The SAPHIRE Server: A New Algorithm and Implementation

William Hersh, M.D. T.J. Leone, M.S.
Biomedical Information Communication Center
Oregon Health Sciences University
Portland, OR USA

SAPHIRE is an experimental information retrieval system implemented to test new approaches to automated indexing and retrieval of medical documents. Due to limitations in its original concept-matching algorithm, a modified algorithm has been implemented which allows greater flexibility in partial matching and different word order within concepts. With the concomitant growth in client-server applications and the Internet in general, the new algorithm has been implemented as a server that can be accessed via other applications on the Internet.

Introduction

The SAPHIRE project was undertaken with the goal of improved document retrieval in the medical domain. It was initially implemented in the single-user Macintosh environment [1, 2] and has been evaluated extensively [3, 4]. The results of these experiments show that while the concept-matching algorithm used for automated indexing does not confer benefit over single-word automated indexing, it can be useful for assisting users in mapping the free text of queries into terms from controlled vocabularies, such as the UMLS Metathesaurus [5].

The problems in current information retrieval (IR) systems that motivated SAPHIRE include the inconsistency of human indexers [6], the underlying inconsistency of human language used by authors [7], and the difficulty end-users often have with Boolean operators used in most IR systems [8]. SAPHIRE attempted to address these problems by basing its indexing on a concept-matching approach, such that different string forms of a concept (also called terms) would map to the same underlying concept [2]. Using a vocabulary such as the UMLS Metathesaurus [9], with its breadth of concepts as well as depth of synonyms, would allow many terms of a concept to be recognized for most medical concepts.

This concept-matching approach was used for both indexing and retrieval, with the goal of matching the diverse expressions of concepts in documents and user queries. For indexing, documents were processed by SAPHIRE to identify their underlying concepts, which were weighted by the IDF*TF formula that gave the highest weight to terms occurring frequently in a small number of documents [10]. In retrieval, the user query was processed to obtain their concepts, which were then matched against the indexing concepts in the documents to obtain a weighted list of matching documents.

The original SAPHIRE concept-matching algorithm used a strict pattern-matching approach requiring not only that all words in a matching term be present but also that they occur in the word order of the term in the vocabulary. The motivation for this high-precision approach was to avoid false-positive matching. This resulted, however, in missing some true-positive matches, which was shown in failure analyses to cause nonretrieval of relevant documents [3].

We have subsequently developed a new algorithm to allow partial matching of concepts and not require exact word order. This approach has been used by others [11, 12]. One limitation is that it necessarily results in an increased amount of false-positive matching, which we hope to control with a weighting scheme that would give highest weight to those concepts matching the input document or query most closely. The indexing algorithm or user performing a query could then decide upon the trade-off between false-positive and false-negative matching, based on how far down the weighted list of concepts they chose to go to include terms.

At the time that the new algorithm was being implemented, the growth of client-server applications, especially those running on the Internet, was increasing. The goal of SAPHIRE has always been to interact with other information systems, such as the DeSyGNER environment from the Decision Systems Group (DSG) at Brigham and Women's Hospital (BWH) [13]. By separating the concept-matching algorithm from the IR system, many other applications could utilize the algorithm itself. Thus it was decided to reimplement it in a client-server architecture that would allow any application running on the Internet to have access to it. In the remainder of this paper, we discuss the algorithm itself and its implementation on the Internet.

0195–4210/95/$5.00 © 1995 AMIA, Inc.

Algorithm

As noted above, the goal of the new SAPHIRE concept-matching algorithm was to allow partial matching of concepts and not require the exact word order as it occurs in the Metathesaurus vocabulary. As with the original algorithm, the UMLS Metathesaurus was used, due to its comprehensive list of concepts and their synonyms. The new algorithm also took advantage of a new feature that began with the 1992 version of the Metathesaurus, which was an inverted list of all words that occurred in each concept.

Before describing the specifics of the algorithm, some definitions of potentially ambiguous terminology are helpful. The Metathesaurus is organized into *concepts*, which have a unique identifier (the CUI). Each major synonym form that is not just a simple lexical variant (i.e., plural or word order change) is a *term*, each of which also has a unique identifier (the LUI). There can be one or more LUI's for each CUI. Each lexical variant of each term is a *string* (with a unique identifier SUI), and there can be more than one SUI for each LUI. As an example, consider the concept *atrial fibrillation*, which has terms *atrial fibrillation* and *auricular fibrillation.* The former term has the lexical variants *fibrillation, atrial* and *atrial fibrillations*.

The MRCON file from the UMLS CD-ROM, which is the central table of the Metathesaurus, contains a row for each string along with its SUI, LUI, and CUI. (Each CUI has a canonical or preferred string; for SAPHIRE's purposes the remaining strings are synonyms.) The MRWD file, which is the inverted word list table, contains a row for each word that occurs in an SUI/LUI/CUI triple. The SAPHIRE algorithm uses these tables intact, but adds some additional files to allow their rapid access. In particular, B-Tree files are added that allow quick look-up of words, LUI's, and CUI's.

The algorithm begins by breaking the input string (which can be a sentence or phrase from a document or a user's query) into individual words. Words are designated as *common* if they occur with a frequency above a specified cut-off in the Metathesaurus. The purpose of designating words as common is to reduce the computational overload for words which are occasionally important in some terms but occur frequently in others, such as the word *A* in *Vitamin A* or *acute* in *acute abdomen*. Since the words *A* and *acute* occur commonly in many other terms, calculating weights for these additional terms adds a large and unnecessary computational burden.

For each word in the input string, a list of Metathesaurus terms in which the word occurs is constructed. The Metathesaurus term lists for common words contain only those terms that also occur in one or more of the non-common words in the input string. Using one of the above examples, if the string were *acute abdomen*, the common word *acute* would only contain the term *acute abdomen* and not the term *acute leukemia.*

Once the term lists for each word are created, a master term list is created that contains any term which occurs in one or more individual word lists. Terms in which less than half of the words occur in the input string are discarded. (Thus, a partial match must have half or more of the words from the term in the input string.) The terms are then weighted based on formula that gives weight to terms that are longest, have the highest proportion of words from the term in the string, and have the words of the term occurring in close proximity to each other.

The algorithm has a number of switches that allow modification of its parameters. The common word cut-off is the Metathesaurus frequency threshold to designate terms as common. It can be set at any level but the default value is 270, which means that any word which occurs more than 270 times in all of the Metathesaurus strings is designated as common. This number was set based upon empirical observation of the algorithm's behavior and results in the most frequent 10% of words in the Metathesaurus being designated as common.

The other switches affect the format of the list of matched terms. The CUI switch causes output to be listed by CUI instead of LUI. (All LUI's for a given CUI are merged, with the weight for the CUI becoming the weight of its highest LUI.) Two other switches control the size of the output list, with one a cut-off for the number of terms displayed and the other a cut-off for the lowest weight allowed. A final switch allows a specific source vocabulary to be set (i.e., MeSH or DxPlain), in which all terms not in that vocabulary are discarded from the returned list.

Figure 1 depicts the actual algorithm in pseudocode. Figure 2 lists a number of example strings input to the algorithm. As can be seen, a number of concepts from the input string that are incomplete (i.e., *calcium blockers*) or have altered word order (i.e., *blood pressure is high*) are still properly recognized.

```
function saphire-concept-matching-algorithm

Input:  string of words
        common word cut-off  (default = 270)
        CUI  (default = false)
        size cut-off  (default = 40)
        weight cut-off  (default = 1.0)

Output:  list of Meta terms

for each word in string
  word_frequency = number of Metathesaurus concepts that word occurs in
  if word_frequency is greater than the common word cut-off then
    word is common
  else
    create a list of all Meta terms that contain this word
for each word in string that is common
  create a list of all Meta terms that contain this word alone or
       that contain this word and at least one word that is not common
create a master list of all Meta terms that occur in any of the words
for all Meta terms in the master list
  twis = number of words from this term in string
  wit = number of words in term
  if (wit + 1) div 2 < twis then discard this term
for all Meta terms in master list
  term_words_present = twis / wit
  log_term_length = log (wit) + 1
  intervening_words = number of words between first and last words of term
       in string
  log_intervening_words = log (intervening_words + 1) + 1
  weight = term_words_present * log_term_length / log_intervening_words
sort Meta terms in master list
if CUI is true then
  eliminate all LUI's which have a common CUI with a higher-ranked LUI
eliminate all terms below size cut-off
eliminate all terms below weight cut-off
```

Figure 1 -- Pseudocode for the new concept-matching algorithm

Implementation

The server currently runs under Unix on a Sun Sparc 20, although nothing about the implementation is Sun- or Unix-specific. It is implemented in C, using a B-Tree package (Faircom Corp.) for data management routines. The network communication protocol is TCP/IP, so that any machine on the Internet can access the server. The algorithm processes a sentence to a ranked list of Metathesaurus terms in 3-6 seconds. Any speed bottlenecks are typically caused by network communication slowdowns.

A variety of clients have been created to access the server. Clients written in C and run from the command-line have been created for Unix and the Apple Macintosh. A colleague was able to quickly implement a client running under Macintosh Common Lisp with less than a page of code (personal communication, Douglas Bell). Finally, a Common Gateway Interface (CGI) has been implemented to allow access from a forms page on the World Wide Web (WWW) with URL http://www.ohsu.edu/post-saphire.html. Other applications that want to utilize SAPHIRE's algorithm can embed the calls to the server directly in their code.

```
calcium blockers -

L0006684 1.399075 Calcium Channel Blockers
L0006675 1.000000 Calcium

blood pressure is high -

L0005827 2.098612 Blood Pressure, High
L0005823 1.693147 Blood Pressure <1>
L0210934 1.693147 Blood Pressure <2>
L0005824 1.399075 Blood Pressure Determination
L0005826 1.399075 Blood pressure, abnormal

congestive failure -

L0018802 1.399075 Heart Failure, Congestive
L0182446 1.193147 Rheumatic heart failure (congestive)

vitamin a -

L0042839 1.693147 Vitamin A
L0042841 1.399075 Vitamin A Aldehyde
L0042840 1.399075 Vitamin A Acid
L0042842 1.399075 Vitamin A Deficiency
L0051906 1.399075 anhydrovitamin A(2)

aortic stenosis -

L0003498 1.693147 Aortic Stenosis
L0003499 1.399075 Aortic Stenosis, Supravalvular
L0003500 1.399075 Aortic Subvalvular Stenosis
L0003507 1.399075 Aortic Valve Stenosis
L0182431 1.399075 Rheumatic aortic stenosis
```

Figure 2 -- Sample output from the server, with the matching LUI, weight, and string respectively. The weight is calculated from the algorithm in Figure 1. The output is truncated at five terms.

Strings and commands are sent to the server in ASCII format. Any command not preceded by a backslash ('\') is assumed to be a text string for concept-matching. Commands are entered by using a backslash followed by the letter of the command, and arguments. The commands include:

- \c common word frequency cut-off (default = 270)
  Example: \c500
- \t set CUI/LUI (default = LUI)
  Example: \tcui
- \v set source vocabulary (default = ALL, otherwise use designations from UMLS documentation)
  Example: \vMSH93
- \x cut-off term weight (default = 1.0)
  Example: \x0.85

A final command allows the display of the frequency for each word in a string, with an asterisk identifying whether or not it is designated as a common word:

- \f display frequency of each word
  Example: \facute leukemia
  Returns:
  517 acute*
  232 leukemia

Future Plans

The future plans for the SAPHIRE server include interfacing with other IR systems, different applications, and UMLS resources. The first undertaking will be to connect the server with back ends of other IR systems. We are currently porting the rest of the SAPHIRE search engine to run in client-server mode on the Internet, and will also be able to interact with other search engines, such as Salton's SMART system [14]. Before an operational IR system is created, some questions remain to be answered about the optimal approach for using the new SAPHIRE concept-matching algorithm for automated indexing. For example, optimal settings will need to be determined for the weight and common word cut-offs. It will be particularly challenging to determine the former, as we aim to maximize appropriate partial matches without including inappropriate ones. This task will be done using a large MEDLINE test collection that we implemented last year [15].

The next undertaking will be to integrate the SAPHIRE server with other applications, such as DeSyGNER [13]. Virtually any application running on the Internet can use the server. Our final task will be to interface with other UMLS resources on the Internet, in particular the UMLS Knowledge Sources server being developed at the National Library of Medicine (NLM) [16]. Since the SAPHIRE server does not store any of the other information present in the Metathesaurus, such as attributes and occurrence data, the NLM server will allow applications to obtain that information after doing a SAPHIRE concept-match from our server.

Client software availability

The C source code for the SAPHIRE client can be obtained via anonymous FTP from medir.ohsu.edu in the directory pub/saphire. The OHSUMED test collection can be obtained from the pub/ohsumed directory.

Acknowledgments

This work was supported by Grant LM05307 and Contract N01-LM13539 from the National Library of Medicine.

References

1. Hersh WR, Greenes RA. SAPHIRE: An information retrieval environment featuring concept-matching, automatic indexing, and probabilistic retrieval. Computers and Biomedical Research 1990;23:405-420.
2. Hersh WR. Evaluation of Meta-1 for a concept-based approach to the automated indexing and retrieval of bibliographic and full-text databases. Medical Decision Making 1991;11:S120-S124.
3. Hersh WR, Hickam DH, Haynes RB, McKibbon KA. A performance and failure analysis of SAPHIRE with a MEDLINE test collection. Journal of the American Medical Informatics Association 1994;1:51-60.
4. Hersh WR, Hickam DH. Information retrieval in medicine: the SAPHIRE experience. In: Greenes R, ed. MEDINFO 95. Vancouver, BC: Elsevier Science, 1995:in press.
5. Hersh WR, Hickam DH, Leone TJ. Word, concepts, or both: optimal indexing units for automated information retrieval. In: Frisse M, ed. Proceedings of the 16th Annual Symposium on Computer Applications in Medical Care. Baltimore: McGraw-Hill, 1992:644-648.
6. Funk ME, Reid CA. Indexing consistency in MEDLINE. Bulletin of the Medical Library Association 1983;71:176-183.
7. Evans DA. Pragmatically-structured, lexical-semantic knowledge bases for unified medical language systems. In: Greenes R, ed. Proceedings of the 12th Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: IEEE, 1988:169-173.
8. Borgman CL. Why are online catalogs hard to use? Lessons learned from information retrieval studies. Journal of the American Society for Information Science 1986;37:387-400.
9. Lindberg DAB, Humphreys BL, McCray AT. The unified medical language system project. Methods of Information in Medicine 1993;32:281-291.
10. Sparck Jones K. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 1972;28:11-21.
11. Lin R, Lenert L, Middleton B, Shiffman S. A free-text processing system to capture physical findings: canonical phrase identification system (CAPIS). In Clayton PD, ed. Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 1991: 843-847.
12. Vries JK, Marshalek B, D'Abarno JC, Yount RJ, Dunner LL. An automated indexing system utilizing semantic net expansion. Computers and Biomedical Research. 1992; 153-167.
13. Greenes RA, Deibel SRA. The DeSyGNER knowledge management architecture: A building block approach based on an extensible kernel. Artificial Intelligence in Medicine 1991;3:95-111.
14. Salton G. Introduction to Modern Information Retrieval. New York: McGraw-Hill, 1983.
15. Hersh WR, Buckley C, Leone TJ, Hickam DH. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In: Croft W, vanRijsbergen C, eds. Proceedings of the 17th Annual International ACM Special Interest Group in Information Retrieval, 1994:192-201.
16. McCray AT, Razi A. The UMLS Knowledge Source Server. In: Greenes R, ed. MEDINFO 95. Vancouver, BC: Elsevier Science, 1995: 144-147.