Extracting structure from HTML documents for language visualization and analysis

Robert P. Futrelle, Andrea Elaina Grimes and Mingyan Shao
Biological Knowledge Laboratory
College of Computer and Information Science
Northeastern University
Boston, MA 02115
{futrelle,agrimes,myshao}@ccs.neu.edu

Abstract

Document analysis is shifting from document image analysis to the analysis of electronic documents, especially those available on the Web in HTML and PDF formats. We are analyzing a 250M word collection of HTML formatted papers from the American Society for Microbiology with the ultimate goal of doing query answering and information extraction. Each document is converted to a sequence of token-id items by an invertible process called Extreme Tokenization. A lexicon is constructed with attributes including: token string, tag, capitalized, etc. A number of structures are identified, including section titles, figure captions, document navigation tables and most importantly, running text blocks. An XML descriptive structure is built using JAXB 1.0. Sentence boundaries are discovered. Language framework patterns are visualized in a custom Framework Viewer to identify important patterns of expression for further analysis. This work complements our diagram analysis research (ICDAR03).

1. Introduction

Our group focuses on the biology literature, which is a large and rich source of information for biology research and medicine [1]. The goal is to discover the content of text and diagrams in research papers to support information extraction and concept-based queries. This paper focuses on discovering and extracting running text segments in HTML documents, followed by language pattern visualization as a step towards computational linguistic analysis. The methods use highly efficient integer-based (UID) representations for lexical items and text streams which support the use of relational databases for scalability to multi-billion word textbanks. We are developing a 250M word textbank drawn from the web-based Highwire Press collection. The HTML structures in the collection are consistent and not overly complex.

There have been decades of important work in the analysis of document images [2]. Current document image systems can discover and label components and extract the reading order [3, 4]. Large numbers of documents are now available in electronic format, so that image analysis is bypassed. These electronic documents, and HTML in particular, still present challenges for structural analysis and content extraction. For information extraction from HTML, *wrappers* are developed that describe the structures containing information, e.g., headings or specific table elements. Manually designing wrappers is infeasible for large heterogeneous collections, so wrapper induction procedures have been developed [5]. Entropy measures [6] and "visually" based methods [7] have been devised for identifying content blocks.

XML is a far more expressive means of representing document content, so systems have been developed to convert HTML documents to XML, adding semantics at the same time [8]. The XDOC workbench has been developed for such XML manipulations.

Lessons learned from a biology text mining competition, the KDD Challenge Cup 2002, are reviewed in [9]. An example of a specific task is extracting synonymous gene and protein terms [10]. Still another task is understanding captions in biomedical publications [11], closely related to our own work on diagrams [12].

In analyzing any HTML document, one of the first issues that must be faced is tokenization of the text stream. Some of the standard approaches to the tokenization for natural language, e.g., dealing with hyphenation and other punctuation, are discussed in [13]. One of the first papers that dealt with tokenization and parsing of marked up text was produced by our group [14]; our current approach to these problems uses a method we call *Extreme Tokenization*, discussed below.

The 250M word textbank we use is licensed from the American Society for Microbiology (ASM) and is part of the Highwire Press collection. There are about 50G words in the 12M papers in the Highwire collection, http://highwire.stanford.edu/. They are in essentially the same format as our ASM papers, so our approach should apply to all of them with little modification. These textbanks are static, read-only collections. Once published, papers are never altered. This allows us to build large and efficient data structures to represent any

subset of them, in the form of *Archival Token Sequences* (ATS). All other structures we build, whether permanent or transient, reference the unchanging ATS. A technique we call *Extreme Tokenization* (ET) is used to create the ATS. ET uses a simple and universal strategy that should not require alteration for the foreseeable future.

In the prototype studies reported here, we have analyzed 5M words from the ASM textbank to build XML document instances describing text and caption text sections. Then sentences are identified, analyzed and displayed in a custom browser, the *Framework Viewer*. The objective in building and using the Viewer is to discover the nature of the standard text frameworks that biologists use in their papers to "package" important data and information. Once discovered, such frameworks can be exploited for information extraction and for answering content-based queries.

2. Document layout. ASM HTML format

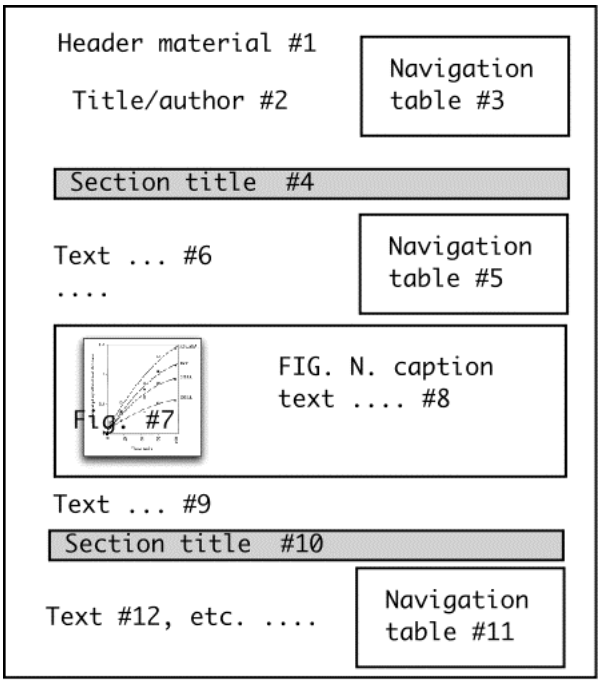

Fig. 1. Schematic screen layout of the HTML versions of papers in the Highwire Press collection, including the ASM textbank. The numbers labeling the components indicate their sequential order in the HTML source. HTML tables are important organizing elements that we use to discover the text and caption text sections.

The major components of a Highwire Press HTML paper include: the title section, including author affiliations and links to journal resources; abstract; section titles; running text blocks; tables appearing at regular intervals for document navigation; figures and tables and their captions; etc. Portions of each paper are contained in HTML tables, e.g., section titles and figures, including their captions, and the navigation blocks (Figure 1).

3. Extreme Tokenization. TIDs and the ATS

Each paper is converted to a sequence of tokens using a process we call *Extreme Tokenization* (ET). The process converts the HTML source text to a sequence of integer Token IDs (TIDs) including tokens for all HTML markup. The result is that the documents becomes a sequence of integers, the *Archival Token Sequence* (ATS), with the sequence indexed by a Sequence ID, (SID). The rationale behind ET is that it does not make complex decisions which might need to be altered later. An example of ET would be the handling of the symbol for the radionuclide, ^{32}P, an isotope of phosphorus. ^{32}P is so common that it might be treated as a single token. ET is conservative and represents it as a sequence of four tokens: ^{, 32,}, P. Note that there is no whitespace in ^{32}P. White space delimiters in the HTML source are reduced to single blanks, which are explicitly represented as tokens in the ATS. The *only* character sequences that are not split by ET are HTML tags and entities and contiguous sequences of the alphanumeric characters, a-zA-Z0-9. Hence the moniker, "Extreme".

One goal of the tokenization process is to allow documents to be stored as a sequence of integer records in relational databases, systems that have no difficulties in scaling to billions of records.

A basic lexicon is constructed which is a pair of maps between TIDs and strings. We also generate more specialized lexicons that contain information such as whether the token is an HTML tag or entity, whether a string is capitalized, is a single capital letter, is a full stop (".", "?"), etc. The design of these lexicons allows us to avoid expensive character-by-character string comparisons in sentence boundary detection, search or display.

4. XML schema for document structure

The particular goal of the work described here is to find the text blocks for linguistic analysis, omitting material such as title and author information, section titles, references, etc. To do this, we build a logical description of the major portions of the document that we want to include or ignore. This is constructed as an XML document instance whose identifiers label the sections of interest and whose leaf elements are the integer SID values that start and end each component. This logical description omits many structural details that are not needed for the sentence boundary detection task. For example, the internal structures in the Reference list at the end of the paper are not described, but simply contained

in a single span which is subsequently ignored in our language analysis.

XML Schema are used because they are more concise and expressive than DTDs. Schema are themselves specified in XML rather than using an independent notation [15]. The algorithms for identifying the schema components are manually designed, rather than attempting to automatically induce a set of rules, a process that could be error-prone. The ATS is then analyzed to discover the schema components for each paper and the corresponding in-memory Java instance of the schema is built and then marshaled by JAXB to a file. The in-memory schema is used directly or regenerated later by unmarshaling the file. The Schema used has the following components, where "+" indicates one or more repetitions of a component:

ASMDoc: Section+

Section: Name, Start SID, (Table or Text)+, End SID

Table: Start SID, End SID

Text: Start SID, End SID, IsCaption

5. Sentence boundary detection

For further linguistic analysis, it is necessary to discover sentence boundaries. Algorithms for doing this are ubiquitous and generally quite successful [16]. As with all such methods, the algorithm reduces to case analysis through a series of rules, e.g., is a "." or "?" followed by a blank and then a capitalized word? A period following a single capital letter is assumed not to be a sentence delimiter. All intervening HTML tags and entities are ignored in the sentence analysis. Using a small set of heuristics, sentence boundary detection accuracy of well over 99% was obtained.

6. Visualizing word-pair Frameworks

Based on various preliminary studies, we have determined that biology research papers contain many language structures which are repeated often in a single paper, in multiple papers and by various authors. Structures that contain important information appear to be made up of high-frequency components which we call *Frameworks* with embedded low-frequency informational elements. An example of this is given by the following with the Framework elements underlined and the informational elements in italics:

"The level of *GTP-bound Rap-1* was measured..."

In this example sentence the framework is the split bigram, "The level of" and "was measured", which brackets the informational item "GTP-bound Rap-1". The Framework Viewer we have built [17] allows us to study the occurrence of such patterns with the Framework elements, vertically aligned from sentence to sentence, much as in KWIC displays (Keywords In Context). Figure 2 shows the Framework Viewer.

7. Software implementation

All software is developed in Java 2 (1.4.1). In the prototype, sequences and the lexicon are stored in ArrayLists and HashMaps, respectively, and saved as Java serialized files. For larger datasets we use Oracle 8i. The prototype studies use a textbank of about 5M words. All analyses use the integer SIDs and TIDs.

.

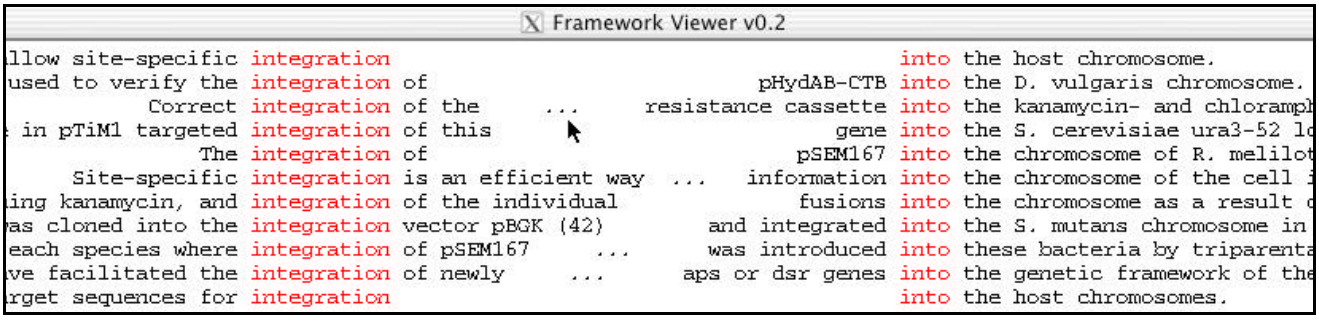

Fig. 2. A portion of a Framework Viewer window, showing the 11 instances of the Framework, "integration ... into", the total found in the full text of 100 ASM papers (500K words) in HTML format. Note the inclusion of low-frequency (low occurrence), highly informational elements in the framework, "PhydAB-CTB", pSEM167" and "pBGK". Versions of the Viewer currently under development will allow more than such literal matching, e.g., finding Frameworks based on word frequencies alone, by the category of the components (such as gene or protein names) and part of speech. The Viewer will also allow user interaction to classify instances for further analysis or for training for machine learning or to reveal further details about the entities or their context.

The XML schema are supported by JAXB [18]. JAXB processes the schema to build the Java classes which are in one-to-one correspondence with the schema as well as creating an object factory to generate the appropriate Java objects. JAXB is designed exclusively to deal with XML Schema rather than DTDs

The Viewer is written in Java Swing with the text display in Java 2D to allow the greatest flexibility, interaction, etc. The scrolling text is created on demand as scrolling proceeds, a "lazy" approach that enhances responsiveness and space and time efficiency and allows scaling to arbitrarily large collections.

8. Results and Discussion

The first result of our current project reported here is the development of the *Extreme Tokenization* process that leads to an archival integer-based format for the text and lexicon. Next we developed a system to identify the important text components of our ASM textbank and record it in XML documents. We then were able to delimit sentences in the body and caption text. Finally, we developed an efficient *Framework Viewer* for visualizing text patterns in the biology papers.

The systems described here are being developed further and scaled up. They form a firm foundation for the next steps in our major program for information retrieval and text mining from the large collections of biology research papers that are increasingly available in electronic form.

Acknowledgements

Emine Yilmaz helped with this paper. Supported in part by NSF DBI-0211047, IIS-9978004 and the Northeastern University *Institute for Complex Scientific Software*, http://www.icss.neu.edu/

References

[1] R. P. Futrelle, "BioNLP.org Natural language processing of biology text (http://bionlp.org)," 2001.

[2] G. Nagy, "Twenty Years of Document Image Analysis in PAMI," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 38-62, 2000.

[3] S. Klink, A. Dengel, and T. Kieninger, "Document Structure Analysis Based on Layout and Textual Features," in *DAS - International Conference of Document Analysis Systems*. Rio de Janeiro, Brazil, 2000, pp. 99-111.

[4] M. Aiello, C. Monz, L. Todoran, and M. Worring, "Dcoument Understanding for a Broad Class of Documents," *Intl J. Document Analysis and Recognition*, vol. 5, pp. 1-16, 2002.

[5] W. W. Cohen and L. S. Jensen, "A Structured Wrapper Induction System for Extracting Information from Semi-Structured Documents," in *IJCAI 2001*, 2001.

[6] S.-H. Lin and J.-M. Ho, "Discovering Informative Content Blocks from Web Documents," in *SIGKDD '02*. Edmonton, Alberta, Canada, 2002.

[7] Y. Yang and H. J. Zhang, "HTML Page Analysis Based on Visual Cues," in *ICDAR '01*, 2001.

[8] S.-J. Lim and Y.-K. Ng, "A Heuristic Approach for Converting HTML Documents to XML Documents," in *Computational Logic - CL 2000*, vol. 1861, J. Lloyd, Ed. Berlin: Springer-Verlag, 2000, pp. 1182-1196.

[9] A. S. Yeh, L. Hirschman, and A. A. Morgan, "Evaluation of text data mining for database curation: Lessons learned from the KDD Challenge Cup," *Bioinformatics*, vol. 19, pp. i331-i339, 2003.

[10] H. Yu and E. Agichtein, "Extracting synonymous gene and protein terms from biological literature," *Bioinformatics*, vol. 19, pp. i340-i349, 2003.

[11] W. W. Cohen, R. Wang, and R. F. Murpy, "Understanding Captions in Biomedical Publications," in *KDD 2003*, 2003.

[12] R. P. Futrelle, M. Shao, C. Cieslik, and A. E. Grimes, "Extraction layout analysis and classification of diagrams in PDF documents," in *ICDAR (Intl. Conf. Document Analysis and Recognition)*. Edinburgh, 2003.

[13] David D. Palmer, "Tokenization and Sentence Segmentation," in *Handbook of Natural Language Processing*, R. Dale, H. Moisl, and H. Somers, Eds. New York: Marcel Dekker, Inc., 2000, pp. 11-35.

[14] R. P. Futrelle, C. C. Dunn, D. S. Ellis, and M. J. Pescitelli, Jr., "Preprocessing and lexicon design for parsing technical text," in *Proc. 2nd Intnl Workshop on Parsing Technologies*. Morristown, New Jersey: Association for Computational linguistics, 1991, pp. 31-40.

[15] E. van der Vlist, *XML Schema. The W3C's Object-Oriented Descriptions for XML*. Sebastopol, CA: O'Reilly, 2002

[16] A. Mikheev, "Periods, Capitalized Words, etc.," *Computational Linguistics*, vol. 28, pp. 289-318, 2002.

[17] A. E. Grimes and R. P. Futrelle, "Text Pattern Visualization for analysis of biology full text and captions," in *IEEE Computer Society Bioinformatics Conf.* Palo Alto, CA: IEEE-CS Press, 2003.

[18] Sun_Microsystems, "Java TM Architecture for XML Binding (JAXB) http://java.sun.com/xml/jaxb/," 2003.