Business Insight from Collection of Unstructured Formatted Documents with IBM Content Harvester

Biplav Srivastava and Yuan-chi Chang

IBM Research
New Delhi, India and Hawthorne, NY, USA
{sbiplav@in, yuanchi@us}.ibm.com

Abstract

In this paper, we report the development and experiments of IBM Content Harvester (CH), a tool to analyze and recover templates and content from word processor created text documents. CH is part of a bigger effort to collect and reuse material generated in business service engagements. Specifically, it works on unstructured formatted documents and works by extracting content, cleansing off sensitive information, tagging it based on user-defined or domain-defined labels, and making it available for publishing in any open format and flexible querying. As a result, one can search for specific information based on tags, aggregate information regardless of document source or formatting peculiarities and publish the content in any format or template. CH has been applied to a broad variety of document collections containing hundreds of documents, including live engagements, to promising effect.

1 Introduction

With the proliferation of office productivity tools, it has been recognized that users need better ways to search, organize and reuse their content when appropriate. The introduction of text indexing and search addressed certain challenges in finding relevant content. However, it is still very time consuming and labor intensive to crawl through the search results in order to identify reusable content piece by piece.

A common scenario is team-based document creation, which is wide-spread in Information Technology (IT) software and services business. The documents are created with commercial word processors like Microsoft Word, Lotus Symphony and Open Office. The teams start from mandated templates and team members add content. In the process, they invariably change a document's structure if expedient to capture some specific information. It is not uncommon for teams to create 100s of documents created on a IT project. Such documents are stored in repositories (including file systems) that traditionally provide only keyword based search. Over time, we have a large collection of documents following different templates: documents from a single client but multiple types of documents (e.g., design, test, specification), documents from different clients and same or multiple types of documents.

A potential consumer of the document will be interested in the content in the document but usually not the template because the latter changes from one client setting to another. For example, many teams will be interested to reuse the design of an online checkout and international shipping feature as implemented on a retail website, but the template in which the information is documented is of little consequence. Consider we have information about this feature from two clients projects in the repository. Since the technology is complex, we would want to compile the full list of features that previous teams have considered. However, the documents available will have an inter-mix of content and the formatting information, and it is very daunting for a third team to sift through the complete feature documents to compile the information they need.

A large services organization does thousands of projects in a year. Given the scale of documents they produce, it is practically impossible today to refer to a subset of design documents, all created in different enagements, and potentially with different templates, and try to obtain content from them in an integrated form. The resulting content can then be published with any template that the new client may want.

We present the Content Harvester framework to solve the challenges. It works on collections of unstructured formatted documents and requires the user to specify the textual segment of information they want to extract, what identifiers they want to replace and what to label the extracted content. The tool first separates the textual and non-textual (including formatting) content[1], and then uses segment specification to extract information of interest. It maintains basic structure of extracted content - paragraph, list and table, and represents it in XML. Next, sensitive information is cleansed off and then extracted content is

15th International Conference on Management of Data
COMAD 2009, Mysore, India, December 9–12, 2009

[1] But keeps record of what was found.

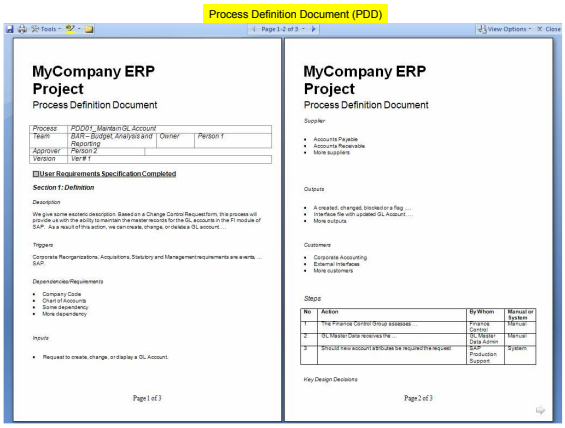

Figure 1: The original Word file of a running example.

tagged with labels from the segment specification. The result is available for publishing in any open format like WordML, HTML or PDF by simple XML transformations, and flexible XML queries can be done over it. As a result, one can search for specific information based on tags, aggregate information regardless of document source or formatting peculiarities and publish the content in any format (Word, pdf, html) or template. A scaled-down version of the CH tool is publicly available[2]. In addition, methods are provided to learn templates and segment specifications. Content Harvester has been applied to a broad variety of document collections containing 100s of documents, including live engagements, to promising effect.

Our contributions in the paper are: (1) A format-independent methodology to segment unstructured formatted documents into units of text for cross-document processing; (2) An architecture to extract content with user annotated format-independent landmarks for repurposing and reuse; (3) A statistical method to recommend segmentation (or landmark) boundaries for user annotation in a cluster of documents; (4) A statistical method to analyze and identify clusters of similar documents that are likely to stem from the same templates, using parsed text units; (5) Experimental results and pilot experience on the efficacy of the approach.

We now describe 1, 2 and 5 from above; please see details of the rest in the longer version of the paper[3]. We will present the problem, describe our approach and an implementation, discuss initial empirical & pilot results, illustrate CH's ability to bring new insights and conclude with a review of related work.

2 Problem

The problem setting is that we have a collection of original documents produced with a document processor like Microsoft Word. A document can consist of headings, paragraphs, lists, tables, images, non-textual generic objects, and any of their combinations (e.g., lists inside tables). The document's content is annotated with formatting/visual cues. A person can identify a particular content of interest by textual markers that serve as content boundaries. Since a marker is also part of the document, it can consist of text and formatting styles.

More precisely, we use the following terms:

- Marker: A piece of text that a user can view in a document, or pre-defined special positions in a document – document start and end.
- Formatting instructions: bold, italics, underline, font, list type, table, cells, section hierarchy, embedded objects, images, and so on.
- Document fragment: A contiguous fragment of original content of a document including formatting instructions, demarcated by start and end markers.
- Segment or extracted content: A contiguous fragment of content along with its demarcating start and end markers. The formatting instructions are absent in a segment.
- *Original* document: A sequence of one or more document fragments.
- *Harvested* document[3]: A sequence of one or more segments.

Consider the example original document shown in Figure 1. Text *Process* and *Team* are markers. The process name between the two markers is a piece of content the user may be interested in. The two markers and the process name constitute a segment. Note that the three texts are different cells of a table in the example.

The goals of harvesting content from original documents of a collection are to: (1) extract segment(s) with content of interest (2) cleanse extracted content off any sensitive context (3) tag extracted segments with a set of given labels (including the case when the labels come from a known model) (4) enable tag-based and content-type search on extracted content (5) allow output to be published in any encoding format or document type, and (6) identify documents following a common template in a pool from different sources. The challenges of the problem are explained in the longer version of the paper[3].

3 Solution for Harvesting Documents following a Common Template

The pseudo-code for Content Harvester (CH) is shown in Figure 2 and the architecture of a prototype system is shown in Figure 3. CH works on collections of unstructured formatted documents (D) and requires the user to specify the textual segment of information they want to extract and what to label the extracted content with (L), and

[2] At: http://www.alphaworks.ibm.com/tech/contentharvester

[3] The prefix *original* or *harvested* will be dropped when the type of document is clear from context.

what sensitive identifiers they want to replace (R^c). We now explain the main steps in subsequent subsections.

Algorithm: **HarvestContentFromSimilarDocs**
Inputs: A set of documents D following a common template,
Landmark specification of what to extract, L
Rules expressing what phrases to cleanse and the new phrase, R^c
Flag for output format
Output: A set of cleansed documents, D^c
Cleansing report, R

Pre-processing:
1. For each document d_i
2. Convert D_i to XML representation
Main Steps for each document
1. For each document d_i
2. Parse d_i
3. Group characters along word boundaries
4. Group words along paragraph (formatting) boundaries
5. Record paragraphs with basic source formatting information
6. Remove non-textual and non-formatting content
7. Uselandmarks L on textual paragraphs
8. Use markers to identify segments
9. Use formatting information to identify basic content structure
10. Use heuristics to overcome noisy/ missing text
11. Apply R^c on extracted content
12. Use labels from L to tag extracted content
13. Publish D_i^c in XML
14. Record statistics about d_i for reporting
15. Publish final R with overall and per-file statistics

Figure 2: Pseudo-code of Content Harvester.

3.1 Parsing Word Documents

In recent years, modern word processing software began to adopt XML as a supported file format, which allows easier access to text content stored in the files. While XML is self-describing, these XML file formats primarily focus on the presentation and rendering styles of the text content, such as character formatting, paragraph spacing, lists, tables and figures, etc. There is a lack of provision for user-annotated constructs to describe the content. Hence a tool like ours is still required to analyze and identify bodies of semantically similar content.

We use two commonly applied standards as examples to describe our approach to segment text content for further analysis, i.e. Office Open XML (OOXML) and OpenDocument Format (ODF). Microsoft Office suite software supports OOXML while Star Office, Google Docs and IBM Lotus Symphony support ODF. In this paper, we will focus on word processing section of the above standards but our approach may be more generally applicable to other sections such as spreadsheet and presentation.

Our approach to extract the raw text content (steps 2-6 in Figure 2) is to identify paragraph boundaries and reassemble texts falling within the same paragraph boundary as a single text block for subsequent analysis. If one views a text file as a sequence of character streams, we find paragraphs to be the basic and natural segmenting markers to group together characters semantically. This view is fairly different from say, indexing a text document for text search, where the natural segmentation will be at the word level.

In the WordProcessing ML section of the OOXML, paragraphs are identified the <w:p> tags, where w is the namespace declaration of WordML. Under each <w:p> tag, there may be styling information about the paragraph such as headings, bulleted lists or numbered lists. The paragraph may also contain one or more character formatting

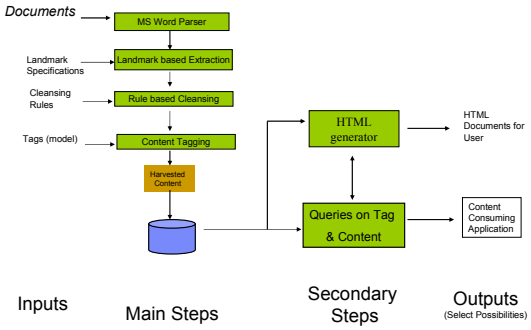

Figure 3: Architecture of Content Harvester.

instructions under the <w:r> tags, which define styling information such as bold, italic, underline or color for the associated characters under the <w:t> tags. An example of key tags used by our parser is shown in Figure 4.

```
...
<w:p wsp:rsidR="00000000" wsp:rsidRDefault="0033549C">
    <w:pPr>
        <w:pStyle w:val="Heading2"/>
    </w:pPr>
    <w:r>
        <w:rPr>
            <w:rFonts w:ascii="Arial" w:fareast="MS PGothic"
                w:h-ansi="Arial" w:cs="Arial"/>
            <wx:font wx:val="Arial"/>
            <w:color w:val="000000"/>
            <w:b/>
            <w:sz w:val="20"/>
            <w:sz-cs w:val="14"/>
        </w:rPr>
        <w:t>Section 1: Definition</w:t>
    </w:r>
</w:p>
<w:p>
...
```

Figure 4: An example showing the relationship between <w:p>, <w:r>, <w:t> tags and their associated style tags.

We have also identified how documents in OpenDocument Format can be harvested. Due to space limitation, we defer the details to the longer version of the paper[3].

3.2 Landmark Based Extraction

The previous section described how textual and non-textual (including formatting) content is separated. Now, textual content of interest in the document has to be identified. For this, segment specification called *landmark* is used to extract information of interest. We differentiate landmarks from the conventional notion of segments in image and text extraction literature because there markers, or segment boundaries, are seen in the input's bit stream. In our case, the characters in the markers may be fragmented.

We define *landmark* as a specification of a segment whose start pattern is known and its end pattern is optional. If the end pattern is missing, the end of the segment is marked by start pattern of any known landmark specification. It is easy to see that:

- If end is known, the segment becomes neighborhood dependent.

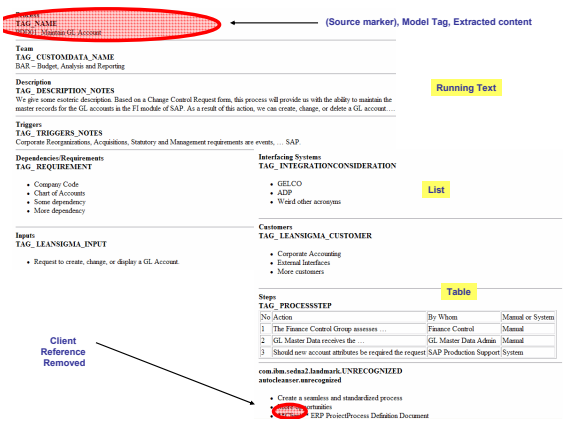

Figure 5: Extracted output in the example.

- If end is unknown, the segment is neighborhood independent.

We assume that the user will look at a sample of the documents and create landmark specifications. Consider documents from a Pharmaceutical engagement describing business process descriptions. Figure 1 shows one of the documents (slightly masked). Suppose we want to get the process name and steps from these documents. Process name is contained within a cell of a table and its predecessor cell has text marker, *Process*, and successor cell has text marker, *Team*. Figure 6 shows an example specification of the landmark. Here, *startMarker* is specified and *isStartMarkerSep* notes that the content is separated from the marker by non-textual separator. The end marker is optional. The field *modelReferenceTo* specifies the label to assign the extracted content and *isRepeating* flags absence of tabular content.

```
<landmark
        landmarkId="P1"
        isRepeating="no"
        isOptional="no"
        startMarker="Process"
        isStartMarkerSep="yes"
        modelReferenceTo="processTitle"
        isEndMarkerSep="yes">
</landmark>
```

Figure 6: An example of specification of a basic landmark.

For tables, *isRepeating* is set to 'yes' and *header*s are specified for each column of the table. An example for steps is in Figure 7. Note that the first column of the table has an empty value in the header. The output of the extracted content in shown in Figure 5.

The extraction of content is described in steps 7-10. CH maintains basic structure of extracted content - consecutive text, list and table. It also uses rules on formatting information from parsing phase (step 10) to handle noise. Some rules are:

- Ignore empty white spaces that are not part of any segments.

```
<landmark
        landmarkId="RC1"
        isRepeating="yes"
        isOptional="yes"
        startMarker="Steps"
        isStartMarkerSep="yes"
        modelReferenceTo="OtherSection/title"
        isEndMarkerSep="yes">
        <headers>
                <header name="" order="0">
                        </header>
                <header name="Action" order="1">
                        </header>
                <header name="By Whom" order="2">
                        </header>
                <header name="Manual or System"
                        order="3"> </header>
        </headers>
</landmark>
```

Figure 7: An example of specification of a tabular landmark.

- If an empty white space is within a cell, consider it as valid content.

- If an item is part of a list and is empty, ignore it.

In Figure 5, the middle rule was used on *Steps* table so that the first column header (empty) is recognized. Note that the basic structure of *Process* that it is a piece of continuous text, and of *Steps*, that it has tabular description, is preserved in extracted content.

3.3 Post-processing Extracted Content

In Step 11, a regular expression rule processor applies R^c on extracted content to remove references that are either non-relevant in a new context (e.g., version number) or privacy-related (e.g., client name). Then extracted content is tagged with labels from L (step 12) and then published in a XML representation (step 13).

The result is available for publishing in any open format like WordML, HTML or PDF by simple XML transformations, and flexible XML queries can be done over it. As a result, one can search for specific information based on tags, aggregate information regardless of document source or formatting peculiarities and publish the content in any format (Word, PDF, HTML) or template.

4 Experiment

We now discuss how the methods perform in practice by conducting an evaluation on a diverse dataset as well as verifying the method in the field by undertaking a pilot study.

4.1 Extracting Content from Documents

Here, we investigate how *HarvestContentFromSimilarDocs*() performs across different data sets with diverse characteristics: documents from day-to-day activities to software/IT business, average page lengths from very small (even 1) to large (≈60-70), different scale in number of documents in a dataset (ranging from a couple to 242), differing fidelity to their common template, and the scale of number of tags of interest (3-24). In the analysis that follows, we consider 5 data-sets corresponding to process design in SAP projects in different industries, 1 from detailed process design in an Oracle project, 1 dataset of personnel evaluations and another on recipes for cooking dishes. Note

that the tool has been released publicly and we are aware that it has been downloaded uniquely $\succ 60$ times. Furthermore, we are aware of CH being tried on $\succ 50$ different datasets. The analysis presented is only for a controlled set spanning diverse dataset characteristics.

Table 1 presents a preliminary evaluation. The columns represent average size of documents, # docs for the experiments, the ratio of the number of landmark specs created for each tag, the avg. % of document's content extracted and retained, the average number of tags applied, avg. processing time and finally a review of whether the dataset is amenable to content extraction cost-effectively. All columns are self-explanatory except the fourth one which we call *variability ratio*. The ratio measures how many landmarks are needed on an average to extract content for each tag in the dataset. Hence, the value conveys how disparate documents in a dataset are. If the ratio is 1, a single landmark is sufficient to get content for a tag from the whole dataset. Hence, the dataset indeed follows the underlying template consistently. The higher the value from 1, the more likely are the documents in the dataset to vary from a common template. We note that 5 of the 8 datasets in the experiment were conforming to a common template with their *variability ratio* (fourth column) in [1, 2] and another at 2.6. In fact, *Pharma-1* has 242 documents and yet has the ratio at 1.04. In contrast, the two data sets of *High Tech* have significant variability in their documents.

The results are very encouraging and show that the users could easily harvest content for a large proportion of tags of interest ($\succ$ 70% and even 100%) across the range of datasets. The content corresponding to these tags could be very specific and low (e.g., 8% of total content in *High Tech*) or as high as 93% in *Pharma-1*. The time to process a document varies with page length but it is about a minute/ document for a typical 10-page document. The extensive runs on large datasets of *Pharma* and *High Tech* indicate that the CH method is robust.

The experience of Oil&Gas-1 dataset is peculiarly interesting. Although the Word documents here were template-wise consistent, they were hard to work with due to *hidden/vanish* feature of Microsoft Word whereby invisible text is included in a document. The user would look at an original document to determine markers and content of interest but the tool may or may not encounter the same marker pattern and content. So, either extraction would fail or different content than what the user expected would come. We had to provide a separate tool to the users to expose hidden text and this solved both the problems.

4.2 Pilot Study of Content Harvester on Design Documents

We conducted a pilot study of Content Harvester to understand how feasible the tool's approach is in harvesting and cleansing large document collections in practice. The pilot ran for 5 weeks and involved a user group within IBM that manually cleansed design documents and formatted them to a standard form. The pilot's version of the tool was

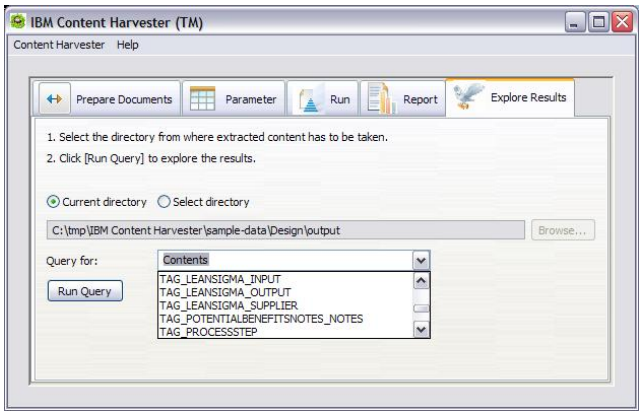

Figure 8: Selecting content based on tags.

an enhancement of Content Harvester presented here – the tool was aware of design tags and hence could enforce consistency checks, and could generate output in more formats. The pilot was designed to find where most time would be spent using the Content Harvester tool and for what types of documents this method would be cost-effective. The complexity of a document set was measured in terms of the number of documents and their average number of pages. The similarity of the documents were in terms of the number of common segments/ landmarks (estimated by approximate number of sections, tables, etc). 6 datasets of equal sizes but different characteristics (#pages, formatting and template consistency) were used.

We found that writing landmark specifications was the most time-consuming part of using the tool. However, this was a one-time cost and would be amortized if the number of documents to be processed was not very small and the documents followed a basic level of similarity. For documents of about 10 pages and moderate similarity, which is common for software design, the tool would be more cost-effective than manual cleansing and re-formatting. CH has the added benefit of tagging the harvested content and allowing it to be published in any open format using style-sheets. This makes the output seamlessly usable by other software tools.

5 Business Insight with CH

We now illustrate the kind of business insight possible with CH. Recall the original document shown in Figure 1. It is available as part of the *Design* dataset available from CH website. In the released tool, the user can use tags to select harvested content of documents in a dataset and post XML queries. In Figure 8, the user is looking for content available based on all the tags declared in the dataset[4]. She selects the tag *TAG_PROCESSSTEP*.

In Figure 9, the content associated with the tag is shown for all the files. The documents in the dataset were differing in common structure as the two tables have different first column; hence multiple landmarks were needed. But once

[4]The tool manual gives the details of how this can be done.

Dataset Name	Doc Size (in pages)	# Docs Processed	Ratio (Raw #s): # Landmarks/ # Tags	Parsed & Retained Content(%)	Tags Found (%)	Avg. Proc. time per doc (secs)	Comments
Pharma-1	4-10	242	1.04 (25/24)	93	86	40	Good
Pharma-2	6-10	27	2.6 (18/7)	84	83	81	Good
Oil&Gas-1	10-12	8	1 (10/10)	29	≈80	103	Good (difficult due to hidden text)
High Tech-1	35-60	21	4.3 (13/3)	8	≈70	520	Bad (variability)
High Tech-2	35-60	63	4.3 (13/3)	8	≈70	329	Bad (variability)
Oracle-Des	15-20	2	1.7 (5/3)	16	84	360	Good
Misc-1	4-7	3	1 (6/6)	81	100	44	Good
Misc-2	1	3	1 (5/5)	75	87	3	Good

Table 1: Evaluation of Content Harvester on Different Datasets.

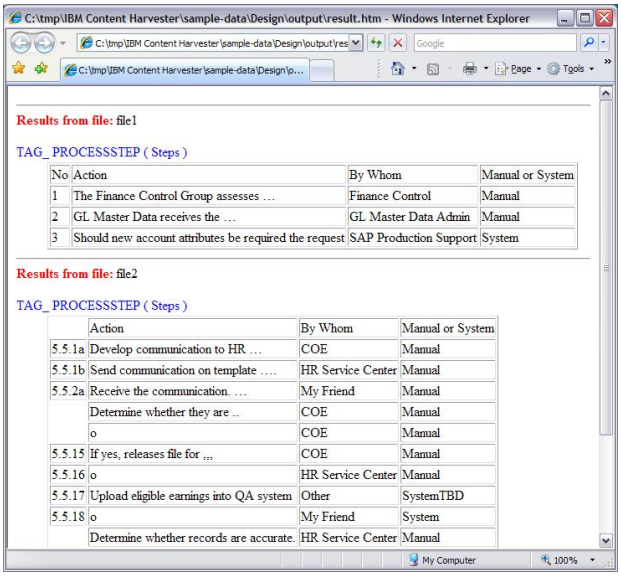

Figure 9: Example of an application enabled by CH. The result for XML query for tag *TAG_PROCESSSTEP* is shown across documents.

extracted and tagged with CH, the content can be searched across the dataset and new insights be found. Note that besides content, the tabular structure of the process steps is retained during extraction and this can be manipulated by applications.

6 Discussion and Conclusion

CH tackles a pressing hurdle in asset reuse which is how to get information from documents and improve consumption. CH allows harvesting of unstructured, formatted documents by extracting content, cleansing off sensitive information, tagging based on user-defined names and making it available for publishing in any open format and flexible querying. The current version works on MS Word but the approach extends to Open Doc standard also as explained. CH has been applied to 100s of documents from a broad variety of sources, including live engagements, to promising effect. We presented experimental results on document harvesting's effectiveness, pilot experience of the tool's feasibility, and statistical methods to work with unorganized documents to find subsets of similar documents on which the tool can work effectively. The approach is a stepping stone to gain business insights into collections of unstructured documents, and to that end, the released tool supports tag-based querying while integration of the harvested content with other analytical tools is shown in [4].

However the tool can be limiting in some situations. The user has to know about the documents at some level and specify the landmarks in terms of low level information. Additionally, the landmark specification is context-independent since selective behavior based on specific instances of landmark occurence is not supported.

In the past decade, there was significant amount of research in the domains of information retrieval, data mining and XML that touched upon the challenges of extracting content or analyzing document structure [1, 2]. There are approaches of mining hierarchical trees [5] and subtrees[6]. However, we find these approaches of limited help in the context of word processing XML standards that encode common business documents.

More details on the above issues can be found in [3].

6.1 Acknowledgements

We thank Swaroop Chalasani and Sridhar Maradugu for their help in tool implementation, Kathy Byrnes for pilot guidance, and Richard Goodwin, Juhnyoung Lee, Debdoot Mukherjee and Vibha Sinha for useful discussions.

References

[1] S. Abiteboul. Querying semi-structured data. In *Intl. Conf. Database Theory*, pages 1–18, 1997.

[2] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. *SIGMOD Rec.*, 27(2):295–306, 1998.

[3] B. Srivastava and Y. chi Chang. Business insight from collection of unstructured formatted documents with ibm content harvester. In *IBM Research Report at http://domino.research.ibm.com/library/cyberdig.nsf/index.html*, 2009.

[4] B. Srivastava and D. Mukherjee. Organizing documented processes. In *In IEEE Services Computing Conference, Bangalore, India*, 2009.

[5] K. Wang and H. Liu. Discovering structural association of semistructured data. *IEEE Trans. Knowl. Data Eng.*, 12(3):353–371, 2000.

[6] M. J. Zaki. Efficiently mining frequent trees in a forest. *ACM SIGKDD Conf.*, pages 71–80, 2002.