Information extraction from online XML-encoded documents

Patricia Lutsky
ArborText, Inc.
1000 Victors Way
Ann Arbor, MI 48108
USA
plutsky@arbortext.com

Abstract

Online reference documents tend to be semi-formatted in that they contain repeated sections with similar structure, and have free-text inside each section. XML (extensible markup language) enables document designers to design rich tag sets where tags for section headings contain information about each section. This contextual information, coupled with the fact that the free-text sections of the documents use a limited sublanguage, mean that simple natural-language-based techniques can be used to extract facts from semi-formatted online documents. The SIFT document parser system has demonstrated results for this type of extraction in the area of software testing.

Introduction

Documents such as reference manuals are increasingly available online. These reference documents are often semi-formatted; they are organized into specific repeated sections for each entity being described with free-text in each section. Work has already been done on extracting information from online HTML documents. (Hsu and Yih 1997) This extraction requires a good deal of inferring of the meaning of the document markup. XML, the Extensible Markup Language, allows the markup for section headings to be richer than the tags that HTML provides. These heading tags can contain valuable contextual and structural information that can aid semantic processing in a natural language processing system. With domain specific markup, the step of inferring semantic document structure goes away.

XML

Extensible Markup Language (XML) is a simple dialect of SGML (standard generalized markup language). "The goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML." (World Wide Web Consortium 1998) A key advantage of XML over HTML is that XML allows users to create custom tags. Also, it separates content information from presentation formats.

XML provides a mechanism, the document type declaration, for defining constraints on the logical structure of a document. Within the definition is the markup declaration, which declares the type of tags available for a class of documents. Tags can have attributes associated with them, where domain-specific information can be specified in the markup. For instance, in an operating system routine manual (Digital Equipment Corporation 1988), there can be a <ROUTINE> tag for the headings of routine descriptions and an <ARGITEM> tag for the description of each argument of a routine. Further, the <ARGITEM> tag can contain descriptive information about the argument such as name, logical type, physical type, whether it is read-only, read-write, or write-only, and what parameter passing method is used for it.

XML has been developed by a working group of the World Wide Web Consortium and has been designed for ease of implementation and for interoperability with both SGML and HTML. Major vendors are providing support for XML in current or future versions of their web products.

Information Extraction

Information extraction concerns extracting specific facts from documents. Unlike automated indexing, the precise semantic content of sentences must be understood to extract facts. In online reference documents, there are often specific types of facts that would be valuable if they could be extracted from the documents.

For software testing, concepts that are relevant to automatic test case generation are valuable. The job of a software tester is to compare a software system to an expectation of how it should perform, usually in the form of an English document. Discrepancies indicate a change should be made to the software, to the document, or to both. If test-related facts can be automatically extracted from documents and used to generate tests, testers' time can be focused on the more complex tests. Test-related facts for operating system testing include: formatting and value constraints for parameters, and prerequisites for operations, such as operations that require that privileges be set. Other applications will have other types of information to extract.

Parsing Semi-structured Documents in XML

XML tags allow for domain-specific information in section headings and this context information is sufficient for information extraction. As described in Section 2, the following tags for descriptions of arguments to an OpenVMS (Digital Equipment Corporation 1988) operating system routine include the name, logical type (vms-usage), type, access method, and parameter passing mechanism.

```
<ARGITEM
  name="pidadr"
  vms-usage ="process_id"
  type="longword (unsigned)"
  access ="write only"
  mechanism ="by reference"/>

<ARGITEM
  name ="image"
  vms-usage ="logical_name"
  type="character-coded text string"
  access="read only"
  mechanism ="by descriptor"/>
```

This information is used to generate a header in the document. It is also used to provide contextual information to a document parser, providing much more information than a simple <H1> or <H2> HTML tag would.

For example, if the sentence

> All undefined bits in the longword must be 0

occurs in the description of the pidadr argument that is headed by the above pidadr ARGITEM, the SIFT document parser would know from the

> type="longword (unsigned)"

information in the ARGITEM tag that the type of the argument is longword(unsigned). Then, when the parser identifies the referent of "the longword," the heading correctly directs the semantic processing to choose the argument that is being described.

Another example is the processing of the sentence

> SYSPRV is required to specify the system directory table

Here, the argument structure for the verb "specify" indicates that it requires a value and the recipient of the value. "the system directory table" is the value, but there is no explicit mention of the recipient of the value. It is the OpenVMS parameter that is being described, partab, and the parser can ascertain that from the context information that is contained in the ARGITEM tag. Further, the vms-usage information would indicate that "the system directory table" is a valid value for the type of this parameter.

Tags for other section headings may provide valuable information as well. For example, one OpenVMS data type is the *item list*: a list structure where each piece of information is represented in an item, and different item codes are relevant for different parameters. If the sentence

> This item code may appear anywhere in the item list.

is contained in a section with headings

```
<ARGITEM
  name = "itmlst"
  vms-usage="item_list_3"
  type="longword (unsigned)"
  access="read only"
  mechanism="by reference"/>

<PARAMITEM
  name = "LNM$_TABLE"
  type="item_code"/>
```

The sentence processing resolves the reference for "this item code" to be LNM$_TABLE from the name value of the PARAMITEM tag, and resolves the reference for "the item list" from the name value of the ARGITEM tag.

To further simplify parsing of online reference manuals, the text inside the free-text sections uses a restricted sublanguage (Grishman and Kittredge 1986) (Kittredge and Lehrberger 1982), so simple domain-specific parsing techniques can be effective. A sublanguage is a semantically constrained version of a natural language spoken by a particular group of people. A sublanguage is not a subset of a natural language, but rather has its own grammar that reflects the way the group of people communicates. For reference manuals, linguistic constructs such as puns and fanciful metaphors will not be used. The tone is simple and uniform. Specific concepts tend to be described the same way throughout a document. In fact, as I will show, certain concepts are expressed in a limited number of identifiable sentence types.

SIFT

SIFT (Lutsky 1998), which stands for "specification information from text," is a natural-language-based information extraction tool that can extract test-related information from the free-text portion of semi-formatted documents. It uses simple techniques for information extraction from texts, yet has been able to improve tests for the domains of OpenVMS operating system testing (Lutsky 1994) and XCON expert system database testing (Barker and O'Connor 1989) (Schlimmer

1991).

Figure 1 shows the architecture of SIFT. The input to the system is software documentation; the output is additions to a test system. These additions will take different forms depending on the test system. SIFT contains four modules. These include both domain-independent components (a testing knowledge module and a general purpose parser) and domain-dependent components (a sublanguage grammar and a domain model). The domain-independent modules are robust enough to work for multiple domains so that a tester wanting to use the SIFT document parser in a new domain only has to develop the domain model and sublanguage grammar based on the specific way that information is expressed in the documents.

Input and Output

The input to SIFT is a semi-formatted online document. SIFT is most valuable for documents that have a long lifespan, where the system being described will be continually adding new features. That way, the grammar can be written once, and then run repeatedly on the document as it changes with new versions.

The output of the SIFT document parser is a canonical form of the relevant information that can be easily translated into the correct format for additions to the test system. For example, if allowable limits on an integer parameter are being extracted, the sentence

> The maximum value you can specify with the BUFQUO argument is 65355.

is translated to a canonical form such as

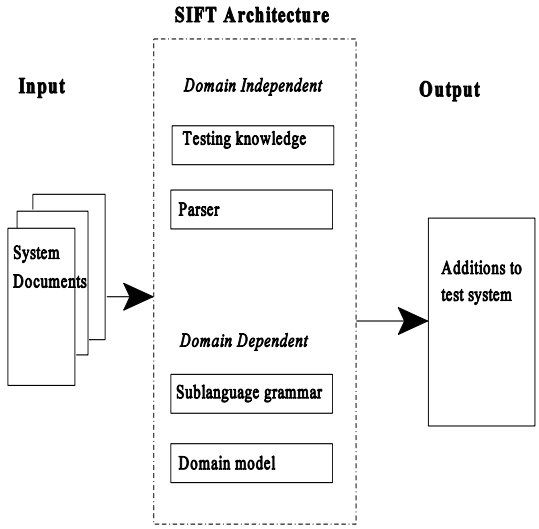

Figure 1. Architecture of SIFT

> The maximum value for BUFQUO is 65355.

and an encoded canonical form such as

> (maximum_value BUFQUO 65355)

that can be mechanically translated into the format used by the testing system. The required output format is described as part of the sublanguage grammar for the domain.

Sentence types

In SIFT, a *sentence type* is a group of sentences of a sublanguage that are structurally the same. For example, the following sentences are of the same type:

> The box is on the counter
> The pen is on a box
> The shoe is under the counter

A sentence type can be identified with a grammar as the following description of the three example sentences:

> NP is PREP NP

Components of the grammar may be domain-specific as in the following sentence type description:

> PRIVILEGE is required to VERB NP

where PRIVILEGE is a noun phrase describing an OpenVMS operating system privilege. A corresponding sentence is

> SYSNAM privilege is required to specify executive or kernel mode access for a logical name table.

SIFT modules

The domain-independent general-purpose parser has four parsing steps, and the sublanguage grammar parts correspond to them. The four parts of the grammar are:

- preprocessing directives
- phrase structure grammar
- heuristics for identifying sentence types
- heuristics for extracting semantic information.

The grammar only covers the structures of the sentence types that are used to convey testable information in the target document. It reflects the sentence type grammar as in

NP MODAL CONTAIN-VERB no more than NUMBER characters.

for sentences such as

The string may contain no more than 31 characters.

An equivalence name can contain no more than 255 characters.

The corresponding parser is based on an ATN (augmented transition net) (Woods 1970). It looks at each appropriate section of the document and its contextual information. Within each section, it works on a sentence-by-sentence basis, although a "sentence" of the sublanguage may only be a sentence fragment of English. If it is able to parse a sentence and derive a semantic representation for it, it returns the corresponding semantic expression. If not, it simply ignores the sentence and moves on to the next one.

The phrase structure output is a parse tree. Its format is domain specific and may contain semantic information as well as syntactic groupings.

This tree is passed to two types of heuristics: one set of heuristics identifies the sentence type based on the phrase structure description of the sentence and the other set of heuristics converts the semantic contents of the sentence to a domain-specific useful form.

This second set of heuristics takes advantage of the XML-encoded knowledge in the document for semantic processing. As shown in Section 4, this knowledge can be used in reference resolution and in determination of the correct values for the argument structure of lexical items. It can also help with missing references in informal documents, such as if the sentence fragment

Only relevant for software.

occurs in a specification document. The missing sentence subject is obtained from the heading tag of the section for the attribute that is being described.

The domain model is based on a linguistic formalism, the generative lexicon (Pustejovsky 1995). By using a linguistic formalism, the language-related information is combined with general semantics of domain model entries.

The other domain-independent module is the testing knowledge module. It contains general knowledge about testing and common test case scenarios. Currently it is used only when translating canonical sentences into the correct format for additions to the test system. In the future its role might be expanded to assist further in SIFT's processing of the document.

Results

The SIFT system has been measured by comparing the test information that it can extract from the reference pages of the OpenVMS System Services Reference Manual (Digital Equipment Corporation 1988) to a gold standard set of tests that were written by a human tester. (This document was actually written using SGML, but it could have been encoded in XML.) The tests, with a total of 174 test cases, are for the subset of the OpenVMS operating system that concerns logical name translation. SIFT can extract information that accounts for 25 test cases, or 14% of the gold standard. Fourteen percent is quite good for automatic test generation. It represents a significant time saving for testers and would allow testers to concentrate on the harder tests.

SIFT can extract facts about argument restrictions, such as

This attribute may be specified only for process-private logical name tables.

LNM$_CHAIN must be the last item code in the current item list.

It can also extract facts about required arguments, and required privileges.

While SIFT has been used primarily for software system testing, the general parsing technique can be adapted for use on other types of semi-structured online documents. As another example, consider extraction of information from use cases. Use cases are tools used in object-oriented design that model the interaction between different agents (Jacobson 1992). Use cases are part of object-oriented analysis because they show how the objects are used in scenarios. Example tasks for SIFT are to locate all use cases that involve a specific type of operation or to extract information about use case steps with specific prerequisites.

Conclusions

Valuable facts can be extracted from online documents encoded in XML. The domain-specific tags that XML enables allows documents to contain contextual information that facilitates information extraction from semi-formatted documents. And, since the free-text of the document is written in a restricted sublanguage, simple domain-oriented parsing techniques are sufficient for parsing. As more and more world wide web documents are encoded in XML, rich domain-oriented tag sets will be developed that will ease information extraction from these documents; tag set design teams can include information extraction experts in order to insure the maximum usefulness of the document markup.

References

Barker, V. & O'Connor, D. 1989. Expert systems for configuration at DIGITAL: XCON and beyond. Communications of the ACM, 32. 298-318.

Digital Equipment Corporation 1988. OpenVMS System Services Reference Manual Version 5.0.

Grishman, R. & Kittredge R. (Eds.). 1986. Analyzing language in restricted domains: Sublanguage description and processing. Hillsdale, NJ:Lawrence Erlbaum Associates.

Hsu, J. & Yih, W. 1997. Template-based information mining from html documents. Proceedings of the AAAI 97 Conference. 256-262.

Jacobson, I. 1992. Object-Oriented Software Engineering. ACM Press.

Kittredge, R., & Lehrberger, J. (Eds.). 1982. Sublanguage: Studies of language in restricted semantic domains. New York:Walter de Gruyter.

Lutsky, P. 1994. Using a document parser to automate software testing. Proceedings of the 1994 ACM Symposium on Applied Computing. 59-63.

Lutsky, P. 1998. Automating natural-language-based processes of software testing. Unpublished PhD dissertation, Brandeis University.

Pustejovsky, J. 1995. The Generative Lexicon. Cambridge, MA:MIT Press.

Schlimmer, J. 1991. Learning meta knowledge for database checking. Proceedings of AAAI 91. 335-340.

World Wide Web Consortium 1998. Extensible Markup Language (XML) 1.0. W3C Recommendation 10-February-1998. http://www.w3.org/TR/REC-xml.

Woods, W. 1970. Transition network grammars for natural language analysis. Computational Linguistics. 13. 591-606.