A Query-by-Similarity Indexing Strategy for Web Forms

Willian Ventura Koerich, Ronaldo dos Santos Mello

Universidade Federal de Santa Catarina, Brazil
{willian.vkoerich, ronaldo}@inf.ufsc.br

Abstract. Search engines do not provide specific searches for Web forms related to the Deep Web, in particular, similarity search. To deal with this lack, we propose a query-by-similarity system called WF-Sim, and this paper presents the indexing strategy adopted by WF-Sim for querying-by-similarity Web forms. It is centered on suitable index structures to the main kinds of queries posed on Web forms, as well as some optimizations in order to reduce the number of index entries. To evaluate the indexes performance, we ran experiments on two WF-Sim persistence strategies: file system and database. We also compare the performance of our indexes against the traditional keyword-based index, and the results were promising.

Categories and Subject Descriptors: H.2 [**Database Management**]: Miscellaneous; H.3.1 [**Content Analysis and Indexing**]: Indexing Methods; H.3.3 [**Information Search and Retrieval**]: Miscellaneous

Keywords: Deep Web, Web form, indexing, query-by-similarity

1. INTRODUCTION

The increasing volume of data available on the Deep Web [Madhavan et al. 2009] increases, in turn, the need for accessing this kind of information by users. The problem of finding, integrating and organizing Web forms that serve as entry points to hidden databases has been received substantial attention by the database community [Barbosa and Freire 2007; Barbosa et al. 2007; Fang et al. 2007; Madhavan et al. 2008; Chuang and Chang 2008; Wang et al. 2009; Hong et al. 2010; dos Santos et al. 2012]. However, there is a lack of solutions regarding query support for exploring Web form collections. Some relevant efforts on this direction are *Deque* [Shestakov et al. 2005] and *DeepPeep* [Barbosa et al. 2010], but they do not have support for similarity search. This is a real drawback, considering that Deep Web data sources in a same domain, in particular, data presented in Web form fields, usually are heterogeneous in terms of the definition of their *labels* and *values*. Some few existing work that proposes query-by-similarity over Deep Web data are limited in terms of query capabilities [Qiu et al. 2003; Dong and Halevy 2007].

These limitations had motivated the development of a more efficacy solution called *WF-Sim*. WF-Sim is a system that provides query-by-similarity for Web form data [Goncalves et al. 2011]. Searching at WF-Sim occurs over indexed form fields, which are called *elements*. Index structures are defined for data clusters generated by an element matching process, allowing the retrieval of forms with similar elements. These indexes have been properly designed to facilitate the most frequent types of queries posed on Web forms.

This paper presents the indexing-by-similarity strategy for Web form data developed for WF-Sim, aiming at accessing data hold on two types of persistence schemata supported by the project: file system and database. We evaluate not only the performance of the index structures for each type of persistence, but also which index(es) structure(s) had obtained better performance.

This paper contains more five sections. Section 2 presents the WF-Sim system. Section 3 details the *Indexing* module and the proposed index structures. Section 4 describes an experimental evaluation. Section 5 comments related work and section 6 is dedicated to the conclusion.

2. WF-SIM SYSTEM

Searching methods for Web forms usually performs the exact matching between terms specified in the query input and existing terms on the forms, like labels and values' contents. Aiming at adding query-by-similarity capabilities to Web forms, we had designed the *WF-Sim* system. It is inspired by the assumption that data available in Deep Web are relevant for users who want to find Web forms that meet your needs in terms of online services for several purposes.

WF-Sim is a query-by-similarity processing system for Web forms based on its elements. Given an input query, i.e., a set of elements called *form template*, the system returns a set of Web forms that have similar elements. The system has modules for searching, indexing and clustering. The *Clustering* module was the focus of a previous work [Goncalves et al. 2011]. It applies similarity metrics to group similar elements into clusters. The *Search* module allows the definition of a form template, the submission of the request for elements similar to the template elements (to be searched in the indexes), and the generation of the ranked list of Web forms that are similar to the form template. Details about how these modules work are out of the scope of this paper.

The *Indexing* module is the focus of this paper. It is responsible for creating and updating the index structures, as well as for providing access to them in order to retrieve the relevant forms. Given a form template, the search process for similar forms accesses a synonyms database that directs the form template elements' labels to synonyms elements called *centroids*. A centroid is the element label that represents a cluster of similar elements, as generated by the *Clustering* module. For example, an element with label *"Make"* could be the centroid of a cluster composed by elements with labels *"Make"*, *"Brand"* and *"Select a Make"*. All the centroids are indexed, and each index entry indicates the list of Web forms that have exact or similar elements. Next section details the indexing module and the proposed index structures. In fact, the proposed indexes here are an upgrade of the work of [Goncalves et al. 2011], which had defined a single index structure with a larger index entry composed by an element label and all of its values.

3. INDEXING MODULE

Three index structures have been defined to access Web form data by similarity: *keyword*, *context* and *metadata*. The notion of these structures was borrowed from a previous work [dos Santos Mello et al. 2010] and adapted to the problem of similarity search for Web forms. These structures, in particular, context and metadata indexes, are suitable to the frequent types of queries posed on Web forms.

Context queries search for Web forms based on the contextualization of their fields, i.e., they filter forms that have specific values for a given element label. *Metadata queries* retrieve Web forms that hold a specific metadata content, i.e., a content that acts as a field label or a field value.

Figure 1 shows examples of entries for the developed indexes. The keyword index has entries for any term that may appear in a Web form element, i.e., an existing value or label. The context index defines entries for each combination of a label and one of its allowed values, with a *value###label* format. The metadata index is based on the type of metadata the user is interested in, with entries in the format *term###VALUE* or *term###LABEL*. We prioritize element components whose content variation is higher, like label values in the context index, at the beginning of the index entry string in order to provide an index entry ordering more suitable to the searching. This ordering is particular relevant for inequality or range queries, which require sequential scanning of the index structure[1].

[1]We intend to provide inequality and range queries in a next version of the WF-Sim system.

The synonyms index is an auxiliary structure that allows the searching for similar terms. At label indexing time, a synonym database is queried to check if the label is a synonym of a centroid. If so, a record is inserted in the synonym index containing label name and centroid name. For future queries, if an element label in the template is not found in the indexes, but it is a synonymous of a centroid, it is possible to find out similar forms by searching for synonyms, ensuring, in this way, a similarity search. The synonyms index is important in our indexing strategy because it allows that only labels of centroid elements be indexed in the other index structures, reducing their number of entries.

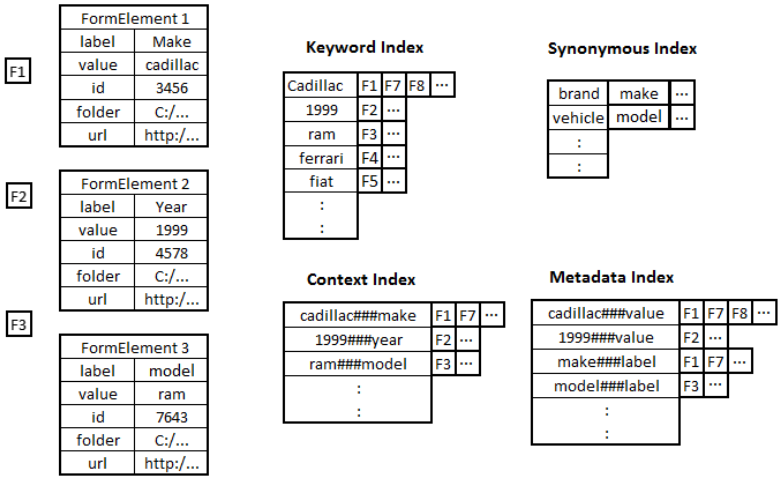

Fig. 1. Indexing structures and considered element information.

Our indexing strategy also supports data cleaning by running a parser that makes term standardization. It converts all letters to lowercase and removes undesirable variations through the process of stemming and removal of stop words. Removal of stop words reduces the size of the indexes and the stemming process reduces the amount of entries, since only the radical terms of desirable element labels are indexed. The parser is also executed during the searching process for similar Web forms. In this case, the template labels are also cleaned before their checking in the indexes.

The access to the indexes is based on the traditional boolean model of information retrieval [Baeza-Yates and Ribeiro-Neto 1999]. In order to illustrate this process, suppose a user wants to find forms that have *GM* brand and a label *year*, i.e., a form template with these two elements' features. The WF-Sim *Search* module then generates the following query: *GM###brand* `AND` *year###LABEL*. Each generated filter is then passed to the *Indexing* module. Considering the filter *GM###brand*, the *Indexing* module recognizes it as a context filter (formed by two terms: a label and a possible value for it), verifies if data cleaning is necessary and then access the index of synonyms. Assuming that the term *"brand"* has only one centroid as a synonym (*"make"*, for example), the filter is converted to "*GM###make*" and the entry corresponding to this filter is accessed in the context index. If there is more than one synonym for *"brand"*, the index entries corresponding to all the synonyms are accessed and it is made a union of the Web form URLs accessed by each entry. The same reasoning applies to the filter *year###LABEL*, and the Web form resulting set of each filter is further processed by the indexing module according to the logical operators defined in the query, being the final Web form result set returned to the Search module.

The index structures were implemented as inverted indexes [Baeza-Yates and Ribeiro-Neto 1999] through the *Lucene* tool [luc 2013]. In the context of this paper, the inverted indexes map a term, for example, a value of an element, to the Web forms containing that particular term. As shown in Figure 1, WF-Sim system stores the following element properties: labels, allowed values, Web

form URL, and a database/file system ID. The ID is the identification of the centroid that holds the element. The indexes point to the location of these elements stored in the file system or database.

File system was the first persistence strategy adopted by the WF-Sim system, being initially conceived for Deep Web domains with a reduced number of Web forms, like *Biology*. It maintains a data file for each cluster of similar elements, being the centroid ID the file name (the file system ID). A data file holds a serialized set of element records, and each record stores the element properties previously described. A relational database was further adopted to keep the element clusters. The database schema is basically composed by a *cluster*, *element* and *value* tables. The element table contains an ID (database ID), a label, a Web form URL and a reference to a cluster. The value table basically holds a description and a reference to an element. The cluster table maintains the cluster ID and the data settings used to configure the similarity metrics that generate the cluster.

Besides *Lucene*, we use *WordNet* [wor 2013] as the basis for the creation of the synonym database. It was used to determine the similarity between elements as well as to build the synonyms index.

4. EXPERIMENTAL EVALUATION

Our indexing strategy was evaluated through preliminary experiments that measure the query processing performance accessing the proposed index structures. Two versions of the indices were created, aiming at accessing Web form elements stored in the file system and database. The experiments were conducted on a computer desktop with an *Athlon II X2 2.9 GHz*, 4 GB memory, 500 GB hard drive, operating system *Windows 7 Home Basic* SP1 64-bit, *MySQL* database, and a sample with 1090 Web forms and 6157 elements.

Figure 2 shows the average runtime (in miliseconds) of the same set of query filters, specific for each kind of index, accessing file system or the database. Figure 3 shows the results of the experiments with an incremental number of query predicates in the *Auto* domain, i.e., filters over one, two and five elements in order to access the context index. We focus on context index because it maintains the query filters most commonly used in Web form querying. In all tests, we ran each query 10 times and got the average processing time.

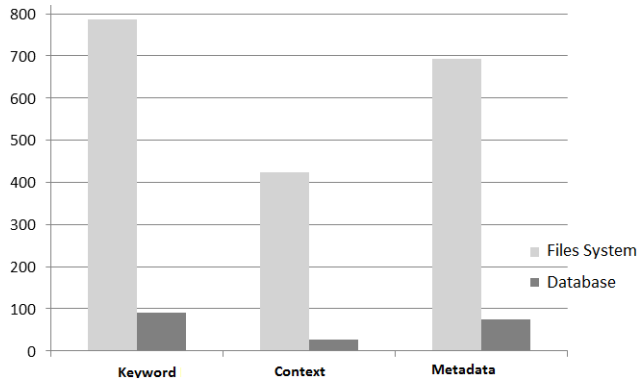

Fig. 2. Average processing time of queries that access the index structures for data in file system and database.

Figure 2 highlights a large difference between file system and database performance. In fact, the execution of the queries spent much more time (8 times slower on average) to access indexed data in the file system. This behavior is due to the fact that file systems do not provide efficient access optimization strategies like database management systems do.

An important contribution of this work, in the context of queries over Web forms, is the reduced processing time for queries accessing context and metadata indexes if compared to the performance

using traditional keyword index, as also shown in Figure 2. This better performance is justified by the fact that, during indexes creation, data about elements are analyzed and filters by context and metadata are automatically defined and stored in the corresponding indices. These two structures are very suitable for further queries over Web forms since they ensure a simplified mapping from a template to these indexes'entries.

With respect to the tests shown in Figure 3, we note that both database and file system query processing time increase were proportional to the increase in the number of conjunctive filters. However, the processing time increase was smaller for the file system, including a processing time decrease from 2 to 5 query filters. This situation needs to be better evaluated through new experiments over larger volumes of data. Even so, processing time difference between file system and database had remained very high, being database access 10 to 30 times faster than file system access for conjunctive filters. Also, we have only a linear increase in database processing time with respect to the number of filters, demonstrating a good index performance.

CONTEXT	# WF	Files system Time	Database Time
jeep###make	355	381 ms	11 ms
corolla###model	155	118 ms	10 ms
jeep###make AND corolla###model	86	486 ms	19 ms
jeep###make OR corolla###model	424	484 ms	21 ms
jeep###make AND NOT corolla###model	269	484 ms	20 ms
acura###make AND audi###make AND chevrolet###make AND 1000###price AND maverick###vehicle"	0	459 ms	36 ms
acura###make OR audi###make OR bentley###make OR 1964###year OR 1999###year	464	490 ms	49 ms
acura###make OR audi###make AND bentley###make OR 1964###year AND NOT 1999###year	355	488 ms	53 ms

Fig. 3. Performance of a set of query predicates that requires access to the context index.

5. RELATED WORK

As far as we know, indexing solutions for Web form query-by-similarity is a research topic not so much addressed in the literature. The closest related work is the approach of [Qiu et al. 2003], which also extracts and stores hidden database information in a relational database. The database maintains a table for each Web form field that holds its values and pointers to clusters of similar values. We see several limitations in comparison to our proposal: *(i)* A great number of indexes must be created if we want to index values of several fields; *(ii)* They assume that a schema matching for Web forms in a same domain was previously accomplished to generate a relational global schema using a mediation system. Such a support increases the complexity of the approach; *(iii)* They are able to answer only queries over field values. Our approach has a broader scope, being able to query field labels by similarity (and their values), as well as terms that acts as a specific metadata for a given Web form.

Other relevant approach is the work of [Dong and Halevy 2007], which deals with the more general problem of indexing dataspaces and supports only field-value (context) queries. Our approach gives additional index support to query form metadata.

6. CONCLUSION

This paper presents and evaluates an indexing strategy for supporting Web form query-by-similarity in the context of the WF-Sim system. This strategy considers the indexing of element data on suitable index structures called metadata and context, besides the traditional keyword indexing. The first two structures provide access optimizations since they index, in a straightforward way, the more frequent query filters over Web forms. If a keyword index would be used for processing a context search (a

"label = value" predicate), for example, we would have to get the Web form result set that contain the term corresponding to the label, then the Web form result set containing the desired value, and compute an intersection of the two result sets. This overhead is unnecessary with the introduction of the context index. The same reasoning holds for metadata search and the metadata index.

Few related work in the literature define indexing structures for similarity search over Web forms, and the existing solutions have limitations in terms of query capabilities and/or seems to be not so effective. Our experimental evaluation has shown the good performance of our specific index structures for Web form data, being the main contribution of this paper. Even so, we intend to compare our solution with related work in terms of processing time, if we have access to their source codes.

Besides that, we intend to evaluate performance for a larger sample of Web forms. Through a partnership with New York University, a sample of approximately 40,000 forms in several domains will be soon available for new experiments. Another future work is to consider searching for Web forms that hold dependencies between fields, such as a field *"Make"* whose values determine the values of a field *"Model"*, assuming an *Auto* domain. The intention is to consider filters that test the existence of such dependencies and define indexing structures that facilitate queries-by-similarity. This kind of query is relevant if the user wants to access Web forms that have a chain of dependencies between fields.

REFERENCES

Apache Lucene. http://lucene.apache.org/core/, 2013.

Wordnet. http://wordnet.princeton.edu/, 2013.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

Barbosa, L. and Freire, J. An adaptive crawler for locating hidden web entry points. In *Proceedings of the International World Wide Web Conferences*. Banff, Canada, pp. 441–450, 2007.

Barbosa, L., Freire, J., and da Silva, A. S. Organizing hidden-web databases by clustering visible web documents. In *Proceedings of the IEEE International Conference on Data Engineering*. Istambul, Turkey, pp. 326–335, 2007.

Barbosa, L., Nguyen, H., Nguyen, T. H., Pinnamaneni, R., and Freire, J. Creating and exploring web form repositories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data Conference*. Indianapolis, IN, USA, pp. 1175–1178, 2010.

Chuang, S.-L. and Chang, K. C.-C. Integrating web query results: Holistic schema matching. In *Proceedings of the International Conference on Information and Knowledge Engineering*. Napa Valley, CA, USA, pp. 33–42, 2008.

Dong, X. and Halevy, A. Y. Indexing dataspaces. In *SIGMOD Conference*. Beijing, China, pp. 43–54, 2007.

dos Santos, L. B., Dorneles, C. F., and dos Santos Mello, R. An approach for extracting web form labels based on distance analysis of html components. In *Proceedings of IADIS International Conference WWW/Internet*. Madrid, Spain, 2012.

dos Santos Mello, R., Pinnamaneni, R., and Freire, J. Indexing web form constraints. *Journal of Information and Data Management* 1 (3): 343–358, 2010.

Fang, W., Cui, Z., and Zhao, P. Ontology-based focused crawling of deep web sources. In *Proceedings of International Conference on Knowledge Science, Engineering and Management*. Melbourne, Australia, pp. 514–519, 2007.

Goncalves, R., Dagostini, C. S., Silva, F. R., Dorneles, C. F., and dos Santos Mello, R. A similarity search method for web forms. In *Proceedings of IADIS International Conference WWW/Internet*. Rio de Janeiro, RJ, Brazil, 2011.

Hong, J., He, Z., and Bell, D. A. An evidential approach to query interface matching on the deep web. *Information Systems* 35 (2): 140–148, 2010.

Madhavan, J., Afanasiev, L., Antova, L., and Halevy, A. Y. Harnessing the deep web: Present and future. In *Proceedings of International Biennial Conference on Innovative Data Systems Research*. Asilomar, CA, USA, 2009.

Madhavan, J., Ko, D., Kot, L., Ganapathy, V., Rasmussen, A., and Halevy, A. Y. Google's deep web crawl. *Proceedings of the VLDB Endowment* 1 (2): 1241–1252, 2008.

Qiu, J., Shao, F., Zatsman, M., and Shanmugasundaram, J. Index structures for querying the deep web. In *Proceedings of the Workshop on Web and Databases*. San Diego, CA, USA, pp. 79–86, 2003.

Shestakov, D., Bhowmick, S. S., and Lim, E.-P. Deque: Querying the deep web. *Data & Knowledge Engineeering* 52 (3): 273–311, 2005.

Wang, Y., Lu, J., and Chen, J. Crawling deep web using a new set covering algorithm. In *Proceedings of International Conference on Advanced Data Mining and Applications*. Beijing, China, pp. 326–337, 2009.