An Open-Source, Interactive Java-Based System for Rapid Encoding of Significant Events in the ICU Using the Unified Medical Language System

J Shu[1], GD Clifford[1], WJ Long[2], GB Moody[1], P Szolovits[2], RG Mark[1]

[1]Harvard-MIT Division of Health Sciences & Technology, Cambridge MA, USA
[2]Computer Science and Artificial Intelligence Laboratory, Cambridge MA, USA

Abstract

The MIMIC II project, previously described at Computers in Cardiology, is creating a massive annotated collection of recorded cardiovascular and related signals and accompanying clinical data from intensive care units (ICUs), to support research aimed at improving critical care monitoring. Annotating this database requires describing its significant clinical events using standardized terminology. We present an interactive Java-based application that encodes events by retrieving the clinical concepts that most closely match a free-text input phrase. We chose to code the events using a subset of the National Library of Medicine's Unified Medical Language System (UMLS), a standard, freely available collection of medical vocabularies for identifying diseases, symptoms, and other clinical concepts. Some common difficulties that occur in the process of coding are spelling errors, abbreviation ambiguities, and combinations of events for which there is no UMLS code. Among the features our system has to deal with these issues are an integrated spell-checker and a personalized dictionary allowing each user of the system to keep a unique list of frequently used abbreviations. Additionally, utilizing the UMLS concept hierarchy and using "fuzzy" string matching both help to find the UMLS concepts that most closely match the free text description. We ran an unsupervised, non-interactive batch test of the program with the UMLS stored in a MySQL database on a 3GHz Pentium 4 PC. We used as input a list of almost 200 distinct medical phrases (a total of 453 words, or 2.3 words/phrase) that were manually extracted from a sample of hospital discharge summaries and then reviewed by a clinician. The program successfully coded approximately 85% of the terms, as judged by two clinicians, and it took an average of 2.56 seconds to complete each query. We present methods for reducing the bottlenecks and improving the completeness and accuracy of the coding, including subject-specific abbreviation dictionaries. Although our application is written to support the MIMIC II project, similar applications will be important components of future real-time medical decision support systems. The use of a flexible API and freely available dictionaries facilitates open source distribution and integration.

1. Introduction

The Laboratory for Computational Physiology at MIT is currently developing a large, comprehensive multi-parameter database of patient signals and clinical data (MIMIC II) [1, 2] in order to support research in intelligent patient monitoring. The MIMIC II database includes physiologic signals, laboratory tests, nursing flow charts, clinical progress notes, and other data collected from patients in the intensive care units of a local hospital. Expert clinicians are reviewing each case and annotating clinically significant events, which include, but are not limited to, diseases (e.g., gastrointestinal bleed, septic shock, hemorrhage), symptoms (e.g., chest pain, nausea), significant medication changes, vital sign changes (e.g., tachycardia, hypotension), waveform abnormalities (e.g., arrhythmias, ST elevation), and abnormal laboratory values. The annotations will be used to train and test future algorithms that automatically detect significant clinical events, given a patient's recorded data. Thus, to make the annotations useful for machine analysis, each annotation must be labelled with a machine-readable, or standardized, code.

Assigning such a code to a clinical event involves automatically translating a free-text description of the event (provided by the annotator) into one or more codes from a medical vocabulary. Each unique clinical concept is assigned a concept code (a unique alpha-numeric identifier), and the concept generally has several different synonyms. For example, *heart attack* and *myocardial infarction* represent the same concept, and both strings are mapped to the same concept code.

The database of medical terminology, or codes, chosen for this task is a subset of the 2004AA version of the National Library of Medicine's Unified Medical Language System (UMLS) [3], a collection of over 100 source vocabularies available (usually freely) from the National Library of Medicine. This subset is effectively equivalent to SNOMED-CT [4], a hierarchical medical nomenclature formed by merging the College of American Pathologists' Systematized Nomenclature of Medicine (SNOMED) with the UK National Health Service's Read Clinical Terms (CT). SNOMED-CT contains a collection of concepts, descriptions, and relationships and is rapidly becoming an international standard for coding medical concepts. Each concept in the vocabulary represents a clinical concept, such as a disease, symptom, intervention, or body part. Each concept can be described by one or more terms (synonyms). In addition, there are many types of relationships that link the different concepts, including hierarchical (*is-a*) relationships and attribute relationships (such as a body part being linked to a disease through the *finding site* relationship). The UMLS captures all of the information contained in SNOMED-CT, but is stored within a different database structure.

There are several challenges to translating free-text phrases into standardized terminology. The search for concept codes must be accurate and rapid enough that annotators do not lose patience. Annotators also tend to make spelling mistakes and use abbreviations that have more than one meaning. The same concept may be described in various different ways, and annotators might wish to code a concept that simply does not exist in the UMLS. Sometimes the annotator might not be satisfied with the level of specificity of codes returned and may want to look at related concepts. This article addresses these issues and compares the accuracy and search times for a variety of medical phrases.

2. Methods

The interface for assessing search performance is an interactive, Java-based application that searches for UMLS concepts that encode a free-text medical phrase. The system's features include an open-source spell-checker and a personalized abbreviation dictionary, along with a large list of commonly used medical abbreviations.

2.1. Special Features

This section describes some of the special features that have been incorporated into the coding algorithm.

2.1.1 Common Abbreviations

The UMLS contains a table of abbreviations and acronyms and their expansions [5], but the table is not adequate for a clinical event coding algorithm because it contains many irrelevant (non-medical) abbreviations, yet lacks many abbreviations that an annotator might use. Therefore, an open source list of medical abbreviations and acronyms [6] is used instead. The list simply contains a textual list of abbreviations and their expansions.

2.1.2 Personal Abbreviations

When reviewing a patient's medical record, annotators will probably wish to code the same clinical concept multiple times. Thus, they have the option at any time to link a term or abbreviation directly to one or more UMLS concept codes, which are saved in a MySQL [7] table and available in later lookups. For example, the annotator can add the abbreviation *mi*, linked to the concept code *C0027051*, the identifier for *myocardial infarction*. On a subsequent attempt to code *mi*, *myocardial infarction* is guaranteed to be one of the concepts returned. This feature also addresses the fact that the open source common abbreviation list sometimes does not contain desired abbreviations.

2.1.3 Spell Checker

Annotators, and clinicians in general, tend to make spelling errors due to being rushed or not knowing the spelling of a complex medical term. An open source spell checker [8] is therefore incorporated into the coding process. The dictionary word list used with the spell checker consists of a standard spelling dictionary [9], augmented with the words from the UMLS normalized word table, which contains medical terms that are not in the standard dictionary. Additionally, the words from the common and personal abbreviation lists are added to the dictionary so that they are not mistaken for misspelled words.

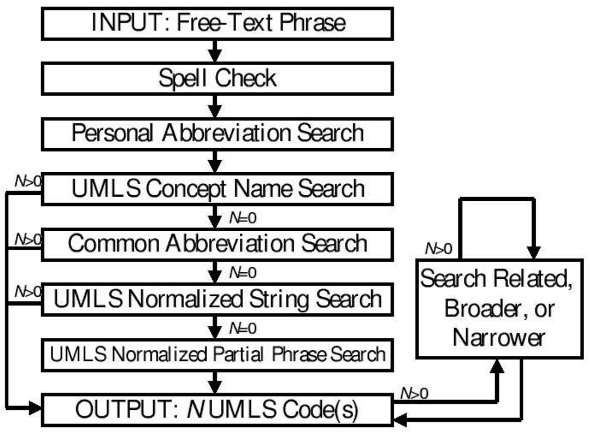
INPUT: Free-Text Phrase
Spell Check
Personal Abbreviation Search
UMLS Concept Name Search
Common Abbreviation Search
UMLS Normalized String Search
UMLS Normalized Partial Phrase Search
OUTPUT: N UMLS Code(s)
Search Related, Broader, or Narrower
N>0
N=0
N>0
N=0
N>0
N=0
N>0
N>0

Figure 1. A flow chart of the search process, where *N* is the number of UMLS codes found by the algorithm.

2.2. Search Procedure

The input to the coding algorithm is a free-text phrase highlighted or typed by the annotator. The phrase is run through a spell checker before a series of resources is consulted to find a concept code matching the phrase. First, the user's personalized abbreviation dictionary is checked, and if it contains a mapping for the input phrase, then those concepts are added to the preliminary results. The results of a concept name lookup in the UMLS concepts table is also added to the results, so that the search is not always limited to the personal abbreviation list. From this point on, the algorithm continues searching only if no potential codes have been found thus far, ensuring that the user receives codes back as rapidly as possible. The next step is to check the common abbreviation list to see if the input phrase is an abbreviation. If still nothing is found, the algorithm then tries to match the input phrase against the UMLS normalized string table, which consists of all UMLS concepts in normalized form, i.e., with common words such as *the*, *and*, and *of* removed and the remaining words alphabetized. Finally, if there are still no concepts found, the algorithm breaks the input up into the largest subsets of words that it can code, and codes each subset separately. At any point, when potential concepts are displayed, the user has the option of searching for *related*, *broader*, or *narrower* terms; these relationships correspond to the *is-a* relationship hierarchy in the UMLS and are helpful for finding a more or less specific concept than the one presented. See Figure 1 for a flow diagram of the search algorithm. *N* represents the number of concepts found after each part of the algorithm.

2.3. Algorithm Testing Method

The initial test of the coding algorithm consisted of running a non-interactive batch test of 194 medical phrases and having two clinicians adjudicate the results. This first test coded 85% of the phrases accurately, taking an average of 2.56 seconds to code each phrase. As a further test of the speed and accuracy of the coding algorithm, a fully interactive version of the software was used by three clinicians to code over 1400 medical phrases. The phrases were taken from a corpus of nursing admission and progress notes from a local hospital's ICU. Specifically, the focus was narrowed to three types of clincal information (*medications*, *diseases*, and *signs or symptoms*) to realistically simulate a subset of phrases that would be coded in an annotation situation. Three clinicians were asked to pick out all such phrases from the nursing notes and invoke the UMLS coding software to code each phrase. The SNOMED-CT subset of the UMLS database was stored in MySQL (MyISAM) tables on a 3GHz Pentium 4 PC. The clinicians each began with an empty list of personal abbreviations (also stored in a MySQL database), which they could add to as necessary. For each phrase, the clinicians were asked to choose the correct resulting UMLS code(s) (if any) and judge whether the results: 1) captured the full meaning of the phrase, 2) were codeable but did not capture the concept exactly, or 3) were simply wrong. To identify bottlenecks, the time to run each type of search (shown in Figure 1) was also logged.

Table 1. Testing Results, including coding success rate, average length of phrases, personal abbreviations added, and misspellings corrected.

Type	Medication	Disease	Sign/ Symptom
Total Phrases	597	261	619
% Success	96.6%	92.3%	76.9%
Avg. Words/Phrase	1.1	1.7	2.0
New Abbreviations	64	23	21
Misspellings	33	19	52

3. Results

The clinicians reviewed a total of 302 nursing notes and found a total of 1477 phrases (597 medications, 261 diseases, and 619 signs or symptoms). The clinician-judged accuracy results show that 90-95% of medications and diseases were coded successfully (i.e., captured the full meaning of the phrase), whereas only 77% of signs and symptoms were coded successfully. Several misspellings were corrected for all three phrase types. Medications were most frequently added as personal abbreviations, because many drug trade names could not be found in the SNOMED-CT subset of the UMLS and were coded with their generic drug names instead. These results are summarized in Table 1.

Timing results were recorded for the five types of searches shown in Figure 1 (i.e., personal abbreviation, concept name, common abbreviation, normalized string, and partial phrase). All personal abbreviation searches took less than 1ms to complete. 75% of concept name searches took less than 4ms, and all took less than 60ms. 95% of common abbreviation searches took less than 5ms, but some phrases took several seconds to complete, most likely due to abbreviations that expanded into phrases with multiple words. Almost all (92%) of the normalized string searches took between 1.35s and 1.75s, while partial phrase searches often took several seconds to complete, depending on the number of words in the input phrase. The largest bottlenecks occurred when exact phrases could not be found and wild-card characters had to be used in the MySQL searches.

4. Discussion

The spell checking and personal abbreviation features were shown to have a positive impact on the coding of medications, diseases, and signs and symptoms. Both the probability of the user finding the correct code on the first try and the amount of time it took to code an abbreviated phrase improved dramatically as new abbreviations were added to the personal dictionaries. The spell checker successfully helped to correct spelling mistakes in many instances.

A possible explanation for signs and symptoms being more difficult to code than medications and diseases is that signs and symptoms are often subjective descriptions of a patient's state and thus have a complex structure, whereas diseases and medications are more well-defined. However, many specific drug names could not be found in SNOMED-CT, and instead had to be coded by searching for the generic drug name. This extra step increased the time required to code many medications initially, but once they were added to the personal abbreviation list, the time to code these medications was negligible.

Most parts of the search algorithm were performed quickly because the lookup tables (for common and personal abbreviations) were small enough to read into memory and because the MySQL tables were indexed. The slower searches used wild-card characters and thus could not make use of the MySQL indices. A possible way to improve some of the search times is to find a different method to replace MySQL wild-card lookups, or leave this part out completely, possibly sacrificing completeness for improved speed.

Further methods to improve the accuracy and speed of the coding algorithm will be investigated. For example, adding a vocabulary of drug trade names might make the coding of medications faster. A method to remove irrelevant results from searches might be to filter the search space by semantic category (e.g., by organ systems or diseases). Another problem to consider is that although searching the normalized string table often helps to find an approximate concept match, it does not consider word order and thus might not capture the actual meaning of the input phrase. To help preserve semantics, UMLS tools such as part-of-speech tagging might be utilized.

5. Conclusion

The system presented for coding free-text medical phrases into UMLS concept codes is shown to code medications and diseases very accurately, and signs and symptoms moderately accurately, with times ranging from under 1ms to several seconds to code a concept. The interactive features of the system, including personal abbreviation lists and spell checking, had a positive impact on the coding process. Personal abbreviations were used extensively and improved the speed of coding, even though there was some learning time because the personal dictionaries were initially empty. Several spelling mistakes were fixed, leading to successful coding.

The test results will be used to resolve bottlenecks and improve the efficiency of the search algorithm. Feedback gathered from the clinicians will be used to improve the accuracy and usability of the coding software. Incorporating new medical vocabularies and filtering results by semantic or syntactic category are methods that might be explored to filter out irrelevant codes. These techniques will be helpful in developing an algorithm to automatically extract and code clinical phrases such as medications, diseases, symptoms, treatments, and laboratory tests from nursing progress notes. The Java source code and a coded corpus of text will be posted on Physionet[10, 11].

Acknowledgments

This publication was made possible by Grant Number R01 EB001659 from the National Institute of Biomedical Imaging and Bioengineering (NIBIB).

References

[1] Saeed M, Lieu C, Raber G, Mark R. MIMIC II: A Massive Temporal ICU Patient Database to Support Research in Intelligent Patient Monitoring. Computers in Cardiology 2002;29:641–644.

[2] Mark RG. Integrating data, models and reasoning in critical care, 2003. National Institute of Biomedical Imaging and Bioengineering Proposal R01 EB001659.

[3] National Library of Medicine. UMLS Knowledge Sources, 16th Edition - July Release: 2004AB Documentation, 2004.

[4] College of American Pathologists. SNOMED Clinical Terms Technical Specifi cation: Revision 23, 2000.

[5] National Library of Medicine. The SPECIALIST LEXICON: UMLS Documentation, 2004.

[6] Berman JJ. Pathology abbreviations and acronyms, May 2001.

[7] http://www.mysql.com.

[8] SourceForge.net. Jazzy - Java Spell Check API.

[9] ftp://metalab.unc.edu/pub/Linux/libs/linux.words.2.tar.gz. Standard Linux dictionary on /usr/share/dict. Redhat 9.0, 2003.

[10] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. Circulations 2000;101(23):e215–e220.

[11] http://www.physionet.org/.

Address for correspondence:

Jennifer Shu
Laboratory for Computational Physiology
Harvard-MIT Division of Health Sciences & Technology
Rm E25-505, 45 Carleton St.,
Cambridge MA 02142 USA
jshu@mit.edu