6-2002

Semi-Automatic Content Extraction from Specifications

Krishnaprasad Thirunarayan
Wright State University - Main Campus, t.k.prasad@wright.edu

Aaron Berkovich

Dan Z. Sokol

Semi-Automatic Content Extraction from Specifications*

Krishnaprasad Thirunarayan[1], Aaron Berkovich[2], and Dan Sokol[2]

[1] Department of Computer Science and Engineering,
Wright State University, 3640, Col. Glenn Hwy, Dayton, Ohio-45435, USA.
tkprasad@cs.wright.edu
http://www.cs.wright.edu/~tkprasad
[2] Cohesia Corporation,
First National Plaza, Suite 1414, Dayton, Ohio-45402, USA.
{aberkovich, dzsokol}@cohesia.com
http://www.cohesia.com

Abstract. Specifications are critical to companies involved in complex manufacturing. The constant reading, reviewing, and analysis of materials and process specifications is extremely labor-intensive, quality-impacting, and time-consuming. A conceptual design for a tool that provides computer-assistance in the interpretation of specification requirements has been created and a strategy for semantic-markup, which is the overlaying of abstract syntax ("the essence") on the text, has been developed. The solution is based on the techniques for Information Extraction and the XML technology, and it captures the specification content within a semantic ontology. The working prototype of the tool being built will serve as the foundation for potential full-scale commercialization.

1 Background

Materials and process specifications are a fact of life for any organization involved in complex manufacturing, particularly for companies in industries such as aerospace, automotive, and materials. Specifications ("specs") are often comprehensive and voluminous covering hundreds of different key characteristics such as the material properties, testing parameters, marking requirements, etc. Specs are basically presentations of numeric data with accompanying text and graphics that explain the quantitative information [1, 2].

To illustrate the impact that specs have on the cost of a product, one only needs to consider the on-going activities within a typical materials producing organization. Sales personnel read the specs at the time of design to identify every processing step and component that impacts the price. Design Engineers refer to specs to determine the requirements for creating a design. Manufacturing Engineers analyze spec requirements to define the appropriate operations

* This work was supported in part by NSF SBIR Phases I and II Grants DMI-0078525 (1999-2002). It does not necessarily reflect the opinions of NSF.

and detailed data to include in the process plan. Quality Assurance personnel interpret the specs to define the acceptance limits and methods required for inspecting and testing. The use of paper-based specs is extremely labor-intensive, quality-impacting, and time-consuming.

Specs historically have been handled as simple textual documents. In order to do any meaningful analysis, comparison, and integration, one should extract numerical, tabular, or graphical data and operate on that data in the fashion intended by its meaning.

Typically, a spec issued by societies such as SAE, ASTM, AMS, etc is identified by the name of the issuing organization, title, spec number, issue/revision date, etc. The spec describes requirements on the processing of a material (alloy) in the mill, and the capabilities that it should possess eventually. The former includes its composition, melt method used, heat treatment, dimensions, marking and packaging requirements, etc. The latter is ascertained using tests such as tensile test, stress rupture test, macro- and micro-examinations, hardness test, etc. As part of defining an "intelligent" structure for specs, Cohesia created the Specification Definition Representation (SDR) as an ontology to articulate the semantic view of the components that comprise a spec, and capture the user's interpretation of it [3]. SDR introduced constructs such as *Procedures* to indicate boundaries for standards requirements such as chemical composition, tensile test, melt method, etc. Procedures are composed of elemental *Characteristics* that describe the requirements that are essential for performing the associated process (e.g., carbon content, yield strength, minimum temperature range, etc). In fact, a procedure encapsulates collections of characteristic-value pairs glued together using suitable connectives. The SDR technology has been incorporated into a commercial software system called MASS (Management and Application of Specifications and Standards). SDR permits defining a common library of industry terms (domain library) and a consistent representation structure that allows MASS to retrieve, compare, and combine specifications to drive manufacturing process.

Users of MASS capture their interpretation of spec requirements by using an intellectual process of reviewing the hard copy specs and manually converting the data into SDR. The SDR is a simple low-level tree language that is efficient for symbolic manipulation, but is very removed from the spec text. In order to raise the level of abstraction, Specification Definition Language (SDL) was designed and implemented. The Specification Studio is an Integrated Development Environment (IDE) for creating and editing specifications in SDL, and compiling them into an equivalent lower-level SDR for use by MASS. The IDE provides a convenient interface to search and use the Domain Library of controlled terminology. From experience, the analysts have observed an emergence of a pattern of "semantic markup", which is composed into the more rigorous SDL. This process of conversion is still laborious, and is subject to the competence of the individual performing the conversion.

2 Problem Statement

The focus of manual content extraction is to convert paper-based specs (using an appropriate domain library which defines standard vocabulary) into an "equivalent" formalized description in SDL. The extractor relies on his/her experience in interpreting the specs, and for rendering it in the SDL form using domain library vocabulary. Even though there is no rigid requirement either on the technical vocabulary used, or on the format, related specs with common origin share similar structure.

Semi-automatic content extraction involves recognition of phrases in spec that are associated with requirements, and subsequent synthesis of SDL fragments, to assist an extractor. Conceptually, this can be carried out in two steps: (a) Automatically recognize domain library terms present in spec and mark them up appropriately, and (b) Manually organize this information to be able to generate SDL from it. The entire approach is ontology driven, since the ontology is captured by the SDL and the domain library of terms. Semi-automatic approach was explored to improve the quality and efficiency of extractions by minimizing transcription errors, and by automating some of the routine mechanical tasks. A realistic yardstick of success is the extent to which mechanical and routine aspects of extraction can be codified and automated, while simultaneously deferring the more difficult and irregular portions to a human extractor.

This paper describes the approach taken, the progress made, and hurdles encountered in our efforts.

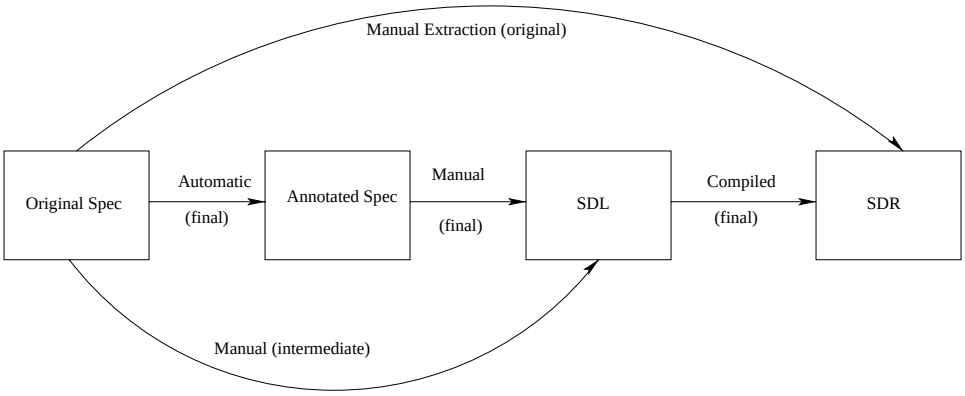

Fig. 1. Semi-automatic Approach to Semantic Markup

3 Related Work in Information Extraction

It is useful to view specs in terms of the "writing styles" [4].

Structured Text: Database Tables, Weather reports found on the web where the fields are delimited with HTML tags, etc.

Free Text: Grammatical texts such as newspaper articles, magazine stories, etc, that contain complete English sentences.

Semi-structured Text: Ungrammatical texts such as Advertisements, Seminar announcements, etc.

We reviewed research on Information Extraction pertinent to content extraction. These technologies included pattern matching systems [4, 7, 8], sentence analyzers [9], automated dictionary builders [4, 10, 13, 21], W3C related initiatives [20] etc. The DARPA-sponsored Message Understanding Conferences (MUC) provided a major source of insight into the progress in information extraction technology [5]. The MUC tasks consisted of skimming through free-text articles in a limited domain, and filling in well-specified templates.

IE technology has evolved greatly in the last decade and there are also several commercial products and search engines that classify documents based on the subject matter. Unfortunately, these tools could not be used directly because there is significant difference between processing general topic documents and highly technical documents: (1) Specs usually contain much more domain specific vocabulary. (2) Ambiguities in specs are of a different nature than those in general-topic documents. (3) Technical terms are often created by combining existing terms [15].

4 Overall Approach

In order to understand the theoretical limitations of our strategy, and the practical problems to be overcome, the following approach was taken:

- Develop a catalog of ambiguities present in specs that can stymie even a human extractor. This catalog sheds light on situations impossible to deal with automatically, and that can potentially jeopardize the soundness of any extraction algorithm.
- Investigate the nature of association between the original spec and its translation that facilitates semi-automatic extraction and verification.
- Improve domain library searches and make it robust with respect to semantics preserving variations.
- State a practically important variant of the general extraction problem that can be used to demonstrate semi-automatic techniques.
- Study the role and application of XML technology in transforming a spec [14–17].
- Finally, describe practical problems to be tackled for semi-automatic fine-grained extractions of extant specs into SDL.

5 Details

We attempted to systematically evolve the manual extraction tool (Spec Studio) into a computer-assisted extraction tool with focus on improving extractor

productivity. The approach taken has an "industrial" rather than "academic" flavor because we attempted to deal with real-world specs rather than contrived ones.

5.1 Extraction Ambiguity

Informally, ambiguity means "open to interpretation". Ideally, one should have a unique interpretation for a spec, but in reality, a spec can be incomplete (due to missing information) or inconsistent (due to over-specification or multiple definition), or unclear (due to convoluted wording).

Manual extraction consists of: (1) Identifying and understanding requirements described in a spec, typically written in English. (2) Mapping specs terminology into Domain Library Terms (DLTs). (3) Identifying appropriate SDL connectives to tie the requirements together.

Problems arising in carrying out the above tasks are called extraction ambiguities. The motivation for studying these is to better understand aspects that are outside the realm of automation, to develop a nomenclature for describing ambiguity in the context of extraction from specs, and to build concrete examples to assist an extractor trainee. We skip the details and illustrative examples, for brevity.

5.2 Literal Translation

Conceptually, every piece of information in SDL and SDR translation owes its existence to a phrase in spec (and possibly, in domain library). Thus, for verification (proofing) purposes, it is important to maintain a one-to-one correspondence between fragments of spec and its translation. Such translations are said to be literal.

The existence of a literal translation depends on the target language and the translation scheme. For example, SDR extractions of fine granularity cannot be literal, while SDL extractions can be made literal to a reasonable extent. To see this dependence, consider the sources of non-linearity: (1) The extractions can contain duplicated information that are shared in spec. (2) The extractions can rearrange information present in spec. (3) Tables and footnotes abbreviate information in irregular and complex ways.

Further, a semi-automatic approach to extraction is feasible only if the automatically generated partial translations are intelligible to the human extractor in the context of the original spec, for subsequent modification.

5.3 Domain Library Search Engine

In the past, an extractor mapped a phrase associated with requirements in a spec to a Domain Library Term (DLT) by guessing the corresponding term and verifying it against the domain library. A typical domain library[3] has about

[3] Cohesia creates and maintains domain libraries for in-house use and for use by its clients such as GE, Alcoa, Allvac, etc.

10,000 phrases. The effectiveness of the search can be substantially improved by having the Spec Studio accept an arbitrary phrase and automatically determine the relevant DLT(s) contained in it. To this end, a domain library search engine was developed that improves the hit rate and enhances the flexibility of DL searches, while balancing recall with precision [18, 19].

Algorithmic details: Expression matcher. The algorithm was designed with the following observations in mind: (1) An input phrase may contain multiple DLTs. (2) The words of a DLT contained in the input phrase may not appear contiguous. (3) Consonants are significant, and "correct" spellings may differ in the vowels. (4) Robustness with respect to misspellings such as transposition of letters or missing vowels is desirable. (5) Ordinary stemmers do not work for the words appearing in the DLTs. Instead, customized tables of prefixes and suffixes were developed to deal with situations that arise in practice. (6) The algorithm was skewed towards recall than precision as a human extractor is overseeing the outcome.

The overall goal of the matching algorithm is to map a phrase to a list of approximately semantically equivalent DLTs. Its main steps are sketched below.

```
List[Phrase] dl, dlwm, inwm;
Phrase ip;     Integer mt;
List[Phrase] dlts;

Step 1: Read list of Domain Library terms (phrases) into dl.
Step 2: Tokenize and construct list of (non-stop) words in the
        domain library terms into dlwm, maintaining links from
        each word back to the DLTs it appears in.
Step 3: Read-in input phrase and match-threshold into ip and mt,
        respectively.
Step 4: Tokenize and construct list of non-stop words into inwm.
Step 5: Determine matching words in dlwn corresponding to each word
        in inwm.
Step 6: Construct list of DLTs that contain these matching words.
        Evaluate each DLT and filter good approximations included in ip.
```

A word is matched at four levels: exact, case-insensitive, normalized (based on consonants), and sorted normalized (based on sorted string of consonants). The exact match requires the case and whitespaces to match. Case-insensitive match converts the inputs to lowercase prior to matching. Normalized match deletes all vowels and special characters (such as "-", "?", etc) prior to matching. Sorted normalized match orders the consonant string before matching. If these checks do not yield a match, the strings are then tested for variations in prefixes and suffixes. For this purpose, sorted normalized representations are compared and common letters are eliminated. Of the remaining letters, if they constitute differences accountable using a customized table of viable suffixes or prefixes, then a

match is declared. (Note that the general algorithm also deals with abbreviations by expanding them and aliases by normalizing them to preferred terms.)

5.4 Automatic Tagging

The text of a spec can be annotated to different levels of detail in order to make explicit mechanically processable information. The following table summarizes the various extraction methods (ontologies[4]) used in the industry (and companies such as GE, Alcoa, etc) today, reflecting the different granularity of extractions — from the most abstract to the very fine:

Method	Qualifiers	Requirements	Procedures	References
D	Spec Class Only	In notes	Not used	In notes
C	Spec Class, Product, Alloy	In notes	Not used	In notes
B	Many Qualifiers	In CV pairs and notes	Used	Retrieved
A	Many Qualifiers	In CV pairs, etc	Used	Retrieved

In order to understand the practical problems with semi-automatic extraction, we attempted to automate Method C Approach to Extraction, which involves generating SDL form that contains Qualified Notes labeled with a Procedure name or a Paragraph heading and the note bodies containing paragraphs with information relevant to the label. The qualifiers are the characteristics — product, product type, spec class, and alloy. Later, we wish to attempt Method B Approach to Extraction.

Two implementations of Method C Approach to extraction were pursued: the first involved direct translation of the spec into SDL, and the second involved indirect translation of the spec into SDL via an XML Master. The former implementation enabled convenient integration into the existing Spec Studio, while the latter implementation enabled applying XML Technology to further enrich literal translation strategy by embedding extraction fragments into the original spec. In other words, the spec can be annotated using XML-tags (derived from the ontology and the DLTs) that abstract the requirements, and keep the spec fragment and its abstraction tightly coupled. Subsequently, suitable views of the XML source can be generated using filters and transformations written in XSLT. In the long run, this technique can be used to apply other extraction methods to the (sufficiently annotated) XML Master.

Algorithmic Details: Design of Software Modules. The syntax of nested paragraphs can be expressed as extended regular expressions, and recognized using a lexical analyzer generated by FLEX. The matching algorithm for phrases has been coded in C++. The generated intermediate representations are in XML, with multiple "views" obtained using XSLT.

[4] According to Gruber, an ontology is an explicit specification of a conceptualization, which is an abstract, simplified view of the world that we wish to represent for some purpose.

The XML markup for Method C Approach can be carried out in four steps: (1) Automatically recognize nested paragraph structure of the spec and generate suitable tags. (2) Automatically recognize and tag qualifiers (and procedures) present in the spec. (3) Automatically generate various views of the XML source using transformations described in XSLT. (4) Finally, verify, and possibly amend, the resulting translations manually.

Preprocessing. Save the spec available in MS-WORD format as a plain text with line breaks. Alternatively, scan and OCR paper-based spec to obtain the text in electronic form.

Structure markup of spec. Paragraphs (numbered sections) are enclosed within tags `<section level=‘‘...’’ id=‘‘...’’>` and `</section>` to mark the beginning and the end of a (possibly nested) paragraph respectively, where level indicates the nesting level starting at 1, and id is the paragraph number string given as a period-separated sequence of numbers. The paragraph heading is enclosed within `<sectionHeading heading=‘‘...’’>` and `</sectionHeading>`. In addition to preserving the logical organization of the spec, if the original layout information is also desired, then the XML master must also retain indentation information. In practice, the FLEX code has to deal with subtle problems and idiosyncrasies of the saved spec.

Tagging phrases corresponding to DLTs for Method C extraction. Document heading contains information about specification name, revision, products, spec class, and alloy name. The paragraph text can contain references to products, spec class, and alloy name. These are recognized using the appropriate fragment of domain library and tagged as follows:

Products: `<product DLT=‘‘...’’> ... </product>`
Spec class: `<specClass DLT=‘‘...’’> ... </specClass>`
Alloy: `<alloy DLT=‘‘...’’> ... </alloy>`

Method B extractions will need identification of characteristic-value pairs and grouping them under relevant procedures.

Generating various views from the XML Master. It is possible to generate various views of the spec from the XML master by applying suitable XSLT stylesheets using Apache's Xalan processor. The views of interest are:

1. The original text of the spec, with the identation preserved as much as possible, for verification.
2. The HTML version of the same for display in a Web-browser.
3. The Method C Extraction that generates syntactically correct SDL with the paragraph text presented as qualified notes (details of which are skipped for brevity).
4. The Method C Extraction that generates syntactically correct SDL with the paragraph presented as qualified notes using paragraph numbers in place of paragraph bodies, to deal with copyright restrictions that prohibit duplication of the spec text verbatim in another document.

5.5 An Example of Method C Extraction

Due to space limitation, we have condensed the example drastically. A fragment of the original specification, GE.txt, is shown below.

```
            SPECIFICATION NO.  B50TF104
                   ISSUE NO.  S4+AM1
                        DATE  January 9, 1996
                   PAGE 1 OF  10
                   CAGE CODE  07482
                  SUPERSEDES  B50TF104-S3
...
ALLOY BAR, FORGINGS, AND RINGS
(INCONEL ALLOY 706)
...
3.2.1  Material Condition.  Material shall be delivered in the following
condition specified on the purchaser order:

(a) Bar and rod shall be hot finished, heat treated per CLASS A, and
descaled; round bars shall be ground or turned.  Forgings shall be
as ordered.

(b) Flash welded rings shall not be supplied unless specified or
permitted on Purchaser's part drawing.
...
```

GE.txt is tagged to obtain GE.xml which is then transformed into the following Method C extracted GE.sdl with text suppressed using XSLT stylesheet.

```
document [B50TF104]
{
  title = "ALLOY BAR, FORGINGS, AND RINGS(INCONEL ALLOY 706)";
   org  = "GE Aircraft Engines";
   type = "specification";
  define APM
  {
       [Products] is "Bar";
       [Products] is "Forgings";
       [Products] is "Rings";
  }
  using APM;

  revision [S4+AM1]
  {
...
  if
       ( [Product Type] is "Bar"  or [Product Type] is "Rod"  or [Product Type] is "Bar"  or
          [Product Type] is "Forgings"  or [Product Type] is "Rings" )  and  ( [Spec Class] is "A" )
    then
    {
         note " = Material Condition."
         "Shall be in accordance with paragraph 3.2.1  "
    }
...
  }
}
```

The following script summarizes the actual processing that a WORD document GE.doc, saved as GE.txt, goes through. Specifically, the tagged file GE.xml is processed using the various stylesheets to generate the SDL forms and recover the original text. (Even in this simplified setting, "context-free" analysis causes tagging errors such as misidentifying "section" as a product type.)

```
flex structTagger.flex
gcc lex.yy.c -lfl
a < GE.txt > GE.xml
java org.apache.xalan.xslt.Process -in GE.xml -xsl CSDLStylesheet.xsl -out GE.sdl
java org.apache.xalan.xslt.Process -in GE.xml -xsl CExpSDLStylesheet.xsl -out GE.exp.sdl
java org.apache.xalan.xslt.Process -in GE.xml -xsl OriginalStylesheet.xsl -out GE.org.txt
```

5.6 Practical Problems for Fine Grain Extraction

Here is a list of hard problems to be overcome for full-scale commercialization of this technology:

Expression Matching: Recognition of and mapping to a DLT. One of the fundamental steps in content extraction is the recognition and mapping of spec phrases that convey requirements, to the corresponding DLTs. A core subproblem is the formalization of DLTs. To appreciate the practical difficulties in recognizing a DLT, consider the following string that illustrates the variety of entities that can be present in a DLT:

`(Cu + Ag) - 0.2% Al Chemical Composition (Final) (GEAE 42-Delta Phase)`

Specifically, a DLT can contain a formula, a sequence of words, a parenthesized sequence of words, or a reference to another spec. The formula itself can contain chemical element names, arithmetic operator symbols, other special symbols, etc. Even though the syntax of a formula resembles that of an arithmetic expression in isolation, there are many examples of formula that are difficult to discriminate from other fragments because of the ambiguous use of symbols such as "–" for "minus" and "dash", "/" for "division", "ratio" and "or", and the form of references. Once the syntax of DLTs has been formalized, their concrete manifestation in spec needs to be addressed. All this is further complicated if the phrase that maps to a DLT is not contiguous or several DLTs share words. Subsequent to recognizing DLTs, it is also necessary to recognize their inter-relationships.

Semantic Markup: Procedure descriptions. Specialized recognizers need to be built for each procedure that can select coherent pieces of text, and organize the information contained in it. Specifically, a list of characteristics and their values relevant to a procedure can be specified and identified using strategy similar to the one employed for recognizing DLTs. However, procedure descriptions also require guessing and inferring information not explicitly provided in spec but needed to express the extraction unambiguously. What is unclear here is the effect of these customized recognizers on each spec, and if their use will decidedly outweigh the tedium of manual extraction for complex specs. A reasonable approach is to build such recognizers for a small set of well-understood procedures

Semantic Markup: Representing and Processing Tables. Tables in spec represent collections of constraints on characteristics and encapsulate procedure descriptions very efficiently. Footnotes modify these tables in interesting ways, often

incrementally and sometimes substantially. The complexity of the table stems from their irregularity. In order to formalize tables, it is important to understand techniques employed for sharing pieces of information in a spec, and propose regular data structures that can hold the same information compactly and be translated into SDL automatically.

Integration: XML Technology. XML Technology is beneficial if it is possible to develop a target language and an extraction discipline that is more or less literal. However, at this time, the semi-automatic approach relies on a human extractor to remedy any inadequacy in the automatic markup by modifying the resulting SDL output. Unless these updates can be reflected back into the XML Master sensibly, the XML intermediary will not be viable. Furthermore, literal translation also requires unconventional use of certain SDL constructs, and unless these nested constructs can be properly handled by the SDR compiler, these will create problems downstream.

6 Conclusions

This project attempted to systematically evolve the manual extraction tool into a computer-assisted extraction tool. To this end, we studied extractions from specs – of different complexities and origin – carried out by highly trained metallurgists, and appreciated the importance of making domain library searches flexible. The recognition of semantically equivalent phrases that map to the same Domain Library Term was attempted using a normalization procedure starting from verbatim searches, to using minor variations on words, to using aliases, to eventually using much richer "patterns of equivalences". This functionality can be incorporated as a separate module, spliced between the IDE and the Domain Library.

For a semi-automatic approach to succeed in practice, the partial results obtained through automation should be intelligible, in relation to the original spec. Otherwise, the extractors will turn-off automation, in the interest of verifiability. This motivates the need to maintain a one-to-one correspondence between the spec and its translation. The XML Technology seems appropriate to extraction to the extent that the XML Master provides a way of tying together the spec and the extracted information, and the use of XSLT stylesheets in transforming this information as desired by the context.

Based on our first experiences with the prototypes we have built, the generated Method C extractions seem reasonable, though not perfect. However, improvements in precision will be needed to scale it to Method B extractions, from human extractor's perspective.

Acknowledgements The authors would like to thank the reviewers for their feedback.

References

1. Sokol, D.Z.: Concurrent Engineering in the Materials Industry: Case Study in the Application of Information Technology, Fourth Annual Conference on Management of Technology, 1994.
2. Sokol, D.Z., Rowe, J.: Integrating STEP and SGML for Concurrent Engineering, CALS 95 International Expo.
3. Sokol, D.Z.: Concurrent Engineering Design System for High Technology Material Suppliers, NSF Phase II Final Report, 1997.
4. Soderland S. G.: Learning Information Extraction Rules for Semi-structured and Free Text, Machine Learning, Vol. **34**, No. 1-3 (1999) 233-272.
5. Proceedings of the Third to Seventh Message Understanding Conference, Morgan Kaufman (1991) (1992) (1993) (1996) (1997).
6. UNISYS: Description of the UNISYS System used for MUC-3, Procs. of MUC-3, 1991,212-222.
7. Hobbs J., Appelt D., Bear J., Israel D., Kameyama M., Stickel M., and Tyson M.: FASTUS: Extracting Information from Natural-Language Text, 1996. (http://www.ai.sri.com/natural-language/projects/fastus-schabes.html)
8. Grishman R.: The NYU System for MUC-6 or Where's the Syntax?, Procs. of MUC-6 (1995).
9. Lehnert W.G., Cardie C., Fisher D., McCarthy J., Riloff E., and Soderland S.: Evaluating an Information Extraction System, Journal of Integrated Computer-Aided Engineering, 1(6) (1994) 453-472.
10. Riloff, E.: Automatically Constructing a Dictionary for Information Extraction Tasks, Proceedings of the Eleventh Annual Conference on Artificial Intelligence (1994) 811-816.
11. Fujii, A., and Ishikawa, T.: Cross-Language Information Retrieval for Technical Documents (1996).
12. Grishman R.: Information Extraction: Techniques and Challenges, Information Extraction (International Summer School SCIE-97), ed. Maria Teresa Pazienza, Springer-Verlag, 1997.
13. Soderland S.G.: CRYSTAL: Learning Domain-specific Text Analysis Rules, CIIR Technical Report # 43, University of Massachusetts at Amherst.
14. Du Charme B.: XSLT Quickly, Manning Publications Co. (2001).
15. Tidwell D.: XSLT, O'Reilly (2001).
16. Harold E. R.: XML Bible, Hungry Minds Inc. (1999).
17. Dietel H. M.: et al, XML: How to Program, Prentice Hall Inc. (2000).
18. Porter M. F.: An Algorithm for Suffix Stripping, Program, Vol. **14**, No. 3, (1990), 130–137.
19. McCarthy J.: A Trainable Approach to Coreference Resolution for Information Extraction, PhD Thesis. Dept. of Computer Science Technical Report # 78, University of Massachusetts, Amherst.
20. van Harmelen F. and Fensel D.: Practical Knowledge Representation for the Web. Practical Knowledge Representation for the Web. In Proceedings of the Workshop on Intelligent Information Integration (III99), (1999) IJCAI-99.
21. Muslea, I.: Extraction Patterns for Information Extraction Tasks: A Survey, In Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction, (1999) AAAI-99.