WikiAnalytics: Ad-hoc Querying of Highly Heterogeneous Structured Data

Andrey Balmin #1, Emiran Curtmola *2

#IBM Almaden Research Center, USA
1abalmin@us.ibm.com

*UC San Diego, USA
2ecurtmola@cs.ucsd.edu

***Abstract*— Searching and extracting meaningful information out of highly heterogeneous datasets is a hot topic that received a lot of attention. However, the existing solutions are based on either rigid complex query languages (e.g., SQL, XQuery/XPath) which are hard to use without full schema knowledge, without an expert user, and which require up-front data integration. At the other extreme, existing solutions employ keyword search queries over relational databases [3], [1], [10], [9], [2], [11] as well as over semistructured data [6], [12], [17], [15] which are too imprecise to specify exactly the user's intent [16].**

To address these limitations, we propose an alternative search paradigm in order to derive tables of precise and complete results from a very sparse set of heterogeneous records. Our approach allows users to disambiguate search results by navigation along conceptual dimensions that describe the records. Therefore, we cluster documents based on fields and values that contain the query keywords. We build a universal navigational lattice (`UNL`) over all such discovered clusters. Conceptually, the `UNL` encodes all possible ways to group the documents in the data corpus based on where the keywords hit.

We describe, WIKIANALYTICS, a system that facilitates data extraction from the Wikipedia infobox collection. WIKIANALYTICS provides a dynamic and intuitive interface that lets the average user explore the search results and construct homogeneous structured tables, which can be further queried and mashed up (e.g., filtered and aggregated) using the conventional tools.

I. MOTIVATION

Growing popularity of Wikipedia and other wikis raises the issue of querying this data to extract insights that span multiple pages. Although most of Wikipedia is free text, it also contains a large amount of structured information in tables, list, categories, and infoboxes. A number of ongoing efforts [7], [5], [4], [13] aim to harness this information.

We focus on querying Wikipedia infoboxes, which are essentially typed records of field-value pairs. Infoboxes appear on over a million Wikipedia pages and often contain the most vital information about the entity described by the page. For example, an infobox on Arnold Schwarzenegger's page (Figure 1) contains information about his office, family, birthday, party and religious affiliation, and more.

A major challenge in querying infoboxes is the diversity of their structure. Every infobox instance has an equivalent of a type – wiki template that renders the infobox wikitext into HTML. However, new templates can be introduced, and old templates can be extended relatively easily. Moreover, enabling query processing was never a requirement for the authors of templates and infoboxes. As a result, templates often allow for many ways of representing the same information. For example, a very popular "officeholder" template has both `date of birth` and `birthdate` fields. Figure 3 conveys the heterogeneity of the infoboxes. There are about $2,500$ distinct infobox types (templates), with over $50,000$ distinct <type, field> pairs. However, there is a clear long tail in the distribution of the number of occurrences of the fields, with almost $20,000$ fields occurring in exactly one infobox and only 300 fields occurring in over $4,000$.

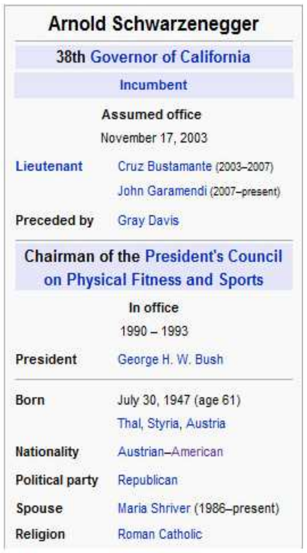
Many other types of data, such as product catalogs and electronic forms collections, exhibit similar structural diversity. These sources are also often designed for human consumption with structural flexibility as the key feature and query processing as an afterthought. As a result, many products in a catalog may have rare or unique fields, and most fields on any given form may be optional or filled with different information by different users.

Such structural diversity presents major problems when queries need to access many objects (infoboxes) in order to extract lists of results. For example, if a user wants to construct a list of all Governors of California, a good heuristic may be to look for infoboxes with type `governor` and `office` field with value "Governor of California." However, this constructed list will be only 90% correct. The list will miss four former governors, including Ronald Reagan whose infobox type is `president` with value "33rd Governor of California" hidden in the `order2` field. Other missing governors are Hiram Johnson and Jerry Brown, for whom the value "Governor of California" occurs in `office2` and `office3` fields, respectively. We call such results *structural outliers*. They are critical for deriving a complete and precise answer.

II. WIKIANALYTICS APPROACH AND DEMO SCENARIO

It is hard to imagine *a priori* reliable integration of information from all large clusters and outliers for the entire dataset - either heuristic or manual. Instead, we adopt a "pay as you go" approach, where only the objects potentially relevant to the result are interactively integrated at query time.

Our system, WIKIANALYTICS, provides multiple clusterings of all potential results, based on the names and values of fields that contain the query keywords. We call such fields and their values *features*. Conceptually, the features define



Arnold Schwarzenegger

38th Governor of California

Incumbent

Assumed office

November 17, 2003

Lieutenant	Cruz Bustamante (2003–2007)
	John Garamendi (2007–present)
Preceded by	Gray Davis

Chairman of the President's Council on Physical Fitness and Sports

In office

1990 – 1993

President	George H. W. Bush
Born	July 30, 1947 (age 61)
	Thal, Styria, Austria
Nationality	Austrian–American
Political party	Republican
Spouse	Maria Shriver (1986–present)
Religion	Roman Catholic

(a) Infobox visualization in Wikipedia.

```
{{ Infobox Governor
| name = Arnold Schwarzenegger
| nick = Governator
| image = Arnold Schwarzenegger 2004-01-30.jpg
| imagesize = 200px
| order = 38th
| office = Governor of California
| term_start = November 17, 2003
| lieutenant = {{nowrap|[[Cruz Bustamante]]<small> (2003-
  -2007)</small>}}<br/>{{nowrap|[[John Garamendi]]
  <small>(2007-present)</small>}}
| predecessor = [[Gray Davis]]
| successor =
| order2 = Chairman of the [[President's Council on
  Physical Fitness and Sports]]
| term_start2 = 1990
| term_end2 = 1993
| president2 = [[George H. W. Bush]]
| birth_date = {{birth date and age|1947|07|30}}
| birth_place = [[Thal, Austria|Thal]], [[Styria]], [[Austria]]
| nationality = [[Austria]][[United States|American]]
| party = [[Republican Party (United States)|Republican]]
| spouse = {{nowrap|[[Maria Shriver]] (1986-present)}}
| religion = [[Roman Catholic]]
...}}
```

(b) WikiText source code that generates Figure 1a.

```
<infobox type="Governor">
  <field>
    <field_name>name</field_name>
    <field_value>Arnold Schwarzenegger
    </field_value></field>
  <field>
    <field_name>nick</field_name>
    <field_value>Governator
    </field_value></field>
  <field>
    <field_name>order</field_name>
    <field_value>38th</field_value>
  </field>
  <field>
    <field_name>office</field_name>
    <field_value>Governor of California
    </field_value></field>
  ...
  <field>
    <field_name>religion</field_name>
    <field_value> Roman Catholic
    </field_value></field>
  ...
</infobox>
```

(c) Infobox as an XML document.

Fig. 1. Sample Wikipedia Infobox: "Arnold Schwarzenegger"

the relevant dimensions on the data specifying the matching context for the query keywords.

WIKIANALYTICS heuristically finds large clusters that are likely to contain the results. It also enables users to easily identify the outliers by exploring and interacting with all the clusters. This allows users to disambiguate the query based on the structure of the results. The intuition is that occurrence of the same keyword in different fields or in different field values is likely to have different meanings. For example, a group of `governor` infoboxes with "California" in the `office` field, which we denote as the $``California'' \in office$ feature, is semantically different from a group where the same keyword occurs in the `birthplace` field. Furthermore, even within the $``California'' \in office$ cluster, it is important to differentiate the infoboxes based on the actual textual values that contain keywords. For instance, there is a significant semantic difference between infoboxes with values "Governor of California" in the `office` field and "Governor of Baja California" in the same field.

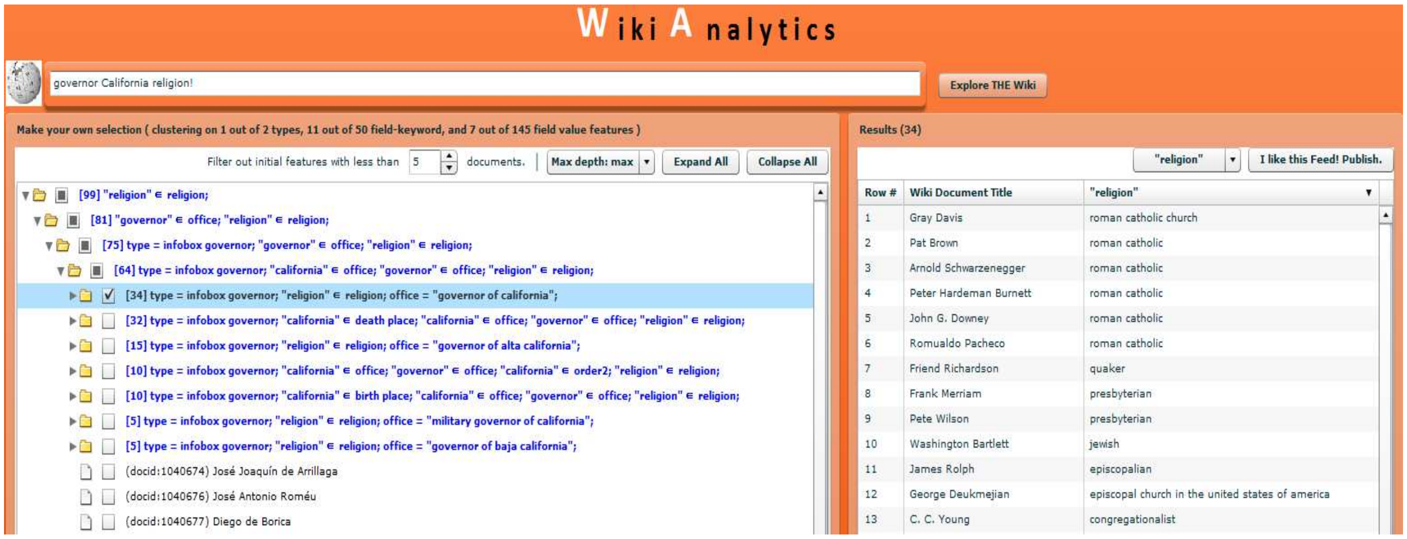
We compute the initial set of infoboxes for clustering by using an off-the-shelf keyword search system that supports stemming, term expansion, and other standard recall-enhancing techniques. Note that the keyword search by itself is not sufficient for building result lists since it cannot provide 100% precision. Imprecise results are tolerable for "point" queries because the user can browse a few candidates to identify a single result. However, if a user needs to compile or aggregate a list of tens or hundreds or thousands of infoboxes, browsing each candidate individually becomes infeasible. In this case, clustering results simplifies the browsing process and enables users to accept and reject semantically similar results as a group.

In order to give users a full picture of the possible clusterings of the query results we adopt a notion of *concept lattice* [8] over the clusters of infoboxes. We devise a data structure, called *universal navigational lattice* (UNL), which encodes all possible ways to group the infoboxes in the query result based on where the keywords hit, i.e. their features. A node is created in UNL for every distinct subset of infoboxes that shares some features. Edges in the UNL correspond to containment relationships between the infobox sets. We developed a GUI that allows users to navigate the UNL and iteratively interact with it by including and excluding clusters from the result list.

After the initial UNL construction we further help the user by heuristically pre-selecting the largest meaningful cluster of infoboxes for the result list. Our heuristic follows the edges in UNL by always picking the largest sub-cluster, until each keyword is found in either the same type or field name for each infobox, or in the same value of the same field. In general, this heuristic performs reasonably well in our experience, but the WIKIANALYTICS GUI allows the user to un-select the entire heuristically selected cluster or some of its sub-clusters.

Consider a scenario where the user wants to find a comprehensive and precise list of all Governors of California and to extract their religious affiliations. In WIKIANALYTICS *the user might start by typing in the following query keywords: "California governor religion." Over the* UNL *corresponding to this query, the default heuristic picks a cluster of infoboxes that all have three features in common. First, they have type* `governor` *– this covers keyword "governor". Second, they have a field* `religion` *with any value – this covers "religion". Finally, they have value "Governor of California" in field* `office` *– this turns out to be the largest feature that covers query keyword "California". This cluster accounts for* 34 *out of* 99 *search results. As mentioned above, the heuristic finds* 34 *out of a total of* 38 *Governors of California, missing out* 4 *structural outliers. Even with good domain knowledge, the user will find it hard to locate them.* $\diamond$

In general, the UNL grows super-linearly with the size of the result and it is rarely practical to present the full lattice as it is to the user due to its large size. We introduce a number



governor California religion!

Make your own selection (clustering on 1 out of 2 types, 11 out of 50 field-keyword, and 7 out of 145 field value features)

Filter out initial features with less than 5 documents. | Max depth: max

- [99] "religion" ∈ religion;
  - [81] "governor" ∈ office; "religion" ∈ religion;
    - [75] type = infobox governor; "governor" ∈ office; "religion" ∈ religion;
      - [64] type = infobox governor; "california" ∈ office; "governor" ∈ office; "religion" ∈ religion;
        - ✓ [34] type = infobox governor; "religion" ∈ religion; office = "governor of california";
        - [32] type = infobox governor; "california" ∈ death place; "california" ∈ office; "governor" ∈ office; "religion" ∈ religion;
        - [15] type = infobox governor; "religion" ∈ religion; office = "governor of alta california";
        - [10] type = infobox governor; "california" ∈ office; "governor" ∈ office; "california" ∈ order2; "religion" ∈ religion;
        - [10] type = infobox governor; "california" ∈ birth place; "california" ∈ office; "governor" ∈ office; "religion" ∈ religion;
        - [5] type = infobox governor; "religion" ∈ religion; office = "military governor of california";
        - [5] type = infobox governor; "religion" ∈ religion; office = "governor of baja california";
        - (docid:1040674) José Joaquín de Arrillaga
        - (docid:1040676) José Antonio Roméu
        - (docid:1040677) Diego de Borica

Results (34)

"religion"

Row #	Wiki Document Title	"religion"
1	Gray Davis	roman catholic church
2	Pat Brown	roman catholic
3	Arnold Schwarzenegger	roman catholic
4	Peter Hardeman Burnett	roman catholic
5	John G. Downey	roman catholic
6	Romualdo Pacheco	roman catholic
7	Friend Richardson	quaker
8	Frank Merriam	presbyterian
9	Pete Wilson	presbyterian
10	Washington Bartlett	jewish
11	James Rolph	episcopalian
12	George Deukmejian	episcopal church in the united states of america
13	C. C. Young	congregationalist

Fig. 2. GUI: Faceted Search-like Interface over the Documents Clustered by Feature Sets

of pruning techniques to keep the UNL size under control. The pruning serves two goals: it keeps UNL construction fast enough for on-line processing, and makes the result easier for users to comprehend and work with. First, in order to separate homogeneous patterns from outliers, the feature pruning technique filters out clusters with fewer objects than a user-defined *feature support threshold* (FST). Typically, this eliminates a majority of the features and allows the users to focus on large clusters of structurally homogeneous records. Second, the user can accept or reject any of these clusters, which consist entirely of results or non-results, respectively.

For instance, using the default $FST = 5$ *in our running example, in order to focus on the outlier answers, let us "accept" the heuristically selected large result cluster and also reject three obviously irrelevant neighboring clusters described by features:* `office` *contains value "Governor of Alta California", field* `office` *contains "Governor of Baja California", and field* `office` *contains "Military Governor of California", which contain* 15, 5, *and* 5 *records, respectively. These clusters are shown in the* WIKIANALYTICS *GUI in Figure 2.* ◇

Finally, the user can recompute the UNL over the remaining objects, and with a lower FST. These steps can be repeated iteratively allowing the user to zoom in on progressively smaller clusters in order to identify the structural outliers.

Let us now recompute the UNL *for the remaining* 40 *records with* $FST = 1$. *Analyzing this shorter list of records, it is easy to spot clusters with suspicious field names like* `order2`, `office2`, *and* `office3`. *They contain the four outlier records, which are Governors of California that were not picked up by the heuristic, as described in Section I.* ◇

The final result of a WIKIANALYTICS search process is a table or a data feed with a key column (name of the wiki page), and a value column for every keyword specified as an extraction by using the special "!" character out of the current cluster selection. For example, the query "California governor religion!" returns pairs of page name and religious affiliation.

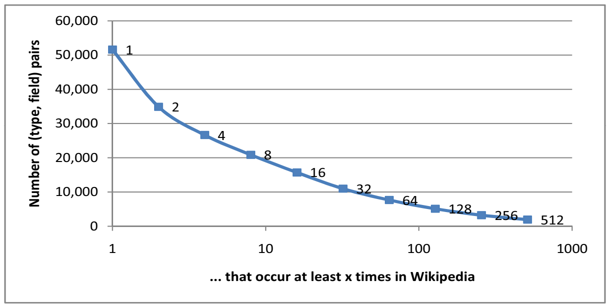


Fig. 3. Wikipedia is sparse: Distribution of fields per number of times they occur in Wikipedia infoboxes.

The resulting feeds can later be joined and aggregated by mashup tools like Yahoo Pipes[1] and Damia[14], and/or visualized by services like Swivel[2] and Many Eyes[3].

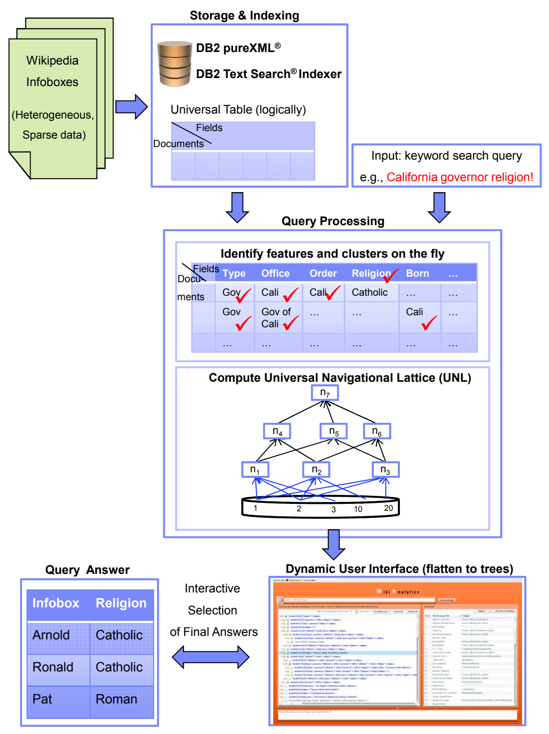
We propose to demonstrate interactively the WIKIANALYTICS tool as a proof-of-concept for our new search exploration approach to ad-hoc querying of highly heterogeneous Wikipedia infoboxes. We will demo this query example along with other similar querying scenarios on the Wikipedia dataset.

III. SYSTEM ARCHITECTURE

In this section, we describe our design goals and the system architecture of WIKIANALYTICS. Our design is constrained by three main requirements: (i) an easy to use and an effective search interface to explore and disambiguate answers by making data selections while navigating heterogeneous collections, (ii) enable the user to select a complete and precise set of answers according to intentions expressed initially as a keyword query, and (iii) the search should not modify nor markup the original data corpus in a way to facilitate data discovery.

Based on these design decisions, we chose the architecture of Figure 4 consisting of the following parts: data storage and

[1]http://pipes.yahoo.com/
[2]http://www.swivel.com
[3]http://manyeyes.alphaworks.ibm.com/manyeyes


Storage & Indexing
Wikipedia Infoboxes (Heterogeneous, Sparse data)
DB2 pureXML®
DB2 Text Search® Indexer
Universal Table (logically)
Fields
Documents
Input: keyword search query
e.g., California governor religion!
Query Processing
Identify features and clusters on the fly
Fields
Documents
Type
Office
Order
Religion
Born
…
Gov
Cali
Cali
Catholic
Gov
Gov of Cali
Cali
Compute Universal Navigational Lattice (UNL)
n7
n4
n5
n6
n1
n2
n3
1
2
3
10
20
Query Answer
Interactive Selection of Final Answers
Dynamic User Interface (flatten to trees)

Infobox	Religion
Arnold	Catholic
Ronald	Catholic
Pat	Roman

Fig. 4. WIKIANALYTICS Architecture

indexing, query processing, and a dynamic user interface with cluster selection.

For the first part, we extract the infoboxes from a Wikipedia snapshot of 2008 and we convert them to XML by using the Wiki2XML tool of the Texterra project[4]. For instance, Figure 1 shows a sample record of Arnold Schwarzengger's infobox in XML format. In order to query infoboxes, we store them in IBM's DB2 pureXML® database which comes with native XML storage and querying support. We leverage the power of DB2 Text Search® engine for XML full-text search.

The query processing part consists of two modules: feature extraction and construction of the universal navigation lattice UNL. The text index produces a set of infobox documents that match all query keywords. The first module conceptually views these infoboxes as a (sparse) universal table, with a row for each infobox and a column for every field that occurs in at least one of them. A feature with a corresponding cluster of documents is created for each field name and field value that contain a given query keyword. The second module builds the UNL lattice graph. Intuitively, UNL encodes all possible meaningful clusters of documents by all sets of features. We create relevant clusters and links between them so as to use the features as dynamic structural dimensions that slice and dice in the data collection to facilitate document exploration and selection.

The third part takes care of the lattice presentation and of the user interaction for the scope of document selection. The lattice is exposed into a tree interface that creates experience similar to that of the multi-faceted dynamic search over the feature sets. This facilitates complete access to the data without dropping any query answers and enables smart querying as well as easy data selection. Lastly, we extract the final result for further data processing, e.g., business intelligence data analytics or data mashups. Figure 2 shows our target search GUI interface.

[4]http://modis.ispras.ru/texterra/download/index.html

WIKIANALYTICS provides a web-based interface to the data. We implemented the backend using Java servlet technology to extract the features from DB2 and to construct the UNL in memory. For presentation, the lattice is flattened into trees and serialized to a Flex frontend application. The GUI allows interactive visualization of the UNL with support for (de)selection of record clusters.

IV. CONCLUSION

Large quantities of structured data are being created by online communities in wikis and other highly heterogeneous data sources. In this paper we presented WIKIANALYTICS, a tool to support on-line ad-hoc querying over these data. We demonstrate effective methods within a smart interactive user interface that facilitates exploration and disambiguation of search results in order to compile complete and precise answers that span multiple records or pages.

REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: a system for keyword-based search over relational databases. In *18th International Conference on Data Engineering (ICDE'02)*, 2002.
[2] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Authority-based keyword queries in databases using ObjectRank. In *International Conference on Very Large Data Bases (VLDB'04)*, 2004.
[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *18th International Conference on Data Engineering (ICDE'02)*, 2002.
[4] K. Bollacker, R. Cook, and P. Tufts. A platform for scalable, collaborative, structured information integration. In *6th Intl. Workshop on Information Integration on the Web (IIWeb'07)*, 2007.
[5] M. Cammarano, X. L. Dong, B. Chan, J. Klingner, J. Talbot, A. Y. Halevy, and P. Hanrahan. Visualization of heterogeneous data. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1200–1207, 2007.
[6] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In *International Conference on Very Large Data Bases (VLDB'03)*, 2003.
[7] DBpedia. http://www.dbpedia.org.
[8] B. Ganter and R. Wille. Formal concept analysis: Mathematical foundations. In *Springer-Verlag*, 1999.
[9] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *International Conference on Very Large Data Bases (VLDB'03)*, 2003.
[10] V. Hristidis and Y. Papakonstantinou. Discover: keyword search in relational databases. In *International Conference on Very Large Data Bases (VLDB'02)*, 2002.
[11] F. Li, C. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *International Conference on Management of Data (SIGMOD'06)*, 2006.
[12] Y. Li, C. Yu, and H. V. Jagadish. Schema-Free XQuery. In *International Conference on Very Large Data Bases (VLDB'04)*, 2004.
[13] Powerset. http://www.powerset.com/.
[14] D. E. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: data mashups for intranet applications. In *SIGMOD Conference*, pages 1171–1182, 2008.
[15] M. Theobald, R. Schenkel, and G. Weikum. An Efficient and Versatile Query Engine for TopX Search. In *International Conference on Very Large Data Bases (VLDB'05)*, 2005.
[16] Z. Vagena, L. Colby, F. Ozcan, A. Balmin, and Q. Li. On the effectiveness of flexible querying heuristics for XML data. In *Fifth International XML Database Symposium (XSym'07)*, 2007.
[17] Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. In *International Conference on Management of Data (SIGMOD'05)*, 2005.