IndexFinder: A Knowledge-based Method for Indexing Clinical Texts

Qinghua Zou, MS, Wesley W. Chu, PhD

Dept of Computer Science,
University of California,
Los Angeles, CA 90095

Extracting key concepts from clinical texts for indexing is an important task in implementing a medical digital library. Several methods are proposed in the literature for mapping free text into terms controlled by the Unified Medical Language System (UMLS). They are, however, not appropriate for building a fast online application. MatMap and other methods use natural language processing (NLP) techniques to map identified noun phrases into concepts. We present a new algorithm for efficiently generating all possible UMLS phrases in a text from which key concepts are identified by using syntactic and semantic filtering. We have implemented the algorithm as a web-based service that provides a search interface for researchers and computer programs. During preliminary manual examinations of the 456 concepts for 100 topic sentences, we noticed that our method has discovered 18 (4%) more phrases that are not obtained from one single noun phrase, and no improper combinations are in the results. Our empirical experiment shows that the algorithm is effective at discovering relevant UMLS concepts while achieving a throughput of 43K bytes text per second. The tool can extract key concepts from clinical texts for indexing.

INTRODUCTION

The Unified Medical Language System (UMLS) is a collection of over 100 medical knowledge sources [1]. The most recent edition of UMLS(2003AA ed.) contains 875,255 concepts and 2.14 million phrases [5] with lengths ranging from 1 to 100 words, and an average of 4.5 words. Our goal is to have a fast method to extract key UMLS concepts automatically from clinical texts for indexing.

A significant amount of research has aimed at developing effective methods for mapping free text into UMLS concepts. Examples of such efforts include *SENSE* [1], *MicroMeSH* [6], *Metaphrase* [7], *KnowledgeMap* [4], *PhraseX* [2], *MetaMap* [3]. Many of these efforts use natural language processing (NLP) techniques to parse passages of free text to generate noun phrases, which in turn are mapped into UMLS phrases. This approach achieves some success. They are, however, some shortcomings to this general technique:

First, some important concepts can never be discovered through the identification of noun phrases. Table 1 provides examples of texts that reveal the shortcomings of the use of noun phrases.

- Example 1: A word from the heading phrase with a word from the description phrase forms the key concept, *prostate hyperplasia* (*C0033577*).
- Example 2: A word from the subject and two words from the location phrase combine to form the key concept, *left lung mass* (*C0746117*).
- Example 3: Words from two sentences combine, forming the key concept, *left lung mass* (*C0746117*).

Example Text	
1	**Prostate**, right (biopsy) - fibromuscular and glandular **hyperplasia**
2	A small **mass** was found in the **left** hilum of the **lung**.
3	A large **mass** was identified. It is in the **left** side of the **lung**.

Table 1. Problems with mapping noun phrases individually.

Secondly, deploying NLP applications is nontrivial and typically requires significant computing resources. Most of the NLP systems work in an offline mode. NLP methods are not generally suitable for a real time web tool that maps large volumes of free text into UMLS concepts. Although MetaMap is implemented using Java, there are no web-based interfaces to map from arbitrary free text to UMLS concepts.

In this paper, we propose a novel approach to discover candidate phrases in a sentence or other text unit and then use syntactic and semantic filters specified by the user to filter out irrelevant conceptual terms. The approach avoids using the expensive NLP process. An empirical analysis shows that our algorithm can process text at a throughput of 43K bytes text per second, which is much faster than *NLP-based* approaches like *MetaMap*. Our system can extract conceptual terms from clinical texts and group them by their corresponding semantic types, which can be used for indexing in medical digital libraries.

BACKGROUND

Mapping of UMLS Concepts We first postulate UMLS concepts in the form shown in table 2, which can be calculated from UMLS normalized string table *MRXNS.ENG*. Here, each row contains a phrase that is given as a set of words where the ordering is overlooked. For any text of *m* distinct words, e.g. *T={A,B,C,D}*, the mapping problem is to find all the phrases in the table that are subsets of the text *T*.

pno	Set of words	#word	concept
1	A	1	C1
2	B	1	C2
3	A, B	2	C3
4	C	1	C4
5	B, C, D	3	C5
6	A, D, E	3	C6
7	E, D	2	C7
8	A, C	2	C8

Table 2. Phrase table

For above *T*, for example, the satisfied phrases are {*A*}, {*B*}, {*A,B*}, {*C*}, {*B,C,D*}, and {*A,C*}. The two phrases {*A,D,E*} and {*E,D*} are not considered relevant since *E* is not in *T*.

Computation Complexity Since UMLS has 2.14 million phrases, Table 2 will contain 2.14 million rows. For a text of *m* distinct words, the mapping problem may be solved in two naïve ways:

First approach: For each *s* of the 2^m-1 non empty subsets of the text, we determine whether *s* is a phrase in the table. Let the average time for the testing of whether *s* is contained in the table be *a*, then this approach has an average time complexity of $O(a2^m)$.

Second approach: For each phrase of the *N* rows (2.14M) in the table, we determine whether the phrase is a subset of the text. Supposing the average time for testing whether a phrase in the table is contained in the text equals constant *b*, then the average time complexity will be $O(bN)$.

Clearly, when $m>20$, both approaches require millions of comparisons. Therefore they are not appropriate for designing a real time web application.

In the following sections, we propose an efficient mapping algorithm that is significantly faster than the above approaches.

METHODS

In this section, we will first present the basic searching idea of generating concept candidates from permuting the set of words in the input text. Then we show how to remove word inflections and how to add synonyms. Finally, we present several syntactic and semantic filters to remove the irrelevant conceptual terms introduced during the generation of phrase candidates.

The Mapping Algorithm

Data structures The Phrase table in Table 2 can be sorted according to the increasing number of words, as in Figure 1(a). We can then use it to populate the four indexing data structures as shown in Figure 1(b-e).

pid	words	#	cui
0	A	1	C1
1	B	1	C2
2	C	1	C4
3	A, B	2	C3
4	A, C	2	C8
5	E, D	2	C7
6	B, C, D	3	C5
7	A, D, E	3	C6

a. Phrase table

word→wid
A→ 0
B→ 1
C→ 2
D→ 3
E→ 4

b. wordHt

wid	pids
0	0, 3, 4, 7
1	1, 3, 6
2	2, 4, 6
3	5, 6, 7
4	5, 7

c.wid2pids

pid	cui
0	C1
1	C2
2	C4
3	C3
4	C8
5	C7
6	C5
7	C6

d. cuis

len	upPid
0	0
1	3
2	6
3	8

e. pLen

Figure 1. Indexing structures for matching text to UMLS concepts

- ***wordHt***: a hash table mapping a word to a unique identifier *wid*, as in Figure 1(b). In the UMLS 2003AA, there are 431,200 distinct words. Suppose that the average characters per word are 10, and *wid* is a 4 byte integer. The total size for the *wordHt* is about 6Mega bytes.
- ***wid2pids***: an array mapping *wid* to a list of phrase identifiers *pids*. It is an inverted index indicating the phrase list where a word occurs, as in Figure 1(c). The average number of phrases per word in the table is 21.3. Therefore the data size for this table is 431200*21.3*4=36.7M.
- ***cuis***: an array that maps *pid* to UMLS *cui*, as in Figure 1(d). Since the total number of phrases in UMLS is 2.14M, this array size is 2.14M* 4= 8.6M bytes.
- ***pLen***: an array indicating the upper bond for a given phrase length, as in Figure 1(e). For example, the *pid* for phrases of length 1 will be less than 3. Using this table, we are able to tell the length for each *pid*. Since the maximal phrase length is 100, the memory size for this array is about 400 bytes.

Since the total memory for the above four data sets is less than 50M bytes, they can reside in the main memory.

Searching reduces to a counting process Given the above indexing data structures, mapping a text *T* into concepts becomes a simple counting process, as shown in Figure 2. It first adds each word into its occurring phrase's queues, then we output those phrases that reach the expect number of words. More specifically, at Line 1, the text *T* is tokenized into a list of words, which are transformed into lowercase. Repeating words are dropped, as in Line 2. The unique words in *wl* are mapped into *wids* through the hash table *wordHt*, as in Line 3. At Lines 4 and 5, we use hash table *countHt* to collect information for phrases and their word lists. If the number of words for a phrase is less than the expected length, some word of the phrase is absent, and the phrase will be removed, as in Line 6. Finally at Line 7, phrase identities are mapped into concepts, and we output the results.

```
//input: text T
//output: list of cui & phrase
1. list of words of T → wl
2. to low case & remove
   repeating words in wl.
3. words in wl →wids
4. foreach wids, get pids
5.   foreach pids, add countHt
6. remove if |words|<exp(pid)
7. replace pid to cui, output
```

Figure 2. Algorithm

pid→words
0 → {A}
1 → {B}
2 → {C}
3 → {A, B}
4 → {A, C}
5 → {D}
6 → {B, C, D}
7 → {A, D}

(a) countHt

word:	wid:	pids
A:	*0:*	**0, 3, 4, 7**
B:	*1:*	**1, 3, 6**
C:	*2:*	**2, 4, 6**
D:	*3:*	**5, 6, 7**

(b) pids

Hashing and counting

Figure 3. Counting for each *pid.*

For example, Figure 3 shows the counting process for the input text *T*={*A,B,C,D*}. In Figure 3(b), a word is mapped into *wid* and *pids*. For instance, word A has *wid*=0 and *pids*={*0,3,4,7*} meaning that A occurs in four phrases. We then add A to the four *pid* word lists in hash table *countHt*, as in Figure 3(a). Figure 3(a) shows the results after processing all the words in *T*. Then *pid* 5 and 7 are removed since their lengths are 1 and 2 less than the expected lengths 2 and 3, respectively.

For an input text of *m* distinct words, since the average length of the phrase list of a word is 21.3, we need to perform *21.3*m* operations at Line 4-5 in Figure 3. Therefore the average time complexity will be O(21.3*m*) which is significantly lower than the complexity of the naive approaches.

Word Normalization

Since we build indexing data using the normalized string table *MRXNS.ENG* where word inflections are removed, word inflections will not be contained in the table *wordHt* as in Figure 1(b). For example, we will not find the concept *C0043209* for *women* unless we normalize it to *woman*.

We use two data structures to store special word inflections, as in Figure 4. Hash table *specialHt* maps a special word to a base. For example, *children* maps to *child* by *bid*=0; *arose* and *arisen* map to *arise*.

word →bid	
children	→ 0
women	→ 1
men	→ 2
brought	→ 3
arose	→ 4
arisen	→ 4
…	

(a) specialHt

bid	base
0	child
1	woman
2	man
3	bring
4	arise
…	

(b) base

Figure 4. Data structure for removing inflection.

When a word has no entry in *wordHt* in line 3 of Figure 2, normalization starts. It follows these two steps:

1) If removing regular inflection, the word has an entry in *wordHt*, then returns the entry; otherwise, it continues to step 2.
2) If the word has an entry in *specialHt*, then return the corresponding base word; otherwise, the word is overlooked

Adding Synonyms

Synonym mapping is useful since people do not usually know the exact terminologies if they are not in the field. For example, a patient query may consist of no medical terms unless synonyms are considered. UMLS provides a synonym list [8]; two data structures *synHt* and *gid2wids* are used for adding synonyms, as shown in Figure 6. The table *synHt* maps a word to a synonym group. For example, the words *eye*, *optic*, and *oclar* are synonyms since they have the same group identifier 0. The *gid2wids* is an array that maps a synonym group to its member words. Given *synHt* and *gid2wids*, it is easy to find the list of synonyms for a given word.

word: **wid→gid**
eye: 21 → 0
optic:34 → 0
oclar:67 → 0
…

(a) synHt

gid	wids
0	21,34,67
…	

(b) gid2wids

Figure 5. Data structure for adding synonyms.

Filtering

Since applications usually have a certain focus, filtering out the results that people are not interested in is very useful. For example, a doctor wants to know what kind of diseases a patient suffers. Rather than returning all concepts to the doctor, several disease-related UMLS phrases are much more desirable. We consider six types of filters as shown in Figure 6.

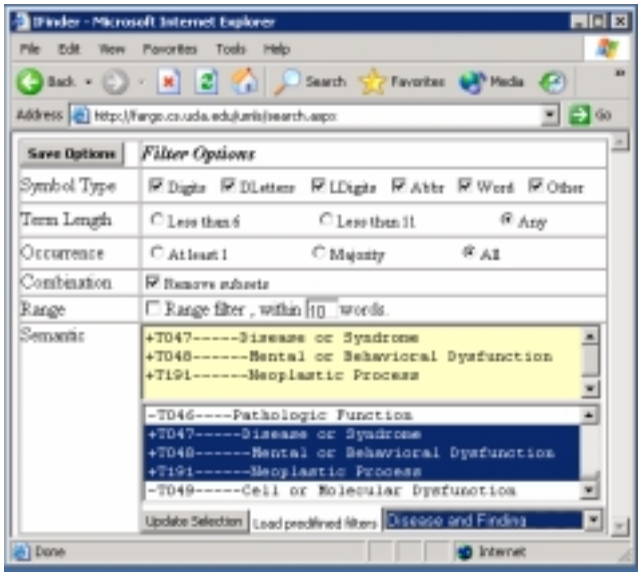

Figure 6. Configuration options.

The first three filters are applied during the mapping process. They are:

- *Symbol Type filter*: to specify the symbol types of interests. For example, if a user wants to ignore digits like *MetaMap* did, he can simply not check the Digits box as in Figure 6.
- *Term Length filter*: to specify the length limitation of candidate phrases.
- *Coverage filter*: to specify the coverage condition for a candidate phrase. It has three options, *at least one*, *majority*, and *all*. By default, it is *all*

where every word in a candidate phrase should be present in the input text.

The latter three filters are used for further pruning the candidate phrases.

- *Subset filter*: to remove phrases if they are subsets of some other phrases. For example, if results are {*lung cancer*} and {*cancer*}, then {*cancer*} will be removed since it is a subset of the former.
- *Range filter*: to remove a phrase if the phrase is found from words in the input text to exceed a specific distance. We can set the filter within one sentence.
- *Semantic filter*: to remove the phrases of semantic types that the user is not interested in. In UMLS, 134 semantic types are defined and each concept maps to one or several semantic types. For example, the user can select *Disease or Syndrome* and its two sub types, as shown in Figure 6, so that the resulting phrases will be of these three types. As a result, the filter also eliminates those irrelevant phrases from the set of phrase candidates. Note that UMLS ISA relationship may also be used to filter out more general phrases.

RESULTS

We have implemented the algorithm as a web-based service named *IndexFinder* that provides web interfaces for users and programs at the following links respectively:

- *http://fargo.cs.ucla.edu/umls/search.aspx*
- *http://fargo.cs.ucla.edu/umls/service.asmx*

Our experiment shows that *IndexFinder* can process 43K bytes text per second. Our preliminary manual examination shows no irrelevant phrases were returned. *IndexFinder*, written in *C#*, is running at the above sites on a 1.2GHz PC machine with 512MB main memory.

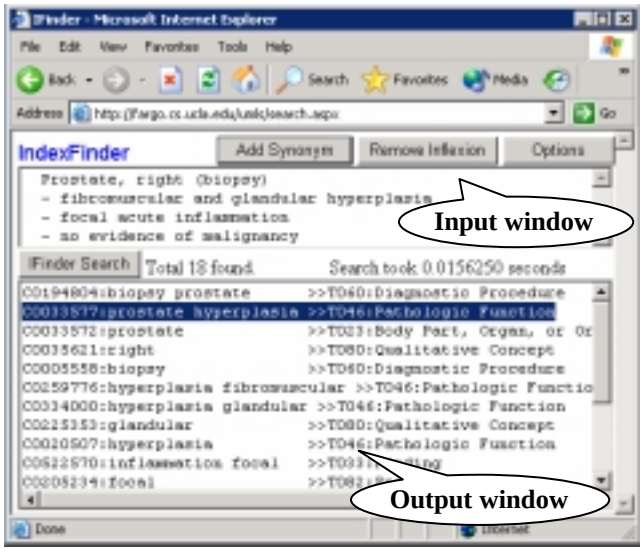

Figure 7. IndexFinder web interface.

Example Figure 7 shows the web interface for users. There are two windows on the web page, one for input text and the other for output results. Three buttons for adding synonyms, removing inflection, and configuring options are at the top of the input window. When a user clicks the *IFinder Search* button below the input window, results will show up. Figure 8 shows 18 phrases found when no filters were applied. Each line has a UMLS concept identifier, phrase text, and corresponding semantic type.

Filtering Figure 8 shows filtering result for the sample input in Figure 7, also shown in the top of Figure 8. When a *subset filter* is used, 8 phrases are returned.

Input of text unit	
Prostate, right (biopsy) - fibromuscular and glandular hyperplasia - focal acute inflammation - no evidence of malignancy	
Filt	Figure 7. Concept finder web interface.
Subset	C0194804:biopsy prostate C0033577:prostate hyperplasia C0035621:right C0259776:hyperplasia fibromuscular C0334000:hyperplasia glandular C0522570:inflammation focal C0333361:inflammation acute C0391857:no malignancy evidence
Pathologic Function (T046)	C0033577:prostate hyperplasia C0259776:hyperplasia fibromuscular C0334000:hyperplasia glandular C0333361:inflammation acute
Body parts & Spatial (T023, T082)	C0033572:prostate C0205234:focal
Diagnostic Procedure (T060)	C0194804:biopsy prostate

Figure 8. Result filtering.

If *Pathologic Function* is selected, four answers will be returned. The two phrases *prostate* and *focal* will be given if the user wants to know body parts or spatial characteristics. There is only one diagnostic procedure used, which is *prostate biopsy*.

Throughput Experiment We tested the web service using 5,783 reports of 128 patients from the UCLA Hospital. The total size of the documents is 10,8M bytes. The service was called for each document when no filter was applied. There are 910K concepts found in 254 seconds. Therefore, the throughput is about 42.7 K bytes per second.

Relevance Evaluation We manually examined the mapping results for 100 topic sentences from the above reports one at a time. A phrase is considered relevant to a sentence if the sentence implies the phrase when ignoring negation. For example, both "*evidence*" and "*no evidence*" are relevant to the input "*no evidence of malignancy*" since negation is ignored. There are total 456 UMLS phrases found for the 100 topic sentences. We noticed that 18 concepts are not from a single noun phrase and there are no irrelevant phrases.

APPLICATION

As a specific clinical application for this research, we have focused on using the *IndexFinder* to intelligently identify the key terms in an electronic medical record for documents that specifically mention brain tumor related content. These documents consist of primary care clinical notes, specialist clinical notes, pathology reports, laboratory results, radiology reports, and surgical notes. Figure 9 shows an excerpt from a radiology report.

> *"The right frontal convexity meningioma is slightly larger now than on the prior examination. The left frontal meningioma is unchanged. There are three other small enhancing nodules seen along the frontal convexities bilaterally, as described above. There are no new lesions seen. There is no mass effect caused by these lesions. There is bifrontal encephalomalacia."*

Figure 9. Free-text excerpt from a radiology report.

Since our interests focus on brain tumor related concepts, we can specify a semantic filter worklist of pertinent documents based on brain tumor characteristics including: cancer type, anatomical location, and medical interventions. These characteristics are then mapped to relevant UMLS semantic types as shown in Table 3 to define semantic filters.

Brain Tumor Characteristics	Relevant UMLS semantic types
Specific Cancer	Neoplastic Proccess
Medical Intervention	Therapeutic Procedure
Anatomical location	Body Part, Organ or Organ Component

Table 3. Using UMLS semantic type to define interests.

A clinician looking for specific documents that address a certain type of brain tumor (i.e. *meningioma*) would have to carefully search the individual documents. With *IndexFinder*, only two key terms, *meningioma* and *encephalomalacia*, are returned for the above text excerpt as shown in Table 4. The two concepts, in fact, are important in the excerpt and thus are good terms for indexing.

Semantic Descriptor	ULMS code
T191:Neoplastic Process	C0025286:meningioma
T047:Disease or Syndrome	C0014068:encephalomalacia

Table 4. Output from IndexFinder for the text in Figure 9.

CONCLUSION

In this paper, we proposed a new efficient approach for generating the conceptual phrase candidates in clinical texts and identifying important phrases for indexing. The method uses a set of data structures to reduce the problem of searching UMLS concepts to a simple counting process. Syntatic and semantic filters are used to eliminate the irrelevant candidates. Our experiment shows that it can process free text at a speed of about 43K bytes text per second. As a result, *IndexFinder* is able to extract key *UMLS* concepts from clinical texts in real time. Preliminary manual evaluation shows that no irrelevant concepts are returned. *IndexFinder* can be used for indexing of clinical texts for a medical digital library.

ACKNOWLEDGEMENTS

This research is supported by NIC/NIH Grant #4442511-33780.

REFERENCES

1. Yuri L. Zieman and Howard L. Bleich. Conceptual Mapping of User's Queries to Medical Subject Headings. Proc AMIA 1997.
2. Suresh Srinivasan, Thomas C. Rindflesch, William T. Hole, Alan R. Aronson, and James G. Mork. Finding UMLS Metathesaurus Concepts in MEDLINE. Proc AMIA 2002.
3. Alan R. Aronson, Effective Mapping of Biomedical Text to the UMLS Metathesaurus: The MetaMap Program. Proc AMIA 2001.
4. Joshua C. Denny, Jeffrey D. Smithers, Anderson Spickard, III, Randolph A. Miller. A New Tool to Identify Key Biomedical Concepts in Text Documents. Proc AMIA 2002.
5. National Library of Medicine. Documentation, UMLS Knowledge Sources, 14 th Edition, January 2003.
6. Elkin PL, Cimino JJ, Lowe HJ, Aronow DB, Payne TH, Pincetl PS and Barnett GO. Mapping to MeSH: The art of trapping MeSH equivalence from within narrative text. Proc 12th SCAMC, 185-190, 1988.
7. Tuttle MS, Olson NE, Keck KD, Cole WG, Erlbaum MS, Sherertz DD et al. Metaphrase: an aid to the clinical conceptualization and formalization of patient problems in healthcare enterprises. Methods Inf Med. 1998 Nov;37(4-5):373-83.
8. Hole W. T, Srinivasan S. Discovering Missed Synonymy in a Large Concept-Oriented Metathesaurus. Proc AMIA Symp 2000:354-358
9. Morioka CA, El-Saden S, Duckwiler, G. et al, Workflow Management of HIS/RIS Textual Documents with PACS Image Studies for Neuroradiology, Proc AMIA Symp 2003 (submitted for publication).