MEDINFO 2004
M. Fieschi et al. (Eds)
Amsterdam: IOS Press

CliniViewer: A Tool for Viewing Electronic Medical Records Based on Natural Language Processing and XML

Hongfang Liu[a], Carol Friedman[b]

[a]*Department of Information Systems, University of Maryland at Baltimore County*
[b]*Department of Biomedical Informatics, Columbia University*

Abstract

With the evolving use of computers in healthcare, the electronic medical record (EMR) is becoming more and more popular. A tool is needed that would enable physicians to accurately and efficiently access clinical information in multiple medical records associated with a particular patient. Both natural language processing (NLP) and the eXtensible Markup Language (XML) have been used in the clinical domain for capturing, representing, and utilizing clinical information and both have shown great potential. In this paper, we demonstrate another use of XML and NLP through CliniViewer, a tool that organizes and presents the clinical information in multiple records. We also describe the flexibility and capability provided when combining XML and NLP to summarize, navigate, and conceptualize structured information. The tool has been fully implemented and tested using patients with multiple discharge summaries.

Keywords

Natural Language Processing, XML, Electronic Medical Record, Summarization

Introduction

With the evolving use of computers in healthcare, the electronic medial record (EMR) is becoming more and more popular. However, the promise of the EMR for saving physicians time in chart review is far from achieved because of the lack of tools that enable physicians to quickly identify and locate information they need for a particular patient when the information occurs in multiple medical records.

To avoid medical errors and improve health care, it is critical for physicians to be familiar with a patient's clinical history so that they can quickly ascertain previous problems, medications, and procedures. One way for a physician to obtain this information would be to manually review all the medical records associated with the particular patient. However, that is generally very time-consuming and impractical. An alternative way would be to use an automated system to retrieve and summarize all the pertinent information, and to present it using an effective user interface. For example, WebCIS [1], a Web-based clinical information system used at New York Presbyterian Hospital (NYPH), allows physicians to electronically view individual reports. However, more intelligent processes would be needed to summarize the data and to organize it according to concepts, such as medications, and problem lists, etc.

Currently, with the growing need for exchanging clinical information, XML [2,3] has been used for the representation and utilization of clinical information. For example, Health Level 7 (HL7) [4] has published an ANSI accredited document standard CDA (clinical document architecture) using XML [5], where CDA is a three-level clinical document architecture with each higher level adding more specificity to the markup of the documents. An XML marked up report has been shown to offer great potential for sharing and exchanging clinical information across different organizations [3,5].

NLP information extraction systems in the clinical domain offer unique opportunities for extending the use of the EMR because they structure and encode narrative text in the EMR so that the information can be used by various automated systems [6,7]. However, most NLP systems in the clinical domain are used to deliver information in narrative text to decision support systems, or coding systems [8,9], and the end users are rarely physicians.

In this paper, we present a novel method for displaying clinical information to physicians that involves combining XML and NLP techniques. This method offers flexibility when viewing, summarizing and navigating information extracted from multiple reports of a single patient. This method was implemented in a system named CliniViewer, an electronic medical record viewer, which utilizes an NLP system, MedLEE [9], and an XML navigation interface called TreeViewer [10].

There has been some work on presenting, summarizing and navigating EMRs associated with a single patient. In a paper written by Estrada et al. [11], a tool called Puya was described that was used to attract a physician's attention to abnormal physical findings in clinical notes by eliminating sentences describing normal physical findings. Krauthammer et al. [12] designed a knowledge model to condense the rich representation of information generated by MedLEE as a result of processing a single report in order to present different predefined views according to the desired level of granularity. One view was designed for clinical purposes and the other for data mining. QCIS [13], which is a multiple-view generation system, could also generate views with multiple orientations (i.e., source-oriented, time-oriented, and concept-oriented) from multiple reports associated with a given patient A user must first type the term of interest and then select a concept from a list of matched controlled vocabulary concepts

that is subsequently provided. Thus, it only provides clinical information associated with the selected concept and the information can only be displayed according to the three predefined orientations. CliniViewer is different from related work in several ways: i) it utilizes XML tree structure to provide a view that summarizes all clinical findings identified by NLP and a view that arranges original text reports in a tree structure; ii) it contains a feature that enables the two views to communicate with each other; and iii) users can specify orientations of views dynamically. In the following, we first present background information of the method. We then describe the design and the implementation of the CliniViewer system. The experiment and the results are discussed next. Finally, we provide a discussion and conclude the paper.

Background

At NYPH, an NLP system, MedLEE [9], is used to extract, structure, and encode clinical information in domains that include radiology reports, pathology reports, and discharge summaries. MedLEE was also integrated with automated systems and used for decision support [9] and vocabulary development [10].

The output format of MedLEE is XML, and is organized according to sections of the report, where each section contains two children: one contains **structured** conceptual representations of the clinical findings and the other contains XML tagged text (**tt**) of the original report. For example, for a section of a discharge summary (i.e., *history of present illness*) with one sentence, as shown in Figure 1(a), the XML formatted output of MedLEE is shown in Figure 1(b). A clinical finding is categorized using an XML tag (e.g., **problem**) and the associated value (e.g., *pain*) is recorded using an attribute **v**. A clinical finding can have modifiers (e.g., **bodyloc** *arm* or **status** *develop*), where each modifier can also have modifiers. For example, **bodyloc** *arm* in Fig 1b) also has a modifier **region** *right*. Additionally, references (e.g. **idref)** to the original text are also included in the structured output. For example, the problem *pain* in Fig 1b) has an **idref** attribute *p2,* which refers to the tagged textual portion. In this example, it refers to the text with the **phr** tag which has an **id** attribute with the value *p2*, and which occurs in the sentence corresponding to sentence identifier **sid** *s1.1.1*. The original text of the report is enriched with tags to delineate and identify the elements **sent** (marking sentences), phrases **phr** (marking textual phrases), and undefined words **undef**. Each occurrence of the elements **sent** and **phr** is associated with an identifier so that the element can be referenced by the structured conceptual component.

A tree is an effective navigation structure, and can be seen in applications, such as Windows Explorer or registry editors. In a tree, the information is displayed in a hierarchical order, usually with the broader topic displayed at the top level while related items are displayed as children. An XML document, by nature, forms a tree. We have previously developed a JAVA application called TreeViewer [10] which presents a tree containing terms physicians use in reports. It provides a navigation interface that allows users to view and sort the tree for the purpose of vocabulary development and maintenance. It enables users to locate items easily, to see compositional components of the terms, and to see their frequencies. The tree displayed by TreeViewer is generally obtained from multiple trees, which were originally XML structured output generated by MedLEE as a result of processing a large corpus of reports associated with many patients. The current implementation of TreeViewer [14] allows users to dynamically obtain views with different orientations. For example, a list of clinical findings can be viewed based on their main semantic categories (e.g., problem, finding, etc) or can be viewed based on body locations or certainty information of these findings.

History of present illness: Intermittent pain in right arm developed on 12/1/99. **(a)**

```
<section c = "history of present illness" >
  <structured form = "xml">
    <problem v = "pain" idref = "p2>
      <bodyloc v = "arm" idref = "p5">
        <region v = "right" idref = "p4"/>
      </bodyloc>
      <onset v = "intermittent" idref = "p1"/>
      <status v = "develop" idref = "p6"/>
      <date v = "19991201" idref = "p8"/>
      <sid idref = "s1.1.1"/>                                    (b)
    </problem >
  </structured>
  <tt>
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"> 12/1/99 </phr>. </sent>
  </tt>
</section>
```

Figure 1 - An example of the structured component of the output form generated by MedLEE

System Design and Implementation

Figure 2 depicts an overview of CliniViewer. It contains four functional components: MedLEE, Tree Generator, TreeViewer and Communicator. MedLEE and Tree Generator run on the server side while TreeViewer and Communicator run on the client side. On the server side, the original textual reports are first parsed using MedLEE, and the XML output of MedLEE is then modified, transformed, and merged into two different XML trees by Tree Generator: one tree is a conceptual tree that provides a summarized view of the concepts in all the reports, and the other tree is a report tree that consists of the original reports, organized as XML trees. These two trees are then transferred to the client side, and each tree is loaded using the TreeViewer interface. A communicator is also constructed, which enables the two trees to communicate with each other.

The input to the system is a set of medical reports for a single patient, where each report is identified by a corresponding event start date (**sd**), event end date (**ed**), exam type (**et**) and report

type (**rt**). The reports are parsed using MedLEE to generate structured XML output. Report identifiers and the XML output of MedLEE are used by Tree Generator to generate the two trees. In order to create the conceptual tree, the structured clinical information extracted by MedLEE from all the reports is merged. In order to create the report tree, the reports are arranged according to report identifier, sections and sentences. In the conceptual tree, positive findings are separated from negative findings. This is possible because the findings generated by MedLEE have positive and negative modifiers. A finding is shown as negative in the conceptual tree if it appears with negative modifiers (e.g., **certainty** with the value *no*) in all reports; otherwise, it is shown as positive.

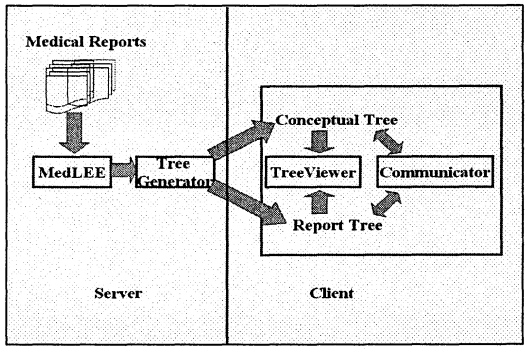


Figure 2 - The overview of Cliniviewer

Figure 3a) illustrates the conceptual tree that was generated from the structured component shown in Fig 1a). A tag called **item** was inserted to separate an original tag (i.e., **problem**) from its **v** attribute (i.e., *pain*). This was done so that multiple values for the same type of clinical finding could be summarized and viewed as children for that type using TreeViewer. For example, in order to represent a different **problem** *tender* in Fig. 3a), an item whose value is *tender* would be inserted as a sibling of *pain*. A tag named **modifier** was also created and for each modifier tag an attribute was generated where the value of the attribute was the original **v** value. For example, the modifier **bodyloc** became an attribute **bodyloc** with a value *arm*. Additionally, an element called **report**, which is associated with the identifier attributes (i.e., **et**, **rt**, **sd**, and **ed**) was inserted between the clinical findings and the corresponding modifiers in order to associate the findings to particular reports. Section information and source information corresponding to the findings were also attached after the modifiers. In the report tree as shown in Fig. 3b), text is arranged according to the document hierarchy (i.e., **report**, **section** and **sent**) captured by MedLEE.

After the two trees are generated, they are transferred to the client side and displayed using TreeViewer. The communicator is established when loading the trees so that they can communicate the shared information with each other (i.e., **report**, **section**, **sid**, and **phr**). The communicator becomes activated by the interface when the user clicks the right button of the mouse. When a node is selected in the conceptual tree, a click on the right button causes all sentences that contain the selected concept to be displayed and the appropriate phrase to be highlighted. Similarly when a node is selected in the report tree, a click on the right button causes the conceptual tree corresponding to that sentence to be displayed.

```
<concepttree>
 <positivelist>
  <problem>
   <item v = "pain">
    <report sd = "1999-12-03" ed="1999-12-07"
            rt = "dsum" et = "dsum">
     <modifier>
      <item bodyloc = "arm"
            bodyloc_region = "right"
            onset = "intermittent"
            status ="develop"
            date = "19991201">
       <section c = "history of present illness"
                sectiontext = "History of present
illness:">
        <source sid = "s1.1.1"
                phrset = "p1 p2 p4 p5 p6 p8"/>
       </section> </item> </modifier></report>
   </item></problem></positivelist></concepttree>
```

(a)

```
<reporttree>
 <report sd = "1999-12-03" ed="1999-12-07"
         rt = "dsum" et = "dsum">
  <section c = "history of present illness"
           sectiontext = "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"> 12/1/99 </phr>.
  </sent> </section></report></reporttree>
```

(b)

Figure 3 - An example of the conceptual tree (a) and the report tree (b) generated from Figure 1.

Experiment and Result

We conducted an experiment to determine whether the system functioned properly and promptly. We used all discharge summaries associated with three different patients who were randomly selected where each had more than 10 discharge summaries in the clinical data repository. Pseudo medical record numbers (i.e., *1234567*, *2345678*, and *3456789*) were created for the patients. There were 15, 31, and 38 discharge summaries for each of the three patients respectively. CliniViewer generated the default views for each patient in less than 30 seconds, where the server and the client were running on different computers located in two adjacent buildings. The PC configurations for the server and the client are 3.0Ghz Pentium 4 with 512 MB RAM and 2.4GHz Pentium 4 PC with 512 MB RAM respectively. Figure 4 a) shows a snapshot of CliniViewer after requesting views for patient *1234567*. The top frame in Figure 4 shows the conceptual tree.

Note that there were 294 occurrences of medication items in all the discharge summaries of patient *1234567*, where the most frequent occurrence of a medication was *lasix*.

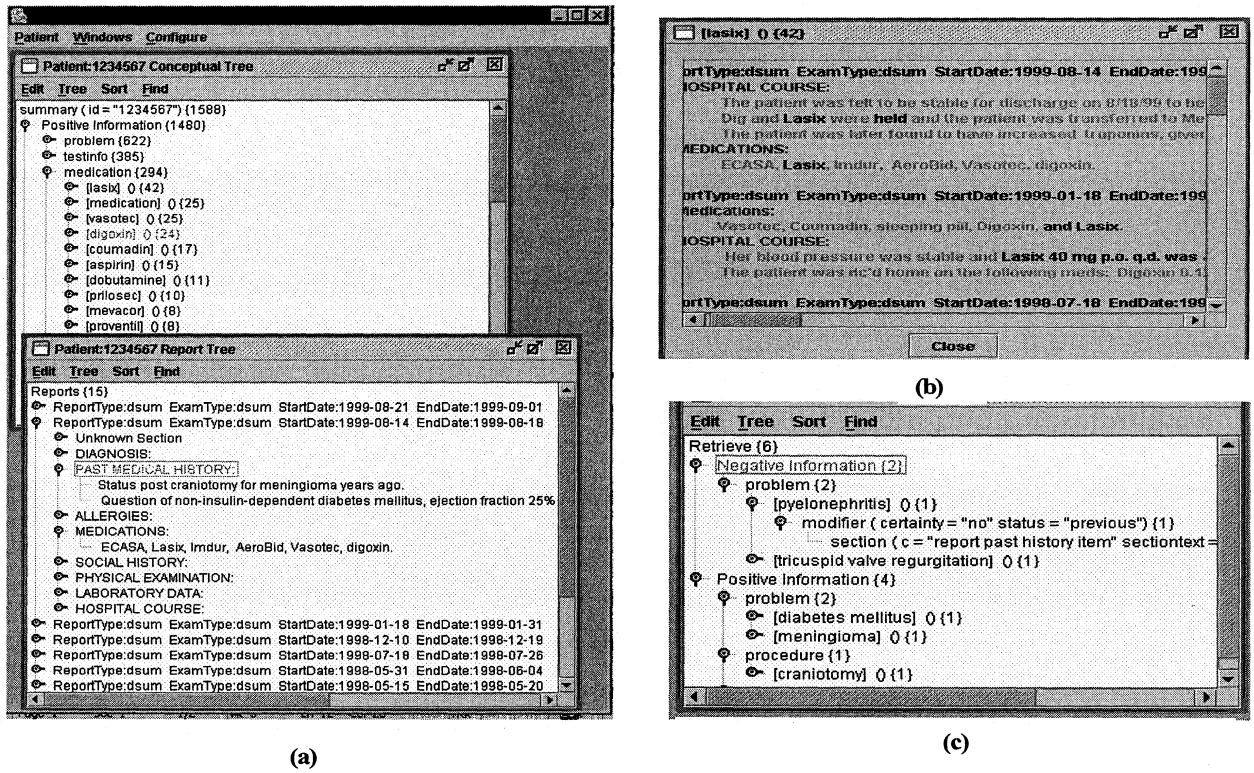


Figure 4 - A snapshot for CliniViewer for patient 1234567 showing various diverse views of the information

In this application, the number of occurrences represents the total number of occurrences in all the reports, and not the total number of reports the findings occur in. When the node *lasix* was selected in the conceptual tree and the right mouse button was clicked, all sentences containing the medication *lasix* together with the exact positions in the original reports were shown within a second and the corresponding phrases were highlighted (see Figure 4b). All reports together with their identifiers appear in the report tree as the children of the root (see the bottom frame of Figure 4 a). When the *past medical history* node was selected in the report tree and the right mouse button was pressed, the associated concepts extracted by MedLEE were shown (see Figure 4 c).

Discussion

In this paper, we presented a tool for physicians to access multiple reports associated with a particular patient using MedLEE and TreeViewer. The tool provides two views of information in the reports by using a communicator between the two trees, which can easily be activated by a mouse click. In the experiment, we only used discharge summaries. The tool could also be used to view diverse types of reports related to the patient, such as radiology reports or pathology reports.

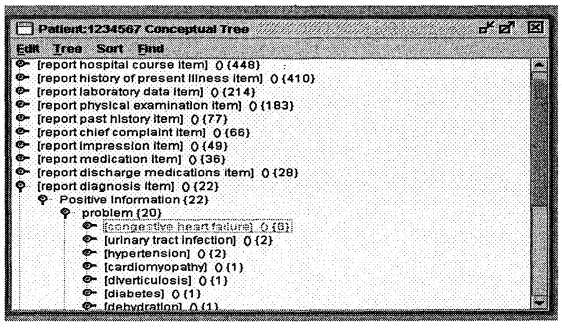
During the implementation of the tool, we found that XML was an extremely flexible mechanism. By transforming the XML documents, the tool can dynamically present many different views of the information according to the user's preference. For example, Figure 5 a) shows a different view of the conceptual tree focusing on sections. Note that in that view *congestive heart failure* occurred the most frequently in the diagnosis sections of patient *1234567*. The communicator enables users to verify the accuracy of the structured clinical findings and to easily navigate between the two views (see Figure 5 b) and see the text corresponding to *congestive heart failure,* which occurred in the discharge diagnosis section).

Additionally, TreeViewer, which is used to view XML documents, is a flexible navigation interface. For example, the only difference in the use of TreeViewer between the vocabulary development tool [10] and CliniViewer is the input because for vocabulary development the input includes reports associated with many different patients.

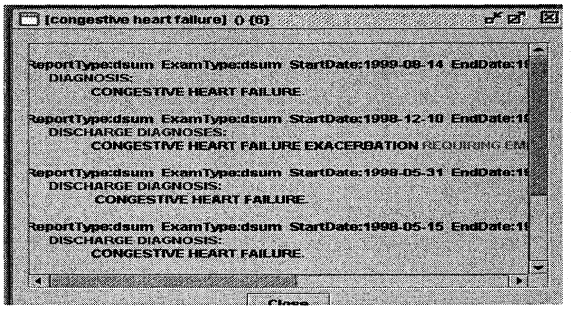
Currently, the frequency information for each node corresponds to the number of occurrences. Thus, findings that are mentioned several times in one report are counted several times. However, the number of different reports a finding occurred in could be easily obtained by adding a new attribute or by changing the way the number of occurrences is computed.

Organizing medical reports via a tree structure is advantageous because physicians can navigate medical reports easily when they are nodes of a tree. More knowledge could be added to the

conceptual viewer to provide more relevant information that could be tailored to the physician's interest. Currently, individual nodes (i.e., sections and sentences) in the report tree are obtained using XML output generated by MedLEE. However, clinical reports that are XML compliant, such as those consisting of CDA formats, can also be loaded into the viewer because it is an interface designed for any XML tree.



(a)



(b)

Figure 5 - A different view of the conceptual tree for patient 1234567

Conclusion

In this paper, we described a tool that allows physicians to conveniently identify and locate clinical information occurring in multiple patient reports. This tool demonstrates that XML is a very flexible mechanism. It also demonstrates that NLP can be used to provide a conceptual level summarization of the information in the EMR.

Acknowledgments

This study was supported in part by grant LM06274 and grant LM7659 from the NLM and Columbia Center for Advanced Technology.

Reference

[1] WebCIS https://webcis.cpmc.columbia.edu. 2003.

[2] XML Web-site. 1998. http://www.w3.org/XML.

[3] Sokolowski R, Dudeck J. XML and its Impact on Content and Structure in Electronic Healthcare Documents. *Proc. AMIA* 1999:147-151.

[4] HL7. 1998. http://hl7.org.

[5] Dolin RH. An Update on HL7's XML-based Document Representation Standards. *Proc. AMIA* 2000:190-194.

[6] Spyns P. Natural Language Processing in Medicine: An Overview. *Meth Inform Med* 1996; 35:285-301.

[7] Chuang J, Friedman C, Hripcsak G. A Comparison of the Charlson Comorbidities Derived from Medical language Processing and Administrative Data. *Proc. AMIA* 2002:160-164.

[8] Fiszman M, Chapman WW Aronsky D, Evans RS, Haug PJ, Automatic Detection of Acute Bacterial Pneumonia from Chest X-ray Reports. *J Am Med Inf Assoc* 2000 7(2):593-604.

[9] Friedman, C. and Hripcsak, G. Natural language processing and its future in medicine. *Academic Medicine.* 1999;74(8):890-895.

[10]Liu H, Friedman C. A Method for Vocabulary Development and Visualization based on Medical Language Processing and XML. *Proc AMIA* 2002:502-506.

[11]Estrada WD, Murphy SN, Barnett G. Puya: A Method of Attracting Attention to Relevant Physical Findings. *Proc. AMIA* 1997:509-513.

[12]Krauthammer M, Hripcsak G. A Knowledge Model for the Interpretation and Visualization of NLP-parsed Discharge Summaries. *Proc. AMIA* 2001:339-343.

[13]Zeng Q, Ciminoj. Providing concept-oriented views for clinical data using a knowledge-based system: an evaluation. *JAMIA* 2002; 9(3) : 294-305

[14]Friedman C, Liu H, Shagina L. Vocabulary Development Visualization Tool based on Natural Language Processing and the Mining of Textual Patient reports. *JBI* 2003; 36 189-201

Address for correspondence

Department of Information Systems,
University of Maryland at Baltimore County,
1000 Hilltop Circle, Baltimore 21250