BMC Medical Informatics and Decision Making

BioMed Central

Software

Resources for comparing the speed and performance of medical autocoders

Jules J Berman*

Address: Cancer Diagnosis Program, National Cancer Institute, National Institutes of Health, Bethesda, MD, USA

Email: Jules J Berman* - bermanj@mail.nih.gov

* Corresponding author

Published: 15 June 2004

BMC Medical Informatics and Decision Making 2004, **4**:8

Received: 13 December 2003

Accepted: 15 June 2004

This article is available from: http://www.biomedcentral.com/1472-6947/4/8

Abstract

Background: Concept indexing is a popular method for characterizing medical text, and is one of the most important early steps in many data mining efforts. Concept indexing differs from simple word or phrase indexing because concepts are typically represented by a nomenclature code that binds a medical concept to all equivalent representations. A concept search on the term renal cell carcinoma would be expected to find occurrences of hypernephroma, and renal carcinoma (concept equivalents). The purpose of this study is to provide freely available resources to compare speed and performance among different autocoders. These tools consist of: 1) a public domain autocoder written in Perl (a free and open source programming language that installs on any operating system); 2) a nomenclature database derived from the unencumbered subset of the publicly available Unified Medical Language System; 3) a large corpus of autocoded output derived from a publicly available medical text.

Methods: A simple lexical autocoder was written that parses plain-text into a listing of all 1,2,3, and 4-word strings contained in text, assigning a nomenclature code for text strings that match terms in the nomenclature. The nomenclature used is the unencumbered subset of the 2003 Unified Medical Language System (UMLS). The unencumbered subset of UMLS was reduced to exclude homonymous one-word terms and proper names, resulting in a term/code data dictionary containing about a half million medical terms. The Online Mendelian Inheritance in Man (OMIM), a 92+ Megabyte publicly available medical opus, was used as sample medical text for the autocoder.

Results: The autocoding Perl script is remarkably short, consisting of just 38 command lines. The 92+ Megabyte OMIM file was completely autocoded in 869 seconds on a 2.4 GHz processor (less than 10 seconds per Megabyte of text). The autocoded output file (9,540,442 bytes) contains 367,963 coded terms from OMIM and is distributed with this manuscript.

Conclusions: A public domain Perl script is provided that can parse through plain-text files of any length, matching concepts against an external nomenclature. The script and associated files can be used freely to compare the speed and performance of autocoding software.

Background

As used in this manuscript, the term "autocoder" refers to a software program capable of parsing large collections of medical records (e.g. radiology reports, surgical pathology reports, autopsy reports, admission notes, discharge notes, operating room notes, medical administrative emails, memoranda, manuscripts, etc.) and capturing the medical concepts contained in the text [1-5].

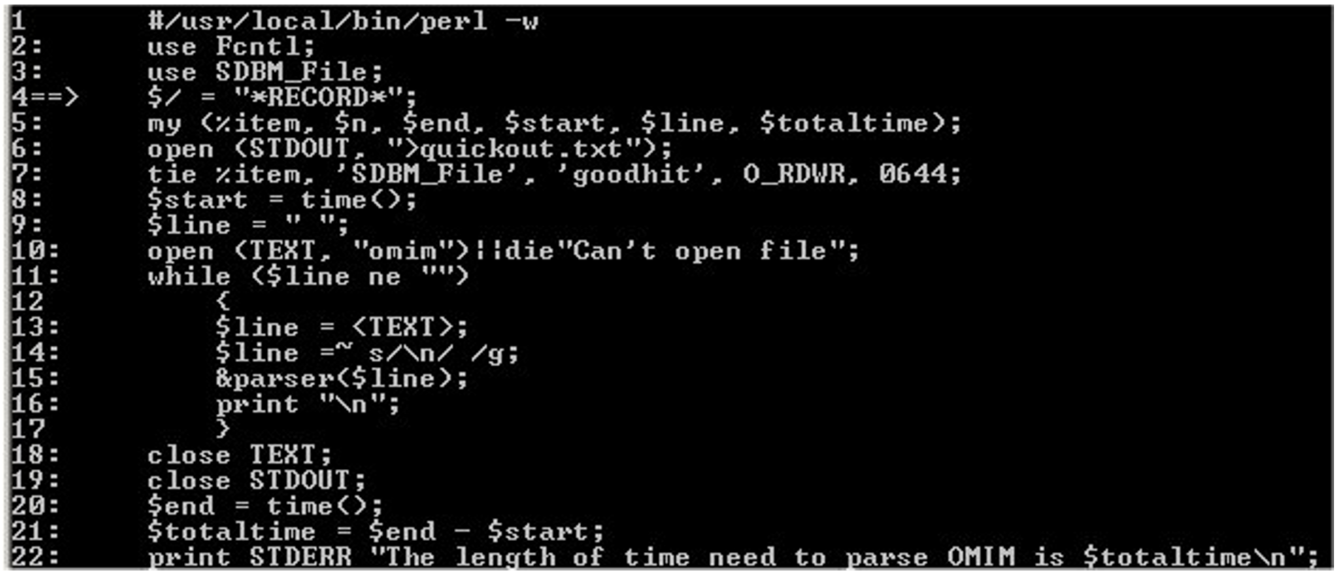
This article distinguishes "autocoding" from "computer-assisted manual coding." Pathologists typically use a software enhancement to Laboratory Information Systems to code the diagnostic line of their reports. Typically, candidate terms are displayed on the same screen alongside or within a single report, and the pathologist edits the proffered codes as he/she prefers. This process should not be confused with "autocoding" and is not equivalent to the fully automatic and large-scale coding required by data miners.

Medical autocoding can be considered a specialized form of machine translation (automated translation from one language into another). Lexical parsers are a simple but somewhat brutish approach to machine translation. The lexical parser depends on terms existing in medical text without internal modifiers. For instance, the term "flat feet" extracted from the first record in OMIM, would have been missed by the lexical parser if it had included an internal modifier, such as "flat erythemic feet." For this reason, much of machine translation work depends on the creation of elaborate grammar rule systems and exception lists that account for idiomatic language.

Finding all the concepts in a report is a necessary and early step in all data mining efforts. The autocoded terms can be used individually as index terms for the document, on a record-by-record basis to produce a concept "signature" that is highly specific for each report [6], or collectively to relate the frequency of terms within records with the frequency of terms in the aggregate document [7].

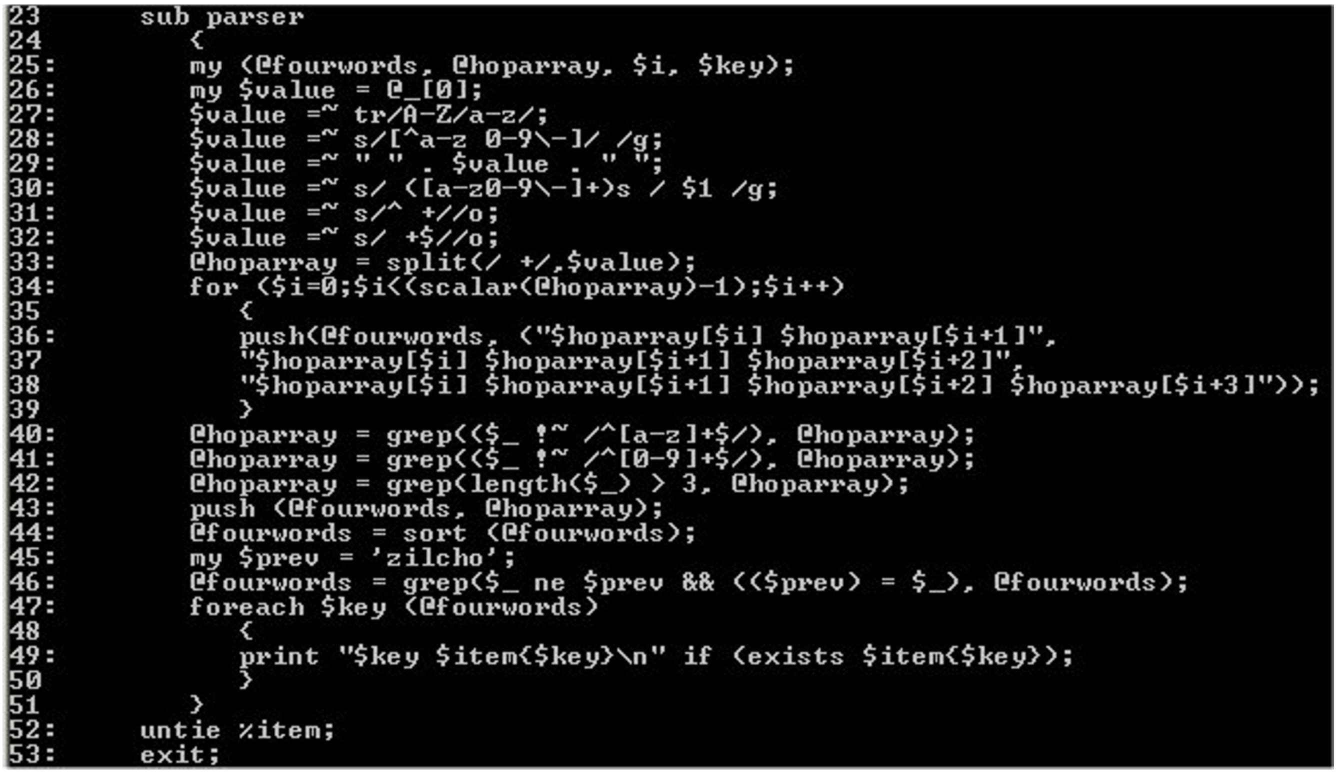
Implementation

The autocoding script, omim18.pl is included as a supplementary file with this article [see Additional file 1]. The script contains 38 command lines, comprising a main script [see Figure 1] plus a subroutine [see Figure 2]. The script is a simple lexical parser that moves record-by-record through a file. The main script opens an input file for parsing (Figure 1, line 10), an output file to receive lists of concepts extracted from the input file (Figure 1, line 6), and creates a hash data structure (also known as dictionary) of all the UMLS code/term pairs contained in an external database file, named goodhit (Figure 1, line 7). The subroutine creates an array of every possible 1,2,3 and 4- word term in the record and matches each of the terms from the array against the entire set of terms contained in the external UMLS dictionary (Figure 2, lines 33–46). When a term is matched, the term and its UMLS code are appended to the output file. The script was designed to be simple, short and fast. Perl programmers should have little or no problem modifying the script to accept their own files with records delimited by any character set or pattern.



```
1       #/usr/local/bin/perl -w
2:      use Fcntl;
3:      use SDBM_File;
4==>    $/ = "*RECORD*";
5:      my (%item, $n, $end, $start, $line, $totaltime);
6:      open (STDOUT, ">quickout.txt");
7:      tie %item, 'SDBM_File', 'goodhit', O_RDWR, 0644;
8:      $start = time();
9:      $line = " ";
10:     open (TEXT, "omim")||die"Can't open file";
11:     while ($line ne "")
12          {
13:         $line = <TEXT>;
14:         $line =~ s/\n/ /g;
15:         &parser($line);
16:         print "\n";
17          }
18:     close TEXT;
19:     close STDOUT;
20:     $end = time();
21:     $totaltime = $end - $start;
22:     print STDERR "The length of time need to parse OMIM is $totaltime\n";
```

Figure 1
omim18m.jpg The Perl autocoding script has two sections: a main section and a parsing subroutine. Omim18m.jpg displays the entire main section of the autocoding script.



```
sub parser
   {
   my (@fourwords, @hoparray, $i, $key);
   my $value = @_[0];
   $value =~ tr/A-Z/a-z/;
   $value =~ s/[^a-z 0-9\-]/ /g;
   $value =~ " " . $value . " ";
   $value =~ s/ ([a-z0-9\-]+)s / $1 /g;
   $value =~ s/^ +//o;
   $value =~ s/ +$//o;
   @hoparray = split(/ +/,$value);
   for ($i=0;$i<(scalar(@hoparray)-1);$i++)
      {
      push(@fourwords, ("$hoparray[$i] $hoparray[$i+1]",
      "$hoparray[$i] $hoparray[$i+1] $hoparray[$i+2]",
      "$hoparray[$i] $hoparray[$i+1] $hoparray[$i+2] $hoparray[$i+3]"));
      }
   @hoparray = grep(($_ !~ /^[a-z]+$/), @hoparray);
   @hoparray = grep(($_ !~ /^[0-9]+$/), @hoparray);
   @hoparray = grep(length($_) > 3, @hoparray);
   push (@fourwords, @hoparray);
   @fourwords = sort (@fourwords);
   my $prev = 'zilcho';
   @fourwords = grep($_ ne $prev && (($prev) = $_), @fourwords);
   foreach $key (@fourwords)
      {
      print "$key $item{$key}\n" if (exists $item{$key});
      }
   }
untie %item;
exit;
```

Figure 2
omim18s.jpg The Perl autocoding script has two sections: a main section and a parsing subroutine. Omim18s.jpg displays the entire parsing subroutine of the autocoding script.

The medical nomenclature used is the unencumbered subset of UMLS. Instructions for obtaining the unencumbered (so-called Category 0) subset of UMLS were described by the author in a prior publication [8]. Although the complete UMLS can be downloaded at no cost from the National Library of Medicine website, users must sign a License Agreement, available at: http://www.nlm.nih.gov/research/umls/license.html.

For this manuscript, a modification of the unencumbered subset of UMLS was created. This subset contains terms that are 1,2,3, or 4 words in length. Terms containing greater than four words were omitted. Also, many of the one-word terms were omitted. One-word terms found in the unencumbered subset of UMLS were preserved only if they consisted of combinations of both alphabetic and numeric characters (e.g., p53, cd-117, and ws-1358a1). Purely alphabetic one-word terms such as iris, cervix and Cushing were excluded. This was done in an effort to preserve the names of biomolecules and markers (which tend to contain numerics) and to reduce the number of homonyms and medical identifiers. The word "iris" can refer to the flower or to the eye structure. The word "cervix" can refer to the uterine cervix or to the neck (connecting head to trunk). The word "Cushing" may refer to the eponymous disease, or it may reveal the name of a patient. In general, specific medical terms consist of two or more words: "C0022078 disease of iris", "C0007847 cancer of cervix", "C0010481 Cushing syndrome." The exclusion of singletons helps achieve a non-ambiguous domain of concepts.

The value of excluding one-word terms is entirely dependent on the intended uses of the concept-index. The autocoding script will accept any nomenclature consisting of term/code pairs. There are virtually hundreds of standard nomenclatures available to medical text data miners and most contain one-word terms. Information related to individual nomenclatures contained in the UMLS metathesaurus are available at: http://www.nlm.nih.gov/research/umls/.

The medical corpus used is the Online Mendelian Inheritance in Man (OMIM). OMIM is a listing of every known inherited condition in man. Each condition has biologic and clinical descriptions in a detailed textual narrative that includes a listing of relevant citations. The OMIM text exceeds 90 MBytes in length and can be downloaded from the National Center for Bioinformatics anonymous ftp site: ftp://ftp.ncbi.nih.gov and subdirectory: /repository/omim/.

OMIM is an ideal and challenging corpus for testing indexing and retrieval algorithms because it contains free-text (paragraphs), structured text (lists), names (in free-text and in citations suitable for testing de-identification algorithms), gene-related terminology (names of genes, cytogenetic descriptors, proteins) and medical terms (co-morbid features of inherited diseases) and both common and obscure medical conditions. It has been used as a test corpus [9]. Additional information on OMIM is available at: http://www.ncbi.nlm.nih.gov/omim/.

The first paragraph of the first record (of over 15 thousand records) of OMIM is shown:

100050 AARSKOG SYNDROME

FIELD TX

Grier et al. (1983) reported father and 2 sons with typical Aarskog syndrome, including short stature, hypertelorism, and shawl scrotum. They tabulated the findings in 82 previous cases. X-linked recessive inheritance has been repeatedly suggested (see 305400). The family reported by Welch (1974) had affected males in 3 consecutive generations. Thus, there is either genetic heterogeneity or this is an autosomal dominant with strong sex-influence and possibly ascertainment bias resulting from use of the shawl scrotum as a main criterion. Stretchable skin was present in the cases of Grier et al. (1983). Teebi et al. (1993) reported the case of an affected mother and 4 sons (including a pair of monozygotic twins) by 2 different husbands. They suggested that the manifestations were as severe in the mother as in the sons and that this suggested autosomal dominant inheritance. Actually, the mother seemed less severely affected, compatible with X-linked inheritance.

Readers may download a copy of OMIM that can be used to replicate this test on their own systems. A "frozen" OMIM file can be used to compare the speed of the provided autocoding script against other programs. In addition, the OMIM file can be used to compare the ouput results when different nomenclatures are used with the same autocoding script (e.g. to test the nomenclature).

Results

Input file

The autocoder script is intended to work with any plain-text file on any operating system. Line four of the main section of the Perl script [see Figure 1] contains the command: $/ = "*RECORD*"; This command sets the input record separator ($/) to *RECORD*, OMIM's indicator for the beginning of a new record. Had this line been excluded, or had a # preceded the line, the autocoding Perl script would have parsed the input file using the default newline record separator. The resulting output file would have produced a concept index for each line in OMIM, rather than for each record in OMIM. When using the autocoding Perl script, it is important to assess the plain-text file to determine if a consistent delimiter separates records, and to substitute the file's delimiter into line 4. In the absence of any particular delimiter, line 4 can be deleted.

Speed

The length of time to autocode OMIM is 1,236 seconds on a 1.6 GHz processor or 823 seconds on a 2.4 GHz processor. Since OMIM is approximately 92 Megabytes in length, this provides an autocoding speed of 1 Megabyte in less than 10 seconds. The autocoder can be adapted to accept any corpus of text and any terminology. The autocoder was also tested using neocl.xml, a taxonomy containing over 68,000 names of neoplasms [10]. The modified autocoder script (omimcan.pl), using neocl.xml (about 16% the size of the UMLS-derived nomenclature used in this study), autocoded OMIM in 810 seconds on a 1.6 GHz processor. Readers can replicate the autocoding test on their own computer systems using the autocoding Perl script and the UMLS unencumbered database provided with this article. The size of the OMIM file increases modestly over time as revisions are made to the 15,000+ entries. Therefore, execution speed will vary slightly with OMIM version. Fastidious readers may wish to truncate their version of OMIM to the version size used in this study (92,471,085 bytes).

Summary of concept frequencies in OMIM

Two Perl scripts calculated term and concept frequencies in OMIM. These scripts are included in the supplementary file collection distributed with this manuscript [Additional file 1].

The total number of different terms in the nomenclature is 529,983

The total number of different concepts in the nomenclature is 290,154

The number of records in OMIM is 15,506

The number of different words in OMIM is 296,345

The total number of words in OMIM is 12,180,657

The number of different concept matches found in OMIM is 23,490

The number of different nomenclature terms found in OMIM is 32,459

The total number of concept matches in OMIM is 367,963

The unencumbered subset of UMLS contains about 1 million terms, and this represents about half of the approximately 2 million terms contained in UMLS [8]. The terms in the unencumbered UMLS were winnowed to exclude terms greater than four words in length and single word terms other than mixed, alphanumeric terms. The resulting external data file of code/term pairs contains about half a million terms and over a quarter million different concept codes. One code may map to several synonymous terms. For example, the following synonymous terms all have the same concept code.

C0338106 colon adenocarcinoma

C0338106 colon carcinoma

C0338106 adenocarcinoma of colon

C0338106 carcinoma of colon

C0338106 carcinoma of the colon

C0338106 adenocarcinoma of the colon

C0338106 colonic adenocarcinoma

C0338106 colonic carcinoma

The number of words in OMIM is 12,180,657 and is contained in a file that exceeds 90 MBytes in length. The number of different words (sometimes referred to as the number of unique words) in OMIM is 296,345. Most large literary texts have fewer than 25,000 unique words. Finnegan's Wake is considered by some to have more different words than any other English language text. A word count of Finnegan's Wake, using a modification of the word counting script used in this manuscript, yielded 56,790 different words. The word counter Perl script used to produce this number is included as a supplemental file [see Additional file 1]. Most English dictionaries contain definitions for approximately 60,000 to 90,000 different words. This indicates that OMIM is a rich source of textual variation and a superb test corpus for an autocoding trial.

When the unique word count of a text greatly exceeds the number of unique words in English language dictionaries, there are only a few possible explanations. The first possible explanation is that the text is riddled with spelling errors. There is no practical limit to the number of incorrect spellings that can be contained in a large text. OMIM is a fastidiously edited text, and orthographic errors are rare. The second source of a large number of unique words is the inclusion of many non-standard words, including names (from literature citations), abbreviations and special terminologies. The field of biology is replete with alphanumeric words (p53, cd-117). These terms and the names of authors listed in the many citations contained in OMIM, account, in large part, for the exceedingly large number of different words found in OMIM.

Output file

The concept indices for each of the records in OMIM are contained in a supplementary file, quickout.txt [see Additional file 1]. An example concept index is shown for the first record (Aarskog syndrome) in OMIM (see Methods).

aarskog syndrome C0175701

birth defect C0220810

cervical spine C0728985

cleft lip C0008924

connective tissue C0009780

connective tissue disorder C0009782

dominant inheritance C0678942

finger joint C0016125

flat feet C0016202

genetic heterogeneity C0242960

genu recurvatum C0152235

imperforate anu C0003466

lower lip C0226938

macrocytic anemia C0002886

maxillary hypoplasia C0240310

monozygotic twin C0041432

palpebral fissure C0229244

portal cirrhosi C0400945

pulmonary disease C0024115

recessive inheritance C0678943

rectoperineal fistula C0240880

short stature C0349588

upper lip C0226930

x-linked inheritance C0241764

A review of the OMIM entry for Aarskog syndrome demonstrates that many of the terms from the first paragraph of the text can be found in the concept index. Of the 24 concepts terms in the index, all are of two-word length or longer. Words in terms ending in the letter "s" were truncated (e.g. cirrhosi, anu), to produce equivalent terminology for many singular and plural terms. Exceptions are words whose plural forms are not produced by the addition of an "s", such as datum/data, mouse/mice, fish/fishes, fox/foxes. In these cases plural and singular forms would occur in the terminology list (data, datum, mouse, mice, fish, fishe, fox, foxe). There were 367,963 coded terms extracted from OMIM and this translates to an average of about 24 extracted concepts for each OMIM record.

Discussion

The work involved in the production of semantic parsers can be prodigious. There is a natural tendency to protect the intellectual property held in machine translators. Nonetheless, the field of machine translation cannot proceed unless there is a way for developers and end-users to perform side-by-side comparisons between different translators. If each developer of autocoding software is reluctant to provide her software to her competitor, there should at least be some available baseline autocoder that can be used by everyone to express the relative speed and performance of their autocoder.

Performance issues

Unlike speed, performance is not an easily defined concept. Informaticians often measure performance with "precision and recall", terms that are roughly analogous to "specificity and sensitivity" used in statistical studies.

Suppose your document contains one million articles, and you want to find all the articles on "renal cell carcinoma." An omniscient entity informs you that there are actually 40 articles relevant to the topic. You do a look-up for autocoded articles that include the UMLS concept unique identifier, C0007134, corresponding to renal cell carcinoma, hypernephroma, Grawitz tumor and other synonyms. Your search yields 20 articles and of those 20 articles, only 10 were included in the list that the omniscient entity has provided. Your recall is 25% (portion of relevant articles successfully retrieved). Half of the 20 articles you found were relevant, so your precision is 50% (portion of retrieved documents that are relevant).

For the purposes of this study, the most difficult issue related to measurements of performance are the varied uses people have for autocoded text. One set of informaticians may prefer "best" or "parsimonious" coding. In "parsimonious" coding, there is a "best" code to representing the ideas contained in a defined section of text. A review article on "liver cirrhosis" may contain many different terms, but the parsimonious coder may only preserve a single code for the entire article. Another informatician, also a parsimonious coder, may not be so strict, but may want only the best term from among a group of subterms. So if "adenocarcinoma of endometrium" appears in text, she may want to preserve this term but omit the so-called atomic inclusive terms, "adenocarcinoma" and "endometrium." Another informatician may want a complete listing of every matching term in a text, including terms that occur within larger terms. Still another informatician may want to include all ancestral terms for each term found in the text (i.e. terms not present in the original text).

A coder may insist on "sense" parsing, preserving the intended sense of the term as used in its context. For example, if the text includes the sentence, "Adenocarcinoma of the endometrium is not present." The "sense" coder might want to exclude the term "adenocarcinoma of the endometrium" or may wish to tag the term with a negation modifier to preserve the intended sense of the term. The author has previously published an "in place" method of inserting codes directly into sentences, preserving modifier terms (including negations) [1].

Finally, some informaticians are "signature" coders. In signature coding a list of the codes from a section of text are used as a textual signature. The relationship of one section of text to another section of text is determined by a quantitative representation of how closely their concept signatures match. Concept signatures are used to retrieve or organize related documents, not specific concepts [6]. Tim Bray has written an excellent article that reviews the many limitations of "precision and recall" as measurements of indexing performance [11].

The criteria for performance would be very different for each of these different kinds of coding strategies, because each coding strategy is designed for a different purpose. The purpose of the tools provided with this article is to provide a baseline approach to autocoding. The idea is for each investigator to come up with a method for performance measurement suitable to her needs. The lexical autocoder can be used to test the relative performance of the investigator's autocoder against the provided lexical autocoder, using performance criteria chosen (and justified) by the investigator.

Conclusions

In the field of medical informatics, no publicly available corpus of autocoder, nomenclature and text has been prepared as a baseline for autocoder comparisons. This paper rectifies this situation. Available with this manuscript are the following tools 1) a public domain autocoder written in Perl (a free and open source programming language that installs on any operating system); 2) a nomenclature database derived from the unencumbered subset of the Unified Medical Language System; 3) a large corpus of autocoded output derived from a publicly available medical text. The simple autocoder provided creates a concept index of medical terms contained in plain-text files. The autocoder consists of 38 lines of Perl code and can be easily installed and executed. Using OMIM as a sample medical text, the autocoder processed text at a rate of approximately 10 seconds/Megabyte and produced a concept list of approximately of 24 terms per OMIM record

Availability and Requirements

The provided tools are designed for simplicity and ease of implementation. The autocoding script is an extremely short program written in Perl. Perl is a freely available open source programming language. Perl interpreters for virtually any operating system are available from several sites on the web, including: http://www.cpan.org and http://www.activestate.com. The autocoding script should execute on any operating system hosting a Perl interpreter.

The nomenclature is provided as a clean database file (in a Perl hash structure). Both files are included [see Additional file: 1]. A large medical corpus (OMIM) is available to the public for ftp download [8]. All three files are intended to be used for side-by-side comparison against an investigator's autocoder, using the investigator's performance criteria.

Competing interests

None declared.

Author's contribution

The work expresses the opinion of the author and does not represent policy of the U.S. government or any of its agencies.

Additional material

Additional File 1

A tarred and gzip-compressed collection of files used in the manuscript. The files include are: 1. omim18.pl (2,404 bytes), the Perl autocoding script 2. quickout.txt (9,540,442 bytes), the autocoded output for omim18.pl, using OMIM and an unencumbered subset of UMLS 3. goodhit.dir (4,096 bytes) and goodhit.pag (33,554,432 bytes), the database files containing the paired concept codes and terms for the unencumbered subset of UMLS 4. tiecount.pl (1,380 bytes), the perl script that counts the number of concepts and terms included in the autocoder nomenclature 5. countcon.pl (1,649 bytes), the Perl script that counts the concepts in OMIM's autocoded output file 6. finneg2.pl (4,630 bytes), the Perl script that counts the different words contained in Finnegan's Wake (by James Joyce). 7. neocl.xml (4,855,690 bytes), a publicly available taxonomy and classification for neoplasms based on the developmental lineage of tumors. 8. omimcan.pl (3,161 bytes), the Perl script that extracts all names of neoplasms contained in OMIM records matching the neoplasm taxonomy file, neocl.xml. 9. quickcan.txt (517,263), the autocoded output of omimcan.pl, using OMIM and neocl.xml Autocode.tar.gz is (9,509,278 bytes), tarred and gzipped (.tar.gz) compressed distribution file containing all the described archive files. This file can be easily decompressed with freely available software available with complete instructions from many web sites http://www.gzip.org.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1472-6947-4-8-S1.gz]

References

1. Berman JJ: **Concept-Match Medical Data Scrubbing: How pathology datasets can be used in research.** *Arch Pathol Lab Med* 2003, **127**:680-686.
2. Berman JJ, Moore GW, Donnelly WH, Massey JK, Craig B: **SNOMED Analysis of 40,124 Surgical Pathology Cases.** *Am J Clin Pathol* 1994, **102:**539-540.
3. Berman JJ, Moore GW: **SNOMED-Encoded surgical pathology databases: a tool for epidemiologic investigation.** *Mod Pathol* 1996, **9:**944-950.
4. Moore GW, Berman JJ: **Automatic SNOMED coding.** *Proc Annu Symp Comput Appl Med Care* 1994:225-229.
5. Moore GW, Berman JJ: **Performance analysis of manual and automated systemized nomenclature of medicine (SNOMED) coding.** *Am J Clin Pathol* 1994, **101:**253-256.
6. Grivell L: **Mining the bibliome: searching for a needle in a haystack?** *EMBO Reports* 2002, **3:**200-203.
7. Salton G, Allan J, Buckley C, Singhal A: **Automatic analysis, theme generation, and summarization of machine-readable texts.** *Science* 1994, **264:**1421-1426.
8. Berman JJ: **A tool for sharing annotated research data: the "Category 0" UMLS (Unified Medical Language System) vocabularies.** *BMC Med Inform Decis Mak* 2003, **3:**6.
9. Cantor MN, Lussier YA: **Putting data integration into practice: using biomedical terminologies to add structure to existing data sources.** *Proc AMIA Symp* 2003:125-129.
10. Berman JJ: **Tumor classification: molecular analysis meets Aristotle.** *BMC Cancer* 2004, **4:**10.
11. **On Search: Precision and Recall** [http://www.tbray.org/ongoing/When/200x/2003/06/22/PandR]

Pre-publication history

The pre-publication history for this paper can be accessed here:

http://www.biomedcentral.com/1472-6947/4/8/prepub