawes, K. A. Kumar, S. Michel, K. Aberer, and M. Lehning, "Sensor metadata management and its application in collaborative environmental research," in *eScience*, 2008, pp. 143–150.

[4] S. Michel, A. Salehi, L. Luo, N. Dawes, K. Aberer, G. Barrenetxea, M. Bavay, A. Kansal, K. A. Kumar, S. Nath, M. Parlange, S. Tansley, C. van Ingen, F. Zhao, and Y. Zhou, "Environmental monitoring 2.0," in *ICDE*, 2009, pp. 1507–1510.

[5] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *WWW*, 2006, pp. 585–594.

[6] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, 1999, Technical Report.

[7] "SPARQL query language for RDF," W3C Recommendation, World Wide Web Consortium, Technical Report, January 2008.

[8] D. F. Gleich, "Models and algorithms for pagerank sensitivity." Stanford InfoLab, 2009, Ph.D Thesis.

[9] G. H. Golub and C. F. V. Loan, *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[10] O. Axelsson, *Iterative solution methods*. Cambridge University Press, 1994.

[11] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of ACM*, vol. 16(9), pp. 575 – 577, 1973.

[12] F. Cazals and C. Karande, "A note on the problem of reporting maximal cliques," *Theoretical Computer Science*, vol. 407(1), pp. 564 – 568, 2008.