Adding Structure to Unstructured Text

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, Ohio 44242
jmaletic@cs.kent.edu

Michael L. Collard
Department of Computer Science
Kent State University
Kent, Ohio 44242
collard@cs.kent.edu

Abstract

An overview of the authors' research program in document engineering is presented. Underlying techniques are being developed for agile parsing of unstructured and semi-structured text to extract metadata. XML technologies are leveraged in novel ways to support complex querying, analysis, and transformation of large text bases. New methods for difference analysis are being developed to support document evolution and maintenance. Additionally, advanced information retrieval methods, namely latent semantic indexing, in conjunction with clustering techniques are used to extract high level features and concepts from large corpora.

1 Introduction

Our research program is centered on the maintenance and reverse engineering of large legacy software systems. To address this difficult problem we must deal with large numbers of text documents. These documents include source code files in various programming languages and dialects, internal documentation normally written in English, external system documentation of various types (e.g., text, diagrams, tables), and possibility user manuals, bug reports, and version histories.

These documents are all unstructured or semi-structured in nature. We have developed a number of fundamental techniques for querying, analyzing, and transforming documents with the explicit goal of recovering and identifying metadata from unstructured and semi-structured text.

The domain of software documents poses many significant problems due to the sheer amount of text along with its heterogeneous nature. That is, commercial software systems are measured in millions (or 10's of millions) of lines of code. To this end we are investigating efficient and flexible methods that can be applied across different languages and in heterogeneous formats. **Figure 1** presents an overview of the problem and what aspects we are addressing. The main research problems we have investigated are related to the representation of unstructured and semi-structured text in XML, parsing and translation methods to go from raw text to XML, and tools for analysis and transformation.

Our general approach is to use and leverage XML technologies to support storage and extraction of metadata. However, translating unstructured text into XML requires specialized custom built parsers. We utilize new parsing methods based on flexible grammar specifications that allow us to skip over uninteresting or ill formed text. These parsing methods are very robust and extremely efficient which allow them to be applied to very large text bases in a practical manner.

With regards to our research on XML representations of text we have chosen a wholly document view. This is opposed to the more prevalent application of XML for data exchange.

We are continuously developing tools and techniques for querying, analyzing, and transforming both raw text and its more abstract XML representation. This is particularly important in automated feature detection or concept location in raw text. At the raw text level we have successfully used information retrieval methods, namely latent semantic indexing, to cluster document parts and automatically identify high-level concepts in large document bases.

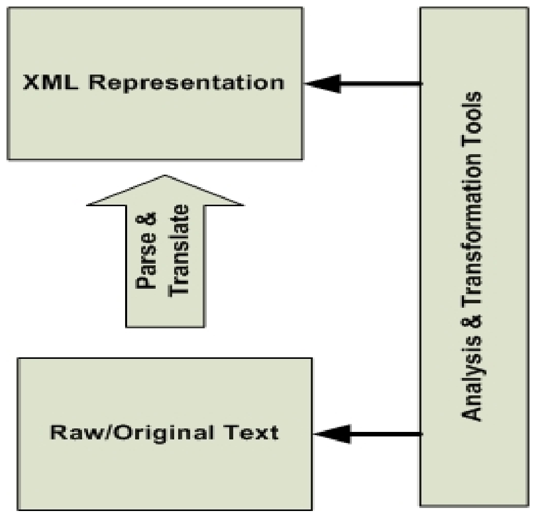

Figure 1. Overview of problem

Finally, we have developed a visualization tool to help support the analysis of large amounts of source code documents. This tool is quite generic with regards to input data and can be applied to any type of text document and other types of data.

This paper is organized to detail the work we've done on a number of major research problems. Section 2 discusses our work with parsing methods; section 3 is on representation issues. Section 4 deals with the analysis tools we've develop to work on raw text and section 5 deals with analysis and translation tools for XML representations. A short description of our information visualization work is in section 6.

We feel that our techniques can be directly applied to other problem domains that have unstructured text documents and in section 7 we briefly describe some proposed research directions along these lines.

2 Agile Parsing Methods

Analysis begins with the extraction of lexical, structural, syntactical, and documentary information from documents (i.e., source code files). Unfortunately, this is more difficult than it should be. On a purely textual level a straightforward lexical approach such as regular expressions can be used. However, robust and efficient regular expressions can be difficult to write especially when determining the matching context. To address this one can make use of current compiler technology however existing compiler-centric parsers have difficulties in the preservation of the original (source-code) text especially with regard to white space, comments, and preprocessor directives. Parsers take a high level AST (Abstract Syntax Tree) view of a program and do not consider these items to be of importance. In addition they are not robust and cannot handle code that is either incomplete or has compilation problems.

In order to analyze all aspects of a source code document we have developed a robust and efficient translator that parses unprocessed C/C++/Java source code and generates srcML, our XML representation of source code (Collard et al. 2003; Maletic et al. 2004) . Our translator preserves all of the source-code text and is able to work with code fragments, e.g., an individual statement. It is very robust and can handle the typical unstable state of source code during development.

Unlike typical parsers, a top-down approach is used that allows for it to be used as part of a stream or pipeline. For srcML this means that the translator can be used a source for SAX processing. The robustness of our translator is due to a selective parsing approach based on the concept of Island Grammars (Moonen 2001). In this approach "islands", i.e., patterns of tokens of specific interest can be discovered even when surrounded by "water", i.e., tokens not of interest. The concept was implemented using the LL(k) compiler generator ANTLR. The top-down parsing of ANTLR was extended to provide stream parsing where XML tokens were inserted into the stream of text tokens as soon as a markup element is identified. This provides for low latency when used as a source for SAX processing.

3 XML Representations

From its original application to text, XML is now used to represent a data of all types. The wide variety of applications of XML has led to two major categories of application: document-oriented and data-oriented. Document-oriented XML uses elements to organize and provide context to text. It has been mainly applied to text documents, e.g., XHTML, DocBook, etc. Mixed content is typically used – where elements contain both text and sub elements. The original document text is preserved (possibly including white space) in its original document order. A full range of XML technologies may than be used, but in particular the schema languages DTD and RelaxNG are preferred over XML Schema. XML Schema is not used due to its inability to handle mixed content and less of a need for data-typing of text elements.

Data-oriented XML stores the textual information in attributes, or in pure elements (i.e., no mixed content). All types of XML technologies may be used, but the deficiencies of DTD and the need to express strong data-type relationships leads to a preference for use of XML Schema or RelaxNG.

We have found a document-oriented approach particularly useful for software engineering applications involving source code. In our *srcML* representation (SouRce-Code Markup Language) (Collard et al. 2003; Collard and Maletic 2004; Collard et al. 2002) XML is used to augment C, C++, and Java source code with syntactic information from the parse tree to add explicit structure and metadata to program source code. Comments, preprocessing information and formatting are preserved and identified for use by other maintenance tools.

The srcML representation is transparent to the original source code text, i.e., there exists a 1-1 mapping between the text of the source-code document and the equivalent srcML document. Source code can be converted from plain text to the srcML representation and back without the loss of any textual information including white space. Locations in the source code can be addressed using XML addressing languages such as XPath and XPointer. In addition to the syntactic meta-data that the srcML elements provide, additional metadata can be embedded as attributes in the syntactic elements, or stored externally with links into the srcML document.

In practice the format has proven itself to be very lightweight. Application to large source code projects, (e.g., Linux kernel), has shown that the srcML representation is on average only 3.5 times the size of the original plain text. This is unlike data-oriented XML representations of source code whose representations are hundreds and even thousands of times larger than the original text, and which often lose essential document information. The speed of our translator is currently at 11,000 lines of source code per second.

The srcML representation has been extended to simultaneously represent multiple versions of a source-code document. This representation, *srcDiff*, (Collard 2004; Maletic and Collard 2004) is a source-code meta-difference format Differences between versions of the document are represented in an XML format based on srcML. The original and modified source code is integrated into a single srcML-based representation with the differences marked in additional XML elements. Both the original and modified versions of the document can be directly extracted from the srcDiff representation. Locations in the both versions and their relationship to the changes can be addressed using XML addressing languages such as XPath and XPointer. This allows direct support for a syntactic description of the differences. Queries may be performed on the difference information involving location of source-code changes, characteristics of the changes, etc. Modeling of differences at this level allows for the validation and transformation of differences. Currently, our srcDiff tool translates at approximately five times the speed of diff. For a project with 60,000 lines of source code it requires less than 14 minutes. This is in sharp contrast to all semantic and most syntactic differencing tools which take this amount of time on a small file.

4 IR and Unstructured Text

We have successfully applied an advanced information retrieval method, namely latent semantic indexing, to a number of problems (Maletic and Marcus 2001; Maletic and Valluri 1999; Marcus and Maletic 2001; 2003; Marcus et al. 2004) dealing with unstructured, heterogeneous, text documents. This novel work deals with identifying abstract concepts and features from large sets of documents.

Latent Semantic Indexing (LSI) (Landauer et al. 1998) is a machine-learning model that induces representations of the meaning of words by analyzing the relation between words and passages in large bodies of text. As a model, LSI's most impressive achievements have been in human language acquisition simulations and in modeling of high-level comprehension phenomena like metaphor understanding, causal inferences and judgments of similarity. For complete details on LSI see (Deerwester et al. 1990). LSI was originally developed in the context of information retrieval as a way of overcoming problems with polysemy and synonymy that occurred with vector space model (VSM) (Salton and McGill 1983) approaches. One of the most successful applications of SVD in information retrieval is the Google search engine (www.google.com).

Most importantly however, is that LSI does not utilize a predefined grammar or vocabulary. As such it can be easily applied different types of unstructured text documents in different languages or formats. Many other information retrieval (IR) and natural language processing (NLP) techniques require a detailed grammar to extract any meaningful information. These grammars are often expensive to construct or brittle. While some of these approaches are more accurate than LSI they cannot compete with regards to its flexibility and low cost of application.

5 XML Applications

The first major application of srcML was to perform queries on source code documents. A combination of the srcML format and common XML tools were used to extract information from source-code documents with results comparable to other heavyweight (full parser-based) parsing approaches according to results obtained using a standard C++ fact extraction benchmark (Collard et al. 2003) . Queries can be performed on source code in a realistic state, i.e., incomplete code, code fragments, etc. The queries are written in XPath and a command-line XPath tool was used to query the source code.

The transparency of our approach makes it ideal for document transformation. Transformation of source code text can be performed at the XML level (Collard and Maletic 2004) with the use of XPath to select the parts of the document that need to be transformed. These non-intrusive transformations of source code can be performed at the XML level using XSLT, DOM, SAX, etc. This allows changes to specific elements in the source code e.g., insertion or removal of an individual statement.

As mentioned previously, the srcML representation allows for the addition of attributes for embedded metadata. However, storing metadata externally is a more flexible approach. In our work (Collard 2003; Collard 2004) associations between entities and regions of source code are represented as an XML model. This model uses the XML linking language XLink for the integration of source models with the source code. Multiple source models, from highly abstract to low-level source-code associations, can be integrated with source code into a single query or transformation.

6 Visualization of Documents

Information visualization techniques are being investigated and applied to visualize the complex abstract relationships and characteristics of large software systems (Maletic et al. 2002; Maletic et al. 2003; Marcus et al. 2003). Our current research prototype, sv3D, was developed specifically as a visualization front-end and is completely independent of any underlying data acquisition or analysis tool. The system accepts simple XML input and be applied to text documents. sv3D allows the user to simultaneously visualize multiple attributes of multiple documents. It supports object level manipulation (rotation, scaling, etc.), filtering, and user selectable mappings of visual features to attributes.

7 Proposed Research

Our research program has given us a large amount of experience and expertise in dealing with and extracting metadata from unstructured text. We feel that many of the fundamental techniques we developed will directly apply to the specific business needs and challenges of industry. Our interest is to identify some of these problems and investigate the applicability of our techniques.

Our techniques are very efficient and scalable to large corpora. The parsing techniques are novel and provide a means to add structure to unstructured documents in various domains.

8 Bio Sketches

Dr. Jonathan I. Maletic is an Associate Professor of Computer Science at Kent State University. He has authored nearly 60 published papers and is an internationally recognized researcher in software engineering and document engineering. Dr. Maletic is on the Steering Committee of the ACM Symposium on Document Engineering. He completed the Ph.D. in 1995 from Wayne State University. His web site is at www.cs.kent.edu/~jmaletic/.

Dr. Michael L. Collard recently graduated (2004) from Kent State University with the Ph.D. He is currently a lecturer within the Department of Computer Science. Dr. Collard's research interests are in document-oriented approaches to single and multiple versions of source code. He is a member of the Program Committee for the 5th ACM Symposium on Document Engineering (DocEng) 2005. See www.cs.kent.edu/~collard/.

References

Collard, M. L., (2003), "An Infrastructure to Support Meta-Differencing and Refactoring of Source Code", in Proceedings of 18th IEEE International Conference on Automated Software Engineering (ASE '03), October 6-10, pp. 377-380.

Collard, M. L., (2004), *Meta-Differencing: An Infrastructure for Source Code Difference Analysis*, Kent State University, Kent, Ohio USA, Ph.D. Dissertation Thesis.

Collard, M. L., Kagdi, H. H., and Maletic, J. I., (2003), "An XML-Based Lightweight C++ Fact Extractor", in Proceedings of 11th IEEE International Workshop on Program Comprehension (IWPC'03), Portland, OR, May 10-11, pp. 134-143.

Collard, M. L. and Maletic, J. I., (2004), "Document-Oriented Source Code Transformation using XML", in Proceedings of 1st International Workshop on Software Evolution Transformation (SET'04), Delft, The Netherlands, Nov. 9, pp. 11-14.

Collard, M. L., Maletic, J. I., and Marcus, A., (2002), "Supporting Document and Data Views of Source Code", in Proceedings of ACM Symposium on Document Engineering (DocEng'02), McLean VA, November 8-9, pp. 34-41.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., (1990), "Indexing by Latent Semantic Analysis", *Journal of the American Society for Information Science*, vol. 41, pp. 391-407.

Landauer, T. K., Foltz, P. W., and Laham, D., (1998), "An Introduction to Latent Semantic Analysis", *Discourse Processes*, vol. 25, no. 2&3, pp. 259-284.

Maletic, J. I. and Collard, M. L., (2004), "Supporting Source Code Difference Analysis", in Proceedings of IEEE International Conference on Software Maintenance (ICSM'04), Chicago, Illinois, September 11-17, pp. 210-219.

Maletic, J. I., Collard, M. L., and Kagdi, H. H., (2004), "Leveraging XML Technologies in Developing Program Analysis Tools", in Proceedings of 4th International Workshop on Adoption-Centric Software Engineering (ACSE'04), Edinburgh, Scotland, May 25, pp. 80-85.

Maletic, J. I. and Marcus, A., (2001), "Supporting Program Comprehension Using Semantic and Structural Information", in Proceedings of 23rd International Conference on Software Engineering (ICSE'01), Toronto, CA, May 12-19, pp. 103-112.

Maletic, J. I., Marcus, A., and Collard, M. L., (2002), "A Task Oriented View of Software Visualization", in Proceedings of 1st IEEE Workshop of Visualizing Software for Understanding and Analysis (VISSOFT'02), Paris, France, June 26, pp. 32-40.

Maletic, J. I., Marcus, A., and Feng, L., (2003), "Source Viewer 3D (sv3D) - A Framework for Software Visualization", in Proceedings of 25th IEEE/ACM International Conference on Software Engineering (ICSE'03), Portland, OR, May 3-10, pp. 812-813.

Maletic, J. I. and Valluri, N., (1999), "Automatic Software Clustering via Latent Semantic Analysis", in Proceedings of 14th IEEE International Conference on Automated Software Engineering (ASE'99), Cocoa Beach Florida, Oct., pp. 251-254.

Marcus, A., Feng, L., and Maletic, J. I., (2003), "3D Representations for Software Visualization", in Proceedings of 1st ACM Symposium on Software Visualization (SoftVis'03), San Diego, CA, June 11-13, pp. 27-36.

Marcus, A. and Maletic, J. I., (2001), "Identification of High-Level Concept Clones in Source Code", in Proceedings of Automated Software Engineering (ASE'01), San Diego, CA, Nov. 26-29, pp. 107-114.

Marcus, A. and Maletic, J. I., (2003), "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing", in Proceedings of 25th IEEE/ACM International Conference on Software Engineering (ICSE'03), Portland, OR, May 3-10, pp. 125-137.

Marcus, A., Sergeyev, A., Rajlich, V., and Maletic, J. I., (2004), "Concept Location using Latent Semantic Analysis", in Proceedings of 11th IEEE Working Conference on Reverse Engineering (WCRE'04), Delft, The Netherlands, Nov. 9-12, pp. 214-223.

Moonen, L., (2001), "Generating Robust Parsers using Island Grammars", in Proceedings of 8th IEEE Working Conference on Reverse Engineering (WCRE'01), Suttgart, Germany, Oct. 2-5, pp. 13-24.

Salton, G. and McGill, M.,(1983),*Introduction to Modern Information Retrival*, McGraw-Hill.