Research Paper ■

Representing Information in Patient Reports Using Natural Language Processing and the Extensible Markup Language

CAROL FRIEDMAN, PHD, GEORGE HRIPCSAK, MD, LYUDA SHAGINA, HONGFANG LIU

Abstract **Objective:** To design a document model that provides reliable and efficient access to clinical information in patient reports for a broad range of clinical applications, and to implement an automated method using natural language processing that maps textual reports to a form consistent with the model.

Methods: A document model that encodes structured clinical information in patient reports while retaining the original contents was designed using the extensible markup language (XML), and a document type definition (DTD) was created. An existing natural language processor (NLP) was modified to generate output consistent with the model. Two hundred reports were processed using the modified NLP system, and the XML output that was generated was validated using an XML validating parser.

Results: The modified NLP system successfully processed all 200 reports. The output of one report was invalid, and 199 reports were valid XML forms consistent with the DTD.

Conclusions: Natural language processing can be used to automatically create an enriched document that contains a structured component whose elements are linked to portions of the original textual report. This integrated document model provides a representation where documents containing specific information can be accurately and efficiently retrieved by querying the structured components. If manual review of the documents is desired, the salient information in the original reports can also be identified and highlighted. Using an XML model of tagging provides an additional benefit in that software tools that manipulate XML documents are readily available.

■ **JAMIA.** 1999;6:76–87.

Information in textual patient documents is a valuable source of clinical data, yet information in textual form cannot be reliably accessed for automated applications. To provide access to the information, medical language processing (MLP) systems have been developed,[1–8] which extract and structure information in patient reports in order to organize and encode the pertinent information appropriately for subsequent clinical applications. The organization of the structured output is critical if the output is to be effectively used for other applications.

Some systems[1,3,4,8] produce output that is used for subsequent automated applications associated with

Affiliations of the authors: Columbia University, New York (CF, GH, LS); Queens College City University New York, New York (CF, HL).

This work was supported in part by grants LM06274 and LM05627 from the National Library of Medicine and by the Columbia Center for Advanced Technology (CAT) in High-performance Computing and Communications in Healthcare with funding from the New York State Science and Technology Foundation.

Correspondence and reprints: Carol Friedman, PhD, Department of Medical Informatics, Columbia University, 161 Fort Washington Avenue, AP-1310, New York, NY 10032. e-mail: ⟨friedma@flux.cpmc.columbia.edu⟩.

Received for publication: 5/18/98; accepted for publication: 9/16/98.

decision support or quality assurance. Other systems[9–11] automatically generate ICD (International Classification of Diseases) codes[12] from text to assist in generating billing codes. The output generated by these types of systems is structured so that it can be used for automated applications, but the output does not directly correspond to the original report.

Some systems enrich the text in patient reports with predefined tags[13,14] so that the reports can be rendered in ways that are useful for highlighting, manual review, and limited automated retrieval (e.g., the term "cardiomegaly" may be tagged in an Impression section). The problem with this type of document model is that documents with specified information cannot be automatically retrieved with sufficient accuracy. For example, if all documents containing a tag for "cardiomegaly" were retrieved, many of the retrieved reports would correspond to the negation of cardiomegaly, as in "cardiomegaly was not observed." A schema that integrates the two output models is highly desirable because it would provide reliable automated access and also be convenient for applications supporting manual review.

This paper describes a document structure, based on XML (extensible markup language,[15] a subset of standard generalized markup language [SGML])[16,17] that was designed for ease of implementation and interoperability with SGML and with HTML (hypertext markup language, which is used by WEB browsers). The model we designed embeds a tagged, structured, and encoded representation of the informational contents in an enriched version of the original report.

With this schema, numerous applications become possible. For example, using the Digital Imaging and Communications in Medicine (DICOM) standard[18] along with the model and method described in this paper, an application using an appropriate user interface could allow radiologists to coordinate textual findings with regions of a digitized image of a radiology examination. The textual findings could then be processed to produce enriched reports. Subsequently, users could retrieve reports with a specified positive finding and have the location of the finding in the corresponding images highlighted. This type of capability could lead to new types of applications for MLP in training and telecommunications, as well as to the more typical uses of MLP in research, decision support, and quality assurance.

For applications not involving digitized images, highly selective reports could also be retrieved and pertinent information in the reports highlighted. In addition, since the processed reports are in XML, a standard format for textual documents, commercial off-the-shelf and publicly available software can be obtained to manipulate the XML output.

Background

A number of MLP systems have been developed that structure and encode clinical information occurring in textual clinical reports so that the information can be used for automated decision support[3,4,8,19–21] and so that document manipulation and viewing by health care workers[9–11,14] for financial and clinical applications can be facilitated. A detailed overview of a broad range of MLP systems can be found in an overview by Spyns.[22]

We have developed an MLP system called MedLEE,[2,23] which has been used at Columbia–Presbyterian Medical Center (CPMC) since February 1995. MedLEE was designed as a general processor in the medical domain. It was initially developed for chest radiography and since has been expanded to mammography, neuroradiology, pathology, and electrocardiography reports. Evaluations of the system as used with chest radiographs and mammography reports[3,24–27] showed that it was effective in identifying specific clinical conditions and was effectively used[19] for improving the quality of patient care. In these studies, MedLEE was used for decision support where ease of access to the structured data was critical for the application to be effective. The output form was designed to achieve a balance between completeness and ease of access to the data. The next section provides background information describing the version of the representational model that was previously generated.

Representation of Structured Information

The formal representational model for the structured output is described in more detail in Friedman et al.[28] We use a frame-based representation that is consistent with the conceptual graph model.[29] Each frame specifies the informational type, the value, and modifier slots which are also frames. Thus, the initial output form for *moderately enlarged spleen*, as shown below, is a frame denoting a **problem**, which has the value **enlarged**; in addition, there are **degree** and **body location** modifiers with the values **moderate** and **spleen** respectively:

[problem,enlarged,[degree,severe],[bodyloc,spleen]]

The output form undergoes several mappings before the final form is created. One type of mapping is re-

```
<address> <street> <number> 124 </number> <street_name> Barrows Lane
</street_name> </street> <city> New York </city> <state> NY </state> <zipcode>
10001 </zipcode> </address>
```

Figure 1 An extended markup language (XML) tagging scheme for address. The address tag contains embedded tags for street, city, state, and zip code. Similarly, the street tag contains embedded tags for number and street name. This scheme facilitates searching for specific information in certain parts of the address, such as a particular zip code.

quired to compose components of multiword phrases that are separated in the original text; another type of mapping is necessary to translate target terms into controlled vocabulary concepts. The components responsible for these mappings are discussed later, under Methods. A final mapping is generally performed in order to translate the frame format to the final format.

The structured output form, as described earlier, is not linked to the words in the original sentences of the report. Modifications to MedLee and to the underlying representational model were made to link the structured form to the text. The new document model is described under The Document Model, below, and the MLP method used to automatically generate enriched documents by processing the clinical reports is described under Methods.

Related Work

The concept of using a standard generalized markup language (SGML)[15–17] to maximize the utility of electronic documents is well established. Hypertext markup language (HTML)[30] and extensible markup language (XML)[15,31] are based on SGML and are used for rendering documents for the World Wide Web. Widespread adoption of markup languages is evidenced by the text-encoding initiative (TEI),[32] which uses SGML to encode literature; the chemical markup language (CML),[33] which involves documentation of chemical compounds using SGML; and open financial exchange (OFE),[34] which is an SGML standard format for interchange of financial transactions. Also, a large collection of software tools,[35–38] many of which are publicly available, have been developed to process SGML and XML documents.

In the health care field a number of articles report the use of SGML and XML to tag medical documents.[39–42] A special interest group, Health Level Seven (HL7) SGML/XML,[43] has been formed to further the use of SGML and XML in the electronic patient record. In particular, the group's effort involves specifications for embedding XML in the HL7 structure and for developing a model of medical documents[44] to facilitate exchange of documents between users. Another goal is to provide automated applications with the capability of processing the documents after the document exchange has been made.

In the MLP community, Sager et al.[13] first described the utility of augmenting textual patient reports using SGML by means of MLP to provide tools to facilitate document handling for health care workers. One relatively simple task would be to tag the words and phrases of the text according to semantic and syntactic categories, and another would be to tag the report according to sections and paragraphs. Sager et al. also proposed the use of a more complex tagging schema, such as tagging occurrences of diagnoses, that could be used later to retrieve pertinent clinical information.

Zweigenbaum et al.[14] also proposed the adoption of an "enriched-document" paradigm based on SGML and natural language processing to further disseminate applications that utilize natural language processing methodology. A number of benefits of using a document-oriented model were discussed. A noteworthy gain would be the ability to use the annotated text as a valuable resource to further the development of language processing systems. Zweigenbaum et al. also proposed embedding a conceptual graph into each sentence of a document.

A major disadvantage of the document-centered approaches is that accurate access of the information is elusive because the tagging schemas are extensively intertwined with the text and are dependent on word order, which is extremely varied and unpredictable. The schema we present in this paper integrates the content-centric and document-centric approaches, because the salient clinical information is represented in a structured XML form that contains references to identifiers in the unstructured report, where the original words and phrases are assigned unique identifiers. This design is optimal both for searching, because it is not dependent on the ordering of the phrases in the text, and for rendering text to users, because the

```
<?xml version ="1.0"?>
<!-- DTD for report style output generated by MedLEE   -->
<!ELEMENT medleeOut  (section)+>
<!ELEMENT section  (structured?,tt)>
<!ELEMENT structured (problem|procedure)*>
<!ELEMENT problem
      (certainty|degree|status|change|bodyloc|region|sid|idref)*>
 <!ELEMENT procedure
      (certainty|degree|status|change|bodyloc|region|sid|idref)*>
<!ELEMENT degree  (degree)*>
<!ELEMENT certainty  EMPTY>
<!ELEMENT change  (certainty|degree|change)*>
<!ELEMENT descriptor EMPTY>
<!ELEMENT status EMPTY>
<!ELEMENT sid EMPTY>
<!ELEMENT bodyloc  (region|bodyloc)*>
<!ELEMENT region (region)*>
<!ELEMENT tt  (#PCDATA | sent )*>
<!ELEMENT sent  (#PCDATA|phr|undef)*>
<!ELEMENT phr  (#PCDATA)>
<!ELEMENT undef  (#PCDATA)>
<!ATTLIST section c CDATA  #REQUIRED>
<!ATTLIST sent id ID    #REQUIRED>
<!ATTLIST phr id ID    #REQUIRED>
<!ATTLIST problem v CDATA  #REQUIRED
                 idref IDREFS #IMPLIED>
<!ATTLIST procedure v CDATA  #REQUIRED
                 idref IDREFS #IMPLIED>
<!ATTLIST region v CDATA  #REQUIRED
            idref IDREFS #IMPLIED>
<!ATTLIST bodyloc v CDATA  #REQUIRED
            idref IDREFS #IMPLIED>
<!ATTLIST degree v CDATA  #REQUIRED
                idref IDREFS #IMPLIED>
<!ATTLIST sid idref IDREFS #REQUIRED>
<!ATTLIST certainty v CDATA   #REQUIRED
           idref IDREFS #IMPLIED>
```

Figure 2 The document type definition (DTD) of a clinical report (medleeOut) generated by MedLEE contains sections that consist of two components—a structured component **structured** containing structured data, and a tagged textual component **tt**.

structured XML form contains references to appropriate portions of the original text.

Extensible Markup Language

XML is a subset of SGML that is computationally less complex than SGML and therefore simpler and more efficient to process. XML makes it possible to add elements of information (i.e., tags) to textual documents so that the documents are machine independent and in a form that can be manipulated better than if the tags were not present. Documents can be structured using various levels of complexity, because tagging is a general mechanism that can be very simple (e.g., use of tags denoting new sections or new sentences) or complicated, particularly when there are many levels of nested embedded tags. Figure 1 represents an address using XML tags. The start of a tagged component T is denoted by ⟨T⟩ and the end is denoted by ⟨/T⟩. Tags may be nested in other tags. In Figure 1, street, city, state, and zipcode tags are nested in the address, tag, and number and street_name tags are nested in the street tag. Having an address in this form provides a way to manipulate documents with address tags in different ways. For example, documents with a specified zip code and street name can be retrieved easily by searching for the text enclosed by the zipcode and street_name tags.

The structure of XML documents is specified using a DTD, which is a set of blueprints related to information about the organization of the document type and consists of specifications concerning the structure of the document. The DTD is used by an XML parser to ensure that a document is valid according to the DTD. The specification involves the positions, attributes, cardinality, and values of the tags. Figure 2 shows an example of a DTD for the document type **medleeOut**. This model represents a simplified version of the model we have designed. The definitions of the elements (i.e., ⟨**!ELEMENT** . . .⟩) delineate the structure of the tags, which are the components of the document. Cardinality is represented using "?" (denoting zero elements or one element), "+" (one or more elements), or "*" (zero or more elements). The names and types of values of the attributes of an element are specified using the statement ⟨**!ATTLIST** . . .⟩. Figure 2 is described in further detail below.

The Document Model

Figures 2 to 4 are simplified illustrations of the document tagging schema we are proposing. Figure 2 is the DTD that defines a simplified version of the structure of the output (medleeOut) generated by MedLEE. The output consists of one or more sections; each sec-

```
<structured>
        <problem v = “pain” idref = “p2>
                <bodyloc v = “abdomen” idref = “p3”>
                        <region v = “lower” idref = “p4”> </region>
                </bodyloc>
                <onset v = “intermittent” idref = “p1”> </onset>
                <status v = “develop” idref = “p6”> </status>
                <date v = “19950304” idref = “p8”>
                <sid idref = “s1.1.1”>
        </problem>
        <problem v = “swelling” idref = “p13”>
                <certainty v = “no” idref = “p12”> </certainty>
                <bodyloc v = “extremity” idref = “p14”> </bodyloc>
                <sid idref = “s1.1.2”>
        </problem>
</structured>
```

Figure 3 An example of the structured component of the output form generated by MedLEE. Two tags correspond to the informational type **problem**. One has the value **pain** with a reference to identifier p2 along with other modifiers, which also have values and identifiers. The second has the value **swelling** and references identifier p13; it also has modifiers **certainty**, body location (**bodyloc**), and sentence identifier (**sid**), whose value is a triple identifying the section number, paragraph number in the section, and sentence number in the paragraph. The identifiers are shown in Figure 4, which illustrates the tagged text component, **tt**.

tion consists of an optional structured component **structured** that is followed by a required tagged textual component **tt**. The new lines and indentations shown in Figures 2 to 4 are not necessary for XML but are used in the diagrams to improve readability. The structured component provides a content-centric view of the report. It contains a structured representation of the contents of the report and is essential for reliable and efficient access to information in the document. The structured representation also contains information that references corresponding textual portions of the report.

As shown in Figure 2, the structured component consists of zero or more components that correspond to primary findings called **problem** or **procedure**. The component **problem** contains zero or more components that correspond to modifiers of the findings and are called **certainty**, **degree**, **status**, **change**, **bodyloc**, **region**, **sid**, and **idref**. The modifier components are also defined in the DTD: Those that have no nested structures are defined using the keyword **EMPTY** (e.g., the definition of **sid**, which specifies a sentence identifier).

The tags representing the primary findings and modifiers also have attributes. For example, **problem** has an attribute **v**, which must be present (**#REQUIRED**) and which consists of character data (**CDATA**), and an attribute **idref**, which is optional **#IMPLIED** and which consists of one or more references (**IDREFS**) to other attributes in the document that are unique identifiers.

The tagged textual element **tt** is also specified in Figure 2. It provides a document-centric view of the report because it consists of the original report enriched with tags that delineate and identify textual elements **sent** (marking sentences) and **#PCDATA**, which is the original textual data. The component **sent** consists of textual data, phrases **phr**, or undefined words **undef**. The component **phr** has an attribute **id** whose value is a unique identifier within the report. The **idref** attributes of the elements of the structured components correspond to the **id** attributes of the phrases. Similarly, the **idref** attributes of the **sid** elements of the structured components correspond to the **id** attributes of the sentences (**sent**).

Figure 3 shows the contents of the structured component that is generated after processing the sentences: *Intermittent pain in lower abdomen developed on 3/4/95. There was no swelling in extremities.* Notice that Figure 3 consists solely of tags that have attribute–value pairs. The name of the tag corresponds to the type of information being represented. For example,

problem is a type of information. It has an attribute **v** whose value is "pain," and an attribute **idref** whose value **p2** is an identifier that refers to a portion of the original text of the report, as shown in Figure 4.

The **problem** tag also has embedded tags that are modifiers. The **bodyloc** modifier has an attribute **v** whose value is "abdomen" and also an **idref** attribute. Tags that correspond to phrases in the original textual report have **idref** attributes. However, some tags do not have an **idref** attribute because they do not correspond to a phrase in the original report but to contextual information added during parsing. For example, **parsemode** specifies the method used to structure the information. The parse mode is a measure of accuracy of the output based on the mode used to interpret the sentence and obtain the structured form. **Mode1** is likely to be the most accurate interpretation, whereas **mode5** is likely to be the least accurate. A description of the parse modes is given in Friedman et al.[28]

In Figure 3, the values of the **v** attribute are frequently the same as the corresponding words and phrases in the report. However, as shown in Figure 5, the value of the **v** attribute can be different from the corresponding phrase in the actual report, because it corresponds to a controlled vocabulary term (e.g., **splenomegaly**), which is different from the canonic textual form (e.g., **enlarged spleen**).

The tagged text component **tt** is shown in Figure 4. It is the original report enriched with tags that uniquely identify sentences and phrases. A tag **sent** notes the beginning of a new sentence. It has an attribute **id** whose value is a triplet that identifies the section number, paragraph number, and sentence number of the sentence in the report. This information is useful for certain applications. For example, in discharge summaries at CPMC, the first and second sentences of the History of Present Illness section generally contain the chief complaint, and the last two sentences of the Hospital Course section contain the discharge plan. Furthermore, sentences that are adjacent and in the same paragraph generally refer to the same body locations and time period, unless another body location or time period is explicitly stated.

The tag **phr** denotes the beginning of a single word or multiword phrase of the report. It has an **id** attribute, whose value is a unique identifier of the phrase in the report. Phrases that are referenced in the structured component are shown in Figure 4, and those that are not referenced are omitted. For example, *no* in *no evidence of* is preceded by the begin **phr** tag identified by "pll," and *of* is followed by the end **phr** tag. **Sent** may also have an element that is an **undef** tag. This tag surrounds words that are not found in the lexicon. This may prove useful for other applications, such as further training in the NLP system or identification of proper names.

The structured representations of certain sentences are not always as straightforward as in the example discussed above, particularly if the text is complex or contains conjunctions. For example, the sentence *the spleen and liver are enlarged* contains the concepts **splenomegaly** and **hepatomegaly**, which are controlled vocabulary concepts associated with the phrases **enlarged spleen** and **enlarged liver**, respectively. In the text, the individual word components of **enlarged spleen** are *enlarged* and *spleen*, which are separated from each other, as are *enlarged* and *liver*. Moreover, *enlarged* pertains to both *spleen* and *liver*. The XML

```
<tt>
 <section c = "history of present illness" > History of present illness: <sent id = "s1.1.1">
<phr id = "p1"> Intermittent </phr> <phr id = "p2"> pain </phr> in <phr id = "p4">
right </phr> <phr id = "p5"> arm </phr> <phr id = "p6"> developed </phr> on <phr id =
"p8"> 3/4/95 </phr>. <sid id = "s1.1.2"> There was <phr id = "p11"> no evidence of
</phr> <phr id = "p12"> swelling </phr> in <phr id = "p14"> extremities </phr>.
</tt>
```

Figure 4 The tagged text component is embedded in the tag **tt**. It contains the original report augmented with tags that delineate sections, sentences, and phrases of a report. The tags have attributes **id**, whose values are unique identifiers for that component. For brevity, **phr** tags are not shown for phrases that are not referenced by the corresponding structured component.

```
<structured>
        <problem v = “splenomegaly idref = “p2 p7”>
                                <bodyloc v = “spleen” idref = “p2”> </bodyloc>
                                <certainty v = “high certainty” idref = p5”> </certainty>
        </problem>
        <problem v = “hepatomegaly” idref = “p4 p7”>
                                <bodyloc v = “liver” idref = “p4”> </bodyloc>
                                <certainty v = “high certainty” idref = “p5”> </certainty>
        </problem>
  </structured>
<tt>
The <phr id = “p2”> spleen </phr> and <phr id = “p4> liver </phr> <phr id = “p5”>
appear to be </phr> <phr id = “p6”> moderately </phr> <phr id = “p7”> enlarged </phr>.
</tt>
```

Figure 5 Tagged representation of the structured and tagged components for the sentence *the spleen and liver appear to be moderately enlarged*. The values of the **id** attributes of the tag **phr** are based on the assumption that the sentence appears at the beginning of the report, so that the first word of the sentence, *the*, is assigned a position 1. The attribute **idref** for **splenomegaly** has two values that reference the individual components **enlarged** and **spleen**, which constitute the concept **splenomegaly**.

structure for the above sentence is shown in Figure 5. In this example, two of the **idref** attributes are multivalued, because they are associated with the noncontiguous single word phrases identified by references to **p2** and **p7**, which correspond to **splenomegaly**, and to **p4** and **p7**, which correspond to **hepatomegaly**.

Methods

Overview of MedLEE

MedLEE is written in Quintus Prolog and can run on most Unix and Windows platforms. It takes an average of 3 sec to process a complete radiologic report using a SUN Ultra-1 Model 170 workstation with a clock speed of 167 MHz. MedLEE requires 32 MB of RAM and 4 MB of disk space.

MedLEE was modified to generate XML output consistent with the proposed model. MedLEE consists of several modular components divided according to functionality. Figure 6 shows the different components. A brief summary of each component in the modified version is given in the following paragraphs. A detailed description of the original version appears in Friedman et al.[2]

The first component of MedLEE is the preprocessor, which delineates the sentences of the report. Lexical lookup is performed to identify and categorize single words and multiword phrases in each sentence. The output of this component consists of a list of word positions, where each position is associated with a word or multiword phrase in the report. For example, if the sentence *spleen appears to be moderately enlarged* were at the beginning of the report, it would be represented as the list [1,2,5,6], where position 1 is associated with spleen, position 2 with the multiword phrase *appears to be*, position 5 with *moderately*, and position 6 with *enlarged*. The remainder of the list of word positions would be associated with the remaining words in the report.

The second component is the parser. It utilizes the grammar and categories assigned to the phrases of a sentence to recognize well-formed syntactic and semantic patterns in the sentence and to generate intermediate forms. The target form generated by the parser for the sample sentence *spleen is moderately enlarged* would be the following frame:

[problem,6,[bodyloc,1],[degree,5],[certainty,2]]

In this form the value of each frame is a number representing the position of the corresponding phrase in the report. In a subsequent stage of processing the number will be replaced by an output form that is the canonic output specified by the lexical entry of the word or phrase in that position. The parser proceeds by starting at the beginning of the sentence position list and following the grammar rules. When a semantic or syntactic category is reached in the grammar, the lexical item corresponding to the next available unmatched position is obtained and its corresponding lexical definition is checked to see whether it matches the grammar category. If it does match, the position is removed from the unmatched position list and the parsing proceeds. If it does not match, an alternative grammar rule is tried. If no analysis can be obtained, an error recovery procedure is followed so that a partial analysis is attempted.

The next component performs phrase regularization. It first replaces each position number with the canonic output form specified in the lexical definition of the phrase associated with its position in the report. It also adds a new modifier frame **idref** for each position number that is replaced. For example, the sample output form shown above would be changed to:

[problem,enlarged,[idref,6],[bodyloc,spleen, [idref,1]],[degree,severe,[idref,5]],[certainty, appear,[idref,2]]]

This stage also composes multiword phrases that are separated in the documents. For example, in the sample sentence the individual components of the multiword term **enlarged spleen** are separated. **Spleen** and **enlarged** are composed during phrase regularization and mapped into the target form **enlarged spleen** so that the ouput at this stage would be:

[problem,enlarged spleen,[idref,[6,1]], [bodyloc,spleen,[idref,1]],[degree,severe, [idref,3]],[certainty,appear,[idref,2]]]

Notice that the value of **idref** for the frame problem is a list **[6,1]** rather than a single value, because two words in the text report that are separated from each other constitute the components of the term **enlarged spleen**.

The next component performs the encoding. This consists of mapping the canonic forms into controlled vocabulary terms if applicable. In the example, we assume the controlled term for **enlarged spleen** is **splenomegaly**, the controlled term for **moderate** is **moderate degree** and the controlled term for **appears** is **moderate certainty**. The target form would be translated into:

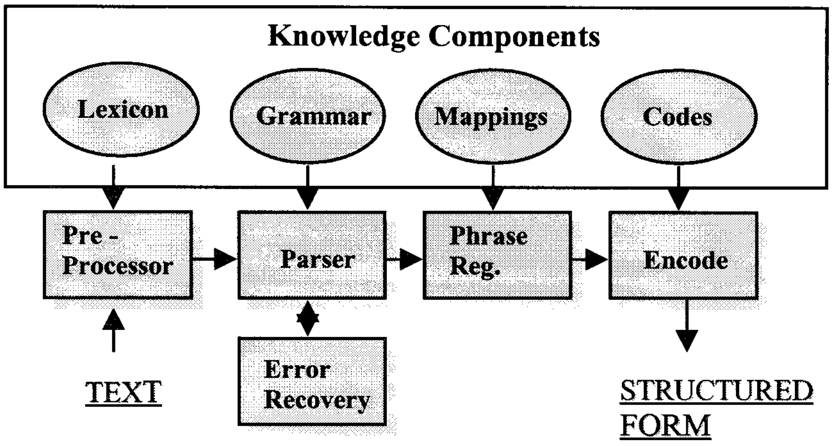


Figure 6 Overview of components of MedLEE. There are five processing phases and four knowledge base components. The first phase of processing is the preprocessor. It determines sentence boundaries and performs lexical lookup. The parsing uses the grammar to determine the structure of the sentence and to generate an intermediate form. The regularization phase composes multiword terms, and the encoding phase maps the output to controlled vocabulary terms.

[problem,splenomegaly,[idref,[6,1]],[bodyloc, spleen,[idref,]],[degree,'moderate degree', [idref,3]],[certainty,'moderate certainty',[idref,2]]]

The last component produces the final output structure. The XML structured part will be:

⟨problem v = "splenomegaly" idref = "p1 p6"⟩ ⟨bodyloc v = "spleen" idref = "p1"⟩ ⟨/bodyloc⟩ ⟨degree v = "moderate degree" idref = "p3"⟩ ⟨/degree⟩ ⟨certainty v ="moderate certainty" idref = "p2"⟩ ⟨/certainty⟩ ⟨/problem⟩

And similarly the tagged text part will be:

The ⟨phr id = "p1"⟩ spleen ⟨/phr⟩ ⟨phr id = "p2"⟩ appears to be ⟨/phr⟩ ⟨phr id = "p3⟩ moderately ⟨/phr⟩ ⟨phr id = "p5⟩ enlarged ⟨/phr⟩

The generation of the structured form is straightforward, as is the generation of the tagged text. A number of possible variations on this scheme are mentioned later, under Discussion.

Generating and Validating Output

Two hundred reports from a previous evaluation[3] were processed using the modified version of MedLEE, and XML output (output A) was generated. To verify that the output generated was valid XML according to the DTD specified for the model, we used an XML validating parser[37] to check the output for the 200 reports.

Another automated test was performed to determine whether the structured output generated by MedLEE contained all the information generated by the previous version of the system. We did not use the original output from the previous evaluation, because the grammar and lexicon had been significantly extended since then. Therefore, differences in the output would be attributable to grammar and lexical changes and not necessarily to the modifications of the MedLEE system. Therefore, the 200 reports were processed once again using the unmodified system along with the recent grammar and lexicon to generate output B. A program was written to automatically convert the XML structured form (output A) into the same form as the original version (output C). For each report, output B was compared with output C to determine whether the two forms were the same.

All 200 reports were processed successfully and converted into the enriched XML format described in this paper. The XML output documents generated for 199 of the reports were parsed successfully using the DTD and the validating parser. A tagging error was found in the tagged text component of one sentence of one document. This was due to an error in MedLEE. There was no significant difference in the time required to process a complete report using the modified system; each report took approximately 3 sec to process using either version of MedLEE.

All 200 reports in output B form were found to be identical to those in output C form.

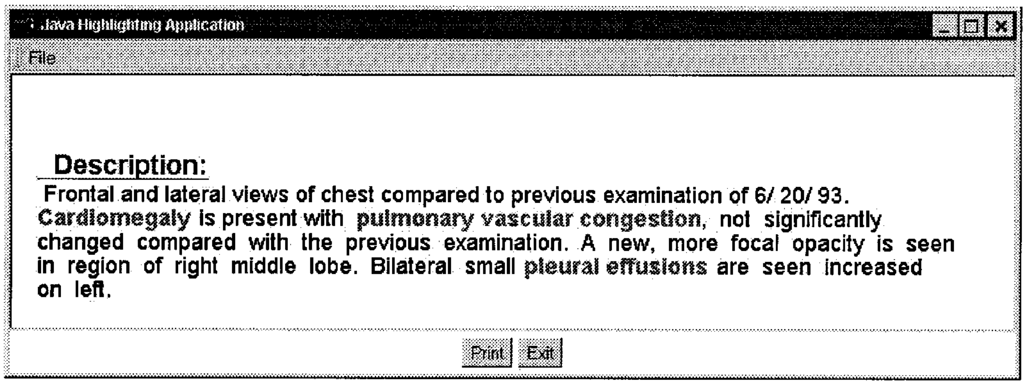
Discussion

The schema we have described extends the functionality of the output generated by MedLEE without changing the ease of access to the data. Retrieval of appropriate documents can be accomplished as before by querying the structured portion of the tagged documents. For example, we wrote a JAVA application to retrieve reports associated with **congestive heart failure** using the same medical logic that was previously written[3] to identify chest radiographic reports that are positive for that condition. The query looked for a positive assertion of a finding of congestive heart failure, pulmonary vascular congestion, or for both the findings cardiomegaly and bilateral pleural effusion. Findings associated with negation and some other modalities were ignored. For example, findings were ignored if they contained a certainty modifier with the value **no**, **rule out**, or **cannot assess**, or a temporal modifier with the value **resolved** or **previous**.

In addition to the retrieval capability, additional functionality was incorporated into the JAVA application in the form of highlighting. When a relevant finding associated with congestive heart failure was detected, the value of the **idref** attribute(s) was used to identify textual phrases to be highlighted.

Figure 7 shows the output that was generated for the description section of one of the reports retrieved as being positive for **congestive heart failure**. Notice that the term **congestive heart failure** is not in the report, but findings suggestive of congestive heart failure are (e.g., *cardiomegaly, pulmonary vascular congestion*, and *pleural effusions*). Also notice that those three phrases are highlighted in the report.

A number of variations of the tagging schema we have presented are possible. One type of variation concerns the placement of the structured output. In the version we have described the structured output was included at the beginning of each section of the report. However, it may be placed at the beginning of



Description:

Frontal and lateral views of chest compared to previous examination of 6/ 20/ 93. Cardiomegaly is present with pulmonary vascular congestion, not significantly changed compared with the previous examination. A new, more focal opacity is seen in region of right middle lobe. Bilateral small pleural effusions are seen increased on left.

Figure 7 Output showing the description section of a radiologic report associated with the clinical condition congestive heart failure, where terms associated with congestive heart failure are highlighted. The report was retrieved and highlighted using a JAVA program and structured output generated by MedLEE. The identifiers corresponding to the structured findings associated with the condition were used to highlight the appropriate phrases in the textual report.

the report so that conceptually it is thought of as an index or codification of the contents of the report. It could also be made to precede each sentence.

Another more substantial variation of the schema involves including more information in the **phr** tags by adding additional attribute–value pairs other than the **id** attribute. For example, the semantic and syntactic categories of the phrases could also be supplied by adding the appropriate attributes **sem** and **syn** to the **phr**ase tag.

This model also raises a nontrivial user interface issue. The ability to retrieve highly specific information means that different types of information (primary findings and also modifiers) within one report may be relevant to a query. That means that different types of information will have to be highlighted, but it is not obvious how to effectively highlight different types of information in a report. For example, when a query retrieves a report, only the primary terms associated with the condition may be highlighted, or some of the modifiers used in the query may be highlighted in different colors, or possibly the whole sentence may be highlighted.

In Figure 7, all the primary terms that were associated with the medical logic query for **congestive heart failure** and were found in the report were highlighted, but body location, degree, and certainty modifiers were not. If they were highlighted, one question would be how to portray the different types of modifiers. If different colors were used, the report could be very distracting. Another issue is that the highlighting example does not adequately reflect that one positive match was made because both **cardiomegaly** and **bilateral pleural effusion** were present and that another positive match was made independently because **pulmonary vascular congestion** was present.

When the conditions to be retrieved are complex and require matching sets of several modifier-finding groups, and when the sentences in the report are long and complicated, the issues will be compounded. One question will be how to best show the association between the modifiers and the findings. For example, in *the spleen and liver are moderately enlarged*, there is a relationship between *enlarged* and *liver*, and between *enlarged* and *spleen*, but highlighting the three words does not show those relations.

The user interface issues that are raised by having a highlighting capability need to be studied further, to determine what types of highlighting helps users read the report more accurately and what types inadvertently mislead the user. By emphasizing limited information, users may be more likely to skip information in the report that is important but that was not highlighted.

Conclusions

We have designed a model that embeds structured encoded information in a textual report using XML, and we have implemented an automated procedure using natural language processing that automatically processes clinical reports and transforms them into a valid XML output form consistent with the model. Associating structured output with portions of the original report adds significant functionality to the report. It means that applications can utilize the structured component of the XML output to obtain highly specific retrieval capabilities and then be able to highlight relevant information, thereby facilitating manual review. For example, a special browser could highlight specific information, such as diagnoses, procedures performed, medications given, or pertinent history, in order to assist the user in the reading of a report.

References ■

1. Sager N, Lyman M, Buchnall C, Nhan N, Tick L. Natural language processing and the representation of clinical data. J Am Med Inform Assoc. 1994;1(2):142–60.
2. Friedman C, Alderson PO, Austin J, Cimino JJ, Johnson SB. A general natural language text processor for clinical radiology. J Am Med Inform Assoc. 1994;1(2):161–74.
3. Hripcsak G, Friedman C, Alderson P, DuMouchel W, Johnson S, Clayton P. Unlocking clinical data from narrative reports. Ann Intern Med. 1995;122(9):681–8.
4. Haug P, Ranum D, Frederick P. Computerized extraction of coded findings from free-text radiologic report. Radiology. 1990;174:543–8.
5. Zweigenbaum P, Bachimont B, Bouaud J, Charlet J, Boisvieux J. A multilingual architecture for building a normalized conceptual representation from medical language. Proc 19th Annu Symp Comput Appl Med Care. 1995:357–61.
6. Baud R, Rassinoux A, Scherrer J. Natural language processing and semantical representation of medical texts. Methods Inf Med. 1992;31(2):117–25.
7. Rassinoux A, Wagner J, Lovis C, Baud R, Scherrer J. Analysis of medical texts based on a sound medical model. Proc 19th Annu Symp Comput Appl Med Care. 1995:27–31.
8. Lenert L, Tovar M. Automated linkage of free-text descriptions of patients with a practice guideline. Proc 18th Annu Symp Comput Appl Med Care. 1994:274–8.
9. Gundersen M, Haug P, Pryor T, et al. Development and evaluation of a computerized admission diagnoses encoding system. Comput Biomed Res. 1996;29:351–72.
10. Lovis C, Gaspoz J, Baud R, Michel P, Scherrer J. Natural language processing and clinical support to improve the quality of reimbursement claim databases. Proc AMIA Annu Fall Symp. 1996:899.
11. Lovis C, Baud R, Scherrer J. A semi-automatic ICD encoder. Proc AMIA Annu Fall Symp. 1996:937.
12. Department of Health and Human Services. International Classification of Diseases. Washington, D.C.: DHHS, 1990.
13. Sager N, Nhan N, Lyman M, Tick L. Medical language processing with SGML display. Proc AMIA Annu Fall Symp. 1996:547–51.
14. Zweigenbaum P, Bouaud J, Bachimont B, Charlet J, Seroussi B, Boisvieux J. From text to knowledge: a unifying document-oriented view of analyzed medical language. Methods Inf Med. 1998;38, in press.
15. XML Web site. Available at: http://www.w3.org/XML.
16. McGrath S. Parseme 1st: SGML for Software Developers. Englewood Cliffs, N.J.: Prentice Hall, 1998.
17. Maler E, Andaloussi J. Developing SGML DTDS: From Text to Model to Markup. Englewood Cliffs, N.J.: Prentice Hall, 1996.
18. Digital Imaging and Communications in Medicine (DICOM). NEMA PS3.1–PS3.12. Reston, Va.: DICOM, 1992.
19. Knirsch C, Jain N, Pablos-Mendez A, Friedman C, Hripcsak G. Respiratory isolation of tuberculosis patients using clinical guidelines and an automated decision support system. Infect Control Hosp Epidemiol. 1998;19(2):94–100.
20. Lyman M, Sager N, Tick L, Nhan N, Borst F, Scherrer J. The application of natural-language processing to health care quality assessment. Med Decis Making. 1991:11(suppl): S65–8.
21. Zingmond D, Lenert L. Monitoring free-text data using medical language processing. Comput Biomed Res. 1993;26: 467–81.
22. Spyns P. Natural language processing in medicine: an overview. Methods Inform Med. 1996;35:285–301.
23. Friedman C, Alderson P, Austin J, Cimino J, Johnson S. A general natural language text processor for clinical radiology. J Am Med Inform Assoc. 1994;1(2):161–74.
24. Jain N, Knirsch C, Friedman C, Hripcsak G. Identification of suspected tuberculosis patients based on natural language processing of chest radiograph reports. Proc AMIA Annu Fall Symp. 1996:542–6.
25. Johnson S, Friedman C. Integrating data from natural language processing into a clinical information system. Proc AMIA Annu Fall Symp. 1996:537–41.
26. Jain N, Friedman C. Identification of findings suspicious for breast cancer based on natural language processing of mammogram reports. Proc AMIA Annu Fall Symp. 1997: 829–33.
27. Hripcsak G, Kuperman G, Friedman C. Extracting findings from narrative reports: software transferability and sources of physician disagreement. Methods Inform Med. 1998;37: 1–7.
28. Friedman C, Starren J, Johnson S. Architectural requirements for a multipurpose natural language processor in the clinical environment. Proc. 19th Annu Symp Comput Appl Med Care. 1995:347–51.
29. Sowa J. Conceptual Structures: Information Processing in Mind and Machine. Reading, Mass.: Addison-Wesley, 1984.
30. Hypertext Markup Language 4.0 Specification. HTML Web site. Available at: http://www.w3.org/TR/REC-html40/. Accessed Oct 20, 1998.
31. Extended Markup Language Specifications. XML Web site. Available at: http://www.w3.org/TR/REC-xml/. Accessed Oct 20, 1998.
32. Text Encoding Initiative (TEI). TEI Home Page: Available at: http://www-tei.uic.edu/orgs/tei/index.html. Accessed Nov 13, 1998.
33. Chemical Markup Language (CML). Available at: http://www.venus.co.uk/omf/cml. Accessed Oct 20, 1998.
34. Open Financial Exchange (OFX). Open Financial Exchange. Web site. Available at: http://ofx.net/ofx/ab-main.asp/. Accessed Nov 13, 1998.
35. Cover R. Home Page. Available at: http://www.oasis-open.org/cover. Accessed Nov 13, 1998.
36. Hood E. Collection of Perl programs for processing SGML. perlSGML. Web site. Available at: http://www.oac.uci.edu/indiv/ehood/perlSGML.html. Accessed Oct 13, 1998.

37. IBM XML Tools Web site. Available at: http://www.alphaworks.ibm.com/formula. Accessed Oct 20, 1998.
38. Microsoft XML tools Web site. Available at: http://www.microsoft.com/xml/parser/jparser.asp. Accessed Oct 13, 1998.
39. Dolin R, Alschuler L, Bray T, Mattison J. SGML as a message interchange format in health care. Proc AMIA Annu Fall Symp. 1997:635–9.
40. Rosenthal D, Sokolowski R. Using SGML for voice-enabled structured medical reporting. Proceedings Graphic Communication Association SGML '96. 1996:191–6.
41. Dilks C, Kutlesa J. SGML-based electronic drug submissions: a case study. Proceedings Graphic Communication Association SGML '96. 1996:583–7.
42. Alschuler L, Dolin R, Spinosa JC. SGML in healthcare information systems. Proceedings SGML Europe '97. May 11–15, 1997; Barcelona, Spain; pp. 195–204.
43. SGML/XML Special Interest Group. Health Level 7 (HL7) Web site. Available at: http://www.mcis.duke.edu/standards/HL7/committees/sgml/. Accessed Oct 13, 1998.
44. The HL7 Document Patient Record Architecture. Health Level 7 (HL7) Web site. Available at: http://www.mcis.duke.edut/standards/HL7/committee/sgml/WhitePapers/Prap. Accessed Oct 20, 1998.