

# A Rank-Rewrite Framework for Summarizing XML Documents

Maya Ramanath

Max-Planck Institute for Informatics  
Saarbrücken, Germany  
ramanath@mpi-inf.mpg.de

Kondreddi Sarath Kumar

Max-Planck Institute for Informatics  
Saarbrücken, Germany  
skondred@mpi-inf.mpg.de

**Abstract**—With XML becoming a standard for data representation and exchange, we can expect to see large scale repositories and warehouses of XML data. In order for users to understand and explore these large collections, a summarized, bird’s eye view of the available data is a necessity. In this paper, we are interested in *semantic* XML document summaries which present the “important” information available in an XML document to the user. In the best case, such a summary is a concise replacement for the original document itself. At the other extreme, it should at least help the user make an informed choice as to the relevance of the document to his needs. In this paper, we address the three main issues which arise in producing such meaningful and concise summaries: i) which tags or text units are important and should be included in the summary, ii) how can the selected tags and text be presented in a concise and coherent manner? and iii) how to generate a semantic summary for different memory budgets? We conduct user studies with different real-life datasets and show that our methods are useful and effective in practice.

## I. INTRODUCTION

With the ubiquity of XML as the format of storage and exchange of data, we can expect to see ever-growing repositories of XML documents. Exploration of these collections requires the use of a diverse set of tools ranging from classifiers, clustering tools, data visualizers to mining software. One of the ways in which *human-centric* exploration can be made easier is to provide the user with a concise, summarized view of the information contained in an individual or in a set of documents. In addition, the tasks of searching for relevant information in a set of documents and deciding whether a given document is relevant to an information need or not, are made much easier if “bite-sized” summaries of the documents in question can be constructed and made available to the user. Moreover, with the advent of mobile devices with very small displays, summaries prove to be even more essential in satisfying the user’s information needs.

There are two kinds of summaries which can be generated. A *generic summary* summarizes the entire contents of the document. A *query-biased summary* summarizes those parts of the document which are relevant to the user’s query. The focus in this paper is the generation of generic summaries of XML documents. In producing generic summaries, the implicit assumption regarding the user’s information need is that, he is interested in knowing “what is in the document”

without having to read the document in its entirety. In this paper we propose techniques to *automatically* generate concise, readable summaries of XML documents subject to a memory budget. As a concrete example of the summaries we would like to generate, consider the IMDB snippet in Figure 1 which describes the movie “The Train: An Immigrant Journey”. All the tags in this snippet have semantics associated with them (they are not tags for formatting or display), and there are short pieces of information at the leaf level, such as the title of the movie, its director, genres, etc. A concise summary of this snippet is shown in Figure 2 where only the “important” structure and value information has been retained – some tags and their text values have been dropped completely (colourinfo) or converted into a more concise representation (keywords, prod\_countries). The resulting summary is shorter but conveys all the important information from the original document.

### A. Challenges in XML Summarization

A summary is useful if, at the very least, it helps the user decide whether a particular document is worth looking into in its entirety or not. The best summary would encapsulate most or all the salient points of the document and could, in many cases, serve as a replacement for the original document. However, generating the “best” summary involves satisfying two contradictory goals: maximum coverage and minimum space. The larger the amount of content to be included in the summary, the larger the size of the summary. The larger the size of the summary, the lesser its utility to the user. And so, a good balance between the size of the summary and its coverage is required.

**Informativeness:** The notion of informativeness is intuitive. For a unit of information (tags, text) to be informative to the user, it has to be important in the document and presented concisely to the user. This follows directly from the goal of summarization which is to present the salient points of the document in a concise manner to the user.

**Non-redundancy:** Redundancy in XML data is very explicit. The same tag, say, <KEYWORD> could occur multiple times, once for each keyword associated with a given movie. Clearly, it is not necessary to repeat all occurrences of the

```

<movie>
  <title>#7 Train:An Immigrant Journey, The</title>
  <aka>#7 Train from Main Street</aka>
  <prod_year>2000</prod_year>
  <prod_countries>
    <country>South Korea</country>
    <country>USA</country>
  </prod_countries>
  <prod_lang>English</prod_lang>
  <prod_location>New York City</prod_location>
  <prod_location>Queens</prod_location>
  <director>Park, Hye Jung</director>
  <genres>
    <genre>Short</genre>
    <genre>Documentary</genre>
  </genres>
  <keywords>
    <keyword>korean</keyword>
    <keyword>pakistani</keyword>
    <keyword>subway</keyword>
    <keyword>manhattan</keyword>
  </keywords>
  <colourinfo>Color</colourinfo>
</movie>

```

Fig. 1. Toy IMDB Example

tag in the summary, but instead, concisely represent it using a single tag.

**Coverage:** Coverage is closely related to informativeness and refers to the amount of information in the summary as opposed to the data. We may choose, for example, to leave out certain tags since they are not important enough. While this may improve the readability of the summary, it simultaneously reduces the coverage.

**Coherence:** The *context* of a tag in terms of its parent and/or sibling may sometimes be of importance. For example, in the context of movies, we cannot have a <PROD\_YEAR> without also having the <TITLE> as well. However, we can have the <TITLE> without the <PROD\_YEAR>.

### B. Our Approach

We regard the problem of generating XML summaries as a two-step problem. First, we need to rank the tags and text according to a notion of their “importance” in the document. The tag and text units will be considered for inclusion in the summary based on their ranking. Second, we need to *rewrite* the selected tags and text in order to make it readable, yet, concise. Our final goal is to propose techniques for summarizing a given XML document in a completely *automated* manner given a highly constrained memory budget. For example, some of the summaries we generate are just 512B long while the original documents

```

<movie>
  <title>#7 Train:An Immigrant Journey, The</title>
  <prod_year>2000</prod_year>
  <prod_countries>
    <country>{South Korea, USA}</country>
  </prod_countries>
  <prod_location>
    {New York City,Queens}</prod_location>
  <director>Park, Hye Jung</director>
  <genres>{Short,Documentary}</genres>
  <keywords>
    <keyword>
      {korean,pakistani,subway,manhattan}
    <keyword>
  </keywords>
</movie>

```

Fig. 2. A Concise Summary

were over 100KB in length.

### C. Contributions and Organization

Our contributions are the following:

1. We propose several general-purpose methods for rewriting text and tags which are suitable in different situations.
2. We propose techniques to rank text values and tags based on their importance in the document.
3. Finally, we propose an algorithm which takes the ranked tags and text values and rewrites them into a summary while strictly adhering to a memory budget.

The rest of this paper is organized as follows. Related work is discussed in Section II. Section III discusses the data and summary model for our work. Section IV discusses various techniques to prioritize tags and text values to be included in the summary. Section V describes the algorithm for generating a summary given a memory budget. Section VI reports on our user study. Finally, we conclude in Section VII.

## II. RELATED WORK

Text summarization is a well-developed field (see, for example, [1] for an overview), dedicated to developing techniques for summarizing text documents. In a nutshell, the aim is to present *important* and *non-redundant* information contained in a document to the user in a concise and readable manner. One way to tackle the problem of summarization is to look at it as a ranking problem – text spans (say, sentences) are ranked according to a certain set of features and only the top-ranked spans are included in the summary. We follow this approach in our work. That is, we rank both tags and text values and consider them for inclusion in the summary according to their rank. However, the techniques in text summarization are not directly applicable in our context for two reasons:

- the structure of an XML document may be as important as the text (the tags and tag hierarchy in XML are meant for providing additional semantics)
- the textual values which appear in XML documents are not always free-flowing text, for which text summarizers are most suitable

The work described in [2] is aimed at summarizing XML documents. However, the goal of the techniques presented here is still to rank and extract the best sentences to be included in a summary. But the structural (XML) features of the document are specifically made use of to improve upon previous techniques. Hence, the kind of XML document that is being summarized is still predominantly text while certain parts of the text are tagged. In contrast, our work deals with documents which may or may not have free-flowing text.

The work presented in [3] deals with XML Schema summarization, where the goal is to present important schema elements to make large XML *schemas* easily readable by the user. Our work, on the other hand, is interested in summarizing XML *documents* of which structure is only one part and the data the other.

Other approaches to representing XML data in a concise manner include compression (for example, [4]) and statistical summaries (for example, [5]). However, the many tools for compression focus on efficient query processing and not on the readability for the end-user. And statistical summaries are used mainly for cardinality estimation which feed into the query optimizer. Work on building tools for database exploration (for example, [6], [7]) is also relevant in our context. However, our work looks at *document-oriented XML* while the focus of these tools is on developing techniques to summarize the entire database (of relational tuples). On the other hand, a document-oriented view is taken, for example, in [8]. But, here the aim is to extract *facets* from a database consisting of *text* documents while we aim at presenting the user with a concise and readable summary of individual XML documents.

### III. SUMMARIZATION MODEL AND DEFINITIONS

We first present some terms and their definitions which are used throughout the paper.

**Definition 3.1 (XML document):** An XML document  $D$  is a rooted, labeled tree  $T = (V, E)$  and consists of elements, attributes and text values. The name of a node  $n$  is denoted  $label(n)$  and the text value of a node  $n$  is denoted  $text(n)$ . If  $n$  is an internal node, then  $text(n)$  refers to the concatenation of all text nodes reachable from  $n$ .  $parent(n)$  denotes the parent node of  $n$  and  $children(n)$  denotes the set of children of  $n$ .  $path(n)$  denotes the sequence of tag names from the root to  $n$ ; If  $n$  is a text node, then  $path(n)$  is the sequence of tag names from the root to  $parent(n)$ .

**Definition 3.2 (Text, Long Text, Short Text):** Any value at the leaf level is “text”. The text may be of two simple types: short text or long text. Long text refers to free-flowing text

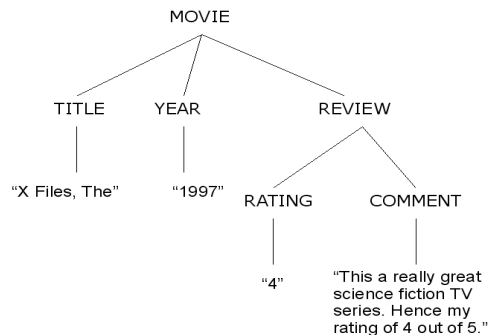


Fig. 3. Examples of Data Types

while short text refers to short values such as names, titles, dates, numbers, etc. This distinction is used to determine the appropriate summarization policies (discussed in Section IV). In the XML tree depicted in Figure 3, the leaf values for tags  $\langle TITLE \rangle$ ,  $\langle YEAR \rangle$  and  $\langle RATING \rangle$  are short text values while  $\langle COMMENT \rangle$  is a long text value.

**Definition 3.3 (Tag Unit, Text Unit):** A tag unit is a set of nodes  $T_e = \{n | label(n) = e\}$  and  $\forall n_i \in T_e, path(n_i) = path(n_j)$  when  $i \neq j$ . A text unit is a set of text values  $R_e = \{text(n) | label(n) = e\}$  and  $\forall text(n_i) \in R_e, path(n_i) = path(n_j)$  for  $i \neq j$ .

Each tag which occurs in different root-to-leaf paths are differentiated. This is important to maintain context – for example, the same tag  $\langle NAME \rangle$  could refer to both a person’s name as well as a company’s name. When no distinction between tag and text units is required, they are together termed *information units*.

**Definition 3.4 (Rewrite):** The “rewrite” operation takes a pair of tag-text units and rewrites them into a more concise form. The form of rewrite depends on the characteristics of the tag-text unit pair and is explained in detail in Section IV-A.

**Definition 3.5 (Select):** The “select” operation comes into play during the construction of the summary and it selects the next tag or text unit to be included (or eliminated) based on the rank of the tag/text and the memory budget.

**Definition 3.6 (Summary):** The summary of an XML document is another XML document constructed using the select and rewrite operators described above.

We first describe rewriting of information units into more concise formats and their ranking in Section IV and then move on to constructing the summary based on a memory budget in Section V.

### IV. INFORMATION UNIT SUMMARIZATION

We use the XML document describing information about the movie “2001: A Space Odyssey” as a running example in this section. It contains various information such as, title,

TABLE I  
SUMMARIZATION CHOICES FOR TEXT UNITS

	Short Text	Long Text
Unique Values Large	Sample	Sample/Shorten
Unique Values Small	Enumerate	Shorten/Enumerate

```
<production.location>
Shepperton Studios,Shepperton,Surrey,England,UK
Borehamwood,Hertfordshire,England,UK
Isle of Harris,Scotland,UK
MGM British Studios,Borehamwood,Hertfordshire,England,UK
Monument Valley,Utah,USA
</production.location>
```

```
<genre>Sci-Fi,Adventure</genre>
```

Fig. 4. Enumerated Text Information Units

year, production location, production languages, alternative titles, plot summaries (3 of them), actors and other people involved in making the movie, trivia items, etc. Hence the document contains many long and short text units, and is about 140KB in length.

#### A. Rewriting Text Units for Summarization

Recall from Definition 3.3, that a text unit consists of a set of text values *each of which* have the same enclosing tag (defined by the name of the tag as well as its path from the root). When the text unit is rewritten, the enclosing tag is included *only once* and the text units are listed within this tag using a suitable separator (for example, a “,” or a newline). We now list the methods for rewriting text units.

**Enumerate** If the average length of the text in the set is “short”, and the number of unique values in the set is small, then the rewrite method will simply *enumerate* these values.

**Sample** If the average length of the text in the set is “short”, but the number of unique values is large, then the rewrite method will include a “few” representative samples according to their rank (ranking text values is discussed later).

**Shorten** If the average length of the text in the set is “long”, then each text value is rewritten using a text summarizer (for example, [9]) to shorten it.

**Shorten/Enumerate** If the text needs to be shortened, but the number of unique such values is small, then the text unit is rewritten by enumerating the shortened versions of the units.

**Sample/Shorten** If the average length of the text is “long” and the number of unique values is also large, then the rewrite method would include a sample of important values and then shorten each of them.

Table I tabulates the various rewriting options. Each of the described options helps to eliminate redundancy in tags and to shorten text that may be too descriptive. Figure 4

```
<production.location>
MGM British Studios, Borehamwood,Hertfordshire,England,UK
Borehamwood, Hertfordshire, England, UK
</production.location>
```

Fig. 6. Sampled Text Values for <PRODUCTION\_LOCATION>

shows a couple of examples from the example movie “2001: A Space Odyssey” where enumeration may be suitable. Consider the tag <GENRE>. The only two genres listed are “Sci-fi” and “Adventure”, but each of them is enclosed in a separate <GENRE> tag. We can eliminate this redundancy by enclosing both words in a single <GENRE> tag. Similarly, the 5 production locations listed in 5 different <PRODUCTION\_LOCATION> tags can be combined into a single unit as shown in Figure 4. Other examples from the same movie include <PRODUCTION\_COUNTRY> (2 values), <PRODUCTION\_LANGUAGE> (2 values), etc.

Similarly, using a text summarizer to shorten long text helps not only in reducing the length of the summary, but also some of the potential redundancy in a long description. A good example of this is <PLOT>, which explains the plot line of the movie. Figure 5 shows the original plot text and a 25% summary of the same derived with the MEAD text summarizer [9]. The shortened text, though not conveying all the information contained in the original summary, nevertheless highlights some of the key points.

Enumeration and shortening are basic solutions to summarize text units, but the challenge actually lies in determining when to enumerate and/or when to shorten text. This decision has to be made when there is a limit set on the summary size. We may decide to enumerate a text unit corresponding to <PRODUCTION\_LOCATION>, but then find that 5 values is too large to fit into the summary given its memory budget. And so, that unit would be changed from being enumerated to being sampled. In effect, what we require is a ranking of text values in the text unit so that when a value has to be included in the summary or deleted from it, there is a definite priority. This aspect comes into play more explicitly in the next part of the paper where we describe solutions for generating good text samples for those text units which have a large number of values.

#### B. Ranking Text Units

1) *Centroid Query Method*: As our first motivating example, refer to Figure 4, which shows the text unit set of 5 values corresponding to <PRODUCTION\_LOCATION>. Suppose the goal is to pick the 2 most *representative* samples. Clearly, we would like the UK locations, and possibly the England locations to be in the sample since these locations dominate. This implies that we choose a sample set similar to that shown in Figure 6. In order to generate such samples automatically, we use the “centroid-query” method.

**ORIGINAL** (plot): Moon explorers encounter a monolith that points them to a destination near Jupiter. In flashback, we see another size of the monolith playing a key role in human evolution, i.e., we learn how to kill. An expedition is launched to investigate the Jupiter possibility. Two young astronauts and a bunch in suspended animation spend months in space, passing their time partly in communicating with the human-like brain of their ship's computer, HAL. HAL malfunctions and causes the death of all the suspended animation passengers as well as one of the "awake" astronauts; the other one barely survives and figures out how to disable HAL. He arrives alone at the Jupiter destination and undergoes a series of cinematographically confusing experiences that amount to his final appearance on the screen as a giant fetus.

**25% SUMMARY:** Moon explorers encounter a monolith that points them to a destination near Jupiter. HAL malfunctions and causes the death of all the suspended animation passengers as well as one of the "awake" astronauts; the other one barely survives and figures out how to disable HAL.

Fig. 5. Summarized TIU

This method ranks the text values in the text unit according to how representative they are of the text unit. The required number of samples can be chosen starting from the top-ranked text value and working downwards. The basic idea is to first generate a "centroid query" based on popular keywords. Then, a relevance score is calculated for each text value with respect to the centroid query. The higher the relevance of the text, the higher its rank. Figure 7 shows the steps required to rank the text. Steps 2 to 6 calculate the number of text values in which a term occurs. The top few of these terms form the centroid query. Next, a relevance score is computed for each text value with respect to the centroid query (steps 7 to 9). We use the basic *tf.idf* measure where the *tf* is the number of times a term in the query occurs in the text and *idf* is the number of text values in which the term occurs. Clearly, the text value which best represents the text unit is ranked the highest.

**Input:**  $L$

$L$  is the list of text values to be ranked

$l \in L$  denotes a text value

**Output:**  $L'$

$L'$  is the ranked list of text values

1.  $T$  = set of terms occurring in  $L$
2. **foreach**  $t \in T$
3.    $df(t) = \text{count}(\{l | l \in L \text{ and } t \in l\})$
4. **endfor**
5.  $CQ$  = top- $k$  terms in  $T$  sorted by their  $df(\cdot)$
6.  $CQ$  is the centroid query
7. **for**  $i$  from 1 to  $\text{length}(L)$
8.    $\text{Score}(L_i) = \text{tf.idf}$  score of  $L_i$  wrt  $CQ$
9. **endfor**
10. **return**  $L' = L$  sorted according to  $\text{Score}(\cdot)$

Fig. 7. Centroid Query-Based Sample Generation

2) *Diverse Text Values:* The above method works reasonably well for certain kinds of text units – those which can be reduced to a set of terms. However, there are other kinds of text values – for example, names – which cannot be meaningfully reduced to terms. Hence we would need to rank these kinds of text values by trying to determine

their importance in the *context* of their occurrence. There are two ways of viewing the context – the document under consideration or the corpus to which the current document belongs. As far as possible, we try to use the document itself as the context. Failing that, we use the corpus. The last option is to select the samples randomly.

As an example, consider the names of actors in a movie. In the context of the document, key actors are often discussed in the trivia items or goofs section or sometimes even in the plot. And actor names occur multiple times in the corpus since they act in many movies. However, in order to determine whether an actor is important *in a particular movie*, we need to consider the document itself as the context. If an actor's name is discussed often, we could make a reasonable guess that this particular actor is important in this context. In the example movie, the lead actor Keir Dullea's name comes up at least 4 times in the document (mainly in the trivia items) – more than any other actor.

However, it is not always possible to use the document as the context. As another example, consider the values for the tag <PRODUCTION\_LANGUAGE> in the example movie – English and Russian. There is no particular context within the document using which we can determine which language should be ranked higher. Instead, we turn to the corpus. Since a majority of movies in the corpus list their production language as English, we rank English higher than Russian.

It is possible to combine the two ways of ranking when required. If it so happens that of multiple text values that need to be ranked, only a few occur more than once in the document, then we rank these few using their occurrence counts in the document. However, to rank the remaining text values, we make use of their counts in the corpus.

3) *Correlated Samples:* So far, we talked about generating samples for text units independently of one another. But, many of the text units are *correlated*. For example, we cannot construct an arbitrary <actor, role> pair in the summary since the two are correlated. However, this does not preclude us from using different, independent methods to determine important actors and important roles. Our method for generating correlated samples, initially treats each text unit independently – that is actors are ranked independent

of their roles and vice-versa. Then, we take (at least) the top sample for each text unit and find the corresponding correlated text value for the remaining tags. However, in order to rank the entire correlated sample, we rank it based on the importance of the tag (described next) and then within each tag, based on the importance of the text value. For the example of actor and role pairs, we rank actor names independently and the role names independently. Then, if we need two samples of <actor,role> pairs, we pick the top actor and find the corresponding role value for the first ranked pair (assuming <ACTOR> is a more important tag than <ROLE>) and then pick the top role value and find the corresponding actor.

4) *Prioritizing the Methods*: In keeping with our goal of automating the summarization procedure without need for user input, we prioritize techniques for ranking as follows: i) rank using the document as the context, ii) rank using the corpus as the context, iii) rank by centroid query .

### C. Ranking Tags

We now turn our attention to the problem of how to determine important tags. By their nature, XML documents can have widely varying structure. However, if we restrict our attention to a single corpus, such as the movie corpus, the actual number of tags are not as diverse as the number of text values, even though the structure can still be widely varying.

As in the previous case of text units, there are two ways of looking at the context of a tag unit<sup>1</sup> – the document and the corpus. However, the popularity of a tag within the document does not correlate directly with its importance. For example, there can be only 1 movie title, but more than one trivia item. And clearly, the movie title is of more importance than the trivia item. Hence, we have to look at the corpus to determine the important tags.

In order to do this, we first enumerate all paths in the document. Then, we count the number of documents in the corpus in which each path occurs (note that we ignore the number of times a path occurs in a document). The intuition is that important tags occur in more documents than unimportant ones. To illustrate this, consider Table II which shows the number of times a tag occurs in the corpus. Clearly, the number of occurrences reflect the importance of tags. For example, *all* documents in the corpus contain a <MOVIE>, <TITLE> and <PRODUCTION\_YEAR> – the three things which basically define the movie. We also see that the <CAST> is more important than <GOOFS> and the <DIRECTOR> is more important than <KEYWORDS> and so on. Hence we can now order the tags in terms of their importance in the corpus.

<sup>1</sup>Recall that a tag unit refers not just to the tag name, but to its path starting from the root.

TABLE II  
OCCURRENCE OF TAGS IN THE CORPUS

Tags	No. of occurrences in corpus (1000 docs)
<MOVIE>	1000
<TITLE>	1000
<PRODUCTION_YEAR>	1000
<DIRECTOR>	746
<LINKS>	146
<GOOFS>	19
<CAST>	647
<KEYWORDS>	385

## V. GENERATING THE SUMMARY

In a nutshell, we have now described the following main methods of rewriting text units: i) Enumeration, ii) Shortening and iii) Sampling. And we have described 3 methods of ranking text units: i) with document as context, ii) with corpus as context and iii) centroid query-based ranking. Each of these methods is applicable to different kinds of text units and sometimes the two different methods may be applicable to the same text unit. And so, we described a prioritization of methods. For tags, we described a method to rank tags based on the corpus.

We now turn our attention to the generation of the summary. We outline an algorithm to *automatically* generate a summary of the required size. The algorithm works in four steps. First, all long text values in a text unit are shortened. Second, tags and text values are ranked. Third, a “first-cut” summary, based on the number of occurrences of tags in the original document, is constructed. This summary is larger than the required size. And finally, this summary is iteratively refined, by eliminating tag and text values until the desired size is reached. Note that the rewrite of the tag-text units is explicitly highlighted in the first step only when a text summarizer is used to shorten the text values. Otherwise, the rewrite is implicit – all text units are ranked and enumerated in the initial stages. Samples are extracted from these ranked text units as the summary is iteratively refined.

*Ranking text and tag values*: We utilize the prioritization outlined in Section IV-B.4 in order to rank text values. Initially, *all* text values in a text unit are ranked. Ranking tags is a straightforward step wherein, we lookup each tag with its corresponding count in the corpus.

*Iterative Refinement of Summary*: This is the main step of the algorithm. The original document is the initial summary. This summary is then iteratively refined to get smaller and smaller summaries until a small enough summary, which does not exceed the memory budget is obtained. In a nutshell, we repeatedly eliminate the least important text value of the least important tag from the current summary until the size of the summary is less than or equal to the memory budget. However, we need to consider two special cases:

First, a tag, and correspondingly, its text value cannot be removed in isolation if it is a correlated value. For example, it makes no sense to remove the <TITLE> of a movie, but leave behind its <PRODUCTION\_YEAR>. And so, we need to remove text values from *all tags of equal rank* if they are correlated. If <TITLE> and <PRODUCTION\_YEAR> have the same rank, then they are considered for removal in the same iteration - that is, pairs of values are removed rather than individual values. However, if the values are correlated, but one tag has a lower rank than the other, then it makes sense to remove only the lower ranked tag. An example of this is the correlated pair <ACTOR> and <ROLE>. The tag <ROLE> is ranked lower than <ACTOR>, and indeed it makes intuitive sense to have only the name of the actor without knowing his role in the movie, but not vice-versa. Second, the proportions of text values corresponding to the tags need to be maintained while at the same time aiming for higher coverage of tags and text. For example, <CAST> is more important than <KEYWORD>. However, if we blindly follow the policy of eliminating the least important text in the least important tag, we may end up with a summary consisting of 20 actors and no keywords. And so, we need to design a policy to maintain the balance between the coverage of tags and of text values. The strategy we utilize involves generating a “first-cut” summary, which is as close in size to the required summary size as possible while still retaining all tags in the original document. That is, for each tag in the document, *at least one* text value is retained in the first-cut summary. We use the following formula to determine the number of text values to be retained per tag:

$$S(t) = \frac{Size_{sum}}{Size_{doc}} * Num_{text}(t)$$

where  $S(t)$  is the number of samples for tag  $t$  under consideration,  $Size_{sum}$  is the desired summary size,  $Size_{doc}$  is the size of the document, and  $Num_{text}(t)$  is the number of text values for the tag  $t$ .

Once we determine the number of samples per tag, we simply pick them from the ranked list of text values from the corresponding text unit. This first-cut summary is then iteratively refined as described above.

## VI. EXPERIMENTS

We performed a user study to evaluate the quality of summaries generated by our techniques as well as to test the mechanism for ranking text values and tags – one of the key operations required for summary generation.

*Datasets:* The datasets used in our study was derived from two sources: IMDB (<http://www.imdb.com>) and DBLP (<http://dblp.uni-trier.de>). The IMDB corpus consisted of approximately 50,000 XML files, each describing a single movie. The DBLP corpus consisted of around 160,000 XML files, each describing a single publication. We chose 4 movies from IMDB and 4 publications from DBLP. There were no particular criteria in selecting these documents except

TABLE III  
XML FILES USED IN THE USER STUDY

Dataset	Doc Id	Doc name	Size
IMDB	I1	Titanic	200KB
	I2	<i>Fear and Loathing in Las Vegas</i>	77KB
	I3	Fight Club	85KB
	I4	<i>2001: A Space Odyssey</i>	129KB
DBLP	D1	Tutorial: Languages for Collection Types (V. Tannen)	4KB
	D2	<i>Data Mining: An Overview from a Database Perspective (M.-S. Chen, J. Han, P.S. Yu)</i>	3KB
	D3	Queries and Views in an Object-Oriented Data Model (U. Dayal)	4KB
	D4	<i>Query Evaluation Techniques for Large Databases (G. Graefe)</i>	20KB

TABLE IV  
EXAMPLE TOP-5 VALUES FOR FIGHT CLUB

Top-5 Tags	Top-5 Actors
movie/production_language	Norton, Edward
movie/production_year	Pitt, Brad
movie/title	Bonham Carter, Helena
movie/production_country	Leto, Jared
movie/colourinfo	Hawn, Phil

that they were about “well-known” movies/publications and were too large to fit into a single screen. Table III gives an overview of the datasets. The IMDB files contained basic movie information such as title and year of the movie as well as information about a large number of actors, miscellaneous crew, trivia items, etc. The DBLP files, in contrast were much smaller (but still too large to fit into one page) and their size was mainly because of the large number of citations provided.

*Evaluation Setup:* For each of the above files, we constructed two summaries of sizes 512B and 1KB - both summaries fit well within a single screenful and could be browsed without having to scroll. Each summary was evaluated by 3 evaluators. The evaluation consisted of two phases. First, the summary itself was evaluated with respect to its content and size. The evaluator filled out a questionnaire for each summary and provided a final grade for the summary ranging from 1 (best) to 5 (worst). Second, to test the ranking of tags and text values, a “Top-5” evaluation was done. Here, the user was shown the top-5 tags that the system generated and was asked to say whether or not each tag belonged in the top-5. Next, the top-5 text values for certain tags (actor names for IMDB, citations for DBLP) were shown and a similar evaluation was performed. An example of a top-5 evaluation for the movie Fight Club is shown in Table IV.

*Results:* For each document, we tabulate the overall grade provided by each user in Table V. Most evaluators liked

TABLE V  
OVERALL GRADES FOR IMDB AND DBLP DOCUMENTS

	IMDB		DBLP		
	512B	1KB		512B	1KB
I1	2,2,3	3,4,2	D1	1,1,1	2,1,4
I2	3,2,3	2,2,3	D2	1,2,1	2,2,3
I3	3,1,2	3,2,2	D3	2,1,2	1,2,2
I4	2,2,2	1,1,3	D4	1,1,1	1,2,3

TABLE VI  
IMDB AND DBLP: RESULTS OF TOP-5 EVALUATIONS

	IMDB		DBLP		
	Tags	Actors		Tags	Cite
I1	4	3	D1	5	5
I2	3	2	D2	3	2
I3	5	3	D3	4	3
I4	3	1	D4	3	2

the shorter, 512B summary for the DBLP documents. For the remaining summaries, both IMDB as well as DBLP, the overall impression was mixed. While most people agreed in their comments that the summary was good in providing a bird’s eye view of the document, there were a few criticisms.

First, the inclusion of certain tags (sometimes at the expense of others which were considered more important) were cited as the main reason in lower grades. For example, the tag <COLOURINFO>, while important if the movie was black and white, is of almost no interest otherwise. However, the inclusion of the tag came at the expense of, say, an extra <ACTOR> or the <PLOT> outline. On the other hand, for the DBLP documents, the summaries were by and large satisfactory, but the inclusion of certain tags, like, <URL>, was deemed unimportant in the context of publications (even though <URL> did not come at the expense of any other tag).

Second, the text values included in the summary were deemed by and large satisfactory, but a few cases were criticised. For example, a low grade was given to a DBLP document summary since the user could not see why the citations that had been generated were the most relevant to the publication. We had used the corpus statistics in order to rank the citations in the absence of any other contextual information. Indeed these criticisms indicate that there is a need to go beyond the corpus (even a large corpus as DBLP) in order to accumulate more contextual knowledge. For example, maybe a full text version of the publication and its citations would help in determining which citations are more important to the publication. However, we need to go beyond DBLP to acquire and process this information.

Table VI tabulates the results of the Top-5 Evaluations. The count in each of the columns corresponds to the number of values, which a majority of the 3 evaluators agreed belonged in the top-5. The tables indicate that while the choice of tags were mostly found acceptable, the same cannot be said for the text value samples. The main criticism for the <ACTOR>

values was due to two reasons: i) the wrong actor (or actors) was indeed ranked higher, and ii) the actor ranked higher was not known to the user. An example for the former was “James Cameron”, who was ranked the number 1 actor in the movie “Titanic”. He got a high score since his name is often cited in the document, but not as an actor, but as the film’s director (even though he also acted in the movie). This shows again, the need for more contextual information. As an example of the latter, the actor “Keir Dullea”, the lead actor in the movie “2001: A Space-Odyssey”, was not known to most evaluators even though he was ranked as the top actor for that movie.

In summary, our techniques have proved to be reasonably good in generating short, useful summaries, but nevertheless, can be improved in quality as far as ranking text values is concerned.

## VII. CONCLUSIONS

Our focus in this paper was to provide general-purpose techniques to generate concise, generic summaries automatically for a given XML document. We used a rank-rewrite approach to achieve this. We proposed different ways of ranking text values – with document as context, corpus as context and centroid-query based ranking. We proposed different strategies for rewriting tag-text pairs - enumeration, shortening and sampling. We also described an algorithm to automatically generate a summary of the required size. Finally, we showed through a user study that our methods are applicable and effective in practice and highlighted some of the drawbacks.

There are several directions of future work. For example, improved algorithms for generating the summary could be useful for better coverage. Context identification needs to be more fine-grained and there is a need to acquire “background” information (for example, full text of publications as in DBLP) to improve the ranking of text values.

## REFERENCES

- [1] U. Hahn and I. Mani, “The challenges of automatic summarization,” 2000.
- [2] M. Amini, A. Tombros, N. Usunier, and M. Lalmas, “Learning-based summarisation of XML documents,” *Information Systems*, 2007.
- [3] C. Yu and H. Jagadish, “Schema summarization,” in *Proc. of VLDB*, 2006.
- [4] P. Ferragina, F. Luccio, G. Manzini, and S. Muthukrishnan, “Compressing and searching XML data via two zips,” in *Proc. of WWW*, 2006.
- [5] J. Freire, J. Haritsa, M. Ramanath, P. Roy, and J. Siméon, “StatiX: Making XML count,” in *Proc. of SIGMOD*, 2002.
- [6] L. Lakshmanan, J. Pei, and Y. Zhao, “Qc-trees: An efficient summary structure for semantic OLAP,” in *Proc. of SIGMOD*, 2003.
- [7] R. Saint-Paul, G. Raschia, and N. Mouaddib, “General purpose database summarization,” in *Proc. of VLDB*, 2005.
- [8] W. Dakka and P. Ipeirotis, “Automatic extraction of useful facet hierarchies from text databases,” in *Proc. of ICDE*, 2008.
- [9] D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang, “MEAD - a platform for multidocument multilingual text summarization,” in *LREC 2004*, 2004.