TagMeds - A Tool for Populating eXtensible Markup Language Documents with UMLS Concept Unique Identifiers of Current Medications

By

Andrew S. Nakrin

B. A. Psychology, New York University, 1973

M. D., Rush Medical College, 1981

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Electrical Engineering and Computer Science at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2001

Signature of Author_____

Department of Electrical Engineering and Computer Science

May 11, 2001

Certified by:_____

Peter Szolovits

Professor of Electrical Engineering and Computer Science

Thesis Supervisor

Accepted by:_____

Arthur C. Smith

Professor of Electrical Engineering and Computer Science

Chairman, Committee for Graduate Students

This research was supported by a National Library of Medicine Training Grant.

TagMeds - A Tool for Populating eXtensible Markup Language Documents with UMLS Concept Unique Identifiers of Current Medications

by

Andrew S. Nakrin, MD

Submitted to the Department of Electrical Engineering and Computer Science on May 11, 2001 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering and Computer Science

ABSTRACT

TagMeds is a system that recognizes and marks textual descriptions of a patient's current medications in the unstructured textual content of consultations letters. Medications are found based on their names and on linguistic patterns describing their dose, form of administration, etc. The UMLS is used as the underlying database of terms, and detected medications are encoded into XML tags consistent with and making use of the Health Level 7 (HL7) Clinical Document Architecture. The specific aims of this research are: (1) to review the literature in order to determine the state of the art in tagging free text for search and utilization, (2) to construct a tool that will reliably generate UMLS Concept Unique Identifier tags of current medications within free text. The methods involved are: (1) creating Perl procedures to recognize patterns in free text to retrieve the UMLS Concept Unique Identifiers and to insert these unique identifiers into XML tagging of the text and (2) statistical analysis of the use of TagMeds on a data base of consultation letters from the Endocrinology Clinic of the Children's Hospital of Boston as compared to manual markup by a group of physicians. The performance of an NLP system is found to be at least as sensitive as the performance of physicians in the extraction of current medications and their attributes. The tagged current medication information has the potential to support a personal electronic medical record system, such as PING. Additional development of TagMeds is likely to bring significant improvements, with modest expenditure of time and effort. TagMeds demonstrates that great utility can be achieved with a medical natural language processing system using simple and unsophisticated techniques.

Thesis Supervisor: Peter Szolovits

Title: Professor of Electrical Engineering and Computer Science

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to the following:

Peter Szolovits, Ph.D., thesis supervisor, for his unswerving faith, moral rectitude, and insightful advice.

Isaac Kohane, MD, Ph.D., and Lucila Ohno-Machado, MD, Ph.D., for their material aid and advice.

The entire MIT Laboratory for Computer Science Clinical Decision-Making Group, who made this work possible, easy, and fun.

The following physicians for manually marking a hundred records and placing their hearts of gold into the creation of a "gold standard" for this experiment: Horner Chen, Phebe Chen, Meghan Dierks, Steven Dierks, Ken Dols, Jeff Gilbard, Steven Greenberg, Ken Mandl, Mary Nash, James Nash, Lionel Nelson and Lisa Penas.

My wife, Teresita King, MD, FACOG, and my daughter, Joy Nakrin, Haverford College Class of 2002, for their limitless love, devotion, support and understanding.

Krassimir Paskalev, MIT Class of 2001, for his generous debugging help.

Hillel Alpert of Vital Science & Health for his explanation of and guidance with the statistical analysis.

Toya Conner for help with data entry.

CONTENTS

CHAPTER 1

INTRODUCTION

One of the problems in medicine today is presenting useful information to the user of a medical record without giving back mountains of data that the user must sort through. A physician or other healthcare giver who takes over the care of a patient either gets the electronic medical record or picks up the chart. They often have to fish through a mountain of data to try to find what they need to know. Once the relevant information is found, the rest of the pile of paper is an obstacle.

A physician wants to know what medications the patient is currently taking and what allergies the patient has. These are the two things that absolutely need to be known before a physician can go on. The physician could read the notes from the previous physician—the history, the physical, and the problem list—or he or she could talk to the patient. The physician could do all of the above and still not know anything about the patient's problem. A simple tool is needed to retrieve the essential information from the mass of irrelevant information.

Physicians need to know the current medications for the following reasons:

- for reordering to provide continuity of care;

- for analysis to see what problems the other physician has addressed; and
- to be aware of what drugs are currently in the patient's body in case they will interact with the drugs about to be given.

The medical group at the MIT Laboratory for Computer Science and the Children's Hospital Informatics Program to attempt to address this problem developed an electronic medical record called PING (Personal Internetworked Notary and Guardian).[i] [ii] These record builders encountered the same problem that has been noted by other builders of electronic medical records: the problem of presenting only information useful to the user of the record and not mountains of data that the user must sort through to understand.

The bulk of data entries often come in the form of free text. Data entries that are not free text can be converted into free text by known methods. Facsimile documents can be converted to free text using any one of a number of readily available Optical Character Recognition (OCR) programs. Telephone and dictation data can be converted to free text using Speech Recognition programs.

A specified electronic medical record is only rarely retrieved for the purpose of examining the original entries. Usually the record is retrieved to learn information that will be immediately useful, such as when a physician requests a medical record to determine a patient's medication.

The most efficient and effective way to meet this need is to process the free text when it is entered into the medical record. The meanings associated with the free text can be inserted, parenthetically, into the text by use of eXtensible Markup Language (XML). This process simplifies the retrieval of needed information. The text is processed upon entry, and never needs to be analyzed again. This is most efficient overall, and even more dramatically reduces the waiting time between the request for information by the user and the display of the needed information.

Tagging

The process of placing labels into free text is called tagging. The terms *free text* and *character data* signify the material that is typed and entered into the electronic medical record before any alteration is performed. Free text entered into an XML-tagged document may not contain opening or closing pointed brackets (<, >).

The free text character data is analyzed to understand its meaning and structure. This is called parsing. While the character data is being analyzed, a piece of information that may be sought after at a later time is found and tagged. We can call this piece of information a *lexeme*.

Tagging a lexeme consists of placing an opening pointed bracket (<), a specified opening label, and a closing pointed bracket (>) before the piece of information and an opening pointed bracket, a specified closing label, and a closing pointed bracket is placed after the lexeme (for example <label>lexeme<label>). These bracket and label

combinations are called the opening and closing tags. Now the text is no longer free text, but parsed character data. The text contains character data plus XML tags. The search process involves searching for the opening tag (<label>) and sorting until the closing tag is found. The piece of information between the pair of tags can then be presented to the user as the response to a request.

A review of the literature indicates that it is not within the reach of present technology to tag all free text with its meaning. Complete meaning analysis is prevented by problems such as simple spelling errors, the impreciseness of the language, elisions that omit the actual subject being discussed, redundant concepts in the medical dictionary, homonyms, acronyms, abbreviations, concepts that were missing from the medical dictionary, and proper names. There are positive examples of researchers who have searched through the free text entries and found specific types of information that were then tagged with their meanings. The Overview of Current Methods section below cites examples of circumscribed natural medical language processing that are more successful, and attempts at more global natural medical language processing that seem to reflect the limits.

TagMeds

TagMeds is another step in this process. Use of TagMeds can answer the following questions:

- Precisely which medications is this patient taking at the present time?
- How much of each medication is being taken with each dose?

- By what route is each dose being given?
- How frequently is such a dose being given?
- For what duration is such a regimen in place?

The scope of the information culled by TagMeds is circumscribed, but the utility of this information is great. A physician or other primary caregiver that undertakes the care of a patient must first give the order to continue the current medications in order to provide continuity of care. She must evaluate the current medications to determine if the current problems in treatment result from the appropriateness of the regimen, and incrementally alter the regimen as indicated. She can determine if new problems may be side effects or adverse reactions to the current medications. She must determine how any planned new therapeutic interventions will interact with the current medication. The current medications act as a pointer to the thought processes of the previous caregivers in relation to what they thought they were treating. Often these sound deductions are left unstated in the medical record and unstated by the patient. Only a review of the current medications can yield such insight. When a physician undertakes to continue a patient's care, she begins by being crystal clear on what the patient's current medications and known allergies are. TagMeds provides a method of quickly and accurately retrieving the information about current medications for the user.

The hypothesis

*The hypothesis of this thesis is that TagMeds can find a patient's current medications in free text as well as a physician can. This hypothesis is tested by comparing the

performance of TagMeds on a sample of 100 letters of consultation against the performance of a group of physicians. If the hypothesis is correct, then TagMeds has utility as a time and labor saving tool. If the null hypothesis is rejected, and the data show that the physicians function better than TagMeds, the utility of TagMeds is open to challenge. If the null hypothesis is rejected, and the data show that TagMeds functions better than the physicians, then there is evidence that TagMeds has definite utility that may save time and labor and improve patient care.

Overview of current methods

PING

The Personal Internetworked Notary and Guardian (PING) is a proposed secure, distributed, scalable lifelong personal medical record. PING will incorporate a patient's complete medical record from all sources, which will be easily viewable by the patient, his healthcare provider, and significant others at any time and from any place. The patient may carry access to his complete lifetime medical record around anywhere he or she goes in the form of a smart card or other authentication tool.

For any given patient, a healthcare provider will make entries into the medical record via a number of routes. The text entry could be e-mailed, faxed, or phoned to the PING server, or the healthcare provider could dictate an entry. This information is given to authorized users such as doctors who have permission to care for the patient, the patient themselves, or family members the patient gives permission to, with proper

authentication. The use of PING requires a user name, password, and a secure location or a secure identification card.

Each PING entry consists of the header portion and the data portion. The header portion will contain information about the entry, including the author (healthcare provider), the owner (patient), the people and their roles who are permitted to make an entry, the date and time of creation, and the type and format of the entry. The data portion of the entry is the author's entry itself, which should include text and which is defined as character data and XML tagging. The data portion may also include data other than text, provided that the data has been provided with XML tags that label its format. In this way, data generated using other programs carries with it the instructions for its translation into the language of PING.

The patient and those with permission to access the records could then easily access the record over the Web, after providing identification to the PING server with, for example, a secure card. The lifetime medical record could then be viewed with a personal computer and a Web browser.

Medical natural language processing

Fiszman et al. used a natural language processing system and two keyword searches to detect the presence or absence of three pneumonia-related concepts and inferred the presence or absence of acute bacterial pneumonia from 292 chest X-ray reports.[iii] A

gold standard consisted of a majority vote of three independent physicians. The reliability of the gold standard was compared to the results of the system. In extracting pneumonia related concepts from chest X-ray reports, the performance of the natural language processing system was similar to that of physicians and better than that of lay persons and keyword searches. This demonstrates the power of matching strings in a dictionary with strings in free text to recognize and understand concepts. In a limited domain, such as concepts expressing acute bacterial pneumonia in Fiszman's experiment or current medications in TagMeds, simple literal string matching can be a powerful tool, limited only by the time it takes to search the text, character by character, for each string in turn.

Hripcsak et al. built a natural language processor that detects the presence or absence of six clinical conditions in admission chest radiograph reports.[iv] The system performed as well as physicians and better than lay people on a sample of 200 chest X-ray reports. This lends support to the hypothesis that natural language processors in restricted domains can be effective tools.

Lovis et al. compared the efficiency of various text pattern-matching algorithms, and found that the Boyer-Moore-Horspool algorithm, achieves the best overall results when used with medical texts.[v] This algorithm is a string-matching algorithm that compares characters from the end of the search string to its beginning. When characters don't match, the search jumps to the next possible match. Perl was selected for TagMeds because of its many efficient intrinsic string searching and matching algorithms.

Perl, which is designed for text processing, is the programming language used to write the code for TagMeds. Perl proceeds line by line; it considers an entire line of text and deals with it as an item of data.

Gabrieli and Speth developed a MUMPS-based text-to-meaning engine that was able to leverage extensive, rapid lookup in assigning meaning to text.[vi] This tool set has become the property of Lernout & Houspie and is not available for research or commercial use. The lessons of extensive lookup form a part of the basis for TagMeds.

Lexicons

Johnson et al. constructed a resource for providing semantic information about words and phrases, called lexemes, in medical narratives.[vii viii ix] This resource used the 1997 Metathesaurus of the Unified Medical Language System (UMLS) of the National Library of Medicine (NLM) to create a semantic lexicon constructed using the CORBA[1] architecture, which extracted about 79% of the meaning from medical narrative. The limitations demonstrated in attempting comprehensive automated text-to-meaning translation have guided the goals of TagMeds toward a more limited scope and reach. The resources and talents placed in the service of the UMLS make it a useable and very promising medical dictionary to use in constructing a text-to-meaning engine.

---

[1] The Common Object Request Broker Architecture (CORBA) is an emerging open distributed object computing infrastructure being standardized by the Object Management Group. CORBA automates many common network programming tasks such as object registration, location, and activation; request demultiplexing; framing and error-handling; parameter marshalling and demarshalling; and operation dispatching.

Prakash et al. used the 1999 UMLS Metathesaurus to process discharge summaries and surgical notes with a concept-identification program; 82.6 percent of concepts identified were true positives.[x] Causes of problems were redundant concepts in the UMLS, homonyms, acronyms, abbreviations and elisions, concepts that were missing from the UMLS, proper names, and spelling errors. The error rate was too high for concept indexing to be the only production-mode means of preprocessing medical narrative. This article contains a good analysis of the complications that should steer the researcher away from an immediate goal of comprehensive text-to-meaning analysis.

Joubert et al. used UMLS to query information databases by using conceptual-graphs pattern matching to operate a semantic integration of information databases using the UMLS knowledge sources.[xi] Bodenreider et al. showed the superiority of UMLS to another lexicon, MAOUSSC.[xii] Many groups of researchers are working with, testing, and perfecting the UMLS.

Kim et al. utilized the statistical properties of word pairs and triples to identify a large list of humanly acceptable phrases in the medical field as a part the UMLS.[xiii] The authors used the UMLS list of phrases as a gold standard for validating their methods. The quality of the method was found to be sufficient to support the automatic placement of hyperlinks in text at the site of highly ranked phrases. As discussed by Lovis et al., multiple word phrases are more difficult to search for, and slow the processing of free text. The advantage, that the exact meaning of a phrase is much less ambiguous than

the meaning of an individual word, is shown here. For this reason, TagMeds attempts to achieve a balance between speed and accuracy by searching for a limited number of phrases.

Bakken et al. tested the adequacy of the Clinical Logical Observation Identifiers, Names, and Codes (LOINC) semantic structure as a terminology model for standardized assessment measures.[xiv] The results supported the adequacy of the Clinical LOINC. Huff et al. described the history of the development of the LOINC vocabulary and the methodology used in its creation.[xv] The scope of the LOINC vocabulary is too circumscribed to be used as the basis for the TagMeds project, by intention of its design. The UMLS contains a table, a one-to-one mapping, that contains the UMLS Concept Unique Identifier and the equivalent LOINC term identifier rendering translation from one medical dictionary to the other a simple process. Nothing in the LOINC vocabulary is unavailable to the UMLS user. The researcher gains advantages of speed and the reduction in the number of redundant entries by using the most circumscribed vocabulary possible. LOINC is so circumscribed that it cannot serve.

Sager was a pioneer in the natural language processing of clinical data. The New York University Linguistic String Project (LiSP) uses syntax, grammar, and structure to enter medical free-text into a database for query. Information precision and information recall was 89.6% and 92.5%, respectively.[xvi xvii] This is an impressive result for a comprehensive text-to-meaning analysis. The use of a database to store processed strings provides a high standard for fast retrieval of processed text.

Ohno-Machado et al. tested the GuideLine Interchange Format (GLIF) model for encoding clinical guidelines. GLIF was sufficient to model the guidelines, but the encodings revealed substantial variability.[xviii] This interesting advance still illustrates the difficulty in assigning meaning to free text. These difficulties stem from the inherent ambiguities in the language as well as redundancies in the medical dictionary. TagMeds is another effort that attempts to use a still more circumscribed goal to facilitate success in the face of the ambiguities GLIF has faced.

Schulz et al. began the process of ensuring quality encoding using the Read Code Thesaurus.[xix] [xx] Cooper et al. have automatically encoded the free text of electronic medical records into Medline Subject Headings (MeSH) terms, which are then used to perform MedLine searches.[xxi] An interesting review of the goals and accomplishments of the Read Code Thesaurus has been published by Tange et al.[xxii] Both the Read Code and the MeSH terms are medical dictionary languages mapped one-to-one to UMLS Concept Unique Identifiers in the UMLS.

Huibert et al. conceptually divided the retrieval of information from clinical narratives into two parts: searching the labeled content, and extracting the meaning. He showed that the finer the granularity of the content labels, the faster meaning could be extracted.[xxiii]

SNOMED

Where do you look up the terms? What medical dictionary do you use to find the meaning of the terms that you encounter? Systematized Nomenclature of Medicine (SNOMED) is a medical dictionary that has been developed by the American College of Pathologists.[xxiv] [xxv] It is complete, clinically oriented, well organized, and has a good level of granularity, and it is the best one around today for purposes of this experiment.

Campbell et al. compared coding using SNOMED, UMLS, and the Read Codes. The authors assembled 1929 source concept records from a variety of clinical information taken from four medical centers across the United States. The source data included ample coding of medications. SNOMED was judged to be significantly more complete in coding the source material than the other schemes and had substantially more duplications of coding assignments associated with a loss of clarity.[xxvi] The researchers concluded that:

> SNOMED International is considerably more complete, has a compositional nature and a richer taxonomy. It suffers from less clarity, resulting from a lack of syntax and evolutionary changes in its coding scheme. READ has greater clarity and better mapping to administrative schemes (ICD-10 and OPCS-4), is rapidly changing and is less complete. UMLS is a rich lexical resource, with mappings to many source vocabularies. It provides definitions for many of its terms. However, due to the varying granularities and purposes of its source schemes, it has limitations for representation of clinical concepts within a computer-based patient record.

All of the terms encoded with UMLS Concept Unique Identifiers during the course of this research had equivalent SNOMED unique identifiers listed for them as well, in the UMLS files. Clearly SNOMED could have been used in place of the UMLS for this research. Based upon the statement of Campbell et al, SNOMED may well be a better choice. It should be noted that the varying granularities and purposes may not be the result of any flaw in the execution of the UMLS dictionary, but may result from the nature of its mission as a metathesaurus encompassing varying language dictionaries of varying purposes. It should also be noted that the most suitable language is probably a function of the purpose for which said language will be use, and not a global, overarching conclusion.

We elected to use the UMLS rather than SNOMED because the NLM is applying such substantial resources into refining, testing, and disseminating the UMLS; licenses the use the UMLS for research purposes available free of charge; and makes the UMLS readily available for research without charge.

The Unified Medical Language System

The U.S. Department of Health and Human Services, the National Institutes of Health National Library of Medicine (NIH NLM) has developed the Unified Medical Language System (UMLS). The UMLS is actually a composite of all of the other medical dictionaries that are available in electronic form. Each medical dictionary is a subset of the UMLS. The UMLS Metathesaurus contains about 800,000 concepts and 1,900,000 concept names. MetamorphoSys is system software provided by the NLM that allows

users to exclude any vocabularies that are not helpful or for which they are not licensed. The authoritative documentation for the UMLS is available from the NLM Web site. (47)

The UMLS contains electronically available Knowledge Sources that can be used by a wide variety of application programs for patient records, bibliographies, factual databases, and expert systems.[xxvii] There are three Knowledge Sources: the Metathesaurus, the Semantic Network, and the Specialist Lexicon. The Metathesaurus is the central vocabulary of the UMLS. The Semantic Network categorizes all concepts and the relationships between them. The Specialist Lexicon is a general English language lexicon that includes commonly occurring English words and biomedical terms also commonly found in medical natural language.

The Metathesaurus

The Metathesaurus is the main vocabulary of the UMLS.[xxviii] It is a database of concepts taken from different controlled vocabularies and classifications used in medicine, many of which were discussed above. The Metathesaurus retains the meanings, attributes, relationships, and hierarchies that are used in the source vocabularies. The Metathesaurus establishes synonyms and relationships between one controlled vocabulary and the others. The synonyms and related terms can help users to locate the same concepts as defined in other selected vocabularies.

The Metathesaurus contains concepts from 60 controlled vocabularies, and often from more than one version of a specified vocabulary. Prominent among these component vocabularies are the following:

- The American Medical Association's Current Procedural Terminology,
- the College of American Pathologists' SNOMED–Systematized Nomenclature of Medicine,
- the National Library of Medicine's MeSH–Medline Subject Headings,
- Beth Israel Deaconess Medical Center's BI98,
- Massachusetts General Hospital's COSTAR–COmputer STored Ambulatory Records,
- the American Psychiatric Association's DSM–Diagnostic and Statistical Manual,
- the World Health Organization's ICD–International Classification of Diseases, and
- the Regenstrief Institute's LOINC–Logical Observations Identifiers, Names, and Codes.

The Metathesaurus contains terms in 13 spoken natural languages.

Many of these source vocabularies are copyrighted, and use of the UMLS involves proper licensing not only from the NLM but also from the holders of the various individual copyrights. MetamorphoSys is useful in removing vocabularies for which appropriate licenses have not been obtained.

The Metathesaurus is organized by concept. A Concept Unique Identifier (CUI) names each concept. The purpose is to link alternative names and descriptions of the same concept to the same CUI. Each name for or description of a given concept has a String Unique Identifier. Each of the same strings describing the concept in different spoken natural languages has a separate String Unique Identifier.

A term is defined as the group of all strings that are lexical variants of each other. One string for each term is designated as the preferred usage. One may use MetamorphoSys to change the string that is designated as the preferred usage. For example, if one wanted to always use the SNOMED term in one's computer-generated messages, then the SNOMED terms could be automatically designated as the preferred usage.

The MetamorphoSys requires the full UMLS distribution to be present. The space required for the full Metathesaurus is 3 gigabytes. The Metathesaurus with all of its subset may require another 3 gigabytes. Eight gigabytes of free disk space and 256 megabytes of RAM are recommended to run MetamorphoSys. These large space requirements and the concomitant time requirements prompted us to search for a way of extracting only what was needed from the Metathesaurus to be used by TagMeds. We devised a procedure that extracted only the English language terms and their CUIs and placed them into a 25-megabyte file.

The Semantic Network

The UMLS Semantic Network keeps track of all the concepts in the Metathesaurus and the relationships between them. The Semantic Network assigns a semantic type to each concept and keeps track of the relationships between semantic types. The semantic types are called *nodes*, and the relationships between them are called *links*.

The Specialist Lexicon

The Specialist Lexicon provides information needed for natural language processing. Words often have inflected forms that are variations of the same word in differing tense and participle. The Specialist Lexicon facilitates understanding of what lies in common throughout these variations, leaving one entry for each part of speech for which the term is used.

XML in Natural Language Processing

Several groups have been working on an interesting approach to the XML parsing of electronic healthcare records.[xxix] This approach uses root tags that outline the document type and source. Further branches of the XML parse tree are generated by automated natural language processing of the text. The natural language processor extracts the meaning of the text; the coding of that meaning, in a standard medical vocabulary, is encoded in a set of XML tags. Having the standard meaning of the text precoded in the XML tags allows for rapid searching of text documents.

Two systems in clinical use today generate root XML tags specifying source documents. These systems go on to use natural language processors to generate attribute tags that encode the meaning of the text. This facilitates later search, research, and utilization of the text.

MEDLEE

Friedman's system, which was tested at Columbia Presbyterian Medical Center, uses a natural language processor known as MEDLEE to generate XML tagging. MEDLEE utilizes Columbia's proprietary medical dictionary system, the Medical Entities Dictionary (MED).[xxx] [xxxi] It uses a document model that provides reliable and efficient access to clinical information in patient reports for a broad range of clinical applications and implements an automated method using natural language processing that maps textual reports to a form consistent with the model. Of the reports so generated, 99.5% were valid XML forms consistent with the Document Type Declaration (DTD)[2]. Because of the potential commercial value of the system, MEDLEE's source code is not available. It is not likely that Columbia University would allow MEDLEE to run on a PING server. Dr. Friedman suggested that for experimental purposes, the PING server, which could then be referred to a CPMC server for XML tagging by MEDLEE and their XML parser, could do preprocessing on text.

[2] Note that DTD is an overloaded acronym. It may specify the Document Type Declaration, a set of strictures in addition to the XML 1.0 recommendations created by the document designers useful in making certain that the document contains all the required parts. DTD is also commonly used by XML programmers to specify the Data Type Descriptor, a set of notations given within a Document Type Declaration defining the nature of the data contained in a particular type of element.

James Cimino has reviewed and summarized a decade's experience with the MED in use at Columbia University and the New York Presbyterian Hospital.[xxxii] The review of MED, the basis for MEDLEE, demonstrates how well it supports the use of coded patient data for a variety of knowledge-based activities, including the improved understanding of patient data, access to information sources relevant to specific patient care problems, the application of expert systems directly to the care of patients, and the discovery of new medical knowledge.

Kurzweil AI

Sokolowski at Magnolia Technologies has a related system based on a natural language processor provided by Kurzweil Applied Intelligence (AI).[xxxiii xxxiv xxxv] The system is the prototype of a voice-enabled, structured medical reporting system. The physician dictates to the system, which then uses automatic speech recognition and medical knowledge bases to produce a structured report. This report is then formatted and viewed on a computer screen, stored in databases of patient information, transmitted to other systems, used to support outcome studies, or viewed on a Web browser. The XML format represents the data in a way that can be read by both computers and humans, and efficiently communicated to a wide range of databases and communications protocols. Kurzweil AI has since been sold to Lernout and Houspie. The product is no longer available.

Other systems

Shiffman et al. have published on GEM, a hierarchical, XML-based guideline model. Shiffman has proposed an XML Guideline Element Model. Tags relating to Major concepts relate to identity, developer, purpose, intended audience, method of development, target population, knowledge components, testing, and review plan.[xxxvi] [xxxvii] Tarczy-Hornoch has applied XML to clinical genetic testing data and text. GeneClinics is an object-oriented database containing a combination of data and semi-structured text that is rendered as an XML document reflecting the underlying database schema.[xxxviii] [xxxix]

The tools developed by Friedman et al. and by Kurzweil AI form a model for the development of TagMeds. TagMeds' use of the UMLS as the preferred medical dictionary instead of the MED yields an advantage in allowing a wider group of users and programmers to access the meanings of various lexemes. The reliable parsing and rapid, readable presentation of information, viewable at any location using a Web browser (common to these two applications) should be a standard for PING and TagMeds.

The PING XML Proposal

The PING XML parser will need to handle text, voice input, and facsimile input. The fax input may be converted into a .tiff file and optical character recognition might be applied in order to create text input. Handling text input would be the place to start in developing the parser, which is what TagMeds does.

Exactly how are we organizing and transmitting our messages over the Web? We have used the Health Level 7 organization's Clinical Document Architecture.[xl] [xli] There is a more thorough discussion of HL7 and its Clinical Document Architecture in a succeeding section of this paper. The XML DTD used to parse PING messages complies with the HL7 Version 3 Clinical Document Architecture (CDA) and adds layers of granularity. The HL7 CDA uses SNOMED terms encoded in XML tagging as an extension. Version 3 is still under development and is changing every day, but version 2.3 is what most commercial systems employ today. HL7's CDA provides interoperability. The same tagging can be used no matter what operating system, what programming language, what database, or what software is running, because the XML tags contain the information needed to look it up, understand it, decode it, enter it into the database, and extract it from the database.

For the purposes of this experiment, the UMLS 2000 Metathesaurus has been used in place of the SNOMED terms used by the HL7. The advantages of a rich lexicon with mappings to many source vocabularies, definitions for many of its terms, ready availability, its software system, and the absence of research license fees are decisive in this decision. The limitations imposed by varying granularities and purposes of the UMLS' source schemes are not of great magnitude when the realm of text-to-meaning processing is restricted to medications. The cross vocabulary mappings between the UMLS and SNOMED, in the restricted domain of medications, would not present a

difficult problem should a substitution of SNOMED terms for UMLS CUIs be required at a later date.

SNOMED versus the UMLS

There are varying degrees of granularity in the UMLS. Some medical dictionaries that are incorporated into the UMLS define every level of detail within a subject; others are just very broad and very general. The UMLS varies with the language of origin of a particular term, what can be coded, and what can not be coded. Conversely, when you code up a term that appears in a number of vocabularies, it is unclear which concept from which vocabulary will match it. The UMLS is a less than perfect dictionary.

One argument for using SNOMED is that the Health Level 7 standards organization has developed a Clinical Document Architecture incorporating SNOMED terms as defined elements of clinical documents. HL7 is setting the standard for making information interoperable and portable, and it would make sense to go along with what they are using. This is yet another reason that SNOMED is the ideal medical dictionary to use for this project.

The American College of Pathologists wants $1,000 for an academic research license to use SNOMED. The National Library of Medicine gives a similar license for free to any bona fide researcher that asks. That was a big argument for using the UMLS rather than SNOMED.

The UMLS contains one-to-one mapping for clinical terms, which is the area that SNOMED covers particularly well, including current medications, dosing, and the route between the UMLS CUI and the SNOMED term. The UMLS contains a file in which both identifiers appear as part of the same entry. It will be a simple matter to look up the equivalent SNOMED terms and substitute them for the UMLS terms in the XML markup. Some UMLS terms are being used in this project for now and can be translated to SNOMED without any problem.

It is possible for the PING parser to handle messages outside of the HL7 Version 3 XML text realm if they carry XML tags declaring the XML DTD used. In the future, incoming messages declared as HL7 Version 2.3 XML or other properly declared XML documents could be handled with references to the appropriate DTD. Other messages that are not readable by the PING parser would have to be labeled as such and stored as an otherwise unprocessed blob.

The PING parser and DTD would need to be evaluated in practice. This effort, using TagMeds, will restrict itself to evaluating a set of consultation letters from the Children's Hospital contained in our database. The clinical trials in which PING is used to store the record of individuals from birth to the completion of the first year of life, the record of patients with diabetes, a Personal Genomic Record, or an immunization record might later be used to evaluate the XML parsing tools.

The Document Type Declaration that was developed for use by TagMeds is reproduced here as Appendix B. The first level of granularity, the header portion, is taken from the HL7 Reference Information Model. Increasing levels of granularity were developed for this project. The DTD is rewritten in Backus Naur Form[xlii] for ease of understanding, as Appendix C.

TagMeds is intended to create an XML tagged record to be stored in PING. A discussion of the manner in which XML and the Health Level 7 Clinical Document Architecture Version 3 has been utilized for this purpose is contained in Appendix A.

Perl

Perl was chosen as the programming language for use in this project because it offers a vast array of tools for the examination and manipulation of text, some of which are exceedingly fast, efficient, and easy to use. Perl procedures are modeled after or paraphrased from The Perl Cookbook by Tom Christiansen and Nathan Torkington. Other ideas, and the Perl compiler, are from http://www.Perl.com.[xliii xliv xlv]

Perl reuses features from C, sh, csh, grep, sed, awk, Fortran, COBOL, PL/I, BASIC-PLUS, SNOBOL, Lisp, Ada, C++, and Python and integrates all its features into one language. Perl has also been described as a digested and simplified version of Unix.

Many programmers feel that Perl is close to miraculous for getting things done quickly and efficiently. Its high-level syntax and powerful implementation of regular expressions make it a great tool for quickly and easily creating applications that perform complex

text manipulation. It is also widely used for writing CGI[3] applications for the World Wide Web. Perl's C-like syntax makes it easy to learn even though the language dispenses with pointers, memory allocation, memory-oriented data types, and all the errors that these features can lead to.

Perl is portable across most Unix systems and is available on the Macintosh and Windows NT platforms. The latest release, Perl 5, introduces object-oriented extensions to the language. Finally, Perl is available free of charge through a GNU General Public License[4xlvi] and is installed on most Unix systems already. There is an unusually good selection of books to learn Perl with.

---

[3] The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers.

[4] I can do no better than to quote from the Free Software Foundation Web site: "The GNU Projectwas launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU system. (GNU is a recursive acronym for ``GNU's Not Unix''; it is pronounced "guh-NEW".) Variants of the GNU operating system, which use the kernel Linux, are now widely used...The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too. "

CHAPTER 2

METHODS

Overview

A database was obtained from the Pediatric Endocrinology Clinic of the Children's Hospital of Boston. This database had been de-identified. The database contained 1,147 letters of consultation written in the Endocrinology Clinic. The names of patients, family members, physicians, the addresses, telephone numbers, identification numbers, and other identifying information had been replaced.

Random selection was used to divide the set of 1,147 letters into two sets. One set, the test set, contained 100 letters and was set aside unexamined until the TagMeds software was completely developed. The remaining 1,047 letters were used as a training set.

By examining the training set, a program was developed that searches through free text letters and attempts to match a set of hand-crafted patterns that identify relevant medications, routes of administration, schedules of administration, and dosages. Medication related terms are recognized by comparison against an appropriate subset of the UMLS.

The program was implemented in Perl. The output of the program was the free text letters that were input, tagged and converted into an XML document referring to the Document Type Declaration shown in Appendix D. The appropriate UMLS Concept Unique Identifier was inserted as a required attribute of the tag for each formulation, drug name, that was found in the free text.

The test set of 100 letters was then sent to a group of 20 physicians. The cover letter instructing the physicians on the nature of the experiment and on how to manually mark the 100 letters of consultation is included here as Appendix F.

The first 10 physicians to return a test set, manually marked where medications and medication related terms appeared, had their data sets used as part of the results. The output of the tagging program, when run on the data set, formed the other part of the results.

Selecting information from the UMLS Metathesaurus

One procedure is designed to extract a small portion of the information in UMLS, i.e., only the English language terms and their associated CUIs. This extraction procedure

deals with the problem of looking up a term for a medication, checking if it is a term specifying a medication, and if it is, specifying the unique UMLS concept it refers to.

Dr. William J. Long provided a Perl procedure that queries the UMLS Knowledge Server via a Secure Shell Connection. The time required for word-by-word query-and-response transmission over the Internet makes this technique ill suited to the current project.

Instead, the 3-gigabyte MetamorphoSys software package was downloaded from the UMLS Knowledge Server and unzipped. The MetamorphoSys package poorly suited for a large number of automated searches, chiefly because of a click-button graphical user interface. It was determined that the file MRCON, a part of the MetamorphoSys package, contains all the human language terms, a specification of what language the text term was taken from, the UMLS Unique Identifier, and identifiers from other vocabularies, in a | delimited file. This is a sample entry:

C0000039|ENG|S|L0296452|PF|S0033295|Dipalmitoyl Phosphatidylcholine|0|

Note that the first column contains the UMLS Unique Identifier (C0000039), the second column lists the language from which the text term originates (ENG), and the seventh column lists the term (Dipalmitoyl Phosphatidylcholine).

The purpose of our procedure is to extract a small portion of the information in the UMLS by searching through the 100-megabyte MRCON file. Our procedure writes all

English language text terms and matching UMLS Concept Unique Identifiers (CUIs) into a file that is our own UMLS English language dictionary. This dictionary will then be usable by the rest of the TagMeds system to check if a term is a UMLS term, and if so, to find its UMLS CUI. Our English language UMLS dictionary is a 25-megabyte file, somewhat more manageable than the 3-gigabyte MetamorphoSys. Entries in our English language medical dictionary, umls.dict, have the form:

Dipalmitoyl Phosphatidylcholine|C0000039

The streamlined dictionary that we have created, umls.dict, has been posted on the World Wide Web on the server of the Massachusetts Institute of Technology Laboratory for Computer Science Clinical Decision-Making Group.[xlvii] The Perl procedures developed for TagMeds are included in this report as Appendix E. Appendix G is included for ease in noting the English lexemes and phrases that TagMeds recognizes in its Perl tagging procedure. Some lexemes are used to determine the relevance of contiguous terms and sections. Some are tagged when found. Some are untagged when found. Some trigger the tagging or untagging of contiguous words, numbers or phrases. Some readers may find utility in viewing the English usages outside of the context of the Perl procedures that recognize them. These readers may wish to view Appendix G.

Converting the Letters of Consultation into Free Text

Next the type of input file format to utilize was considered. The initial and most basic form of input upon which TagMeds must work is the text file. The test database for the TagMeds experiment is CWSscrubbed97.mdb, a 32-megabyte Microsoft Access database. CWSscrubbed97.mdb was collected at the Pediatric Endocrinology Clinic at the Children's Hospital of Boston. A set of 1,147 consultation letters is stored in a single table.

Although the Perl DBI[5] allows each letter to be retrieved, processed, and returned to the database, this seems unnecessary. Since the form of input and output in clinical practice for TagMeds will be text, it seems expedient to use the Microsoft Access Wizard to convert the entire table that stores the consultation letters into a single tilde-delimited 3-megabyte text file. A tilde (~) was chosen as a character not otherwise found in the consultation letters to mark the end of one letter and the start of the next.

Converting Free Text Documents into Tagged XML Documents

The next procedure, tag.pl, inserts the XML tags into the free text of the consultation letters. The next major problem was discovered upon attempting to run an early version of the procedure to insert the XML tags into the free text of the letters. The speed of the processing had been greatly streamlined by eliminating the Internet connection, then winnowing down the 3-gigabyte MetamorphoSys to the 25-megabyte umls.dict, and

[5] The DBI is a database interface module for Perl. It defines a set of methods, variables and conventions that provide a consistent database interface independent of the actual database being used.

similarly winnowing down the 32-megabyte CWSscrubbed97.mdb into the 3-megabyte text file containing the letters of consultation. Still, the naive approach of looking up each word from letters in the dictionary requires on the order of 3,000,000 x 25,000,000 = 75 trillion steps, which is time consuming on a 233 megahertz Pentium MMX with Overdrive and 32 megabytes of RAM. A significant optimization of the search algorithm was clearly needed.

Processing

This entire project was conducted on a 233 megahertz Pentium MMX with Overdrive and 32 megabytes of RAM. Widely available commercial technology is much faster and has much greater memory. It was felt that a set of procedures that ran quickly and efficiently on the machine selected for this experiment would generate few problems on any machine likely to be used for the purpose of processing electronic medical records.

One optimization available is a cache-dictionary, which is actually a pattern matcher that looks for the most common medications. When it finds one, it simply substitutes a correctly tagged version of the term, bypassing our UMLS English language dictionary lookup entirely. The cache-dictionary is checked on a first pass by the procedure to insert the XML tags into the free text of the consultation letters, tagging the bulk of the medication terms directly.

Next the text is checked for common language usage patterns that indicate the presence of a patient's medication. Such terms are placed in a hash table where the

term is tagged with a blank tag. The blank tag lets the next processor know that this may be a medication, whether or not it can be found in our English language dictionary. A similar procedure is followed for a list of patterns.

After the above processing, the temporary file becomes the same as the input file, except that each term that might be a medication has a blank tag. The hash table contains all the terms that might be medications. The terms found are counted and the file is examined line by line. If the current line contains a legitimate English language UMLS dictionary entry, the procedure to insert the XML tags into the free text of the consultation letters puts the current UMLS term and UI into slightly more permanent variables, a second hash table. Note that this is the one and only time that the procedure to insert the XML tags into the free text of the consultation letters passes through the English language UMLS dictionary. Hence the program runs in approximately constant time with respect to the largest file, the 25-megabyte English language UMLS dictionary. This is a significant optimization.

The next procedure inserts the XML tags that move on to work with a copy of the input file, but with a blank tag surrounding entries that may be medications. The procedure to insert the XML tags now has access to the hash table that contains the terms that are actually in the UMLS dictionary, paired with their Unique Identifiers. The procedure to insert the XML tags transfers the tagged text to the specified output file.

If the tagged term is not in the hash table, the UMLS lookup failed and "NOT-FOUND" is entered into the medication's blank tag. If the term is in the dictionary, then tag.pl enters the UI into the tag. When tag.pl is done with the UI insertion, it writes everything to the output file that the user has specified, then closes the files properly and freezes the screen for the user to examine.

The procedure to insert the XML tags uses a binary search to look up the term in the UMLS, as opposed to a sequential search used in an earlier version. In order to use a binary search on the English language UMLS dictionary, the file needed to be placed in order and alphabetized by the term searched.

The POSIX library contains a number of functions that allow character-by-character control of the search and substitution. The subroutine that looks up the UI of a text term utilizes the POSIX library. The subroutine takes the parameter passed to it, checks if this parameter is really a UMLS entry, saves the matching UI, and returns the value of the last statement.

At this point, another routine is used to search the output for tagged terms. When a tagged term is found, the surrounding text is matched against common patterns to tag the dose, route, frequency, schedule, and duration of the previously tagged medication. A sample output file, an XML document created by TagMeds from a single letter of consultation, is included in this report as Appendix D.

Randomly Dividing The Database

Another procedure pulls out a random sample of 100 letters to evaluate by hand and again by automated processing by the TagMeds procedure to insert the XML tags. The random sample has been removed and has not been available for use in developing the procedure to insert the XML tags into the free text of the consultation letters.

Assembling PING, TagMeds, and the HL7 Clinical Document Architecture

HL7 Version 3 is a specialization consistent with XML, and the CDA is specialization of HL7 Version 3. The local medication DTD, which we generate for PING Consultation Letters using TagMeds, is a further increase in the level of granularity of the CDA. The CDA header is derived from the HL7 Reference Information Model (RIM) using the HL7 Version 3 MDF. This includes the creation of a header DTD. The PING Server must directly convert the PING Header into a CDA Header using a direct one-to-one mapping from the PING Header elements to similar CDA Header elements by simply couching the PING Header elements in the associated CDA Header XML tags.

The HL7 RIM, as updated in August 2000, provides several levels of detail for generating proper CDA Header tags down to the level of the clinical document. The PING Server as described above must perform all of this tagging. TagMeds will generate the Data Type Descriptors for elements within the clinical document, which is left entirely unspecified in the HL7 RIM.

Eventually, the clinical document Data Type Descriptor will contain subtypes for all free-text document types and then specify a CUI attribute for all terms used within the body of a free-text document. For this project, the only element found within the clinical Document Data Type will be the consultation letter Data Type. Within a consultation letter Data Type, the only tagged element will be the Medication Data Type. Within the medication data type, subsidiary elements will include the CUI, which will specify the form, route, dose amount, strength, and rate of administration. These, in turn, will each include text or numerical data. The tag containing the CUI is a format for specifying which of the 1,500,000 English UMLS CUIs points to the meaning of this particular term. Each element containing a CUI is used to surround the text characters naming a medication.

The Experiment

One hundred of the 1,147 consultation letters in the Children's Hospital of Boston Endocrinology Clinic database, CWSscrubbed97.mdb, have been randomly selected and set aside. The 1,047 remaining letters were used to develop the templates and algorithms for tagging the free text. When the tagging procedure was developed, the 100 letters were given to a group of 20 physicians, not including the author, for manual tagging. The first 10 physicians to return a set of 100 letters, manually tagged, were used to test the hypothesis. The same letters were tagged using the procedure to insert the XML tags. The null hypothesis is that manual tagging of the 100 selected letters is not better or worse than the tagging done by TagMeds.

The author entered the output of TagMeds and the first five manually tagged data sets to be returned into six spreadsheets.

- The first spreadsheet, Formulations, contained anything in the TagMeds XML output surrounded by a <form_cd> tag and anything in the manually tagged data sets that a physician had highlighted and that was the name of a drug.
- The second spreadsheet, Dose, contained anything in the TagMeds XML output surrounded by a <dose_qty> or a <dose_units> tag and anything in the manually tagged data sets that a physician had highlighted and that was the dosing information of a drug.
- The third spreadsheet, Rate, contained anything in the TagMeds XML output surrounded by a <rate> tag and anything in the manually tagged data sets that a physician had highlighted and that was the schedule information of a drug.
- The fourth spreadsheet, Route, contained anything in the TagMeds XML output surrounded by a <route> tag and anything in the manually tagged data sets that a physician had highlighted and that was the route of administration of a drug.
- The fifth spreadsheet, Other Appropriate, contained anything in the TagMeds XML output surrounded by any tag not mentioned above and anything in the manually tagged data sets that a physician had highlighted and that was in the same sentence as or a contiguous sentence to other highlighted information about a drug.
- The sixth spreadsheet, Other Off-the-Wall, contained anything in the manually tagged data sets that a physician had highlighted and that was not in the same sentence as nor a contiguous sentence to the other highlighted information about a drug.

Ms. Toya Conner, a clerical employee of the Harvard Medical School, entered the next five manually tagged data sets in a manner similar to that of the author. After data entry was complete, the author ascertained that Ms. Conner knew that she was free to enter new columns into the spreadsheets and that she had not done so because she had not encountered any new highlighted terms. As a check on the quality of data entry, letter of consultation numbers 1, 2, 50, 51, 99, 100, and a randomly chosen letter from each of the 10 data sets (a total of 70 letters) was compared with the spreadsheet entries.

The time to tag 100 letters using TagMeds was measured several times. The time to perform manual highlighting of the 100 letters was ascertained from each of the physicians.

CHAPTER 3

RESULTS AND DISCUSSION

TagMeds consistently took about 6 seconds to process 100 free-text letters of consultation and enter them into a tagged XML file. The 10 physicians took between 1.5 and 8 hours to examine the same 100 letters of consultation and highlight the current medications and their attributes. Most physicians took between 2 and 3 hours.

The spreadsheet Other Appropriate contained 44 entries that had been marked by one or more physicians, and had not been marked by TagMeds. In all cases, these entries were contiguous to entries for attributes of the same medication that had been marked by TagMeds. In no case was there a medication that would not have been brought to the attention of the user.

Two of these items contained information on the planned duration of treatment. No specification for duration of treatment was found in any of the letters of consultation that were examined as part of the training process for TagMeds. These data points were data points of an uncommon type that TagMeds missed and was supposed to have found, according to its specification.

The other data points in the Other Appropriate spreadsheet fell into two categories. Firstly there was parenthetical summary information that restated other, contiguous information which TagMeds had marked, such as "(for a total of 0.22 mg/kg/day)." The

second group of data points consisted of relevant information that was not a part of the TagMeds specification, such as the indications for treatment with the medication under consideration.

The spreadsheet Other Off-the-Wall contained 60 data points. TagMeds tagged none of these data points, but they each had been tagged by at least one physician. A few of these data points were irrelevant and may have been erroneously tagged. The rest of these data points consisted of information concerning the patient's next visit with the physician. It appears that one or two physicians feel that duration of treatment is a crucial medication attribute, and that when such information is not provided the marking physician would seek out other information that could be used as a presumptive indication of the duration of treatment. While the logic of this argument is clear, the author does not feel that this was part of the original specification of TagMeds promised area of expertise or performance.

The data points in the spreadsheets Other Appropriate and Other Off-the-Wall were not used in the statistical analysis for this experiment. The two legitimate data points would not make for a significant difference in the results and are more easily dealt with in the discussion. The other data points were either redundant or erroneous, and the author feels that it is not necessary to include them in the Gold Standard.

Results consist of summary diagnostic statistics (sensitivity, specificity, predictive value positive (precision), predictive value negative) for TagMeds and each of five physicians

with respect to the gold standard, which is based on the remaining five physicians' reports. The 95% confidence intervals were also caalculated. McNemar's chi-square method[xlviii xlix] for paired proportions between TagMeds and the five non-gold-standard physicians were calculated. Percent agreement and Kappa statistics were calculated. These were all computed for each of the four variables and for the cumulative total. The two data points for duration are not included in the total, and are addressed in the discussion section.

The McNemar test was computed based on matching and non-matching data between subject (TagMeds or physician) and the gold standard. A Bonferroni correction for 25 multiple comparisons is used when interpreting and reporting the p-values of these tests. Thus, a 0.05 cutoff for statistical significance requires a p-value of less than 0.002 per test. When significant, the odds ratio is a reflection of the relative odds of TagMeds versus physician, of matching or identifying a lexeme when the other does not.

The Kappa statistics are not high, mainly due to the adjustment for chance agreement. The McNemar method is adapted in the first analysis in the only way that makes sense to the author, but are recomputed by use of different Gold Standards. The correlation matrix between TagMeds and physicians shown in Table 1 displays whether TagMeds or individual physicians would appear to be more like other physicians. TagMeds does not appear like a physician. The correlation between TagMeds and any physician is at best fair, indicating that TagMeds is doing something different than what a physician is doing.

Consider TagMeds compared to 5 physicians not in the gold standard, with relation to a gold standard of 5 randomly chosen physicians, regarding formulations. TagMeds had a significantly better sensitivity that physicians 1, 2, and 3, and was indistinguishable from physicians 4 and 5. TagMeds had a significantly lower specificity than any physician did. TagMeds had a significant McNemar's Chi Squared with Bonferroni Correction compared to physicians 4 and 5. The odds ratio was low (about 0.3), showing that these physicians were more likely to match the gold standard than was TagMeds. The percent agreement between TagMeds and the gold standard was low, 78.12%, and showing that TagMeds was doing something different than the physicians in the gold standard. The Kappa Statistic was low, 0.1883, and significant. This is believed to be an artifact of the fact that TagMeds and all physicians agreed on the overwhelming preponderance of data points. This renders the difference between the probability of actual agreement and the probability of agreement predicted by chance to be approximately 0, and forces the Kappa Statistic to be erroneously low. The Kappa Statistic is believed not to be helpful in evaluating this study and this attribute in particular.

Consider TagMeds compared to 5 physicians not in the gold standard, with relation to a gold standard of 5 randomly chosen physicians, regarding dose. TagMeds had a significantly better sensitivity that physicians 1, 3, and 4, and was indistinguishable from physicians 2 and 5. TagMeds had a significantly lower specificity than any physician did. TagMeds was not significantly different from any physician by McNemar's Chi Squared with Bonferroni Correction. The odds ratios are therefore not significant. The percent

agreement between TagMeds and the gold standard was high, 93.88%, showing that TagMeds was doing the same thing that the physicians in the gold standard. The author believe that those detailed attributes of a medication that are more likely to be restricted to the current medications section were found in a similar manner by TagMeds and by physicians. The author believes that this demonstrates that the difference between what TagMeds does and a physician does lies in the care that TagMeds takes in examining sections of the letters of consultation where current medications are not usually expected. The Kappa Statistic was fair, 0.3677, and significant.

Consider TagMeds compared to 5 physicians not in the gold standard, with relation to a gold standard of 5 randomly chosen physicians, regarding rate. TagMeds was indistinguishable from all physicians with regard to sensitivity. TagMeds had a significantly lower specificity than physician 4 and 5, and was statistically indistinguishable from physicians 1, 2, and 3. TagMeds was not significantly different from any physician by McNemar's Chi Squared with Bonferroni Correction. The odds ratios are therefore not significant. The percent agreement between TagMeds and the gold standard was high, 88.12%, showing that TagMeds was doing the same thing that the physicians in the gold standard. The author believe that those detailed attributes of a medication that are more likely to be restricted to the current medications section were found in a similar manner by TagMeds and by physicians. The author believes that this demonstrates that the difference between what TagMeds does and a physician does lies in the care that TagMeds takes in examining sections of the letters of consultation

where current medications are not usually expected. The Kappa Statistic was fair, 0.2817, and significant.

Consider TagMeds compared to 5 physicians not in the gold standard, with relation to a gold standard of 5 randomly chosen physicians, regarding route. TagMeds was indistinguishable from physicians 4 and 5 with regard to sensitivity, and significantly better than physicians 1, 2 and 3. TagMeds had a significantly lower specificity than all physicians did. TagMeds was not significantly different from any physician by McNemar's Chi Squared with Bonferroni Correction. The odds ratios are therefore not significant. The percent agreement between TagMeds and the gold standard was high, 86.11%, showing that TagMeds was doing the same thing that the physicians in the gold standard. The author believe that those detailed attributes of a medication that are more likely to be restricted to the current medications section were found in a similar manner by TagMeds and by physicians. The author believes that this demonstrates that the difference between what TagMeds does and a physician does lies in the care that TagMeds takes in examining sections of the letters of consultation where current medications are not usually expected. The Kappa Statistic was moderate, 0.3919, and significant.

Consider TagMeds compared to 5 physicians not in the gold standard, with relation to a gold standard of 5 randomly chosen physicians, regarding all attributes taken cumulatively. TagMeds was indistinguishable from physicians 4 and 5 with regard to sensitivity, and significantly better than physicians 1, 2 and 3 were. TagMeds had a

significantly lower specificity than all physicians did. TagMeds was not significantly different from physicians 1, 2, and 3 by McNemar's Chi Squared with Bonferroni Correction. TagMeds was significantly different from physicians 4 and 5. The odds ratios are low, .408 and .290. The percent agreement between TagMeds and the gold standard was 84.21%, showing that TagMeds was not doing the same thing that the physicians in the gold standard. The Kappa Statistic was fair, 0.2408, and significant.

Results using the most stringent and least stringent gold standards (any vs. all of ten physicians finding a lexeme) were calculated. The separate analyses for Formulation, Dose, Rate, Route, and Total are indicated by an asterisk and the name of variable at the head of the respective sections. The Bonferroni correction now entails 50 comparisons rather than 25, so the associated significance level is 0.001. Note that Route has 36 observations, and the gold standard is 1 (all lexemes found) for all of them.

The chief benefit of the recalculations is to show that the choice of a gold standard did not influence the results of the experiment. The gold standard included 5 randomly chosen physicians not included in any of the statistical comparisons, the original experimental design, and analyzes the output created by TagMeds and the 5 physicians not in the gold standard. The strictest gold standard includes any attribute marked by any physician, and analyzes the output created by TagMeds and by all 10 physicians. The least strict gold standard includes any attribute marked by all physicians, and analyzes the output created by TagMeds and by all 10 physicians. No matter which gold

standard is used TagMeds is as sensitive as or significantly more sensitive than any physician. No matter which gold standard is used TagMeds is as specific as or significantly less specific than any physician.

Figures 1 through 5 show the sensitivity and specificity subtracted from one, for each of the five physicians not chowen to be in the Gold Standard group, and for the automated output of TagMeds, as compared to the majority response of the five physicians chosen to be a part of the Gold Standard group. The curves are drawn to appear as if they are ROC curves[6], although human beings do not gain specificity by dropping sensitivity, nor sensitivity by dropping specificity.

Figures 6 through 10 show the sensitivity and specificity subtracted from one, for each of the ten physicians, and for the automated output of TagMeds, as compared to a Gold Standard taken to be correct if any of the ten physicians tagged an item. The curves are drawn to appear as if they are ROC curves, although human beings do not gain specificity by dropping sensitivity, nor sensitivity by dropping specificity.

Discussion

The picture drawn by this myriad of statistics is a clear one. TagMeds does something that is significantly different than what a physician does. A just case can be made that

[6] Often the clinical researcher is confronted with the question how accurate a particular laboratory test is in identifying diseased cases. The ability of a test to discriminate diseased cases from normal cases is evaluated using Receiver Operating Characteristic (ROC) curve analysis, described by Metz in 1978 and refined by Zweig & Campbell in 1993. ROC curves can also be used to compare the diagnostic performance of two or more laboratory or diagnostic methods, after Griner et al., 1981.

TagMeds is different in a manner that is most useful as a decision support tool, and in a manner that can be easily improved upon in future versions.

The data points for which TagMeds signaled the presence of a medication or an attribute of a medication and none of the physicians indicated such a conclusion were examined. Twelve such points were found. An additional 25 data points were checked, selected in a haphazard manner. In all cases TagMeds had marked legitimate current medications, either the formulation or the dosage, clearly indicated as current, located in the history section of the letter. It appears that the physicians did not examine the history sections carefully.

It would seem that, although the statistics show that TagMeds is failing to match the Gold Standard, the actual situation may be that TagMeds is doing a more thorough and more correct job than the physicians. An interesting hypothesis, based upon the data points that TagMeds caught and none of the physicians caught:
TagMeds has a variable that is reset at the start of each section, named $relevance. $relevance = 0 by default, 1 in most sections, and 2 in the Current Medications section. if the $relevance, an aggressive parsing algorithm is called. if the $relevance is 1, another parsing algorithm is called. No parsing is done where relevance is not 1 or 2. It may very well be possible to eliminate the lack of specificity of TagMeds with relation to the physicians simply by changing the statement that sets $relevance to 1 when parsing the History section of a letter to a similar statement that sets $relevance to 0. This simple, one character change in the Perl procedure would make the specificity of

TagMeds much more like that of the Gold Standard in this study. It is unclear what this would do to the sensitivity, or whether this change is desirable.

The TagMeds output is consistently and significantly more sensitive than or statistically indistinguishable from every physician, no matter whether the Gold Standard is five randomly chosen physicians, a positive finding for an attribute by any physician, or a positive finding by all 10 physicians. This is most clearly true for Formulations, but it is true for all attributes considered.

TagMeds is significantly less specific than or statistically indistinguishable from every physician, no matter whether the Gold Standard is five randomly chosen physicians, a positive finding for an attribute by any physician, or a positive finding by all 10 physicians. This is most clearly true for Formulations, but it is true for all attributes considered.

The system designer may well be justified in sacrificing specificity for sensitivity in a decision support system. As an illustration, take an anesthesiologist who assumes care of a patient and is about to give an anesthetic medication that blocks sympathetic nervous system activity. She needs to know whether the patient is on sympathetic nervous system blocking drugs that can act synergistically with the anesthetic medications and produce profound hypotension. In this example, she has the choice of spending 2 hours examining the patient's 100-page free text medical record. Her other choice is to run TagMeds for 6 seconds and then to spend 10 minutes looking through

the text surrounding the tagged medications to determine which of these medications the patient is currently taking. She can be confident that she is more likely to miss a current medication if she goes through the record manually. Perhaps TagMeds is a useful aid in her care, in this example.

It is important that TagMeds does not appear to be on the same ROC curve in most of the ROC curve-like scatter plot diagrams. It generally appears to fall below the curve. This is an indication that TagMeds is not like a physician who has simply sacrificed specificity in order to be sure to mark everything that is important. TagMeds is worse than that. The sensitivity of TagMeds is very good. As good, or better than, any physician. As good, or better than, all the physicians taken together. But the specificity of TagMeds is very, very low. As low, or lower than any physician. Nonetheless, if catching every potentially important lexeme is the goal, TagMeds does better.

TagMeds is not so sensitive as to flood the utilizing physician with tons of useless tagged text. The 100-letter sample contains 36,000 words and 935 lexemes are tagged by any of the 11 markers (10 human and 1 electronic). Assume, as a worst case, that TagMeds gave the user 935 terms to examine so that the user had to determine from their context whether they were actual current medications or their attributes. This is still quite a reasonable output for the user to examine.

Conclusions

The performance of the NLP system was at least as sensitive as the performance of

physicians in the extraction of current medications and their attributes from pediatric endocrinology letters of consultation. The tagged current medication information has the potential to support a personal electronic medical record system, such as PING. Additional development of TagMeds is likely to bring significant improvements, with modest expenditure of time and effort. TagMeds demonstrates that great utility can be achieved with a medical natural language processing system using simple and unsophisticated techniques where the domain of application is restricted.

FIGURE 1

Figures 1 through 5 show the ROC-like scatter plots of the 5 non-Gold Standard physicians and TagMeds (marked with a T), with respect to individual attributes, and with respect to any attribute, with the Gold Standard being the 5 physicians not shown in these plots.

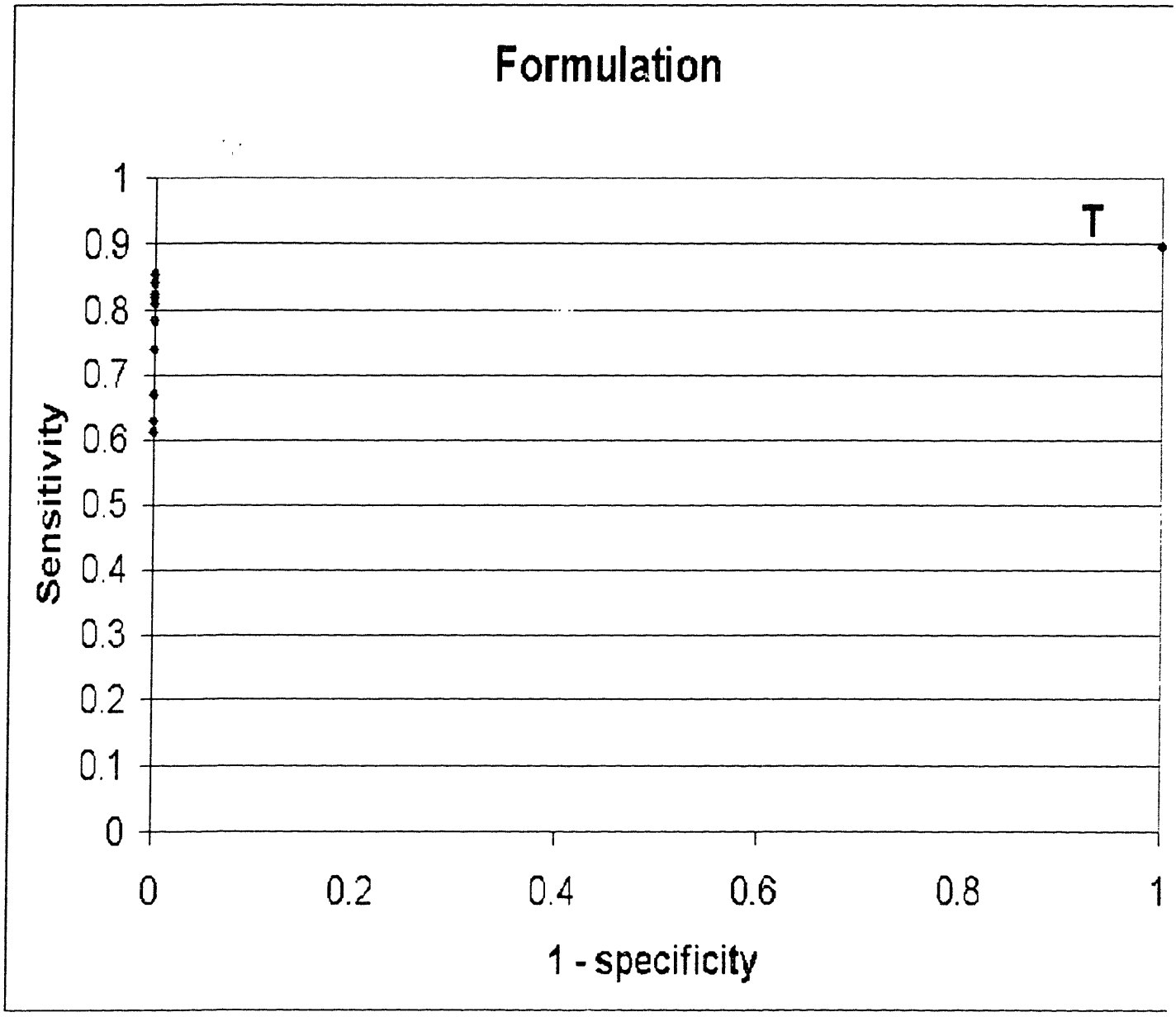

FIGURE 2


Dose
T
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Sensitivity
0
0.2
0.4
0.6
0.8
1
1-specificity

FIGURE 3

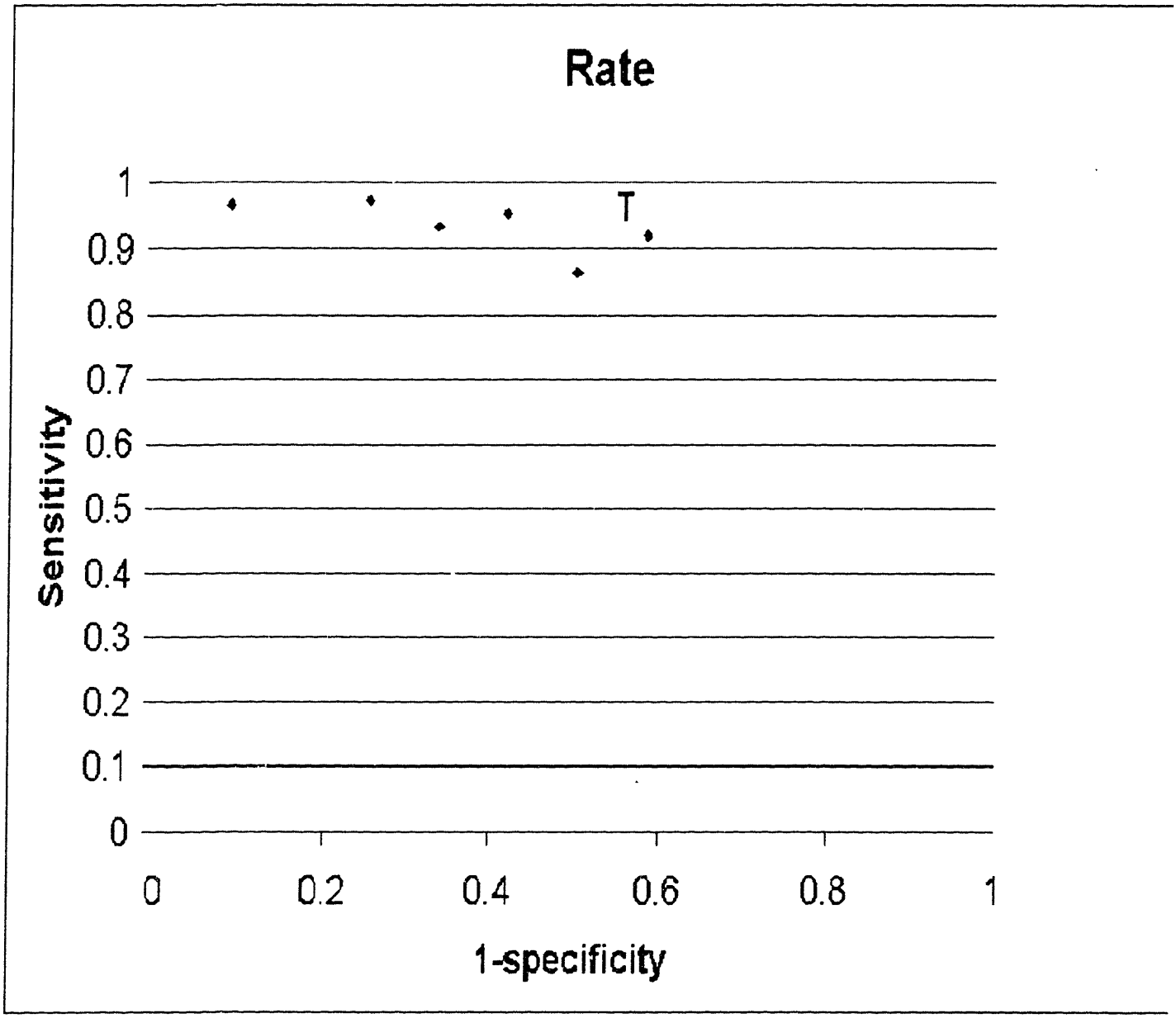
Rate
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Sensitivity
T
0
0.2
0.4
0.6
0.8
1
1-specificity

FIGURE 4

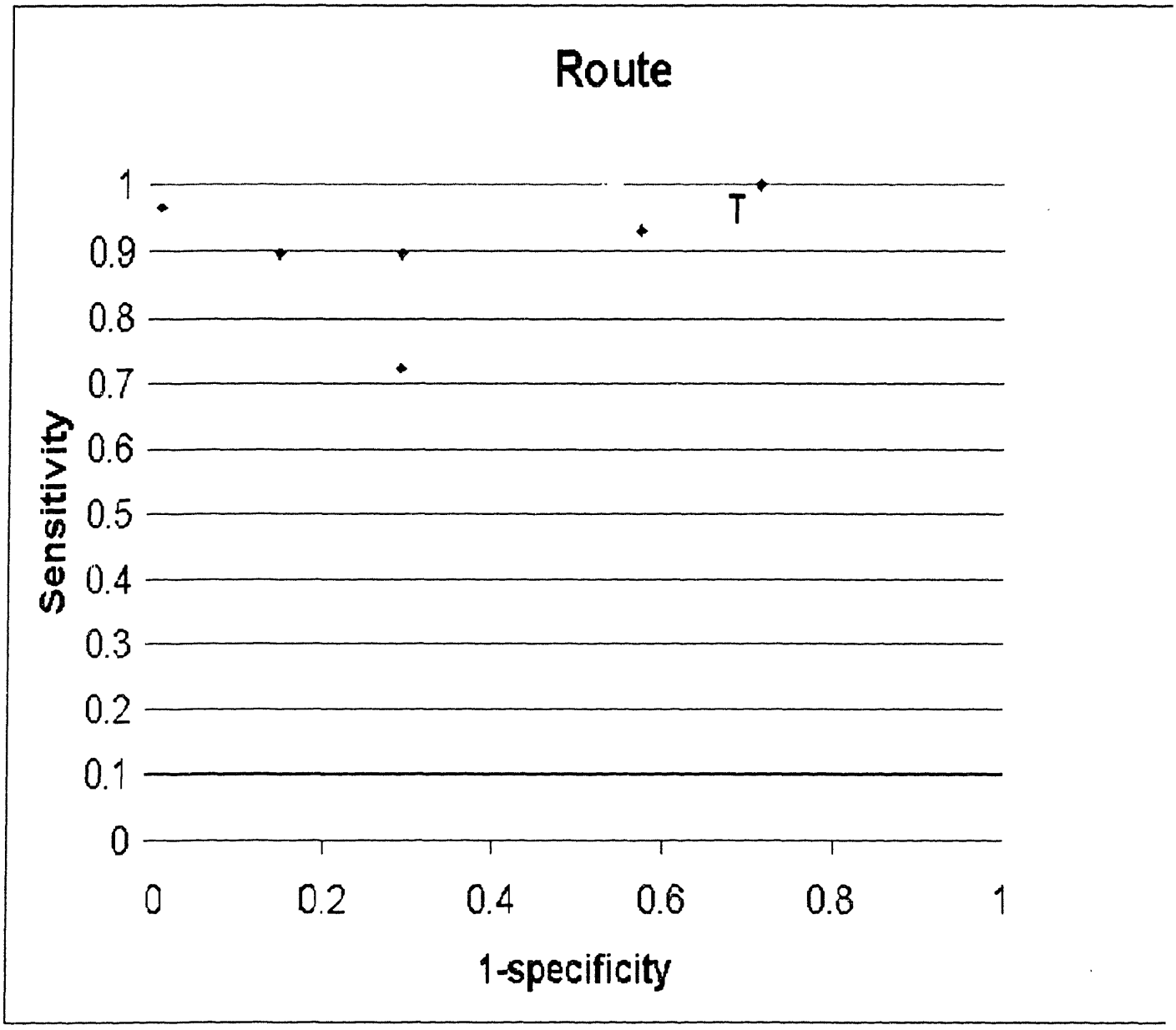
Route
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Sensitivity
T
0
0.2
0.4
0.6
0.8
1
1-specificity

FIGURE 5

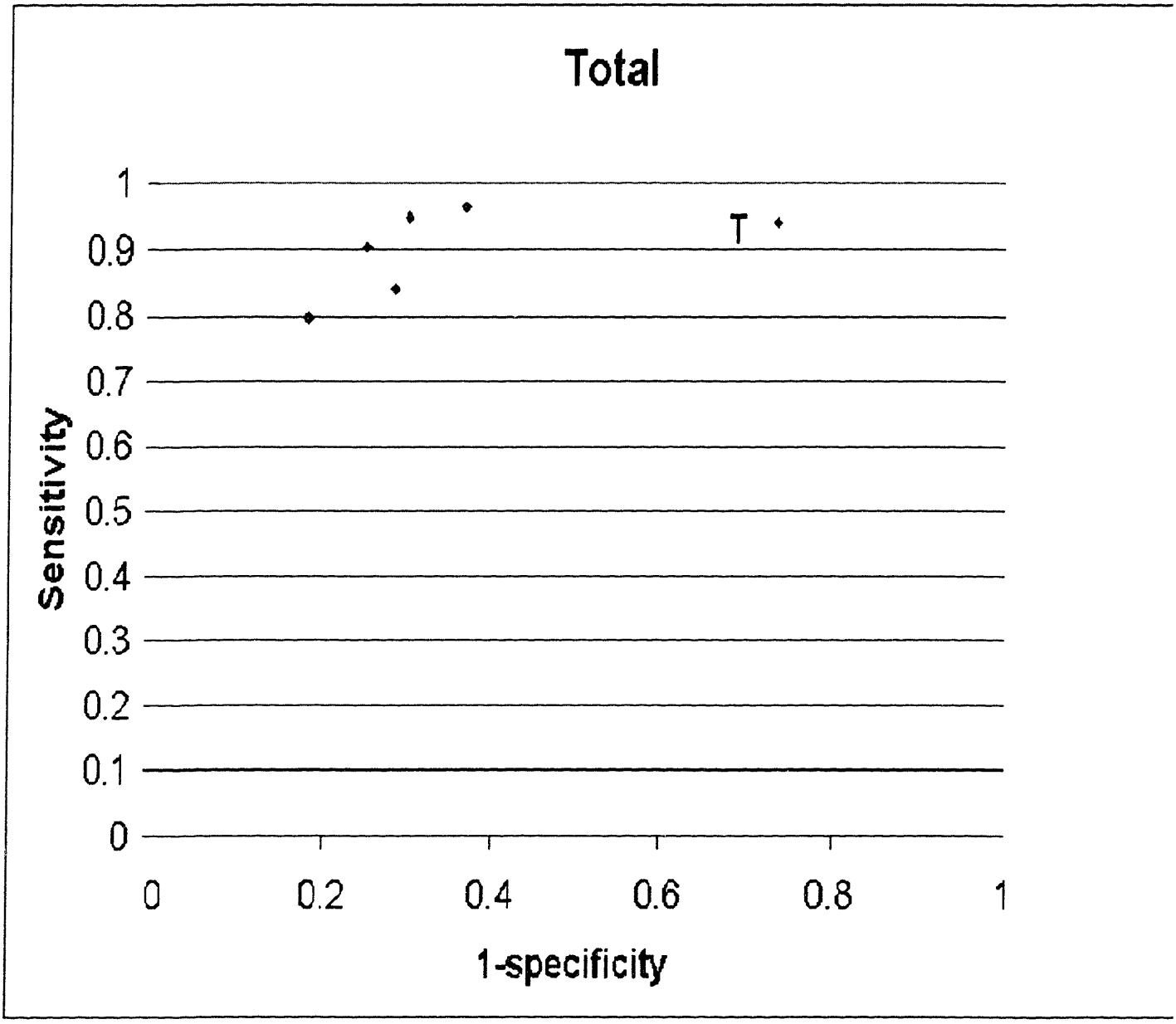
Total
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Sensitivity
T
0
0.2
0.4
0.6
0.8
1
1-specificity

FIGURE 6

Figures 6 through 10 show the ROC-like scatter plots of the 10 physicians and TagMeds (marked with a T), with respect to individual attributes, and with respect to any attribute, using the strictest Gold Standard.

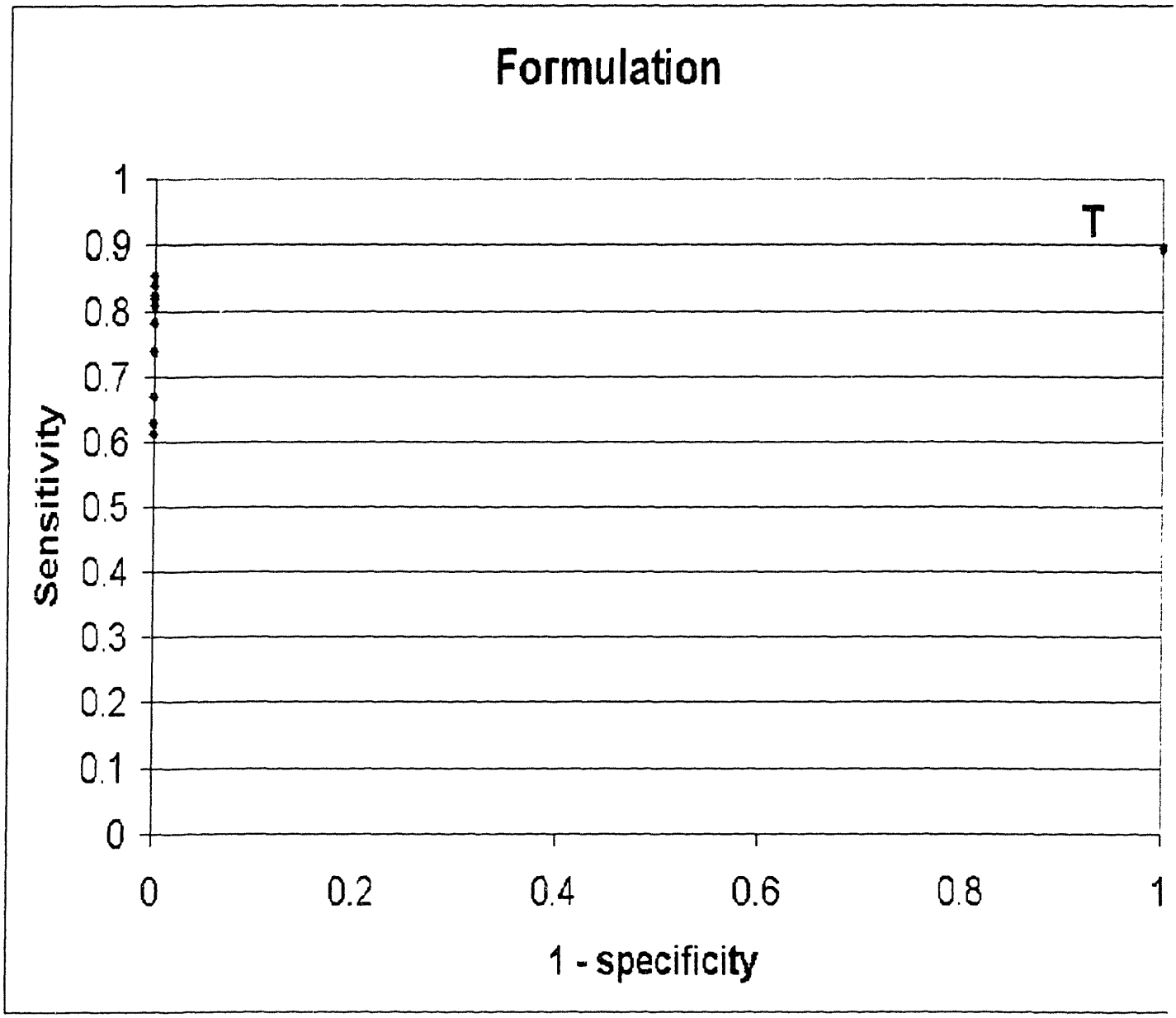

FIGURE 7

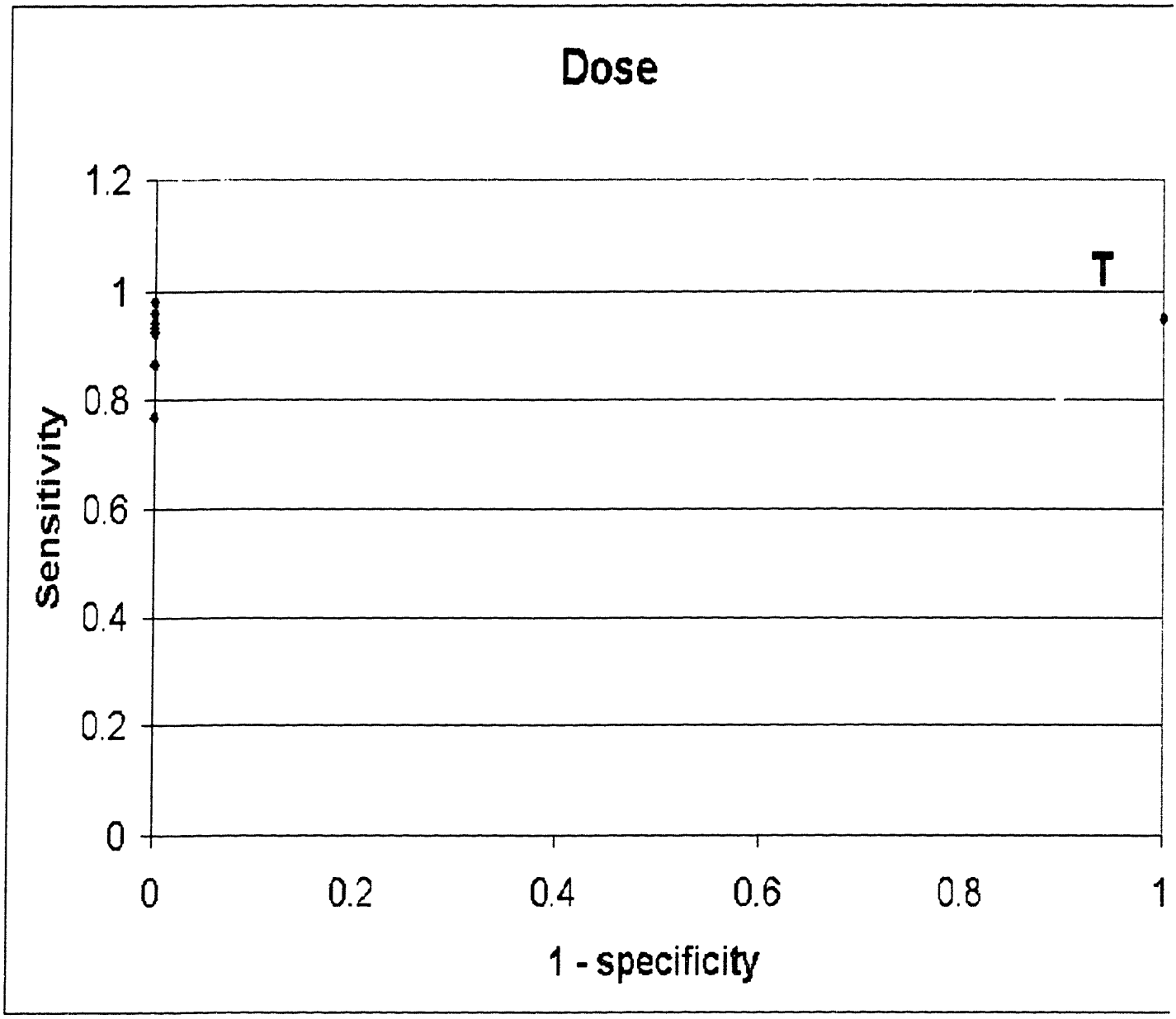
Dose
1.2
1
0.8
0.6
0.4
0.2
0
Sensitivity
T
0
0.2
0.4
0.6
0.8
1
1 - specificity

FIGURE 8

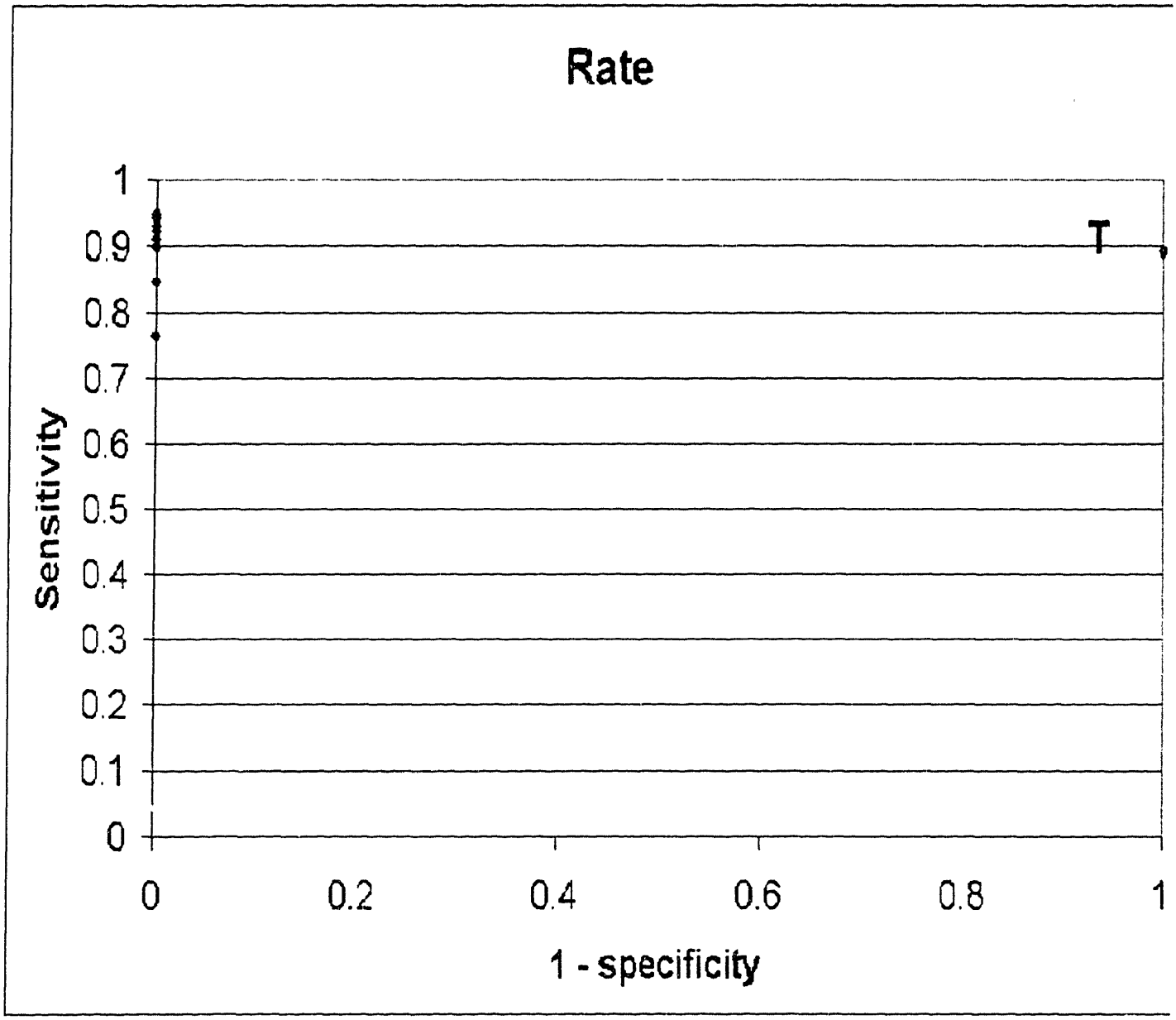
Rate
Sensitivity
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1 - specificity
T

FIGURE 9

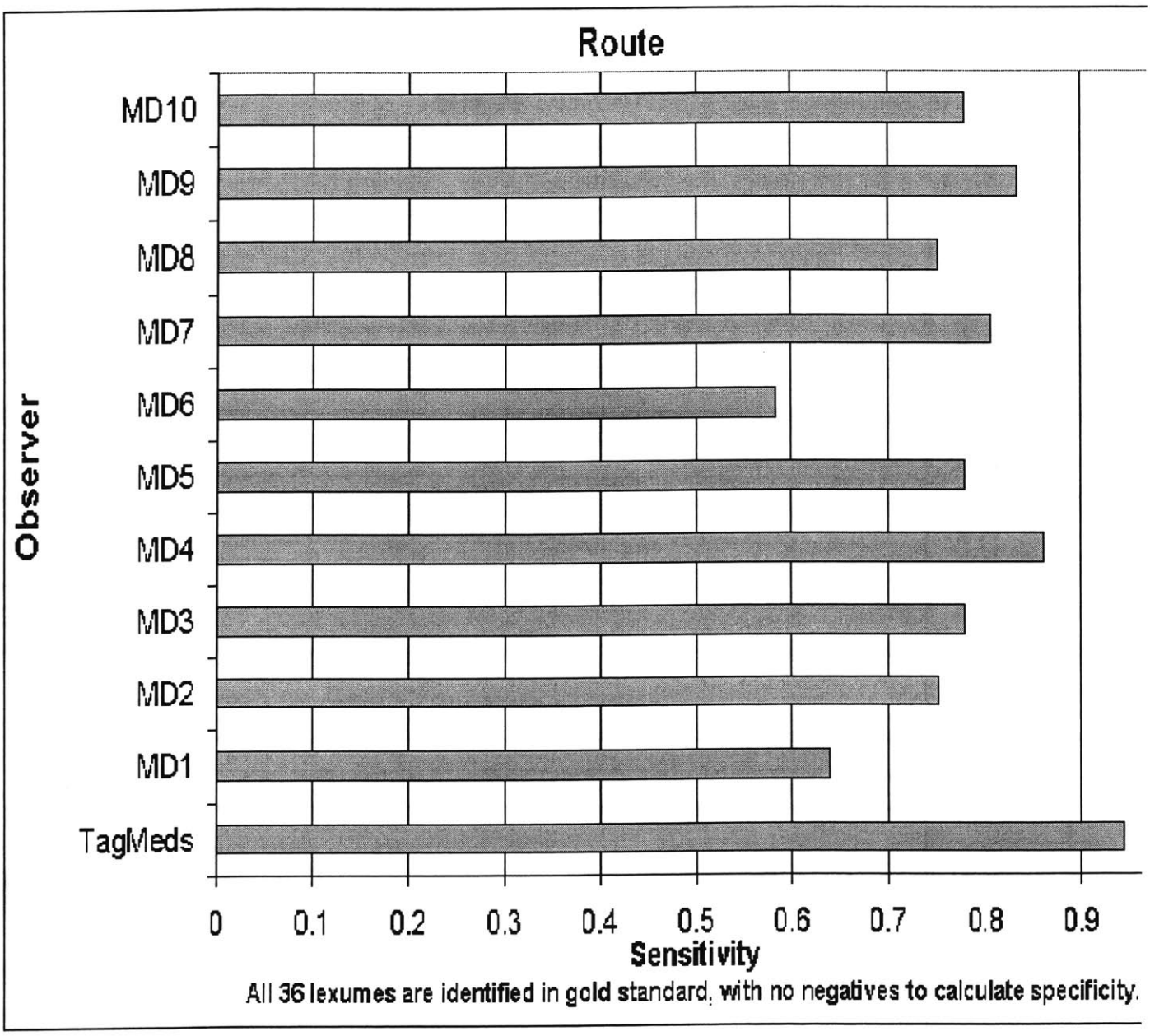
Route
MD10
MD9
MD8
MD7
MD6
MD5
MD4
MD3
MD2
MD1
TagMeds
Observer
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Sensitivity
All 36 lexumes are identified in gold standard, with no negatives to calculate specificity.

FIGURE 10

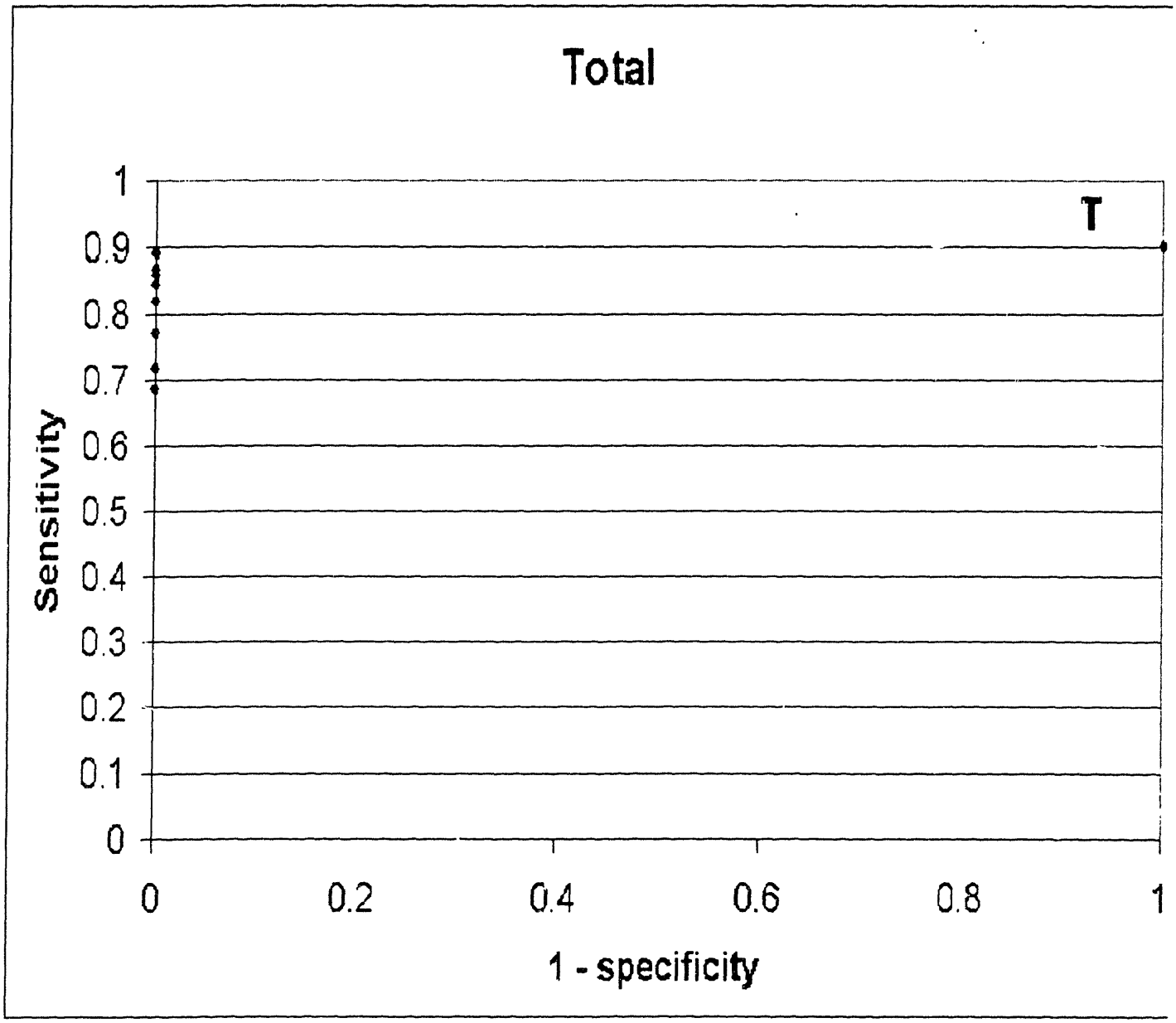
Total
T
Sensitivity
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1 - specificity

TABLE 1

Correlation Matrix: TagMeds (T) and 10 Physicians (N=425)

	T	*MD 1*	*MD 2*	*MD3*	*MD4*	*MD5*	*MD6*	*MD7*	*MD8*	*MD9*	*MD0*
T	1.00										
MD1	0.12	1.00									
MD2	0.28	0.52	1.00								
MD3	0.11	0.51	0.58	1.00							
MD4	0.20	0.39	0.56	0.44	1.00						
MD5	0.29	0.41	0.47	0.38	0.61	1.00					
MD6	0.23	0.43	0.50	0.45	0.39	0.37	1.00				
MD7	0.04	0.35	0.40	0.39	0.40	0.37	0.28	1.00			
MD8	0.22	0.48	0.51	0.44	0.63	0.62	0.44	0.41	1.00		
MD9	0.23	0.39	0.50	0.45	0.61	0.53	0.39	0.32	0.55	1.00	
MD0	0.27	0.40	0.50	0.39	0.55	0.61	0.44	0.26	0.61	0.56	1.00

APPENDIX A - XML PARSING OF NATURAL LANGUAGE TEXT

XML'S main strengths are its ability to store labeled information and the ease with which new labels can be created to represent different kinds of information. (38) A set of labels is often called a *vocabulary*. When large groups of people can accept a common vocabulary or even declare it a standard, applications of different kinds on different computers and networks can share an understanding of the contents of XML documents using that vocabulary. Information that might previously have been stored in a single vendor's proprietary format is now exposed, using a cleanly structured format that labels its contents in plain language. Although understanding and agreeing on the labels and structures is still very much a challenge, XML provides a solid foundation that lets such work get started.

XML offers structured, labeled, and easily exchanged information. It may not be the most efficient format for exchanging or storing information (it is, after all, a text-based format), but it has substantial advantages. XML provides flexibility, ease of processing, and easily documented formats. XML makes achievable the creation of standardized formats that use a similar grammar to express different kinds of information so that applications of all kinds can read them. It builds a foundation on which programs on all kinds of different platforms written in all kinds of different languages can build common understanding and share information.

Abstractions are critical in standards designed for interchange. Using Windows-specific or Java-specific vocabularies on a project that will involve interchange between users of

both platforms is fraught with problems. Fortunately, these platforms have a lot in common, even if they call things by different names and structure them somewhat differently. XML does provide a significant advantage for transforming information among formats.

An XML document can be treated as a set of labeled data, and the user can extract information from that document based on the information's label and its position in the document tree. By choosing a particular parsing model and sticking to it, applications can get a consistent presentation of information, absorb it into their internal structures, and place it back in XML documents if necessary. Although XML documents are designed to hold complex, often arbitrary, document structures, applications that want to use them to store data of any structure can use them for that purpose.

XML is a tool well suited to middleware applications, which are tools that provide a variety of services supporting network interactions, and are commonly used to convert or filter information from one format to another. The ease of processing, filtering, and transforming XML makes it a very useful for applications of this type. Middleware tools can use XML, often invisibly after the initial setup, to connect different systems. They can work at various levels of abstraction, hiding complexity behind a simplified interface.

A database application that needs to request a table from a different database can ask a middleware component to retrieve that table for it. Using XML, the middleware component contacts the middleware component for the other database and makes the

request. The middleware that receives the request changes it into a form appropriate to the database holding the table and sends it as XML to the original middleware making the request. That piece of middleware interprets the XML and passes it to the original database application in a form that it can understand, such as an SQL query.

Using this approach, the database developers on both ends only need to know how to communicate with the middleware. It doesn't matter what kind of database lurks behind the middleware on the other end of the network – the middleware will take care of all the mismatches as it converts the request and responses to and from XML. Similar approaches can be used with a wide variety of applications between clients and servers as well as between databases.

Limitations of XML

The above is a discussion of the advantageous use of XML in describing data and modeling information. There are problems and limitations that must now be discussed.

An external Document Type Declaration (DTD) isn't a document that can be validated by XML. This is a nuisance when attempting to store and process schemas with the rest of the XML information and when referencing the schemas using the same tools as the XML documents.

XML DTDs do a fine job describing hierarchical structured text. They don't understand integers, floating-point numbers, currency, dates, or the other kinds of information most

computing systems handle. Comments are not an especially powerful tool for documenting structures. Documents are valid only if every single element and attribute type used is declared somewhere in the DTD.

In 1999, Tim Berners-Lee stated:

> The threat is that when a company introduces a new document type, no one else will understand it. XML makes it easy for everyone to have his or her own markup languages. We might therefore see an end to the idyllic situation that has prevailed thus far on the Web – the predominance of HTML, which has helped all of us share documents easily. Can it be that, a decade into the Web's existence, XML will give us a freedom that forcibly leads us back toward myriad incompatible languages? This is indeed a serious possibility, but one that has been anticipated.[1]

This fear of anarchy has haunted XML throughout its existence. XML's provisions open the doors to creating new vocabularies on the fly without thought to overall structure. Documents can be created in any vocabulary. The only foundation XML demands is adherence to a basic set of rules regarding type and placement of markup that ensure a neatly named hierarchical structure. Consistency in structures has to be created and enforced by a separate set of processes. Berners-Lee proposes Namespaces and Cascading Style Sheets for managing this anarchy.

With cognizance of these limitations and pitfalls, TagMeds makes use of the solid set of structures that XML provides. In part, this is done in recognition of XML as an emerging standard, one that is already in use by the HL7 CDA. This decision is also made because the labels and structures of XML provide recognizable information that computers can use to convert information from free-text material into internal structures such as databases for additional processing.

HL7 Version 3 Clinical Document Architecture

Data representation and storage in the PING medical record must make use of a standardized data model and set of vocabularies to express these data in a set of common terms. This will be based upon the Health Level (HL) 7 Version 3 XML[li], which is extended to incorporate the specific documents and sections of documents found in a survey of medical documents from a variety of sources that we will examine.

HL7 is an umbrella organization whose mission is to provide standards for the exchange, management and integration of data that support clinical patient care and the management, delivery and evaluation of healthcare services. Specifically, to create flexible, cost effective approaches, standards, guidelines, methodologies, and related services for interoperability between healthcare information systems.
These efforts enable effective, efficient communication between the constituents of the healthcare community as represented by its membership, which consists of an international community of healthcare organizations, vendors, developers of healthcare information systems, consultants and systems integrators, and related public and private healthcare services agencies.

A proposal for a Patient Record Architecture (PRA) based upon the HL7 Version 3 has been developed by the Kona Editorial Group of the HL7 SGML/XML Special Interest Group. The HL7 Version 3 PRA was voted on by the organization in August 2000, and has since been renamed the Clinical Document Architecture (CDA). If HL7 Version 3 is a specialization consistent with XML and the CDA is a specialization of HL7 Version 3, then the local DTD, which we will develop for PING, will be a further increase in the level of granularity of the CDA. A CDA document is a defined and persistent information object that can exist outside of a messaging context and/or can be a payload within an HL7 message.

The CDA header is derived from the HL7 Reference Information Model (RIM)[7] using the HL7 Version 3 Message Development Framework. The CDA header does not preclude the use of a PING header. Tags in the XML header must describe the elements of the PING header. By use of this method, the PING header elements can be identified by the XML parser and presented to the PING parser as the appropriate elements.

The CDA header contains information about the entry, including the author (healthcare provider), the owner (patient), the people permitted to do to the entry, their roles, the date and time of creation, the type and format of the entry, unique identification,

[7] HL7, in its mission is to provide standards for the exchange and integration of data that support clinical patient care, has developed a comprehensive healthcare Reference Information Model (RIM.) HL7's next generation of standards is based on the RIM, which is a remarkably flexible and general model of clinical information.

authentication details, and classification information. A CDA document is invalid and incomplete if the header does not contain the required components.

The CDA header represents the state of document authorization as pre-authenticated, authenticated, and legally authenticated. A pre-authenticated document has been properly transcribed or created but has not been authenticated. A document is authenticated when it has been signed manually or electronically, attesting to its accuracy. The document is legally authenticated and is a complete document when it has been signed. The individual who is legally responsible for the document may sign the document manually or electronically.

The area of application for the HL7 Version 3 CDA is the *clinical document*, defined as a legally authenticated (attested or signed) and persistent entry into a patient record. The HL7 Document is the basic unit of this document-oriented patient record. Reports such as bills, insurance claims, and epidemiological reports are derived views based upon documents.

HL7 Version 3 XML is a method for modeling medical documents and data, and can be extended to facilitate viewing, database entry, database queries, and messaging. Data is structured hierarchically of arbitrary granularity. XML should capture the meaning of the data for rapid, easy parsing. Structured data yields semantic meaning.

The HL7 architecture is a document representation standard designed to support the delivery and documentation of patient care. A document is a defined and persistent information object that can include text, images, and sounds. The document specifications form an architecture that defines the semantics and structural constraints necessary for the exchange of clinical documents. The semantics derive from the HL7 Reference Information Model (RIM). The architecture is vendor-neutral and platform-independent and is specified in XML. Providers express their own clinical and business rules in their local DTDs.

The architecture is structured around three levels that provide increasing granularity of markup:

- Documents at all levels are readable by humans.
- Documents are viewable using widely available and commonly deployed XML-aware browsers and print drivers along with a generic CDA style sheet written in a standard style sheet language.

Each document consists of a header and a body. The header provides metadata that identifies and classifies the document and provides authentication details and information about the encounter, the patient, and the provider. The header utilizes RIM semantics (classes and associations) to define semantics but allows some choice in the expression of the XML element names.

Committees within HL7 are developing the next generation of messages, designated as Version 3. The Version 3 data types utilized by CDA are currently undergoing balloting under the sponsorship of a technical committee. However, Version 2.3 continues to be widely used.

The header uses RIM semantics and has been derived using the HL7 Message Development Framework (MDF) with minor adaptations. Decisions regarding content and structure of the CDA have been complicated because both the RIM and the MDF are still in development. While the HL7 RIM and MDF were initially developed to meet the needs of HL7 Version 3 messaging, the CDA represents their first real world test. Issues stemming from the reliance on the RIM reflect the different timetables of the two projects and the "ownership" of specific RIM classes rather than substantially different requirements.

The scenario of sending CDA documents within HL7 messages imposes unique and sometimes overlapping requirements that have yet to be worked out among the HL7 technical committees and special interest groups. The needs of CDA are currently a driving force for development of domain tables for coded values in the RIM. The HL7 XML Technical Committee anticipates that some features of the CDA will change in the near future as additional standards, including XML Schemas, become available and provide greater functionality.

The interoperability of HL7 XML messages and documents was tested in a prototype exchange network designed, built, and demonstrated on the floor of the Health Information Management and Systems Society (HIMSS) trade show in February 1999 and again in April 2000. HIMSS 2000 demonstration featured healthcare applications and generic XML tools in a scenario that used the CDA, HL7 Version 3 XML messages, and the SNOMED controlled vocabulary. In this scenario, patients were registered on one system, lab orders and encounter records were created on separate applications, and transcribed documents conforming to the CDA generated on another application. All generated records, CDA and Version 3, were collected in an XML database with a simple user interface. Queries against the data were facilitated by the diagnostic and procedure codes included in the lab orders and PRA documents.

A Document Type Definition specifies the elements contained in a class of document and the relationships between these elements. It contains the various elements and attributes that describe a document structure, displayed in a schema. It contains the grammar for that class of document. The Document Type Declaration contains the Document Type Definitions that specify the grammar.

One very important feature of HL7 Version 3 is the existence of coded elements. Coded elements are XML tags that refer to specified terms defined by the SNOMED or LOINC medical vocabularies and that eliminate any ambiguity as to the meaning of the term in question. For the purposes of this paper, the coded elements refer to UMLS-defined terms.

APPENDIX B - THE PING XML DOCUMENT TYPE DECLARATION:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Clinical_document (clinical_document_header, Clinical_document_body)>
<!—The clinical_document_header must be a mandatory element in actual clinical use. For the purposes of this experiment, no headers accompany the clinical_document_body elements, and in order to allow such documents to pass through XML parsing the clinical_document_header has been characterized as an optional element—>
<!ELEMENT clinical_document_header (availability_status_cd, change_reason_cd, completion_status_cd, confidentiality_status_cd, content_presentation_cd, document_creation_dttm, file_dm, last_edit_dttm, reporting_priority_cd, results_report_dttm, storage_status_cd, transcription_dttm, document_change_cd, version_nbr, version_dttm)>
<!ELEMENT document_creation_dttm (#PCDATA)>
<!ELEMENT content_presentation_cd (#PCDATA)>
<!ELEMENT file_dm (#PCDATA)>
<!ELEMENT availability_status_cd (#PCDATA)>
<!ELEMENT change_reason_cd (#PCDATA)>
<!ELEMENT completion_status_cd (#PCDATA)>
<!ELEMENT confidentiality_status_cd (#PCDATA)>
<!ELEMENT last_edit_dttm (#PCDATA)>
<!ELEMENT reporting_priority_cd (#PCDATA)>
<!ELEMENT results_report_dttm (#PCDATA)>
<!ELEMENT storage_status_cd (#PCDATA)>
<!ELEMENT transcription_dttm (#PCDATA)>
<!ELEMENT document_change_cd (#PCDATA)>
<!ELEMENT version_nbr (#PCDATA)>
<!ELEMENT version_dttm (#PCDATA)>
<!—The Clinical_document element and the header branch are taken from the HL7 Reference Information Model. The RIM specifies that there should be a body section to a clinical document, but gives no specification beyond that. All of the branches of the body branch are new in this project.—>
<!ELEMENT Clinical_document_body (consultation_letter*)>
<!ELEMENT consultation_letter (#PCDATA|medication)*>
<!ELEMENT medication (form_cd | strength_qty | dose_qty | dose_units_cd | number_doses_cd | route_cd | rate_qty | duration_treatment_qty | duration_treatment_units | dose_check_qty)*>
<!ELEMENT form_cd (#PCDATA)>
<!ELEMENT strength_qty (#PCDATA)>
<!ELEMENT dose_qty (#PCDATA)>
<!ELEMENT dose_units_cd (#PCDATA)>
<!ELEMENT rate_qty (#PCDATA)>
<!ELEMENT number_doses_cd (#PCDATA)>
<!ELEMENT route_cd (#PCDATA)>
<!ELEMENT duration_treatment_qty (#PCDATA)>
<!ELEMENT duration_treatment_units (#PCDATA)>
```

<!ELEMENT dose_check_qty (#PCDATA)>
<!ATTLIST form_cd UI CDATA #REQUIRED>

APPENDIX C - BACKUS NAUR FORM:

Clinical_document ::= Clinical_document_header
| Clinical_document_body
Clinical_document_header ::= availability_status_cd
| change_reason_cd
| completion_status_cd
| confidentiality_status_cd
| content_presentation_cd
| document_creation_dttm
| file_dm
| last_edit_dttm
| reporting_priority_cd
| results_report_dttm
| storage_status_cd
| transcription_dttm
| document_change_cd
| version_nbr
| version_dttm
Clinical_document_body ::= consultation_letter
consultation_letter ::= medication
medication ::= form_cd
| strength_qty
| dose_qty
| dose_units_cd
| number_doses_cd
| route_cd
| rate_qty
| duration_treatment_qty
| duration_treatment_units
| dose_check_qty

APPENDIX D - A SAMPLE OF THE XML OUTPUT OF TAGMEDS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Clinical_document SYSTEM
"C:\Tags\Clinical_document.dtd">
<Clinical_document>
<Clinical_document_body>
     <consultation_letter> 123 Santa Clause Lane
CH# XX XX XX
Death Valley, CA  90210                                    DOB:
01/01/81

Dear Dr. Neptune:

We had the pleasure of seeing Lucy in the Endocrine Program on
2/2/95 for .

  History: Tinmarie is a 14 year old girl referred to us for
evaluation of exopthalmus and elevated thyroid function tests.
Although the patient and her father have not noted the changes
in her eyes, they have noted a decreased requirement for sleep.
In the summer she may only sleep 2-3 hours.   Her father also
notes large appetite with frequent snacking between meals as
well as a significant thirst.   She drinks up to greater than 2
liters of diet soda per day as well as water and ice tea.    The
patient also notes a 40 pound weight loss over a 5-6 month
period during which time she was dieting actively and was
exercising frequently.    The patient notes that she tends to be
heat intolerant.   At night time she sleeps in pajamas without
any covers or blankets and prefers her room temperature to be 60
degrees.   She does not palpitations after exercise and
occasionally becomes quite anxious when having them leading to
further palpitations she feels.   Her father notes that she has
become more short tempered and more emotional than she had been
previously.  She angers easily.   Her school performance has
deteriorated.   She use to be an A and B student, now she is
doing poorly and receiving F's.   Her teacher has noted that she
also confabulates.

  Current medications: <medication><form_cd
UI="C0593507">Advil</form_cd>
<rate_cd>prn</rate_cd></medication> and <medication><form_cd
UI="C0014806">Erythromycin</form_cd></medication> for strep
throat.

Past medical history:  The patient was a full term infant who
father notes as having been always heavy and tall for age.
```

Review of systems: Significant for frequent headaches without photophobia or nausea, relieved with rest. She denies constipation, diarrhea, nocturia or polyuria or visual changes.

Family history: Significant for maternal hypothyroidism requiring thyroid replacement during her pregnancy. There are no other endocrine or auto immune diseases noted.

Physical exam:
Height: 161 cm Weight: 79.8 kg BP: 132/66
Pulse: 120
General: Slightly obese, alert and cooperative.
Skin: Revealed warm, dry skin without a rash.
HEENT: Significant for slight exopthalmus. Pupils were equal, round and reactive to light. Extraocular movements were intact. There was no notable eye movement lag. Her tonsils were enlarged and slightly erythematous although there was no exudate.
Neck: Supple. Thyroid measured 4.5x3 cm on the right and 3.5x2.5 cm on the left. No nodules were appreciated. There was no adenopathy.
Chest: Revealed clear breath sounds.
Heart: Revealed tachycardia but no murmur.
Abdomen: Significant for quiet bowel sounds and a quite palpable and prominent aorta.
Genitourinary: Tanner IV genitalia.
Extremities: Pulses were bounding throughout.
Neurological: Brisk ankle reflexes. There was also a mild tremor.

Impression: Tinmarie is a 14 year old with hyperthyroidism likely due to Grave's disease. We discussed treatment options with her and have started her on <medication><form_cd UI="C0033511">PTU</form_cd></medication>. We have prescribed 100 mg three times a day and have discussed its associated side effects, especially the lowering of the granulocyte count. She has been instructed to return to your clinic if she develops any fever or mouth sores. We have also started <medication><form_cd UI="C0004147">Atenolol</form_cd> <dose_qty>25</dose_qty> <dose_units_cd>mg</dose_units_cd> <rate_cd>q day</rate_cd></medication> to be increased to bid after one to two weeks for symptomatic improvement of her palpitations.

Laboratory studies pending at this time include: ALKP 211 U/L AST 17 U/L ALT 16 U/L BILI TO 0.4 MG/DL BILI DI 0.1 MG/DL

Disposition: We stressed that it is important for her to avoid gym and extra exercise so as to decrease her cardiac output requirements. For similar reasons, we have suggested taht she avoid caffeine and over the counter cold preparations. We have suggested taht she seek an ophthamology referral for evaluation of her proptosis. We discussed the fact that we would like have to treat her for up to a year with the <medication><form_cd UI="C0033511">PTU</form_cd></medication> and there is a chance that she could relapse later and require further treatment. We also discussed the possibility of treatment with radioactive iodine.

We look forward to see Lucy in one month at which time we will assess her symptoms and determine if any modification of her drug regimen is necessary.

We also reassured the patient and her father that her poor school performance and emotional and behavioral symptoms would likely improve but this would take some time. We also wrote a note to notify the school of the need for her to be excused from gym in that her medical condition required treatment.

Today we have sent her for additional laboratory studies including baseline liver function tests, an anti-peroxidase antibody, TSI and a TBII.
Thank you for letting us share in the care of Lucy.

Sincerely,

Nilto Carter, MD
Associate in Endocrinology

Addendum: Lauren Smith, M.D.
Resident in Pediatrics
Transcribed by Fahey"
"22981"</consultation_letter></Clinical_document_body></Clinical_document>

APPENDIX E - PERL PROCEDURES

```
# Version Final – tag.pl
#Andrew Nakrin
#ANakrin@mit.edu
#22 Apr. 2001
#  In view of the fact that a license to use the 2001 UMLS Knowledge Source Server for research
#purposes was obtained from the NIH NLM, where rights accrue to the author, unencumbered by the
#above, they are protected by a CopyLeft agreement, the GNU General Public License protecting the
#software, and a GNU Free Documentation License protecting the text. The CopyLeft agreement protects
#all software developed for this project, keeping it available for you to use freely.
#The standard CopyLeft agreement is viewable at
#<http://www.GNU.org/CopyLeft.html>.

use POSIX;

# NOTEs:
# the -| issue in .dict files is unclear for now.
# - was used to make DOS's sort.exe sort the dictionary properly
# (?:) is used to skip assignments of the $1, $2, ... variables, thus
# (First)(?:Middle)(Last) when matched with FirstMiddleLast will make
# $1 = "First" and $2 = "Last"

$argv = @ARGV;
($argv >= 2) or die "Tag requires 2 arguments - an input file and an output file.\n";

# DEBUG OPTIONS
#??? to speed up, remove tag_untagged call and simplifly medication pattern
#$run_tagged is turned off for now: the routine was tagging too much junk.
$run_tagged = 0,
$debug = 0;
if($argv > 2) {$debug = 1,}


#  Tagging the input file "<arg 1 - the input file>"
print "Tagging the input file \"$ARGV[0]\" and writing it to the output file \"$ARGV[1]\"\n",
print "If that is correct, press Enter, if it is not, press Ctrl+C\n";
$dummy = <STDIN>,

#initialize all variables
$lines_processed=0,

$fn_cachedict = "cache13apr.dict",
$fn_umlsdict = "umls.dict";
$fn_temp = "uis_not_known_yet";
$fn_temp2 = "temp_file_2";
$ui_not_found= "NOT-FOUND",

#The hash table that holds the UI of each term
my %term2ui;

#words that match the pattern and need to be looked up
my %needs_lookup,
my %cache2ui;                    # keeps the UI's of single word medicines
my %cache3ui;                    # keeps the UI's of multi word medicines
```

```
my %cache3words;                        # keeps all but the first word of a MW medicine
# ??? MultiWord medicines have to have distinct first words

print "Working\n";

print "--- Loading Cache Dictionary terms\n";
&load_cachedict;

print "--- PASS 1 - Collecting possible medications\n";

open(FIN, "<".$ARGV[0]) or die "Cannot open $ARGV[0] for input\n";
open(FOUT, ">".$fn_temp) or die "Cannot open $fn_temp for output\n";
$relevance = 1;
while($cline=<FIN>){
        @cletter=();
        @relevant=(1);
        $cpart="";

        #!!! this must be done first - replace the < and > with &lt; and &gt; to make XML happy. there is
another such code in the while statement below
        $cline =~ s{\&}{\&amp\;}g;
        $cline =~ s{\<}{\&lt\;}g;
        $cline =~ s{\>}{\&gt\;}g;

        while(not $cline =~ m{~}){
                $cline =~ s{\t}{       }g;
                $clinelc = $cline;
                $clinelc =~ tr{A-Z}{a-z};
                $newpart=0;
                $relevance=1;
                # ??? TRADEOFF between newline detection of letter parts and inline
                if ( $clinelc =~ m{current\s+medications}i ){$newpart=1; $relevance = 2; }
                if ( $clinelc =~ m{is\s+currently\s+on}i ){$newpart=1; $relevance = 2; }
                if ( $clinelc =~ m{^\s*disposition}i ){$newpart=1; $relevance = 1;}
                if ( $clinelc =~ m{^\s*impression}i ){$newpart=1; $relevance = 1; }
                if ( $clinelc =~ m{^\s*history}i ){$newpart=1; $relevance = 1;}
                if ( $clinelc =~ m{in\s+summary}i ){$newpart=1, $relevance = 1;}
                if ( $clinelc =~ m{test\s+results}i ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{\s*physical\s+exam}i ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{^\s*sincerely} ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{^\s*past medical history } ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{^\s*review of systems} ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{^\s*family history} ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{^\s*urinalysis} ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{laboratory\s*studies}i ){$newpart=1; $relevance = 0;}
                if ( $clinelc =~ m{^\s*$} ) {$newpart=1; $relevance=1;}
                if($newpart) {
                        $cpart =~ tr{\n}{\t};
                        push(@cletter, $cpart);
                        push(@relevant, $relevance);
                        #print "\n-=-$relevant[-2]-=-$cpart";
                        $cpart = $cline;
                } else {
                        $cpart = $cpart.$cline;
                }
```

```
                $cline = <FIN>;

                # fixes < and > for XML
                $cline =~ s{\&}{\&amp\;}g;
                $cline =~ s{\<}{\&lt\;}g;
                $cline =~ s{\>}{\&gt\;}g;

        }
        # add the line with the ~ to the current part
        $cpart = $cpart.$cline;
        $cpart =~ tr{\n}{\t};
        push(@cletter, $cpart);

        $lines_processed++;
        if( $lines_processed>1) {
                print "\.";
                $lines_processed=0;
        }

        #print "-end of letter--------\n";

        $parts = @cletter;
        for($part=0; $part<$parts; $part++){
                # Process each part of the letter
                $_ = $cletter[$part];

                #print "NEW PART [".$relevant[$part]."] \n";
                #print "$_\n";
                #print "<EOP>";

                if($relevant[$part]!=0) {
                        if($relevant[$part]==2){
                                s{(none)}{<form_cd UI=\"C0549184\">$1<Vform_cd>}gi;
#hi on April 4.
                                s{(phosphorus)}{<form_cd UI=\"C0031705\">$1<Vform_cd>}gi;
                                s{(potassium)}{<form_cd UI=\"C0032821\">$1<Vform_cd>}gi;
#bye for now.
                        }


                        #print "\n\n>>$relevant[$part]> $_";
                        # SPECIAL PATTERN (KNOWN MEDICINES)
                        $_ = " ".$_." ";
                        $cline = $_;
                        $cline =~ s{[^a-zA-Z0-9]}{ }g;
                        @word = split / /,$cline;
                        $words = @word;
                        # Each for step grabs the next word of $cline and
                        # checks if it is a medicine
                        # At the end of each while loop $cline's first word is
                        # removed, so the next $_ word can be processed
                        for($cword=0; $cword<$words; $cword++){
                                $term = $word[$cword];
                                #print "_${term}_";
                                $termlc = $term;
                                $termlc =~ tr{A-Z}{a-z};
```

```
                        if(defined $cache3ui{$termlc}){
                                my(@auis, @awords, @lword);
                                my($k, $medtotal, $lwords, $j, $tocheck, $ltmp);
                                @auis = @{$cache3ui{$termlc}};
                                @awords = @{$cache3words{$termlc}};
                                #print "DOH @auis\n@awords\n";
                                #$t = <STDIN>;
                                $medtotal = @awords;
                                #print "\n$termlc zz $medtotal zz @auis\n";
                                for($k=0; $k<$medtotal; $k++){
                                        $ui = $auis[$k];
                                        $tocheck = $awords[$k];
                                        #print "\nZIZI : $termlc $tocheck\n";
                                        @lword = split(" ",$tocheck);
                                        $lwords = @lword;
                                        if($cword+$lwords<$words){
                                                for($j=0; $j<$lwords; $j++){
                                                        $ltmp = $word[$cword+1+$j];
                                                        $ltmp =~ tr{A-Z}{a-z};
                                                        if($lword[$j] ne $ltmp) {
                                                                #print "ZOZO $lword[$j] <=>
$ltmp\n";
                                                                goto different;
                                                        }
                                                }
                                                $_ =  &tag_term($term."[\\s-]+".$tocheck, $_,
$ui);
                                                #print "\n -=-=- $term -=- $_\n";
                                        }
                                        different:
                                } #for k (forall medicines with same first word $termlc)
                        }
                } # for(cword) @cache3

                for($cword=0; $cword<$words; $cword++){
                        $term = $word[$cword];
                        #print "_${term}_";
                        $termlc = $term;
                        $termlc =~ tr{A-Z}{a-z};


                        if(defined $cache2ui{$termlc}){
                                $ui=$cache2ui{$termlc};
                                # ??? medicines must be hash table. push duplicates entries in
medicines
                                push(@medicines, $termlc);
                                $term2ui{$termlc} = $ui;
                                $_ = &tag_term($term, $_, $ui);
                                #print " =-=- $_\n";
                        } #if cache2ui
                } # for(cword) @cache2

                # PATTERNS
                while(  m{is +on +([a-zA-Z0-9]+)}i ) {
                        $term = $1;
```

```
                                        if(     ($term ne "his") and
                                                ($term ne "her") and
                                                ($term ne "a") and
                                                ($term ne "and") and
                                                ($term ne "no")
                                        ) {
                                                # must have unknown UI
                                                if( not defined $term2ui{$term} ){
                                                        $needs_lookup{$term} = 123;
                                                }
#                                               s{is +on +([a-zA-Z0-9]+)}{QWERTY $1};
                                                s{is +on +([a-zA-Z0-9]+)}{QWERTY <form_cd
UI=>$1<Vform_cd>};
                                        } else {
                                                s{is +on +([a-zA-Z0-9]+)}{QWERTY $1};
                                        }
                                }
                                s{QWERTY}{is on}g;

# PAT: current dose of thyroid hormone

                                while(  m{current\s+dose\s+of\s+(thyroid\s+hormone)}i ) {
                                         s{current\s+dose\s+of\s+(thyroid\s+hormone)}{QWERTY <form_cd
UI=C0040135>$1<Vform_cd>}i;
                                }
                                s{QWERTY}{current dose of }g;

                                while(  m{([a-zA-Z0-9]+) +([0-9]+) +units}i  ) {
                                        $term = $1;
                                        if(
                                                ($term ne "is") and
                                                ($term ne "and")
                                        ) {
                                                if( not defined $term2ui{$term} ){
                                                        $needs_lookup{$term} = 123;
                                                }
#                                               s{([a-zA-Z0-9]+) +([0-9]+) +units}{$1 $2 QWERTY};
                                                s{([a-zA-Z0-9]+) +([0-9]+) +units}{<form_cd UI=>$1<Vform_cd>
$2 QWERTY},
                                        } else {
                                                s{([a-zA-Z0-9]+) +([0-9]+) +units}{$1 $2 QWERTY};
                                        }
                                }
                                s{QWERTY}{units}g;

        # untag some specific patterns fix
        # ??? if any sentence contains one of the patterns below, ALL
        # of the subsequent medicines will be untagged. Ex:
        # Ex: has not been on A, but has been on B. B WILL NOT BE TAGGED
        while(
s{((?:has\s+not\s+been\s+on|hold\s+off\s+on|off\s+of|would\s+recommend|not\s+on|possibility\s+of\s+usi
ng)[^<.]+)(?:<form_cd UI=\"(?:C[0-9]+|$ui_not_found)\">)([^<]+)</form_cd>}
        {$1$2}gi ) {};

        s{(endogenous\s+)<form_cd UI=\"C0021641\">(insulin)</form_cd>}{$1$2}gi;
```

```
        s{<form_cd UI=\"C0021641\">(insulin)</form_cd>((?:-
|\s+)(?:dependent|depended|reaction))}{$1$2}gi;
        s{(one\s+may\s+elect\s+to\s+use\s+)<form_cd UI=\"C[0-9]+\">([a-zA-Z0-9\
]+)</form_cd>}{$1$2}gi;
        s{(off\s+the\s+|off\s+|recommend\s+|advise\s+|suggest\s+|may\s+be\s+changed\s+to\s+)<form_c
d UI=\"C[0-9]+\">([a-zA-Z0-9\ ]+)</form_cd>}{$1$2}gi;
        s{(discontinue(?:d)?\s+the\s+)<form_cd UI=\"C[0-9]+\">([a-zA-Z0-9\ ]+)</form_cd>}{$1$2}gi;
        s{<form_cd UI=\"C[0-9]+\">([a-zA-Z0-9\
]+)</form_cd>(\s+d\s*(?:\\|V)\s*c|\s+can\s+be|\s+deficiency|\s+metabolites)}{$1$2}gi;
        if(!m{not stop\s+<form_cd UI=\"C[0-9]+\">([a-zA-Z0-9\ ]+)</form_cd>}i) {
                s{(stop\s+)<form_cd UI=\"C[0-9]+\">([a-zA-Z0-9\ ]+)</form_cd>}{$1$2}gi;
        }

# If in current medications => tag the remaining untagged text
                        if($relevant[$part]==2 && $run_tagged){ # i.e. current medication part
                                print "\nBEFORE: $_\n";
                                $_ = tag_untagged($_);
s{<medication>([\s\.\-\:\;]*(current\s+medication(s)?\s*(include(s)?)?)*[\s\.\-\:\;]*)</medication>}{$1}gi;
                                print "\nAFTER: $_\n";
                        } # IF REL=2
                } # if relevant
                tr{\t}{\n};
                print FOUT "$_";
        } # for each letter part
} # while FIN

print "\n",
close(FIN),
close(FOUT);

print "--- Getting THE UI's from the database\n",
$terms_found=0;

foreach $term (keys %needs_lookup){
        $ui = &lookup_ui($term),
        if($ui ne "") {
                $term2ui{$term} = $ui;
                $terms_found ++,
                push(@medicines, $term);
        }
}

$terms_needed = keys(%needs_lookup);
print "--- Database lookup completed. $terms_found out of $terms_needed  were found.\n\n";
$msize = @medicines;
print "--- A total of $msize medicines found in the input file.\n";

print "--- PASS 2 - Filling in the UI\'s\n";
open(FIN, "<".$fn_temp) or die "Cannot open $fn_temp for input\n";
open(FOUT, ">".$fn_temp2) or die "Cannot open $fn_temp2 for output\n",
while(<FIN>){
#       foreach $term (@medicines){
#               s{$term}{<form_cd UI\=\"$term2ui{$term}\">$term<\/form_cd>}g;
#       }

        while(  m{<form_cd UI=>([^<]*)<\/form_cd>} ) {
```

```
                $term = $1;
                $termlc = $term;
                $termlc =~ tr{A-Z}{a-z};
                if(not defined($term2ui{$termlc})) {
                        s{<form_cd UI=>([^<]*)<\/form_cd>}{$term};
                } else {
                        s{<form_cd UI=>}{<QWERTY=\"$term2ui{$term}\">};
                }
        }
        s{QWERTY}{form_cd UI}g;

        print FOUT "$_";
}
close(FIN);
close(FOUT);

print "--- SKIPPING Third pass - error correction -missed terms\n";

print "--- PASS 4 - undoing the excess, pull in other tags\n";
open(FIN, "<".$fn_temp2) or die "Cannot open $fn_temp for input\n";
open(FOUT, ">".$ARGV[1]) or die "Cannot open $ARGV[1] for output\n";
print FOUT "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<!DOCTYPE Clinical_document SYSTEM \"C:\\Tags\\Clinical_document.dtd\">
<Clinical_document>
<Clinical_document_body>
        <consultation_letter>";
while(<FIN>){
    if(m{<form_cd UI=\"(C[0-9]+|$ui_not_found)\">}) {
        # LEGEND:

        # ??? Empty patterns that must be skipped
        # (*) - because of the medicine(chemical 1, chemical 2)
          # STUDY #[^\s]*
        # only two weeks

        # !!! NOTE: in order to support the tagging of all words in the "current medications" part more stuff
needs to be coded in (<medication>|)
        # SPECIFIC tags are tagged out of order, by iteratively appending the next tag following or
preceding the medication tag
        ### add the medication arround the form_cd
        if($debug) {print "DOH: $_"; $t = <STDIN>;}
        s{(<form_cd UI=\"(?:C[0-9]+|$ui_not_found)\">(?:[^<]+)</form_cd>)}
          {<medication>$1</medication>}gi;
        do{
                if($debug) {print "DOH2: $_";}
                # NAfter medication and NBefore medicine
                $change = 0;

                ### 1A Units and Dosage
                $change += s
                  {<medication>(.*?)</medication>(\s*(?:dos(?:age|e))?\s*(?:to|of|only)?\s*)((?:[0-9]+[0-9\-
\\]*(?:\.)?\d*|\.\d+)|one\s|two\s|a\s|)(\s*)((?:mg(?:\.)?|mcg(?:\.)?|cc(?:\.)?|meq(?:\.)?|tablets|tablet|units|tsp|))
}

{<medication>$1$2<dose_qty>$3</dose_qty>$4<dose_units_cd>$5</dose_units_cd></medication>}gi;
                if($debug) {print "\nDOH21a: $_";}
```

```
        # FIX 2 empty
        if($debug) {print " 1a: $change";}
        $change -
=s{(\s*(?:dos(?:age|e))?\s*(?:to|of|only)?\s*)<dose_qty>(\s*)</dose_qty>(\s*)<dose_units_cd>(\s*)</dose_
units_cd></medication>}{</medication>$1$2$3$4}g;
                if($debug) {print "\nDOH21af: $_";$t=<STDIN>}

                ### 2A Route
        if($debug) {print " 1af: $change";}
                $change += s

{<medication>(.*?)</medication>(\s*(?:dos(?:age|e))?\s*(?:to|of|only)?\s*)(po(?:\s|\.)?|sq(?:\s|\.)?|nasal\s*s
pray|depot|i[\.\s]*m[\.]*(?:\s|\.)?|inhaler|injection(?:s)?|subcutaneously|intranasally)}
                 {<medication>$1$2<route_cd>$3</route_cd></medication>}gi;
                if($debug) {print "\nDOH22a: $_";}

                ### 3A  Schedule/Rate
        if($debug) {print " 2a: $change";}
                $change += s
                 {<medication>(.*?)</medication>(\s*(?:dos(?:age|e))?\s*(?:to|of|only)?\s*)(m\-w\-
f|qhs|q\.h\.s\.|p[\.\s]*r[\.\s]*n[\.]*|(?:t|q|b)\.i\.(?:d|w)\.|evenings|(?:t|q|b)id|((?:\d+\s+)?(times|x|days)\s+)?(per|
a|every|\\|V)\s*(day|week)|(?:alt)?(?:\.)?q(\.?)\s*\d*\s*(o(\.?)\s*)?(?:d(ay|)(\.?)|week(?:s)?|month)|(?:once)?\
s+daily|mornings|prn|on\s+a\s+prn\s+basis|for\s+sleeping)}
                 {<medication>$1$2<rate_cd>$3</rate_cd></medication>}gi;
                if($debug) {print "\nDOH23a: $_";}

                ##########################################################
                ### 1B Units and Dosage
        if($debug) {print " 3a: $change";}
                $change += s
                 {((?:[0-9]+[0-9\-
\\]*(?:\.)?\d*|\.\d+)|one\s|two\s|a\s|)(\s*)((?:mg(?:\.)?|mcg(?:\.)?|cc(?:\.)?|meq(?:\.)?|tablets|tablet|units|tsp|))
(\s*(?:of)?\s*)<medication>(.*?)</medication>}

{<medication><dose_qty>$1</dose_qty>$2<dose_units_cd>$3</dose_units_cd>$4$5</medication>}gi;
                if($debug) {print "\nDOH21b: $_";$t=<STDIN>}

        # FIX 2 empty
        if($debug) {print " 1b: $change";}
        $change -=
s{<medication><dose_qty>(\s*)</dose_qty>(\s*)<dose_units_cd>(\s*)</dose_units_cd>(\s*(?:of)?\s*)}{$1
$2$3$4<medication>}g;
                if($debug) {print "\nDOH21bf: $_";}

                ### 2B Route
        if($debug) {print " 1bf: $change";}
                $change += s

{(po(?:\s|\.)?|sq(? \s|\.)?|nasal\s*spray|depot|i[\.\s]*m[\.]*(?:\s|\.)?|inhaler|injection(?:s)?|subcutaneously|intr
anasally)(\s*(?:of)?\s*)<medication>(.*?)</medication>}
                 {<medication><route_cd>$1</route_cd>$2$3</medication>}gi;
                if($debug) {print "\nDOH22b: $_";}

                ### 3B  Schedule/Rate
        if($debug) {print " :2b $change";}
```

```
                $change += s
                 {(m\-w\-
f|qhs|q\.h\.s\.|p[\.\s]*r[\.\s]*n[\.]*|(?:t|q|b)\.i\.(?:d|w)\.|(?:t|q|b)id|(?:(?:\d+\s+)?(?:times|x|days)\s+)?(?:per|a|ev
ery|\\|V)\s*(?:day|week)|(?:alt)?(?:\.)?q(?:\.)?\s*\d*\s*(?:o(?:\.)?\s*)?(?:d(?:ay|)(?:\.)?|week(?:s)?|month)|(?:
once)?\s+daily|evenings|mornings|prn|on\s+a\s+prn\s+basis|for\s+sleeping)(\s*(?:of)?\s*)<medication>(.*
?)</medication>}
                  {<medication><rate_cd>$1</rate_cd>$2$3</medication>}gi;
                if($debug) {print "\nDOH23b: $_";}
        if($debug) {print " 3b: $change\n";}
        } while ($change>0);

        if($debug) {print "\nDOH3: $_";$t = <STDIN>;}

        s{<route_cd>(\s*)</route_cd>}{$1}g;
        s{<dose_units_cd>(\s*)</dose_units_cd>}{$1}g;
        s{<dose_qty>(\s*)</dose_qty>}{$1}g;
        s{<rate_qty>(\s*)</rate_qty>}{$1}g;
        s{<medication>([\s\.\-\:\;]*(current\s+medication(s)?\s*(include(s)?)?)?[\s\.\-
\:\;]*)</medication>}{$1}gi;        s{<medication>(\s+)}{$1<medication>}gi;
        s{(\s+)</medication>}{</medication>$1}gi;
        if($debug) {print "DOH4: $_";$t = <STDIN>;}

        } # if <form_cd>
        print FOUT "$_";
}
print FOUT "</consultation_letter></Clinical_document_body></Clinical_document>";
close(FIN);
close(FOUT);
#####


print "Done.\nPress Enter to Quit.";
$dummy = <STDIN>;




# ------------------------------------------------------------------------
# The End



# The subroutine lookup_umls looks up the UI of a text term
# The @_ array holds the variables passed to the subroutine
# NOTE: The search is case insensitive

#LOOKUP_UI V2.0
sub lookup_ui{

        local ($term) = @_;
#       $term =~ tr{A-Z}{a-z};

        local $ui = "";
        local $cui = "";
        local $text = "";
```

```
local $cline = "";

local ($low_bound, $high_bound, $mid, $lstart, $lend); # line start, line end
# $low_bound - $high_bound  range (offset in umls.dict) in which the term we search for could be
# $low_bound should always point to the first character of a line
#$high should always point to the LAST character of a line (BEFORE \n or EOF)

#print "Looking for $term\'s UI...";

open(FTERM, "<".$fn_umlsdict) or die "Cannot open the $fn_umlsdict for input\n";
$low_bound=0;
# make $high equal to the file size of umls.dict
seek FTERM, -1, SEEK_END;
$high_bound = tell FTERM;

$cstep = 0;
while(1){
        $cstep++;

        #temp extra work
        #seek FTERM, $low_bound, SEEK_SET;
        #$binl = <FTERM>;
        #seek FTERM, $high_bound+3, SEEK_SET;
        #$binh = <FTERM>;
        #print "\n---Binsearch step $cstep in range ($low_bound, $high_bound)\n";
        #print "--Low line: $binl";
        #print "--High line: $binh";
        #print "\n";


        $mid = int (($high_bound+$low_bound)/2);
        seek FTERM, $mid, SEEK_SET;
        $cc=getc(FTERM);

        # move back character by character until \n or FILE START is found
        while(($cc ne "\n") && ($mid>=1)){
                seek FTERM, -2, SEEK_CUR;
                $cc=getc(FTERM);
                $mid--;
        }
        if($mid != 0) {
                $mid++;
        }
        $lstart = $mid;
        $cline = <FTERM>;
        chomp($cline);
        $lend = $mid + length( $cline ) -1;
        # print "Cline is $cline from $lstart to $lend\n";

        # process the current line and compare it
        if(not $cline =~ m{^([^\|]*)\-\|(C\d{7}|$ui_not_found)}) {
                print "ERROR: Unknown line : $cline\n";
        }
        $text = $1;
        #$text = substr($1, 0, length($1)-1);
        $text =~ tr{A-Z}{a-z};
```

```
                $cui = $2;

                if( $text eq $term ) {
                        $ui = $cui;
                        goto end_binsearch;
                }

                if( $term lt $text ) {
                # under WIN/DOS use -3. For unix - use -2 only
                        $high_bound = $lstart-3;
                } else {
                        $low_bound = $lend+3;
                }

                if(($lstart == $low_bound && $lend==$high_bound) || $low_bound>=$high_bound){
                        goto end_binsearch;
                }
        }
        end_binsearch:

        if($ui eq "") {
                print "$term \t\t UI not found.\n";
        } else {
                # print "FOUND UI $ui.\n";
        }

        #The subroutine returns the value of the last statement
        $ui;
}

sub load_cachedict{
        open UMLS, "<".$fn_cachedict or die "Cannot open $fn_cachedict cache dictionary\n";
        local($count=0, $text, $cui, $mw=0, $sw=0, $firstw, $restw);
        while($cline=<UMLS>){
                # !!! WHY THAT WORKS - backtracks at the \-
                if(not $cline =~ m{^([^|]*)\-\|(C\d{7}|$ui_not_found)} ) {
                        print "ERROR: Unknown line : $cline\n";
                }
                $text = $1;
                #$text = substr($1,0,length($1));
                $text =~ tr{A-Z}{a-z};
                $cui = $2;
                if($text =~ m{ }) {
                        # multi word $` first word (before the first space), $' the remaining words (after
space)
                        $firstw = $`;
                        $restw = $';
                        if(defined($cache3ui{$firstw})) {
                                push(@{$cache3ui{$firstw}}, $cui);
                                push(@{$cache3words{$firstw}}, $restw);
                        } else {
                                $cache3ui{$firstw}=[($cui)];
                                $cache3words{$firstw}=[($restw)];
                                $mw++;
                        }
                } else {
```

```
                        $cache2ui{$text}=$cui;
                        $sw++;
                }
                $count++;
                if($count%100==0) {
                        print ".";
                }
        }
        close UMLS;
        print "-=-=-=-=- Loaded $sw single word and $mw multi word medicines\n";
        #print keys(%cache3ui);
        #print "=-=-=-=-=------====-=-=-";
}

sub tag_term{
        local ($to_tag, $cline, $ui) = @_;
        local ($s, $e, $part1, $part2, $part12);
        $s = index($cline, "<");
        if($s!=-1){
                $e = index($cline, ">");
                $e = index($cline, ">", $e+1);
                $part1 = substr($cline, 0, $s);
                $part2 = substr($cline, $e+1);
                # KEEP $1 in the substituted pattern, do not use $to_tag
                $part1  =~ s{($to_tag)}{<form_cd UI=\"$ui\">$1</form_cd>}gi;
                $part1 =~ s{(serum\s+)<form_cd UI=\"$ui\">($to_tag)</form_cd>}{$1$2}gi;
                $part1 =~ s{<form_cd
UI=\"$ui\">($to_tag)</form_cd>(\s+(?:level(?:s|)|binding|stores))}{$1$2}gi;
                $part12 = substr($cline, $s, $e-$s+1);
                $part2fixed = &tag_term($to_tag, $part2, $ui);
#               print "\n\n123456-$s $e $part12 $cline\n";
                $cline = $part1 . $part12 . $part2fixed;
        } else {
                $cline =~ s{($to_tag)}{<form_cd UI=\"$ui\">$1</form_cd>}gi;
                $cline =~ s{(serum\s+)<form_cd UI=\"$ui\">($to_tag)</form_cd>}{$1$2}gi;
                $cline =~ s{<form_cd UI=\"$ui\">($to_tag)</form_cd>(\s+level(?:s|))}{$1$2}gi;
        }
        $cline;
}

sub tag_untagged{
# tags all untagged regions of $cline with <medication> tags
        local ($cline) = @_;
        local ($ctag, $s, $e, $part1, $part2, $part12);
        $s = index($cline, "<");
        if($s!=-1){
                $e = index($cline, ">");
                $part12 = substr($cline, $s, $e-$s+1);
                $part12 =~ m{<([^\s>]+)};
                defined $1 or die "ERROR:  <ksjvdfkjsfg>";
                $ctag=$1;
                $e = index($cline, $ctag, $e+1);
                $e = index($cline, ">", $e+1);

                $part1 = substr($cline, 0, $s);
                $part2 = substr($cline, $e+1);
```

```
                $part1  = "<medication>" . $part1 . "</medication>";
                $part12 = substr($cline, $s, $e-$s+1);
                $part2fixed = &tag_untagged($part2);
#               print "\n\n123456-$s $e $part12 $cline\n";
                $cline = $part1 . $part12 . $part2fixed;
        } else {
                $cline = "<medication>" . $cline . "</medication>";
        }
        $cline;
}

exit();
#END_OF_CODE
#pieces of code below

#??? MAKE SURE ALL MEDS BELOW ARE IN THE CACHE DICT
                        #s{(Fer-In-Sol)}{<form_cd UI=\"C0720405\">$1<Vform_cd>}g;
                        #s{(potassium\s+phosphate)}{<form_cd UI=\"C0071778\">$1<Vform_cd>}g;
                        #s{(PTU)}{<form_cd UI=\"C0033511\">$1<Vform_cd>}gi;
                        #s{(calcium\s+citrate)}{<form_cd UI=\"C0108101\">$1<Vform_cd>}gi;
                        #s{(multi-vitamins)}{<form_cd UI=\"C0042890\">$1<Vform_cd>}gi;
                        #s{(humulin\s+insulin)}{<form_cd UI=\"C0020171\">$1<Vform_cd>}gi;
                        #s{(regular)}{<form_cd UI=\"C0205272\">$1<Vform_cd>}gi;
```

```
#_____
#Andrew S. Nakrin, MD,
#Diplomate of the American Board of Anesthesiologists
#
#Clinical Decision-Making Group
#MIT Laboratory for Computer Science
#NE43-418
#200 Technology Square
#Cambridge, MA 02139-1300
#(617) 253-3539
#ANakrin@mit.edu
#
#
#133 Sudbury Road
#Weston, MA 02493-1300
#(781) 647-0806
#AndyNakrin@aol.com
#_____
#
#This procedure is saved as filter_mrcon.pl.
#It is part of a project I call TagMeds.
#This project is in partial fulfillment of the MIT SM degree requirements.
#This research is being supervised by Peter Szolovits, Ph.D.
#At the MIT Laboratory for Computer Science.
#The NIH National Library of Medicine has provided me with generous
#training grants.
#A license to use the 2001 UMLS Knowledge Source Server for
#research purposes was obtained from the NIH NLM.
#Where rights accrue to the author, unencumbered by the above,
#they are protected by a standard CopyLeft agreement.
#The standard CopyLeft agreement is viewable at
#<http://www.GNU.org/CopyLeft.html>.

#The first problem dealt with was where and how to a lookup term
#for a medication, to check if it is, in fact, a term specifying
#a medication. And if it is, what UMLS unique concept it specifies.

#Dr. William J. Long was kind enough to provide me with a Perl procedure
#that queries the UMLS Knowledge Server via a Secure Shell Connection.
#The time required for word by word query and response transmission
#over the Internet makes this technique ill suited to this project. Instead, the #3 gigabyte MetamorphoSys software package with libraries was downloaded from #the UMLS Knowledge Server, and unzipped. The MetamorphoSys package was
#determined to be poorly suited for a large number of automated searches, #chiefly because of a click-button graphical user interface. It was determined #that the file MRCON,a part of the MetamorphoSys package contains all the human #language terms, a specification of what language the text term was taken from, #the UMLS Unique Identifier, and identifiers from other vocabularies, in a | #delimited file. This is a sample entry:
# C0000039|ENG|S|L0296452|PF|S0033295|Dipalmitoyl Phosphatidylcholine|0|

#Note that the first column contains the UMLS Unique Identifier, the second #column lists the language from which the text term originates, and the seventh #column lists the term. The purpose of the filter_mrcon.pl procedure is to #search through the MRCON file, a 100 megabyte file. When a term in the English #language is found, the text term is written on one line of the file umls.dict, #followed by the | delimiter, followed by the UMLS unique identifier. Umls.dict #will then be usable by the rest of the
```

```
TagMeds system to check if a term is a #UMLS term, and if so to find its UMLS Unique Identifer.
Umls.dict is a 25
#megabyte file, somewhat more manageable than the 3 gigabyte MetamorphoSys.
#Entries in umls.dict have the form:
# Dipalmitoyl Phosphatidylcholine|C0000039

#Perl procedures are modeled after or paraphrased from The Perl #Cookbook by Tom #Christiansen and
Nathan Torkington, copyright 1998 by O'Reilly #& associates, #Sebastepol, CA. Other ideas, and the Perl
compiler, are from #<http://www.Perl.com>

#The Open function takes two arguments, a filehandle to open and a string #containing the filename and
the mode. FIN reads from MRCON.
open(FIN, "<MRCON") or die "Cannot open MRCON for input\n";
#and  FOUT writes to umls.dict.
open(FOUT, ">umls.dict") or die "Cannot open umls.dict for output\n";

print "Working on 2M lines\n";
#Let the user know that we got in.

#line count displays to keep the user informed.
$lines_processed=0;
$total_lines=0;
$lines_written=0;

#For as long as we still have input
while(<FIN>){
#input of the form of a UMLS Unique Identifier, a C and 7 digits,
# |and the term is in English
        if( m{^C\d{7}\|ENG} ) {
                if( not m{^(C\d{7})\|ENG\|[^\|]*\|[^\|]*\|[^\|]*\|[^\|]*\|([^\|]*).*$} ) {
                        print "Could not match this line: $_\n";
                    }
#and if it fits the 7 field 6 delimiter pattern
#Then (grab the C plus seven digit UI) and grab the (term)
#and switch so that you've got the term, the | delimiter, and the UI.
        s{^(C\d{7})\|ENG\|[^\|]*\|[^\|]*\|[^\|]*\|[^\|]*\|([^\|]*).*$}{$2\|$1};
#increment the counter of umls dictionary lines
                $lines_written++;
#write term|UI into umls.dict
                print FOUT "$_";
        }

#increment the 10,000 line between print counter
        $lines_processed++;
#increment the lines examined in MRCON counter
        $total_lines++;
        if($lines_processed >= 10000){
                $lines_processed=0;
                print "$lines_written written of $total_lines\n";
#every 10,000 lines let the user know how many lines were examined
#and how many were put into the dictionary.
        }

}
```

```
#don't you hate when the Perl window disappears while you are in the toilet, and #you never find out how
the program progressed. This window won't close until the #user presses Enter.
print "Done.\nPress Enter to Quit.";
$dummy = <STDIN>;

#BetTer be sure we didn't run out of disk space. These are big files. Lets see #if they will close up
properly.
close(FIN) or die "MRCON didn't close";
close(FOUT)or die "umls.dict didn't close";
```

```
$argv = @ARGV;
($argv == 3) or die "Tag requires 2 arguments - an input file and an output file.\n";

print "The input file is \"$ARGV[0]\" and the output file is \"$ARGV[1]\"\n";
print "If that is correct, press Enter, if it is not, press Ctrl+C\n";
$dummy = <STDIN>;

#initialize all variables

#The hash table that holds the UI of each term

print "Example: out of 200 letters 1:5 odds would split the file into \n approx. 40 and 160 letter files\n";
print "Enter the odds (1:X) of a letter being chosen:\n";

$prob = <STDIN>;

open(FIN, "<".$ARGV[0]) or die "Cannot open \"$ARGV[0]\" for input\n";
open(FOUT, ">".$ARGV[1]) or die "Cannot open \"$ARGV[1]\" for output\n";
open(FREST, ">".$ARGV[2]) or die "Cannot open \"$ARGV[2]\" for output\n";
$lc=0;
$cletter=0;
if(rand($prob)<1) {$chosen=1;} else {$chosen=0;}
$total_chosen=$chosen;
while(<FIN>){
        $lc++;
        if($lc>=100){
      $lc=0; print ".";
    }
        if($chosen==1) {
                print FOUT $_;
        } else {
                print FREST $_;
        }
        if( m{\~} ) {
                $cletter++;
                if(rand($prob)<1) {$chosen=1;} else {$chosen=0;}
                $total_chosen+=$chosen;
        }
}

print "\nExamined $cletter letters in order to generate $total_chosen.\n";
close(FIN);
close(FOUT);
close(FREST);
print "Done.\nPress Enter to Quit.";
$dummy = <STDIN>;
# ------------------------------------------------------------------------
# The End
```

```
#_____

#Andrew S. Nakrin, MD,
#Diplomate of the American Board of Anesthesiologists
#
#Clinical Decision-Making Group
#MIT Laboratory for Computer Science
#NE43-418
#200 Technology Square
#Cambridge, MA 02139-1300
#(617) 253-3539
#ANakrin@mit.edu
#
#
#133 Sudbury Road
#Weston, MA 02493-1300
#(781) 647-0806
#AndyNakrin@aol.com
#_____

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#This procedure is saved as <random100.pl>.
#To run this program in a Windows environment, with letters.txt as the input
#file and output.txt as the output file, click Start, click Run, type "Command".
#A DOS window opens. Type "cd c:\TagMeds" and hit Enter. Type "perl #random100.pl letters.txt
output.txt" Note that tag.pl, umls.dict, and #letters.txt must all be in the same folder.
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


#It is part of a project I call TagMeds.
#This project is in partial fulfillment of the MIT SM degree requirements.
#This research is being supervised by Peter Szolovits, Ph.D.
#At the MIT Laboratory for Computer Science.
#The NIH National Library of Medicine has provided me with generous
#training grants.
#A license to use the 2001 UMLS Knowledge Source Server for
#research purposes was obtained from the NIH NLM.
#Where rights accrue to the author, unencumbered by the above,
#they are protected by a standard CopyLeft agreement.
#The standard CopyLeft agreement is viewable at
#<http://www.GNU.org/CopyLeft.html>.

#Perl procedures are modeled after or paraphrased from The Perl Cookbook by Tom
#Christiansen and Nathan Torkington, copyright 1998 by O'Reilly & associates,
#Sebastepol, CA. Other ideas, and the Perl compiler, are from #<http://www.Perl.com>
#This pulls out a random sample of 100 letters.
#to evaluate by hand, as a Gold Standard
$argv = @ARGV;
($argv == 2) or die "Tag requires 2 arguments - an input file and an output file.\n";

print "The input file is \"$ARGV[0]\" and the output file is \"$ARGV[1]\"\n";
print "If that is correct, press Enter, if it is not, press Ctrl+C\n";
$dummy = <STDIN>;

#initialize all variables
```

```
#The hash table that holds the UI of each term

print "Working\n";

my @samples;
for($i=0; $i<100; $i++) {
   $a = int(rand()*1145);
   $samples[$i] = $a;
}
print "--> @samples\n";
@samples = sort {$a <=> $b} @samples;
print "--> @samples\n";
$cpointer = 0;
print "--- PASS 1 .\n";

open(FIN, "<".$ARGV[0]) or die "Cannot open \"$ARGV[0]\" for input\n";
open(FOUT, ">".$ARGV[1]) or die "Cannot open \"$ARGV[1]\" for output\n";
$lc=0;
$cletter=0;
while(<FIN>){
   $lc++;
   if($lc>=100){
        $lc=0; print ".";
     }
   while ($cletter > $samples[$cpointer]) {
      $cpointer++;
      if($cpointer>=100) {
         goto end;
      }
   }
   if($cletter == $samples[$cpointer]) {
      print FOUT $_;
   }
   if( m{\~} ) {
      $cletter++;
   }
}
end:

print "\nExamined $cletter letters in order to generate 100.\n";
close(FIN);
close(FOUT);
print "Done.\nPress Enter to Quit.";
$dummy = <STDIN>;
# ------------------------------------------------------------------------
# The End
```

APPENDIX F - COVER LETTER SENT WITH DATA SET TO 20 PHYSICIANS.

Esteemed colleague:

I have a favor to ask of you. Please help me with a research project. Please take a highlight marker, and go through these 100 letters of consultation.

In Partial Fulfillment of the Requirements for the Degree of Master of Science Massachusetts Institute of Technology, I have written a few procedures called TagMeds. TagMeds sorts through free-text medical records and provides the answers to the following questions. "Precisely which medications is this patient taking at the present time?" "How much of each medication is being taken with each dose?" "By what route is each dose being give?" "How frequently is such a dose being give?" "For what duration is such a regimen in place?"

My hypothesis is that TagMeds can answer these questions as well as a group of physicians can. In order to allow me to test my hypothesis, I have one hundred letters of consultation from the pediatric endocrinology clinic at the Children's Hospital of Boston, courtesy of Dr. Isaac Kohane. Although these are actual letters, Dr. Kohane has changed the names, addresses, ID numbers and other identifying information. Neither TagMeds nor I have looked at this set of letters, although we have looked at a thousand other letters that were originally in the same data set.

Please go through this set of one hundred letters and highlight, or underline, the answers to the above questions. Please return the manually marked set of letters to me as quickly as possible, as I must examine, analyze, and write up the results, and submit my thesis by May 11, 2001. Thank you for your kind consideration.

Sincerely,

Andy

APPENDIX G – ENGLISH LEXEMES RECOGNIZED BY TAGMEDS

This appendix is included for ease in noting the English lexemes and phrases that TagMeds recognizes in its Perl tagging procedure. Some lexemes are used to determine the relevance of contiguous terms and sections. Some are tagged when found. Some are untagged when found. Some trigger the tagging or untagging of contiguous words, numbers or phrases. Some readers may find utility in viewing the English usages outside of the context of the Perl procedures that recognize them.

Lexemes searched for to determine relevance, and aggressiveness of tagging algorithm:
current medications
is currently on
disposition
impression
history
in summary
test results
physical exam
sincerely
past medical history
review of systems
family history
urinalysis
laboratory studies

Lexemes triggering a search for a term to be tagged under certain conditions:
none
phosphorus
potassium
is on his
is on her
is on a
is on and
is on no
is on
current dose of thyroid hormone
current dose of

Lexemes triggering the untagging of a term:
has not been on
hold off on
off of
would recommend
not on
possibility of using
endogenous
insulin dependent|depended|reaction off the
off
recommend
advise
suggest
may be changed to
discontinue(d) the
can be

```
deficiency
metabolites
stop

Lexeme not triggering untagging:
not stop

Triggers tagging:
current medication(s) include(s)

Empty patterns that must be skipped:
because of the medicine (chemical 1, chemical 2)
STUDY
only two weeks

SPECIFIC tags are tagged out of order, by iteratively appending the next tag
following or preceding the medication tag add the medication around the
form_cd
1A Units and Dosage
dos(age|e) (to|of|only) * (one|two|a|)
(mg|mcg|cc|meq|tablets|tablet|units|tsp|)
dos(age|e) (to|of|only)

2A Route
(po|nasal spray|depot|i m|)inhaler|injection(s)|subcutaneously|intranasally)

3A  Schedule/Rate
(m-w-f|qhs|q.h.s.|prn|(t|q|b).i.(d|w)|evenings|(t|q|b)id|(times|x|days)
(per|a|every|/) (day|week)|(alt) qd
qo(d(ay|)|week|month)|(once) daily|mornings|prn|on a prn basis|for sleeping)

1B Units and Dosage
(one|two|a|)(mg|mcg|cc|meq|tablets|tablet|units|tsp|)(?:of)


2B Route
(po|sq|nasal
spray|depot|im|inhaler|injection(s)|subcutaneously|intranasally)(of)

3B  Schedule/Rate
 (m-w-
f|qhs|q.h.s.|prn|(t|q|b).i.(d|w).|(t|q|b)id|(times|x|days)(per|a|every|\|/)(?
:day|week)|(alt)q(.)(o(.))(d(ay|)(.)|week(s)|month)|(once)
daily|evenings|mornings|prn|on a prn basis|for sleeping)(of)

Triggers search for term:
current medication(s) include(s)

Shuts off search for term:
(level(s|)|binding|stores)(serum)

ALL MEDS BELOW ARE IN THE CACHE DICT
Fer-In-Sol
potassium phosphate
PTU
calcium citrate
multi-vitamins
```

humulin insulin
regular
r
n

cache.dict:
Adriamycin-|C0085752
Advil-|C0593507
Albumin-|C9999999
ALBUTEROL-|C0001927
albuterol-|C0001927
Albuterol-|C0001927
Alupent-|C0591074
Amikacin-|C0002499
aminoglycosides-|C0003233
Amitriptyline-|C0002600
AMOXICILLIN-|C0002645
Amoxicillin-|C0002645
amoxicillin-|C0002645
AMOXIL-|C0700524
Ampicillin-|C0002680
ampicillin-|C0002680
antibiotics-|C0003232
anticonvulsants-|C0003286
anticonvulsant-|C0003286
antifungal-|C0003308
antihypertensive-|C0003364
antiseizure-|C0808551
ATENOLOL-|C0004147
Atenolol-|C0004147
Ativan-|C0699194
ATROVENT-|C0591130
Augmentin-|C0591132
Azmacort-|C0699690
BACTRIM-|C0591139
bactrim-|C0591139
Bactrim-|C0591139
BACTROBAN-|C0733863
barium-|C0004749
Beclovent-|C0699071
Beconase-|C0591151
Benadryl-|C0700899
Biaxin-|C0701281
BIAXIN-|C0701281
Bicarbonate-|C0005367
BICARBONATE-|C0005367
BICITRA-|C0718938
Bicitra-|C0718938
bicitra-|C0718938
Bromocriptine-|C0006230
Bromocryptine-|C0006230
capsules-|C0006935
CAPTOPRIL-|C0006938
Captopril-|C0006938
CARAFATE-|C0740157
Carafate-|C0740157
CCNU-|C0687700

ceftriaxone-|C0007561
CEFZIL-|C0728743
CIPROFLOXACIN-|C0008809
cisapride-|C0072916
citrate-|C0376259
clonidine-|C0009014
CLONIDINE-|C0009014
Clonidine-|C0009014
Corticosteroids-|C0001617
Cortisol-|C0020268
cortisol-|C0020268
cortrosyn-|C0701510
Cortrosyn-|C0701510
Coumadin-|C0699129
CYCLOSPORINE-|C0010592
cyclosporine-|C0010592
Cyclosporine-|C0010592
Cytoxan-|C0699319
cytoxan-|C0699319
DDAVP-|C0701195
Decadron-|C0740057
dehydroepiandrosterone-|C0011185
DEPAKENE-|C0700661
DEPAKOTE-|C0719751
Depakote-|C0719751
dextrose-|C0017725
Diamox-|C0591362
Dihydrotachysterol-|C0012319
Dilantin-|C0699512
Ditropan-|C0591395
DITROPAN-|C0591395
DMSA-|C0205739
dopamine-|C0013030
Doxycycline-|C0013090
EMLA-|C0059079
ENALAPRIL-|C0014025
Enalapril-|C0014025
enalapril-|C0014025
Enfamil-|C0726049
Epogen-|C0700704
EPOGEN-|C0700704
Ergocalciferol-|C0014695
Erythromycin-|C0014806
ERYTHROMYCIN-|C0014806
Erythropoietin-|C0014822
ESTINYL-|C0699362
estradiol-|C0014912
Estradiol-|C0014912
Estrogen-|C0014939
estrogen-|C0014939
Ethambutol-|C0014964
Feosol-|C0720403
Fe-|C0022084
Fiorinal-|C0060393
Flintstone-|C0303753
Florinef-|C0060486
Fluconazole-|C0016277

fluoride-|C0016327
Flutamide-|C0016384
FURADANTIN-|C0813801
Furadantin-|C0813801
FUROSEMIDE-|C0016860
GEMFIBROZIL-|C0017245
Glucagon-|C0017687
glucagon-|C0017687
glucocorticoids-|C0017710
glucocorticoid-|C0017710
gluconate-|C0220836
glyburide-|C0017628
humulin-|C0020171
Humulin-|C0020171
hydantoin-|C0020209
hydralazine-|C0020223
HYDRALAZINE-|C0020223
Hydralazine-|C0020223
hydrochlorothiazide-|C0020261
HYDROCHLOROTHIAZIDE-|C0020261
Hydrocortisone-|C0020268
hydrocortisone-|C0020268
Hydroxysteroid-|C0020393
HYTAKEROL-|C0721005
Hytakerol-|C0721005
Ibuprofen-|C0020740
immunizations-|C0020971
Imodium-|C0591635
Imuran-|C0699279
IMURAN-|C0699279
INDERAL-|C0591636
Inderal-|C0591636
inhalers-|C0021461
inhaler-|C0021461
INHALER-|C0021461
injections-|C0021485
injection-|C0021485
Insulin-|C0021641
insulin-|C0021641
Intal-|C0591652
INTAL-|C0591652
interferon-|C0021747
intramuscular-|C0442117
intranasal-|C0442118
intravenous-|C0348016
iodide-|C0021966
iodine-|C0021968
iron-|C0022084
Isomil-|C0726743
jelly-|C0453543
KAYEXALATE-|C0124498
Kayexalate-|C0124498
Labetalol-|C0022860
labetalol-|C0022860
LABETALOL-|C0022860
lanolin-|C0023024
Lasix-|C0699992

LASIX-|C0699992
laxatives-|C0282090
Lente-|C0021659
lente-|C0021659
Levothyroid-|C0733849
Levothyroxine-|C0023589
Levoxine-|C0733850
Levoxyl-|C0721346
Lithium-|C0023870
Lopid-|C0700003
LOPID-|C0700003
Lorabid-|C0721422
lotion-|C0544341
Lupron-|C0701459
MACRODANTIN-|C0700187
Macrodantin-|C0700187
medications-|C0013227
medication-|C0013227
Medication-|C0013227
medicines-|C0025118
medrol-|C0699458
Motrin-|C0699203
Multivitamins-|C0351716
multivitamins-|C0351716
MULTIVITAMIN-|C0351716
Multivitamin-|C0351716
multivitamin-|C0351716
MYLANTA-|C0067044
MYLICON-|C0721886
Nasalcrom-|C0700883
nebulizers-|C0027524
nebulizer-|C0027524
needle-|C0027551
Neurontin-|C0678176
NEURONTIN-|C0678176
nifedipine-|C0028066
Nifedipine-|C0028066
NIFEDIPINE-|C0028066
nitrofurantoin-|C0028156
NIZORAL-|C0699439
nortriptyline-|C0373487
Nortriptyline-|C0373487
Novolin-|C0028467
NPH-|C0020258
Nutropin-|C0699619
NYSTATIN-|C0028741
Nystatin-|C0028741
ointment-|C0028912
Oxandrolone-|C0029995
Pediazole-|C0135837
Penicillin-|C0220892
penicillin-|C0220892
Pepcid-|C0678119
perinatal-|C0178795
Peroxide-|C0031180
Phenobarbital-|C0031412
Plaquenil-|C0699177

Porcine-|C0039005
porcine-|C0039005
pork-|C0452867
PREDNISOLONE-|C0032950
PREDNISONE-|C0032952
Prednisone-|C0032952
prednisone-|C0032952
Premarin-|C0699710
prescribed-|C0278329
prescription-|C0033080
Prescriptions-|C0033080
prescriptions-|C0033080
procardia-|C0700861
PROCARDIA-|C0700861
Procardia-|C0700861
progesterone-|C0373705
Progesterone-|C0373705
Propranolol-|C0202459
propranolol-|C0202459
PROPRANOLOL-|C0202459
Propulsid-|C0722861
Propylthiouracil-|C0033511
Prosobee-|C0727589
ProsoBee-|C0727589
Protropin-|C0699978
Proventil-|C0699770
PROVERA-|C0699702
Provera-|C0699702
Prozac-|C0162373
Pulmozyme-|C0251564
Pyridium-|C0034259
Ranitidine-|C0034665
regimen-|C0677937
Reglan-|C0034977
Regular-|C0205272
Rifampin-|C0035608
RITALIN-|C0728759
Ritalin-|C0728759
ROCALTROL-|C0592076
Rocaltrol-|C0592076
Septra-|C0699595
Serzone-|C0553415
Similac-|C0727850
Solumedrol-|C0701466
solumedrol-|C0701466
solution-|C0037633
Somatomedin-|C0037657
Soma-|C0702216
Spironolactone-|C0037982
Steroids-|C0038317
subcutaneous-|C0443315
sublingual-|C0558302
Sulfamethoxazole-|C0038689
SULFAMETHOXAZOLE-|C0038689
Sulfate-|C0038720
SULFATE-|C0038720
sulfa-|C0749139

SUPRAX-|C0678177
synthroid-|C0728762
Synthroid-|C0728762
SYNTHROID-|C0728762
syrup-|C0458173
tablets-|C0039225
tablet-|C0039225
Tapazole-|C0728778
Tavist-|C0086020
Tegretol-|C0700087
tegretol-|C0700087
terbutaline-|C0039542
Testosterone-|C0039601
Tetracycline-|C0039644
tetracycline-|C0039644
Thyroxine-|C0040165
thyroxine-|C0040165
thyroxin-|C0040165
Titralac-|C0723762
TITRALAC-|C0723762
Tobrex-|C0723768
topical-|C0332237
Trimethoprim-|C0041041
Triphasil-|C0728965
TRIPHASIL-|C0728965
Tums-|C0723950
Ultralente-|C0041616
ultralente-|C0041616
units-|C0439148
vaccines-|C0042210
vaccine-|C0042210
Valium-|C0699187
Valproate-|C0080356
VANCERIL-|C0699073
Vanceril-|C0699073
Vaseline-|C0728774
Vasotec-|C0728763
VASOTEC-|C0728763
Ventolin-|C0033744
VENTOLIN-|C0033744
verapamil-|C0042523
Verapamil-|C0042523
vincristine-|C0042679
vitamins-|C0042890
vitamin-|C0042890
VITAMIN-|C0042890
Zantac-|C0592278
ZANTAC-|C0592278
Zoloft-|C0284660

BIBLIOGRAPHY:

[i] Mandl KD, Riva A, and Kohane IS: A Distributed, Secure File System for Personal Medical Records. International Journal of Medical Informatics 62 (2001) 27-40.

[ii] Riva A, Mandl KD, Oh DH, Nigrin DJ, Butte A, Szolovits P, and Kohane IS: The Personal Internetworked Notary and Guardian. American Medical Informatics Association 2000 Proceedings.

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

[iv] Hripcsak G, Friedman C, Alderson PO, DuMouchel W, Johnson SB, Clayton PD. Unlocking Clinical Data from Narrative Reports: A Study of Natural Language Processing. *Annals of Internal Medicine.* 1995 122(9): 681-688.

[v] Lovis, C and RH Baud. Fast Exact String Pattern-matching Algorithms Adapted to the Characteristics of the Medical Language. *J. Am. Med. Inform. Assoc.* 2000 7: 378–391.

[vi] Gabrieli, ER and DJ Speth. Automated analysis of medical text. II. Cognitive strategy. *J Med Syst.* 1991 Feb;15(1):65–78. PMID: 1748850

[vii]Campbell DA and SB Johnson. A technique for semantic classification of unknown words using UMLS resources. Proc AMIA Symp. 1999;:716–20. PMID: 10566453; UI: 20032963

[viii] Johnson SB. A semantic lexicon for medical language processing. J Am Med Inform Assoc. 1999 May-Jun;6(3):205-18. PMID: 10332654; UI: 99265058

[ix] Johnson SB. Conceptual graph grammar—a simple formalism for sublanguage. Methods Inf Med. 1998 Nov;37(4–5):345–52. Review. PMID: 9865032; UI: 99082551

[x]Prakash N, C Roland and C Brandt. UMLS Concept Indexing for Production Databases: A Feasibility Study. *J. Am. Med. Inform. Assoc.* 2001 8: 80–91.

[xi]Joubert M, M Fieschi, J-J Robert, F Volot and D Fieschi. UMLS-based Conceptual Queries to Biomedical Information Databases: An Overview of the Project ARIANE. *J. Am. Med. Inform. Assoc.* 1998 5: 52–61.

[xii] Bodenreider O, A Burgun, G Botti, M Fieschi, P Le Beux and F Kohler. Evaluation of the Unified Medical Language System as a Medical Knowledge Source. *J. Am. Med. Inform. Assoc.* 1998 5: 76–87.

[xiii]Kim W and WJ Wilbur. Corpus-based Statistical Screening for Phrase Identification. *J. Am. Med. Inform. Assoc.* 2000 7: 499–511.

[xiv] Bakken S, JJ Cimino, R Haskell, R Kukafka, C Matsumoto, GK Chan and Stanley M. Huff. Evaluation of the Clinical LOINC (Logical Observation Identifiers, Names, and Codes) Semantic Structure as a Terminology Model for Standardized Assessment Measures. *J. Am. Med. Inform. Assoc.* 2000 7: 529–538.

[xv] Huff SM, RA Rocha, CJ McDonald, GJE De Moor, T Fiers, WD Bidgood, Jr., AW Forrey, WG Francis, WR Tracy, D Leavelle, F Stalling, B Griffin, P Maloney, D Leland, L Charles, K Hutchins and J Baenziger. Development of the Logical Observation Identifier Names and Codes (LOINC) Vocabulary. *J. Am. Med. Inform. Assoc.* 1998 5: 276–292.

[xvi] Sager, N. *Natural Language Information Processing: A Computer Grammar of English and Its Applications.* ISBN 0–201–06769–2. 1981. Addison-Wesley Publishing Company, Reading, MA.

[xvii] Sager N, M Lyman, C Bucknall and LJ Tick. Natural Language Processing and the Representation of Clinical Data. *J. Am. Med. Inform. Assoc.* 1994; 1:142–160.

[xviii] Ohno-Machado L, JH Gennari, SN Murphy, NL Jain, SW Tu, DE Oliver, E Pattison-Gordon, RA Greenes, EH Shortliffe and GO Barnett. The GuideLine Interchange Format: A Model for Representing Guidelines. *J. Am. Med. Inform. Assoc.* 1998 5: 357–372.

[xix] Schulz E, JW Barrett, and C Price. Read Code Quality Assurance: From Simple Syntax to Semantic Stability. *J. Am. Med. Inform. Assoc.* 1998 5: 337–346.

[xx] Schulz EB, C Price,and PJB Brown. Symbolic Anatomic Knowledge Representation in the Read Codes Version 3: Structure and Application. *J. Am. Med. Inform. Assoc.* 1997 4: 38–48.

[xxi] Cooper GF and RA Miller. An Experiment Comparing Lexical and Statistical Methods for Extracting MeSH Terms from Clinical Free Text. *J. Am. Med. Inform. Assoc.* 1998 5: 62–75.

[xxii] Tange HJ, A Hasman, PF de Vries Robbe and HC Schouten. Medical narratives in electronic medical records. *Int J Med Inf.* 1997 Aug;46(1):7–29. Review. PMID: 9476152; UI: 98136629

[xxiii]Tange HJ, HC Schouten, ADM Kester and A Hasman. The Granularity of Medical Narratives and Its Effect on the Speed and Completeness of Information Retrieval. *Journal of the American Medical Informatics Association* 5:571–582 (1998)

[xxiv] Cote R, D Rothwell and J Palotay. SNOMED. College of American Pathologists, 1994.

[xxv] SNOMED International: The Systematized Nomenclature of Medicine. *http://www.SNOMED.org/* Accessed September 10, 2000.

[xxvi] Campbell JR, P Carpenter, C Sneiderman, S Cohn, CG Chute and J Warren. Phase II Evaluation of Clinical Coding Schemes: Completeness, Taxonomy, Mapping, Definitions, and Clarity. *J. Am. Med. Inform. Assoc.* 1997 4: 238–250.

[xxvii] *UMLS Knowledge Sources, 12th Edition.* National Library of Medicine. January 2001. National Institutes of Health, Bethesda, MD.

[xxviii] UMLS Metathesaurus. *http://www.nlm.mih.gov/research/umls/UMLSDoc.html/* Accessed March 20, 2001.

[xxix] St. Laurent, S. *XML Elements of Style.* ISBN 0–07–212220-X. 2000. McGraw-Hill, New York, NY.

[xxx] Friedman C, G Hripcsak, L Shagina and H Liu. Representing Information in Patient Reports Using Natural Language Processing and the Extensible Markup Language. *J. Am. Med. Inform. Assoc.* 1999; 6: 76–87.

[xxxi] Friedman C, PO Alderson, J Austin, JJ Cimino and SB Johnson. A General Natural Language Text Processor for Clinical Radiology. *J. Am. Med. Inform. Assoc.* 1994; 1(2):161–74.

[xxxii] Cimino JJ. Auditing the Unified Medical Language System with Semantic Methods. *J. Am. Med. Inform. Assoc.* 1998 5: 41–51.

[xxxiii] Rosenthal D, JA Bos, R Sokolowski, J Mayo, K Quigley, R Powell and M Teel. A Voice-Enabled, Structured Medical Reporting System. *J. Am. Med. Inform. Assoc.* 1997; 4:436–441.

[xxxiv] Sokolowski R. XML makes its mark. *J AHIMA.* 1999 Nov-Dec;70(10):21–4; quiz 25. PMID: 10977303; UI: 20328820

[xxxv] Sokolowski R, and J Dudeck. XML and its impact on content and structure in electronic healthcare documents. *Proc AMIA Symp.* 1999;:147–51. PMID: 10566338; UI: 20032848

[xxxvi] Shiffman RN, BT Karras, A Agrawal, R Chen, L Marenco and S Nath. GEM: A Proposal for a More Comprehensive Guideline Document Model Using XML. *J. Am. Med. Inform. Assoc.* 2000 7: 488–498.

[xxxvii] Shiffman S, WM Detmer, CD Lane and LM Fagan. A continuous-speech interface to a decision support system: I. Techniques to accommodate for misrecognized input. *J. Am. Med. Inform. Assoc.* 1995 2: 36–45.

[xxxviii] Tarczy-Hornoch P, P Shannon, P Baskin, M Espeseth and RA Pagon. A Hybrid Text/Data Electronic Publishing Model Using XML Applied to Clinical Genetic Testing. *J. Am. Med. Inform. Assoc.* 2000 7: 267–276

[xxxix] McEntire R, P Karp, N Abernethy, D Benton, G Helt, M DeJongh, R Kent, A Kosky, S Lewis, D Hodnett, E Neumann, F Olken, D Pathak, P Tarczy-Hornoch, L Toldo and T Topaloglou. An evaluation of ontology exchange languages for bioinformatics. *Ismb.* 2000; 8:239–50.PMID: 10977085; UI: 20431597

[xl] Health Level 7 Standards: *http://www.h17.org.* Accessed 9/4/00. Health Level Seven, Inc., Ann Arbor, MI.

[xli] Health Level 7 XML Technical Committee. *http://www.h17.org/special/committees/sgml/sgml.htm.* Accessed September 9, 2000. Health Level Seven, Inc., Ann Arbor, MI.

[xlii] Aho AV, Sethi R, Ullman JD. Compilers: Principles, Techniques and Tools. 1985. Addison-Wesley Publishing Co. ISBN: 0201100886.

[xliii] T Christiansen and N Torkington: The Perl Cookbook. Copyright 1998 by O'Reilly & Associates, Sebastepol, CA.

[xliv] O'Reilly Publishing's Perl.com. *http://www.Perl.com* . Accessed 12 February 2001.

[xlv] A Descartes and T Bunce: Programming the Perl DBI. Copyright 2000 by O'Reilly & Associates, Sebastepol, CA.

[xlvi] What is CopyLeft?–GNU Project - Free Software Foundation. *http://www.gnu.org/copyleft/copyleft.html.* Accessed 12 February 2001. Free Software Foundation.

[xlvii] umls.dict http://www.medg.lcs.mit.edu/umls/umls.dict Accessed May 8, 2001. Nakrin AS.

[xlviii] Fisher LD and van Belle G. Biostatistics: A Methodology for the Health Sciences. Copyright 1993. John Wiley & Sons, New York. ISBN 0-471-58465-7

[xlix] Zar JH. Biostatistical Analysis. Copyright 1974. Prentice-Hall, Inc, Englewood Cliffs, NJ. ISBN 0-13-076984-3.

[I] Berners-Lee T, Fischetti M, Dertouzos M. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web.* 1999. Harper Publishing Co., San Francisco. ISBN: 0062515861

[Ii] Dolin RH, L Alschuler, F Behlen, PV Biron, S Boyer, D Essin, L Harding, T Lincoln, JE Mattison, W Rishel, R Sokolowski, J Spinosa and JP Williams. HL7 document patient record architecture: an XML document architecture based on a shared information model. *Proc AMIA Symp.* 1999;:52–6. PMID: 10566319; UI: 20032829

THESIS PROCESSING SLIP

FIXED FIELD: ill. _____ name _____

index _____ biblio _____

► **COPIES:** Archives Aero Dewey Barker Hum

Lindgren Music Rotch Science Sche-Plough

TITLE VARIES: ► ☐ _____

_____

_____

NAME VARIES: ► ☐ _____

_____

IMPRINT: **(COPYRIGHT)** _____

_____

► **COLLATION:** _____

_____

► **ADD: DEGREE:** _____ ► **DEPT.:** _____

► **ADD: DEGREE:** _____ ► **DEPT.:** _____

SUPERVISORS: _____

_____

NOTES:

_____ cat'r: date:

page:

► **DEPT:** _____

► **YEAR:** _____ ► **DEGREE:** _____

► **NAME:** _____

_____