CodeSlinger: An Interactive Biomedical Ontology Browser

Jeffery L. Painter, Natalie L. Flowers

GlaxoSmithKline, Research Triangle Park, NC 27709, USA

Abstract. CodeSlinger is a highly interactive and semi-intelligent application designed to support the search and navigation of large biomedical coding schemes, thesauri, and ontologies. We discuss how CodeSlinger is used by epidemiologist/physicians in the creation of coding sets for data extraction and analysis, the exploratory nature of the application, and finally, the issues facing our knowledge-representation model and extension of the UMLS.

1 Introduction

There are several issues facing medical informatics today especially with regard to the identification and classification of medical "concepts" in controlled vocabularies, dictionaries, thesauri and ontologies [1].

Coping with the plethora of information available is a matter of utmost importance for both the efficacy with which we are able to make use of medical data, as well as insuring that the results we produce can be viewed with the confidence that reported analysis are accurate and inclusive of the appropriate populations. In response, we developed a multi-user client/server based application called *CodeSlinger* to assist the informaticists with at least one dilemma they face in the process of data extraction. *CodeSlinger* empowers the user to promptly and accurately identify the appropriate set of medical codes relevant to their studies. It also gives them a higher degree of confidence that they are no longer "missing" data which might be relevant to their investigations.

2 Background

In almost every medical records database, one or more "coding schemes" are employed to represent the medical concepts within it. The medical concepts can include drugs, devices, symptoms, conditions, procedures etc. The broad scope of a medical concept is just one of many difficulties when dealing with medical informatics [2].

CodeSlinger helps the user to aggregate sets of codes for the purpose of data extraction and analysis. Until now, the state of the art for this task involved (1) using large medical coding books, (2) relying on one's medical expertise from use or experience with a particular coding scheme, (3) studying the literature to see what sets of codes have been used in a study before and of course (4) using Google and other search engines to locate medical codes.

2.1 Code Selection

Our goal was to provide an application where the user could simply focus on the concepts of interest, and by making use of the highly interactive visual display of codes and their relations, quickly and with confidence develop a set of codes which properly characterizes a medical concept in the database(s) under review.

The databases the informaticists must deal with are also disparate with regard to their format and content; and comparison and analysis of data across such disparate sources requires some way of translating among the coding scheme representations (or "normalizing" the references) so that references to the "same disease", "same condition", "same procedure", or "same drug" may be identified. For example, an epidemiologist may want to extract a cohort of patient data for a study on "heart failure" from two databases that have been coded using ICD-9 [3] in one and MedDRA [4] in the other. This leads to the question of how does one then find the corresponding code maps between coding schemes which would allow the researcher to ensure that the selected populations are comparable? That is, the selected cohorts are representative of the *same* concept or condition.

This challenge led to the development of *CodeSlinger* as a semi-automated code mapping and navigation system. While there are other electronic resources available for mapping codes from one scheme to another, including the UMLS Knowledge Server [5] and TermWorks [6], we focused on developing an application that would be highly interactive and provide visual queues to assist the user in code selection while allowing for the exercise of the user's medical expertise.

In our example (see Fig. 1), the user runs a search for "heart failure", and a set of results is returned in which they will find several codes under both ICD-9 and MedDRA. Findings under ICD-9 include the code "428.1 - Left heart failure". The user may then reveal the possibly related codes by highlighting the code in the user interface. One relation then displayed is the MedDRA code "10024119 - Left ventricular failure". *CodeSlinger* supports several concept mappings that we have exploited from the UMLS system noted in a previous paper [7]. This allows the user to more easily explore the inter-relations among source vocabularies.

3 Interface

As illustrated in Fig. 1, the interface of CodeSlinger is made up of a search box, a results box, coding scheme browser(s), and a final list box (used to compile the code sets that will be used to perform data extraction). The search box at the top of Fig. 1 allows the user to select which coding schemes are of interest for a particular search.

After the search results are displayed, the user can explore each entry by clicking on the code or term to see its related concept maps in each of the sources chosen at the beginning of the search. And, if any alternate codes are associated with that particular code, those codes are made clear in the "Alternate Terms or Codes" window pane.

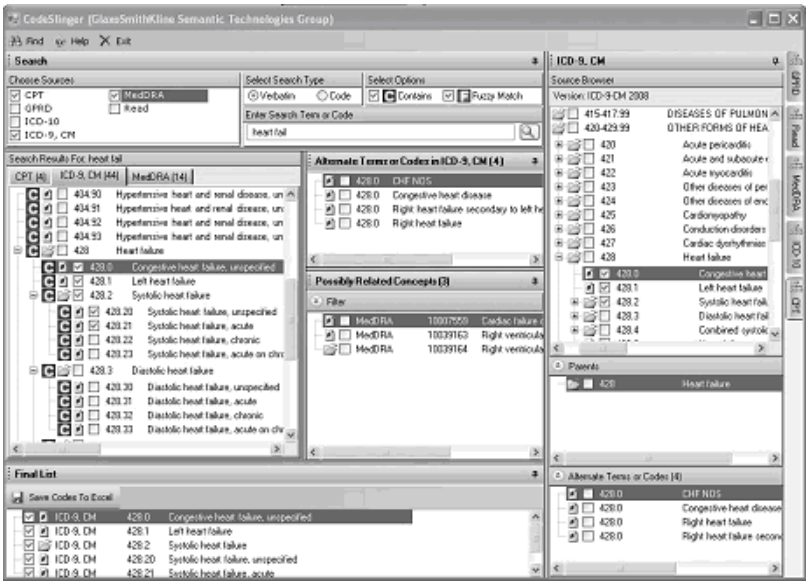

Fig. 1. CodeSlinger Component Layout View

Additionally, a term can be double clicked and the application will open the coding scheme browser (found on the right side of Fig. 1), displaying the code in its hierarchical context, allowing for further visual search and navigation.

At any time, the user can click the check box next to a code to add or remove it from their final list found at the bottom of Fig. 1. Once the code set is collected, the user can export the list as a simple spreadsheet by clicking the icon to save the codes to Excel.

4 Knowledge Representation

Much of the benefit our users experience with *CodeSlinger* is based on the fundamental knowledge representation created to facilitate the search, mapping and exploration of the "concepts" extracted from the UMLS Metathesaurus [8]. Where in most applications based on the UMLS, developers strictly take an unmodified database view of the UMLS content, we have created our own set of tools to extract and manipulate relations directly from the RRF files. We also construct our own custom database, which is comprised of a relatively minimal number of tables used to store source hierarchies, terms and relations of the concept model. The *CodeSlinger* server loads an object model of the hierarchical structures along with active concept nodes which are self-aware of both their attributes (atom identifiers, terms, alternate terms, etc) and relationships to other concept nodes within the terminology server.

The hierarchical trees are indexed to enhance the lookup and retrieval time based on key elements of the concept attributes, including *source, term* and

concept identifiers which are tied to the concept nodes in a cost conscious (in terms of both memory and speed) manner. We preserve the atom, concept and string identifiers found in the UMLS to enable quick retrieval of the information requested by the search interface.

5 Constructing a Search

The search interface allows the user several options to refine and constrain the search results. The user begins by choosing which vocabularies are of interest for the search query. Currently ICD-9, ICD-10, Read (Clinical Terms Version 3), Current Procedural Terminology (CPT), MedDRA, and a customized representation of the OXMIS coding scheme are supported. Next, the user has the option to select whether the search is for a code or term. In some cases the may know a particular code and want to see what other codes may have relevance. Finally, the user may choose from one of two search algorithms or both: (1) a simple *contains* search – indicating that the results must contain either the exact code or search string entered, or (2) a *fuzzy* match.

5.1 Contains Search

Our *contains* search ignores word order and case distinctions. This allows us to match "heart failure" to "Heart Failure", "failure of the heart" and "failure, heart" since each of the terms contains both the individual words "heart" and "failure". While this may return more than the number of items the user may like to see, the stance we chose to take in many aspects of the application was to "cast a wide net" and let the user employ medical expertise to evaluate and discriminate the output.

5.2 Fuzzy Match

The *fuzzy* match generates a Bayesian model to approximate the probability of a lexical match based on the user's search string. The *fuzzy* match is capable of dealing with misspellings and minor word variations. For example, the terms "color" versus "colour" would be matched on either spelling using our *fuzzy* match. The strings are first normalized in the following manner.

1. Tokenize and case fold the terms
2. Remove contractions and parenthetical plurals
3. Apply standard stemming algorithms
4. Remove stop words (customized for our domain)

The normalized string is converted to a sorted bi-gram sequence [9] and stored in our database. When the user submits a new search, the *fuzzy* match first normalizes the search string and converts it to a bi-gram sequence as well. A probability value is then assigned between the search sequence and each of the pre-computed sequences. If the probability of match is high enough, the source

term is matched to a set of possible "concepts" and the corresponding codes are then included in the results. We can tune the output by changing the lower bound required for a match, but this is not a user configurable option. Our best user experience results when the lower bound is set within: $p \in [0.55 - 0.60]$.

6 Results

It should be noted that we are investigating several avenues of releasing the *CodeSlinger* application to the medical informatics community. It is our goal to make this application available to those who would most benefit from its development.

A public demonstration of *CodeSlinger* was given to the UMLS user community in 2008. The application was well received and many favorable comments and feedback were given which will be incorporated in future versions of the application. Also, after giving internal demonstrations, we identified a major initiative where we were able to assist in improving the level of concept coverage for a new internal medical conditions dictionary within GlaxoSmithKline.

7 Acknowledgements

The authors would like to thank Drs. Kathleen Beach, MD, MPH and Hoa Le, MD for help in defining the requirements for *CodeSlinger* and providing valuable feedback during its development.

References

1. Hasman, A: *Challenges for medical informatics in the 21st century.* International Journal of Medical Informatics. 44 (1997) 1-7
2. Cimino, James J: *Desiderata for Controlled Medical Vocabularies in the Twenty-First Century.* Methods Inf Med 1998; 37(4-5):394-403
3. ICD-9 refers to ICD-9, CM the International Classification of Diseases, 9th Revision, Clinical Modification
4. *MedDRA (Medical Dictionary for Regulatory Activities)* is a registered trademark of the International Federation of Pharmaceutical Manufacturers
5. UMLS Knowledge Source Server is a project of the (US) National Library of Medicine, Department of Health and Human Services. Available at: `http://umlsks.nlm.nih.gov/`
6. TermWorks is copyrighted by Apelon, Inc. Available at: `http://www.apelon.com/products/termworks.htm`
7. Kleiner, Merrill and Painter. *Inter-translation of Biomedical Coding Schemes Using UMLS.* Proceedings of the 2006 AAAI Fall Symposium on Semantic Technologies.
8. UMLS Metathesaurus is a project of the (US) National Library of Medicine, Department of Health and Human Services. Available at: `http://www.nlm.nih.org/research/umls/`
9. Adnan El-Nasan, Sriharsha Veeramachaneni, George Nagy: *Word Discrimination Based on Bigram Co-Occurrences*, icdar, p. **0149**, Sixth International Conference on Document Analysis and Recognition (ICDAR'01), 2001