

Xoom: A tool for zooming in and out of XML documents

Maya Ramanath
Max-Planck Institute for Informatics
Campus E1 4
Saarbrücken, Germany
ramanath@mpi-inf.mpg.de

Kondreddi Sarath Kumar
Max-Planck Institute for Informatics
Campus E1 4
Saarbrücken, Germany
skondred@mpi-inf.mpg.de

ABSTRACT

Suppose there is a large corpus of XML documents, each of which describes a movie released in the last 30 years (for example, extracted from IMDB). A movie enthusiast wants to make a list of interesting movies based on various criteria, such as, the genre, lead actors, directors, etc. She first decides to narrow the focus to just thrillers. However, she then has to look into each document individually, since only then is it possible for her to tell whether the combination of actors, directors, etc. interests her. This would be time-consuming if the documents in question contain hundreds of tags each. Instead, she could use our tool *Xoom* (XML-Zoom) which can extract and present the key information in every document. This would drastically cut down the time to go through the documents. She could then use *Xoom* to zoom into specific portions of each of the remaining documents, instead of opening and scanning them from top to bottom. In this proposal, we describe the construction of *Xoom* and outline a demonstration.

1. 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. However, human-centric exploration often boils down to manually inspecting a narrowly focused set of documents. Examples could include complex tasks such as lawyers researching a class of criminal trials and historians wanting to examine the economic consequences World War II to simple tasks such as a movie enthusiast wanting to make a list of interesting movies to watch. The common element in all three cases is that there are several documents to inspect and all these documents focus on a specific topic. It would be much easier to browse through the documents if “bite-sized” summaries of each document or a specific subset of the document could be made available. The user could then decide whether to explore the document in depth or not.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the ACM. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires a fee and/or special permissions from the publisher, ACM. *EDBT 2009*, March 24–26, 2009, Saint Petersburg, Russia. Copyright 2009 ACM 978-1-60558-422-5/09/0003 ...\$5.00

In this proposal, we describe our tool, coined *Xoom* (XML-Zoom), which helps document exploration in two ways. First, it provides generic summaries of the document which helps the user zoom out to a desired granularity. Second, it helps the user zoom in to specific portions of a document to see the most important information available there.

1.1 Zooming in and out

As a concrete example of the type of functionality we would like to provide, consider Figure 1 which shows snapshots from our software. The source (shown on the left) describes the movie “Ocean’s Eleven” (derived from IMDB). All the tags in this document (of which only a very small subset is shown) 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. Concise summaries of 5 and 10 tags (counting only the leaf level tags) are shown in the middle and right-hand side of the figure. Only the most important tags and text values have been retained while unimportant tags (such as `alternate_title`) have been dropped. The result is a bird’s eye view of the source document amenable for a quick glance to filter out unwanted documents.

The key task in zooming (zooming out as well as zooming in), is the ranking of tags and text values. Zooming in requires a ranking of text values belonging to a specific tag which is then displayed when the user asks for it. And zoom out requires a ranking of both tags and text so that the most important tag-text pairs (or spans) are chosen for inclusion in a summary of the desired size. In our system, tags and text values are ranked separately using different scoring methods. Our scoring functions makes use of both the document and the corpus to determine the importance score. The ranked list of tags and text values are then used to either summarize the document or to zoom in to specific portions of it.

Organization. After a brief discussion of related work in Section 2, we outline the architecture of *Xoom* and the techniques we use to rank tags and text in Section 3. We conclude in Section 4 with a description of the proposed demonstration.

2. RELATED WORK

Our techniques have been developed with *data-oriented* XML documents in mind. That is XML documents containing record-like, short pieces of text. As an example, a movie



Figure 1: “Ocean’s Eleven” – Snippet of original document and its 5-element and 10-element summaries. In both the summary snapshots, the numbers in the parentheses before each tag indicate (<no. of children displayed>/<total no. of children>) for that tag.

document could contain titles, actor names, director names, language, etc. Summarization of text documents has a long history (see [4] for an overview of the challenges). Text summarization can be viewed as a ranking problem – sentences (or spans) are ranked according to a notion of importance and then top-ranked sentences are included in the summary. Carrying over this principle of ranking to XML summarization is valid, but since XML documents could contain short pieces of text (such as title, actor names, etc.), it is not possible to reuse the same techniques. Moreover, the presence of explicit structure, in the form of tags requires different techniques of ranking.

To the best of our knowledge, ours is the first summarization work to propose techniques for summarizing data-oriented XML documents where both structure and content are equally important. However, the use of XML markup in text documents to improve summarization quality has been previously studied [1]. These recent techniques still deal with *document-oriented* XML – text documents augmented with XML markup – and not *data-oriented* XML where markup is used as additional input to the document summarization process. Hence, no techniques are developed for summarizing short text values which are required for data-oriented XML.

XML structure summarization has been studied in [7, 2]. But, neither of these proposals take into account the text values. Generating snippets of XML query results is close to our work [5], but our setting is different in that we consider stand-alone XML documents and rank elements without any query bias. Finally, statistical summaries for XML have also been proposed (see for example, [3]), but our zoom-out functionality aims to generate user-readable, semantic summaries.

Our own previous work [6] presented ideas for generic XML document summarization and a user study illustrating its

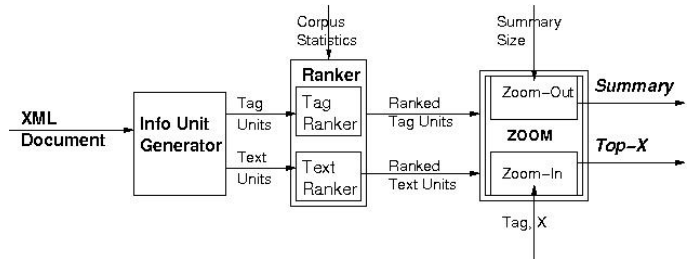


Figure 2: Xoom Architecture

effectiveness. Our recent work has focused on constructing a formal model for summarization. The demo will show this model in action.

3. ZOOMING TECHNIQUES

The architecture of Xoom is shown in Figure 2. The XML document is taken as input into a *Information Unit Generator* module. This module generates two types of information units – tag information units and text information units. Following text summarization techniques, these sets of information units are ranked according to importance by the *Ranker* module which also takes the corpus statistics as input to its scoring functions. The *Zoom* module takes as input the ranked lists of tag and text information units. Based on the desired functionality, that is, zoom-in or zoom-out, it produces a ranked list of text values for a given set of tags, or a generic summary of the desired size.

3.1 Information Unit Generator

The function of the info unit generator module is to separate out tags and text. Tags and text play different roles in a document. Tags provide the structure of documents in a corpus, and the allowable usage may be compactly described by a DTD or XML Schema. On the other hand, text values “instantiate” a document. And so, because of

their different roles, we need to rank them separately with different scoring functions. In addition to separating tags and text values, it is also important to separate text values from each other by grouping text values occurring under the same tag. For example, it makes more sense to compare one actor to another actor to determine which of them is more important, than it is to compare an actor to a production location. Hence the information unit generator generates tag units (basically, the set of unique tags, separate from the text associated with them) and text units (text values grouped according to tag).

3.2 Ranker

3.2.1 Ranking tags

The module for ranking tags takes the tag information units as input and ranks them according to their importance. The most important tags are those which are salient in the corpus. For example, in our movie corpus, the tag `title` is the most salient tag since it occurs in all documents of the corpus – the title sets the context for the remaining portions of the document. Hence `title` is scored the highest. However, while the salient tags tell us a lot about the corpus, we also need to determine tags which may be special to the document. For example, if a movie is an oscar winner, that tag should be ranked high as well, even though it is not salient in the corpus. The tag-ranker produces a score by taking both the salience and the specialty of the tag into account. The advanced user can set a parameter to indicate how to balance the salience score and the specialty score of a tag.

3.2.2 Ranking text

The text ranker scores text values which occur under a given tag (that is, all actors and then all production languages, etc.). First, text values are classified into entities (names, titles, etc.) and plain text (plots, trivia items, alternate versions, etc., which are converted into bag of words). In order to rank entities, we consider their “popularity” (that is, the frequency of occurrence) in the document and the corpus. The more frequently an entity occurs, the more important it is. This holds for many different kinds of entities – for example, George Clooney is mentioned no less than 10 times in the Ocean’s Eleven document, under various tags, such as `trivia_item` and `goof`. Moreover, George Clooney is also popular in the corpus – he acts in several movies and as one of the lead actors in a majority of those movies, is also mentioned frequently in those documents. Hence a combination of his importance in the current movie as well as in other movies gives us a combined importance score. Note that this combination helps to rule out (or rule in) popular actors who merely play a cameo in the current movie. For example, in Ocean’s Twelve, Bruce Willis is not frequent in the document (and hence has low score), but is important in the corpus (where he gets a high score). The advanced user can set a parameter to decide how these scores should be balanced to produce the final score.

In the case of plain text, we make use of the redundancy in the set of text values itself to determine the importance score. For example, suppose we need to rank 10 different trivia items (recall that each trivia item is reduced to a bag of words in order to help in scoring it). Suppose the trivia items talk about George Clooney and Brad Pitt. Then the

trivia item which talks about both of them together would be scored high. In other words, the score of the trivia item is based on the number of high frequency words it contains (here, the frequency is computed within the set of trivia items only).

3.3 Zoom

The Zoom module takes the ranked list of tags and text values as input and uses them to either construct a summary (zoom out) or to provide the user with the desired set of top-x values (zoom in). The zoom in module works in a simple manner. The user simply inputs the number of text values she desires for each tag. The top-ranked text values for the given tags is then chosen and output by the module.

The zoom out procedure is slightly more involved. At first glance, zooming out involves choosing the top ranked tag and text values until the required size of the summary (which is input by the user) is reached. Choosing text values once the tags are given is straightforward (as mentioned in the zoom-in procedure above). However, it is not so straightforward to *automatically* choose the top-ranked tags. To see this, suppose we want to generate a 10 element summary of the movie Ocean’s Eleven. Suppose the tags are ranked in decreasing order as follows: `title`, `production_year`, `actor`, `director` and `colourinfo`. It is straightforward to choose 1 `title` and 1 `production_year` and their corresponding text values, since they occur only once in the original document. However, when it comes to `actor`, there are 68 actors listed in the original document. The entire summary size can be exhausted by simply choosing 8 more actors. But this could hardly be considered a good summary since it does not offer good coverage of the original document. Instead of simply considering the rank of the tags – `actor` is more important than `director` – we additionally need to consider how much more important `actor` is than `director`. Are two `actors` worth 1 `director` or is it 3 `actors`? Our solution considers the magnitude of scores assigned to each tag and chooses the number of tags in proportion to its score. That is, if `actor` has twice the score of `director`, then the number of `actors` and `directors` in the final summary will be in the ratio 2:1. As can be seen from the example shown in Figure 1, this results in summaries with good coverage of important elements for the given space budget.

4. DEMONSTRATION

We have developed `Xoom` as a Java application. The demonstration will use three different corpora – approximately 300,000 movie documents from IMDB¹, approximately 200,000 people documents describing actors, directors, etc. also from IMDB and around 400,000 publication documents from DBLP². Corpus statistics required for scoring (such as entity frequencies and tag frequencies) are previously collected and stored.

The user starts by choosing a corpus and a file. A default zoom-out summary of 10 elements is shown. The user can then choose to take any of the following actions: i) see the original document, ii) increase or decrease the summary size,

¹<http://www.imdb.com>

²<http://dblp.uni-trier.de>

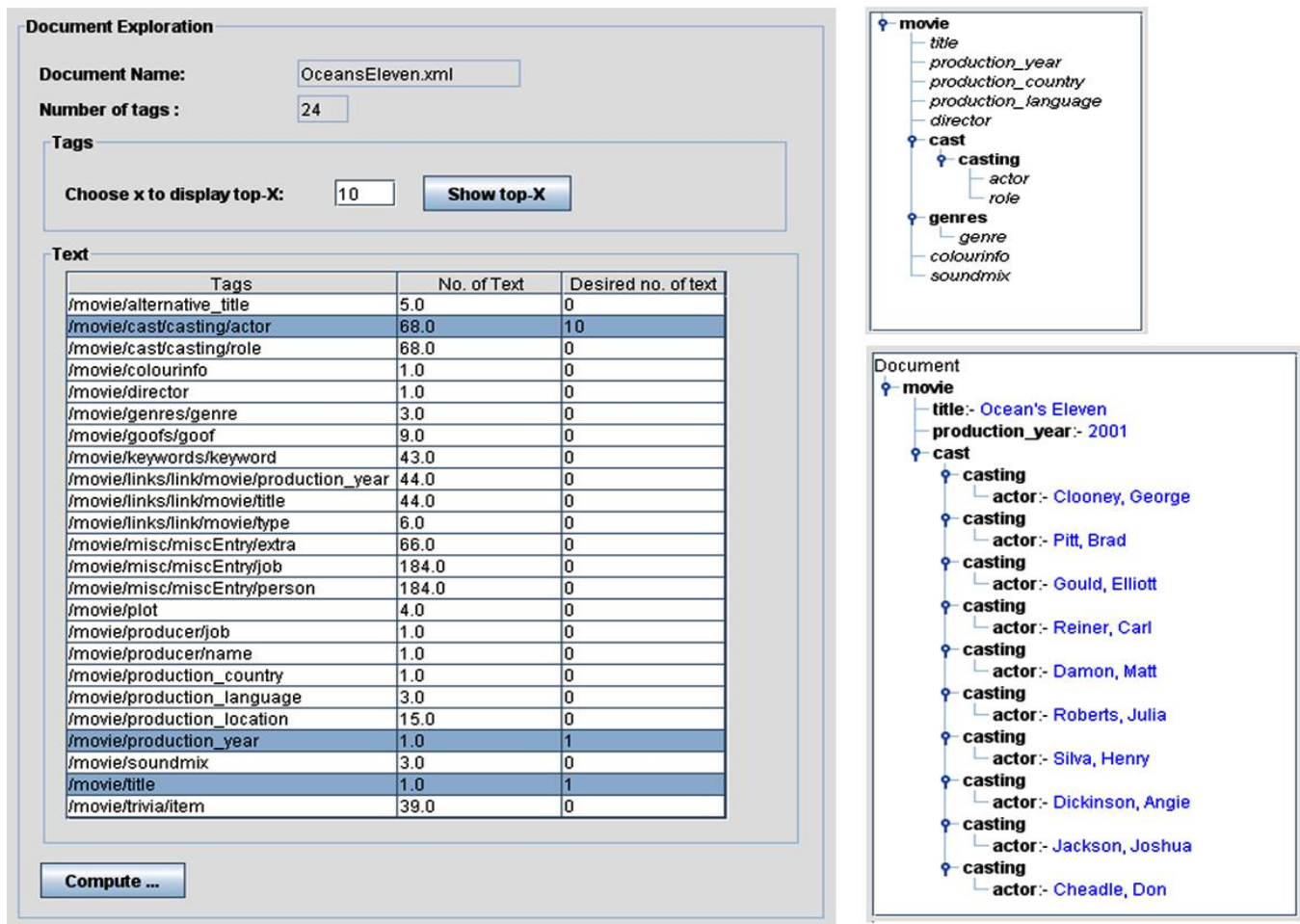


Figure 3: Zooming in: Generating the top-X tags and text values

iii) zoom-in to specific parts of the document or summary by specifying how many top-x values she would like to see.

Previously in Figure 1, we showed the original source, a 5-element summary of the source and another 10-element summary of the source. Though not shown in the screenshot, the user can increase or decrease the size of the summary by pressing a single button or by opening a dialog to specify the number of elements he desires. Figure 4 shows how users can zoom in to specific parts of the document. The dialog on the right hand side has two parts - one for tags and one for text. In the first part for top-x tags, users can specify how many ranked tags she would like to see. The second part for text shows a table with three columns. The first and second columns show the paths available in the document (or the summary, if the user wants to zoom in based on the tags in the current summary) and the total number of text values available for each path respectively. In the third column, users can specify how many text values for each tag should be shown. The results of choosing to zoom in to the top-10 tags (out of a possible 24) and the top-10 actors (out of a possible 68) are shown in the right hand side of the figure (note that these are two independent functions).

5. REFERENCES

- [1] M. Amini, A. Tombros, N. Usunier, and M. Lalmas. Learning-based summarisation of XML documents. *Information Systems*, 2007.
- [2] M. Consens, F. Rizzolo, and A. Vaisman. AxPRE summaries: Exploring the (semi-)structure of xml web collections. In *Proc. of ICDE*, 2008.
- [3] J. Freire, J. Haritsa, M. Ramanath, P. Roy, and J. Siméon. StatiX: Making XML count. In *Proc. of SIGMOD*, 2002.
- [4] U. Hahn and I. Mani. The challenges of automatic summarization. *IEEE Computer*, 11(33), 2000.
- [5] Y. Huang, Z. Liu, and Y. Chen. Query biased snippet generation in XML search. In *Proc. of SIGMOD*, 2008.
- [6] M. Ramanath and K. Sarath Kumar. A rank-rewrite framework for summarizing XML documents. In *Proc. of DBRank*, 2008.
- [7] C. Yu and H.V. Jagadish. Schema summarization. In *Proc. of VLDB*, 2006.