Nomenclature-Based Data Retrieval without Prior Annotation: Facilitating Biomedical Data Integration with Fast Doublet Matching

Jules J. Berman

Cancer Diagnosis Program, National Cancer Insititute, National Institutes of Health, Bethesda, MD, USA
E-mail: bermanj@mail.nih.gov

Edited by E. Wingender; received 12 December 2004; accepted 28 March 2005; published 3 April 2005

ABSTRACT: Assigning nomenclature codes to biomedical data is an arduous, expensive and error-prone task. Data records are coded to to provide a common representation of contained concepts, allowing facile retrieval of records via a standard terminology. In the medical field, cancer registrars, nurses, pathologists, and private clinicians all understand the importance of annotating medical records with vocabularies that codify the names of diseases, procedures, billing categories, etc. Molecular biologists need codified medical records so that they can discover or validate relationships between experimental data and clinical data.

This paper introduces a new approach to retrieving data records without prior coding. The approach achieves the same result as a search over pre-coded records. It retrieves all records that contain any terms that are synonymous with a user's query-term. A recently described fast algorithm (the doublet method) permits quick iterative searches over every synonym for any term from any nomenclature occurring in a dataset of any size.

As a demonstration, a 105+ Megabyte corpus of Pubmed abstracts was searched for medical terms. Query terms were matched against either of two vocabularies and expanded as an array of equivalent search items. A single search term may have over one hundred nomenclature synonyms, all of which were searched against the full database. Iterative searches of a list of concept-equivalent terms involves many more operations than a single search over pre-annotated concept codes. Nonetheless, the doublet method achieved fast query response times (0.05 seconds using Snomed and 5 seconds using the Developmental Lineage Classification of Neoplasms, on a computer with a 2.89 GHz processor).

Pre-annotated datasets lose their value when the chosen vocabulary is replaced by a different vocabulary or by a different version of the same vocabulary. The doublet method can employ any version of any vocabulary with no pre-annotation. In many instances, the enormous effort and expense associated with data annotation can be eliminated by on-the-fly doublet matching.

The algorithm for nomenclature-based database searches using the doublet method is described. Perl scripts for implementing the algorithm and testing execution speed are provided as open source documents available from the Association for Pathology Informatics (www.pathologyinformatics.org/informatics_r.htm).

KEYWORDS: Nomenclature, taxonomy, classification, autocoding, indexing, annotation, metadata, algorithm, data integration, Perl, doublet method, doublet matching

INTRODUCTION

A nomenclature is a comprehensive listing of terms in a knowledge domain. Nomenclatures group together all synonymous representations of a concept. Concepts are typically organized by a hierarchical classification system [1,2]. When textual biomedical data is annotated with a standard nomenclature, data can be queried and retrieved by concept, rather than by contained terms [3,4].

Electronic publication can be found in *In Silico Biol.* **5**, 0029 <http://www.bioinfo.de/isb/2005/05/0029/>, 3 April 2005.

The field of biomedical informatics has been pre-occupied for decades with developing methods to adequately annotate textual data. Everyone who uses biomedical data is familiar with nomenclature-based annotation systems (also known as coding systems): Current Procedural Terminology [5], International Classification of Diseases [6], the Systematized Nomenclature of Medicine Clinical Terminology (Snomed) [7–9], Medical Subject Headings [10], and the Gene Ontology resource [11,12]. The Unified Medical Language System (UMLS) Metathesaurus [13] is a compilation of over 100 biomedical vocabularies available from the National Library of Medicine. Updated versions of the UMLS are released several times each year. The large number of medical vocabularies and their frequent version changes create problems for medical coders. In particular, the time and effort spent preparing well-coded medical records is lost when the coding vocabulary is replaced or changed. Furthermore, the vocabulary chosen to code a particular dataset may be different from the vocabulary chosen to code another dataset. Unless exact mappings are established between the two vocabularies, queries over equivalent terms cannot be integrated over both datasets.

Perhaps the most practical solution to the problem is with automatic re-coding. It is now possible to re-code large datasets very quickly [14,15]. Data managers may find that they can re-code their entire datasets when a user prefers a different vocabulary or when a new version of a vocabulary is issued. Re-coding datasets may become impractical if the vocabularies are frequently modified or replaced.

If the task of re-coding large datasets is impractical, there is another option. The purpose of this manuscript is to describe a new algorithm by which dataset records may be interrogated, quickly retrieving all the records that contain the concept-equivalents of a query term, using any biomedical nomenclature, and using datasets that have not been annotated with vocabulary codes. This algorithm achieves the functionality of a search over an annotated dataset. Because the algorithm requires no pre-coding, it reduces the enormous expenditure of professional time and energy devoted to coding biomedical reports.

METHODS

The algorithm for fast doublet matching is most easily understood by first describing a general approach to conducting code-based searches without pre-annotation. Secondly, the specific method for fast doublet matching can be described. Finally, a software implementation of the algorithm is provided.

Algorithm for code-based searches without pre-annotation

1. The user enters a query term.
2. The user's query term is matched against the terms included in a preferred vocabulary. Any vocabulary is suitable, so long as the vocabulary consists of term/code pairs, where a term and its' synonyms are all paired with the same code.
   For instance, the 2004 version of UMLS has 38 equivalent entries for the code C0206708, nine of which are listed here:
   C0206708|Cervical Intraepithelial Neoplasms
   C0206708|Cervical Intraepithelial Neoplasm
   C0206708|Intraepithelial Neoplasm, Cervical
   C0206708|Intraepithelial Neoplasms, Cervical
   C0206708|Neoplasm, Cervical Intraepithelial
   C0206708|Neoplasms, Cervical Intraepithelial

C0206708|Intraepithelial Neoplasia, Cervical
C0206708|Neoplasia, Cervical Intraepithelial
C0206708|Cervical Intraepithelial Neoplasia
If the user had entered "Cervical Intraepithelial Neoplasms", the query would be compared with the millions of terms included in the UMLS until a match is made against the exact-match term whose code is C0206708.

3. All of the terms in the vocabulary that are equivalent to the query term are collected from the nomenclature. Since we are dealing with term/code pairs, this means that all of the vocabulary terms with the same concept code are collected. In the example given, all 38 of the terms whose concept code is C0206708 would be collected.
4. One-by-one, the equivalent terms are matched against each record in the dataset to determine which record contains character strings that match any of the equivalent terms.
   In the case of the example, this would mean that all 38 UMLS terms equivalent to "Cervical Intraepithelial Neoplasms" would be matched against the entire set of data records. This is the rate-limiting step for the algorithm. This step could not be seriously contemplated prior to the advent of fast computers and fast search algorithms.
5. For each dataset record, if the record contains a character string that matches any of the equivalent vocabulary terms, the record is retrieved. In the case of the example given, all records containing the term cervical intraepithelial neoplasia or any of the term equivalents found in UMLS, would be merged into the query response and annotated with the common concept code C0206708.

Algorithm for fast doublet matching

The following algorithm describes a way to quickly achieve search results for the algorithm described above.

1. A doublet index is prepared for the data records. The doublet index consists of each of the doublets (two consecutive words) in the datasets along with all the locations in the dataset where each doublet occurs.
   In the sample dataset used for this manuscript, the doublet term "vancouver canada" occurs 95 times. Examples of 14 index entries for the "vancouver canada" doublet are:
   vancouver canada = 151198-17
   vancouver canada = 157354-8
   vancouver canada = 166770-13
   vancouver canada = 171565-8
   vancouver canada = 175470-11
   vancouver canada = 178127-8
   vancouver canada = 189527-11
   vancouver canada = 198094-8
   vancouver canada = 201139-11
   vancouver canada = 201398-12
   vancouver canada = 202037-8
   vancouver canada = 204257-14
   vancouver canada = 208131-8
   vancouver canada = 223026-11
   Each entry consists of the name of the doublet, the record number from the dataset in which the doublet occurs (e.g. record number 151198), and the offset position within the record at which the

doublet occurs (e.g. word number 17). The index can be quickly compiled and, once created, never needs to change unless the dataset changes. This index will be used to locate terms composed of any number of words.

2. An associative array is prepared from the doublet index file, consisting of key/value pairs, where the keys are the set of all the different doublet terms present in the dataset, and the values are the byte location in the doublet index file where the doublet first occurs. The doublet index file is alphabetized. Knowing the first occurrence of a doublet in the index allows us to quickly find all the index entries for the doublet by simply going to the first location of the doublet and collecting successive line readings from the index file. The collection of all the index entries for the doublet specifies every record and every position in every record where the doublet occurs.
3. The nomenclature of interest is stored in memory as two associative arrays. One associative array consists of key/value pairs, with nomenclature terms as the keys and corresponding nomenclature codes as values. The other associative array consists of key/value pairs with nomenclature codes as the keys and the list of corresponding nomenclature terms as the values.
   These two associative arrays allow us to quickly match the user's query-term against the entire nomenclature, identifying the code number of the matching term (if it exists) and creating an array of all the equivalent terms that match the code.
4. The array of vocabulary terms that are equivalent to the query term entered by the user are consecutively matched against all the records from the dataset (as described in steps 5, 6 and 7).
5. Each term in the array of equivalent terms is parsed as an array of doublet neighbors. In the case of terms composed of an odd number of words, an overlapping doublet at the end of the term is added.
   Example of a term composed of an even number of words:
   Adenocarcinoma of the lung
   The doublet array is:
   "adenocarcinoma of", "the lung"
   Example of a term composed of an odd number of words:
   refractory anemia with excess blasts
   The doublet array is:
   "refractory anemia", "with excess", "excess blasts"
6. For each term in the array of equivalent terms, entries are collected from the doublet index if they match any of the consective doublets that compose the term. The records that match a term are among the records that match every doublet in the term. For a record in this subset to match the term, it needs to contain the doublets that compose the term in the same text order as the occurrences of the doublets in the term.
   The term "refractory anemia with excess blasts" is composed of three doublets that must occur in the following relative word positions:
   "refractory anemia" . . . position n
   "with excess" . . . position $n + 2$
   "excess blasts" . . . position $n + 3$
   To determine whether term doublets co-occurring in a record actually match a full term, one needs to test whether the doublets occur in relative positions corresponding to their positions in the term. For instance, a record may contain multiple occurrencs of each of the doublets that compose the term "refractory anemia with excess blasts". Imagine that the doublets occur at the following word positions within the record:
   "refractory anemia" locations (15, 92, 105, 234)

"with excess" locations (17, 107, 344)
"excess blasts" locations (18, 108, 992, 1026)
We can be certain that the full term occurs twice in the record, beginning in positions 15 and 105. Only at these two offset positions for the doublet "refractory anemia" are there consecutive occurrences in the same record of the ordered doublets in the whole term (i.e., offsets n, $n + 2$, $n + 3$).

7. All of the dataset records containing all of the consecutively occurring doublets that compose the terms from the array of terms that are equivalent to the query term are collected from the dataset and annotated with the shared concept vocabulary code.

Implementation

The following 4 perl scripts and 2 vocabulary files are made freely available to the public, from the Association for Pathology Informatics [16]:
doub4.pl
bigsort.pl
doubdat2.pl
annotget.pl
neocl.xml
neoself
The steps involved in re-creating an implementation

1. Anyone wishing to use the author's implementation will need to have Perl installed in their computer. Perl is an open source, cross-platform, free and widely available programming language. Perl is very popular among bioinformaticians, and it can be downloaded from The Comprehensive Perl Archive Network [17] or from ActiveState [18].
2. Users will need to have a dataset or plain-text corpus in which every record is separated by the same delimiter. For testing purposes, the author created a 105 megabyte text, as described previously [14, 15].
   The corpus was created by a PubMed query on "pathology [ad] AND neoplasm [all]", at the US National Library of Medicine's website (www.pubmed.org). The query gathered all abstracts from the Pubmed database in which the term neoplasm occurs somewhere in the Pubmed entry, and in which the affiliation of the author contains the word "pathology". The query yielded abstracts that are likely to contain names of neoplasms. The PubMed output file can serve as a good test for an autocoder that uses a neoplasm nomenclature. The PubMed search yielded 66,509 abstracts. All of the abstracts were downloaded into a single file from the PubMed site by setting the "Display" attribute to "Abstract" and the "Send to" attribute to "file". This produced a 105,689,546 byte plain-text file. The file was given the filename tumorab.txt, and this filename was used by the autocoders as a parsing input file. Although this file is not included with this manuscript, anyone in the world with internet access can obtain a near-identical file by repeating the same PubMed query. The records from the text consisted of paragraphs delimited by double return characters (also referred to as double-newline characters or double ascii13-ascii10 character pairs). This is a common way of delimiting textual paragraphs.
3. Users will create a file of all the doublets in the file, one line to each doublet, and each doublet followed by the record number and the record word-offset for the doublet occurrence. When a doublet occurs several times in a single record, each occurrence is indexed, with a different word-offset for each occurrence.

The perl script that creates the doublet index is doubdat2.pl. It may require several minutes to execute, and it produces an index file, that is 400+ Megabytes in length. This file is, in turn, sorted alphabetically by bigsort.pl a short perl script that sorts files of any size. The index file is named doubdat.out. Doubdat2.pl also keeps track of the begin-byte location in tumorab.txt where each text record begins.

Another perl script, doub4.pl, creates a database file containing the byte location in doubdat.out for the first occurrences of doublets.

Annotget.pl performs term searches on 100 terms selected randomly from a chosen vocabulary. In this case, the vocabularies tested are Snomed-CT, extracted from the 2004 version of UMLS, and the "Taxonomy for the developmental lineage classification of neoplasms", made available for public download by the Association for Pathology Informatics [13,18].

RESULTS AND DISCUSSION

Speed

The results of 4 trial runs of 100 randomly generated medical terms using two different medical vocabularies are shown. The trials were executed on a 2.89 GHz CPU.

Source vocabulary: Snomed-CT

1.
   ```
   c:\doublet>perl annotget.pl snomed
   total time for 100 multi-word queries is 5
   total time for 10 single words is 18
   ```
   This software exercise used Snomed-CT as its source vocabuary and selected 100 terms at random from the nomenclature. This simulates a user entering 100 valid terms. For each term, the implementation created an array consisting of every term-equivalent for the original term. For each of these hundred arrays, all of the records in the 105+ Megabyte dataset were searched. Records that matched terms from the arrays were extracted and added to an external file. A special case arises when the term-equivalent of a multiword term is a single word term. For instance, nephroblastoma is a singlet equivalent of Wilms' tumor. Because the doublet index contains no singlets, the implementation creates an array of every doublet containing the word nephroblastoma, accounting for every possible occurrence of nephroblastoma. This means that for single word terms, an entire array of doublets must be searched. This greatly lengthens the execution time for retrieving matches to singlets. In the first run of the software, there were 10 instances of single word terms occurring as term equivalents of randomly selected multi-word terms.

   The speed of the program for 100 randomly chosen terms and all their term equivalents was 5 seconds for the multiword terms (e.g., 0.05 seconds per term). 18 seconds was required to search for 10 single words terms, or 1.8 seconds per singlet.

2.
   ```
   c:\doublet>perl annotget.pl snomed
   total time for 100 multi-word queries is 5
   total time for 8 single words is 14
   ```
   A second run of the software implementation also used Snomed-CT. The results were virtually identical to the first run. The average retrieval time for 100 multiword terms is 0.05 seconds. In the process of the search, 8 single word term equivalents were identified, and the average retrieval time for each single word was 14/8 or 1.75 seconds per singlet.

Source vocabulary: The Taxonomy for the developmental lineage classification of neoplasms

1. c:\doublet>perl annotget.pl
   total time for 100 multi-word queries is 438
   total time for 3 single words is 5
   This exercise used the default nomenclature, the Taxonomy for the developmental lineage classification of neoplasms, as its source vocabulary. As in the prior examples, 100 terms were selected at random from the nomenclature, simulating 100 user-initiated searches.
   The average time for finding multi-word terms is 4.38 seconds, approximately 100 times slower than the average time needed to find a search term using Snomed-CT as the source vocabulary. The reason for the slow execution speed is related to the disparate levels of synonymy in both nomenclatures. The average number of synonyms for each term in the neoplasm taxonomy is 23. This means that a search of 100 randomly selected terms is expanded to 2300 term searches. Snomed-CT is a vocabulary with many terms (exceeding 295,000) but each term has, on average, fewer than two synonyms. This means that a search of 100 randomly selected Snomed-CT terms may expand to fewer than 200 searches. Differences in average term length (number of words in a nomenclature term) among different vocabularies will also contribute to differences in execution speed.
   The search for 100 randomly selected multword terms in the neoplasm taxonomy yielded, after expansion, three single word terms, with an average search time of 1.66 seconds per singlet, nearly identical to the single word search times for the Snomed-CT vocabulary.
2. c:\doublet>perl annotget.pl
   total time for 100 multi-word queries is 520
   This software exercise also used the Taxonomy for the developmental lineage classification of neoplasms. In this case, the average search time for multi-word terms is 520/100 or 5.2 seconds per search term.
   There were no single word searches for this run. The reason why no single word terms were encountered relates to the frequency of occurrence of single word terms in the taxonomy nomenclature. As discussed in detail elsewhere [15], single word terms are rare in the neoplasm taxonomy. In the neoplasm taxononomy only about 250 terms out of a total of about 123,000 terms, are single word terms. Most of the singlets are names of specific tumors ending in the suffix, "oma." "Oma" is short for "tumor." Single-word names of tumors ending in "oma" can be thought of as doublets with the first and second words fused together (i.e. osteoblastoma is "osteoblast" + "oma"). Some examples of "oma" terms are: acanthoma, adamantinoma, adenofibroma, adenomyoepithelioma, adenomyoma, adenosarcoma, ameloblastoma, etc.). Because singletons are scarce in the vocabulary, the expansions of many searches will not include singlets.

Other performance issues

In a prior manuscript by the author, issues of performance measurement were discussed [14]. A lexical parsing algorithm, such as the doublet method, looks for exact matches between text and vocabulary. There are no false positive matches (because the software does not extract terms that are not actually present in the vocabulary). When a lexical parser fails to extract a relevant term, it is always due to insufficiences in the text (e.g., a misspelled term in a medical report) or in the vocabulary (e.g., the omission from the vocabulary of a legitimate term or concept). The quality of narrative text can be greatly improved by avoiding abbreviations, carefully checking orthography, and by using software to ensure that relevant data items are included in reports. Good sentences are short and simple [8]. Good vocabularies contain all the variant terms for any given concept, with no term appearing under more than one concept.

It is the author's position that when discussing the performance of a lexical parser, such as the doublet method, speed is the only performance measurement that can be substantially improved by software. Coding accuracy is largely determined by the adequacy of the nomenclature.

Different philosophical approaches to term-based data retrieval

Advocates of data coding believe that raw textual data is poorly organized and needs to be interpreted, structured and classified before it can be usefully interrogated as a database record. Coding traditionally requires human input. As described in a prior manuscript by the author, humans do not code with great consistency or completeness [8]. Humans also do not write narrative text without introducing ambiguities. This means that humans cannot consistently interpret narrative biomedical text [19]. Although human coding can be improved with training and by hiring dedicated professionals whose primary responsibility is to maintain complete and well annotated records (e.g. Tumor Registrars), such efforts come at great expense.

Human coding, for all practical purposes, makes it impossible to update coded datasets when vocabularies are updated or when the rules for coding are changed. Historically, standard nomenclatures such as ICD or Snomed-CT come out with new versions at least every decade. More importantly, the choice of a vocabulary makes it virtually impossible to integrate the coded data in one dataset with the data in another dataset that has been coded with a different vocabulary or a different style of coding. Published attempts to map between nomenclatures would indicate that precise mappings are impossible [12].

After an institution has devoted thousands of professional hours to coding a large collection of biomedical records, it is sometimes impractical to ask professionals to go back and re-code the data. It is the author's perception that in most cases, the costs of updating a legacy dataset with new codes has been prohibitive. When new nomenclatures emerge, institutions typically either ignore them, or they deploy the new nomenclature prospectively, ignoring legacy data.

Manual coders are hopelessly outpaced by the avalanche of textual data collected by hospital information systems. Consider the narrative text produced or collected by health professionals: email communications, progress notes, nursing notes, operation notes, surgical pathology reports, radiology reports, journal articles, white papers, data analyses, meeting summaries, etc. Most of this data simply goes uncoded.

Automatic coding using natural language techniques holds enormous promise. A basic assumption of natural language parsing is that synonymous terms are closely related word phrases. For instance, the term adenocarcinoma of lung has much in common with alternate terms such as adenocarcinoma of the lung, adenocarcinoma of the lungs, lung adenocarcinoma, lung carcinoma, etc. A program that takes into account minor variations in word order, grammatic constructions, word roots [stemming], syntax variation, or string similarity scores may be able to pull all the equivalent terms from a text automatically.

A limitation of this approach is that term synonyms in medical vocabularies often have no etymologic relationships. Consider the term "renal cell carcinoma." The neoplasm taxonomy contains 54 synonyms, including adenocarcinoma of kidney, hypernephroma and Grawitz tumor. It would not be possible to discover these synonyms by employing a natural language parser. The only way of obtaining adequate synonymy is through a complete nomenclature.

In the neoplasm taxonomy, synonyms for renal cell carcinoma are all coded as "C9385000". The 54 synonyms for "renal cell carcinoma" are listed here:

"adenoca arising from kidney, adenoca arising in kidney, adenoca involving kidney, adenoca of kidney, adenocarcinoma arising from kidney, adenocarcinoma arising from the kidney, adenocarcinoma arising in kidney, adenocarcinoma arising in the kidney, adenocarcinoma involving kidney, adenocarcinoma involving the kidney, adenocarcinoma of kidney, ca arising from kidney, ca arising in kidney, ca involving kidney, ca of kidney, cancer arising from kidney, cancer arising in kidney, cancer involving kidney, cancer of kidney, carcinoma arising from kidney, carcinoma arising in kidney, carcinoma involving kidney, carcinoma of kidney, grawitz neoplasm, grawitz tumor, grawitz tumour, hypernephroid tumor, hypernephroid tumour, hypernephroma, kidney adenoca, kidney adenocarcinoma, kidney ca, kidney cancer, kidney carcinoma, kidney with adenoca, kidney with adenocarcinoma, kidney with ca, kidney with cancer, kidney with carcinoma, renal adenoca, renal adenocarcinoma, renal ca, renal cancer, renal carcinoma, renal cell adenoca, renal cell adenocarcinoma, renal cell ca, renal cell cancer, renal cell carcinoma, stage unspecified renal cell adenoca, stage unspecified renal cell adenocarcinoma, stage unspecified renal cell ca, stage unspecified renal cell cancer, stage unspecified renal cell carcinoma"

Natural language parsing is computationally intensive and much slower than simple hash look-ups (such as those employed by the doublet method). Natural language parsing may be suitable for retrieving terms from a small corpus. When gigabytes or terabytes of information need to be processed, faster techniques may be required.

The general problem of term identification in medical text is a central, unresolved issue in medical informatics. A recent review summarizes the different approaches and tools now available to informaticians [3]. The following suggestions may be helpful when choosing a preferred method for querying medical datasets.

When is Pre-Annotation useful?

- When there is the need for a global analysis of the dataset.
  (In a global analysis of a dataset, all the data elements are examined at once. Typically, the researcher is looking for relationships among the different data elements.)
- When the query response time must be very rapid.
- When the expected number of queries may be very large.

When is fast doublet matching useful?

- When the dataset is typically searched one item at a time.
- When the dataset does not change or changes only by the addition of records.
- When the dataset is being searched using many different types of vocabularies.
- When the dataset is searched with a single vocabulary that is constantly changing.
- When one dataset needs to be integrated with another dataset, and the datasets have not been annotated with the same nomenclature.

CONCLUSIONS

Nomenclature-based data retrieval without prior annotation, using the fast doublet method, has the potential of eliminating much of the time and money now spent on data annotation. The method supports retrieval with any version of any nomenclature, thus permitting researchers to integrate data among heterogeneous sources. An open source implementation of the algorithm, in the Perl programming language, is publicly available (www.pathologyinformatics.org/informatics_r.htm).

ACKNOWLEDGEMENTS

This work was conducted as part of the author's duties as a US government employee. the statements in the manuscript are the opinion of the author and do not represent US government policy.

REFERENCES

[1] Berman, J. J. (2004). Tumor classification: molecular analysis meets Aristotle. BMC Cancer 4, 10.
[2] Berman, J. J. (2004). Tumor taxonomy for the developmental lineage classification of neoplasms. BMC Cancer 4, 88.
[3] Krauthammera, M. and Nenadicb, G. (2004). Term identification in the biomedical literature J. Biomed. Informatics 37, 512-526.
[4] Lu, X., Zhai, C., Gopalakrishnan, V. and Buchanan, B. G. (2004). Automatic annotation of protein motif function with Gene Ontology terms. BMC Bioinformatics 5, 122.
[5] CPT Current Procedural Terminology. (http://www.ama-assn.org/ama/pub/category/3113.html).
[6] International Classification of Diseases (ICD). (http://www.who.int/classifications/icd/en/).
[7] Systematized Nomenclature of Medicine. (http://www.snomed.org).
[8] Berman, J. J. and Moore, G. W. (1996). Snomed-encoded surgical pathology databases: a tool for epidemiologic investigation. Mod. Pathol. 9, 944-50.
[9] Medical Subject Headings (MESH). (http://www.nlm.nih.gov/mesh/meshhome.html).
[10] Moore, G. W. and Berman, J. J. (1994). Performance analysis of manual and automated systematized nomenclature of medicine (Snomed) coding. Am. J. Clin. Pathol. 101, 253-256.
[11] Camon, E., Magrane, M., Barrell, D., Lee, V., Dimmer, E., Maslen, J., Binns, D., Harte, N., Lopez, R. and Apweiler, R. (2004). The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. Nucleic Acids Res. 32, D262-D266.
[12] Cantor, M. N., Sarkar, I. N., Gelman, R., Hartel, F., Bodenreider, O. and Lussier, Y. A. (2003). An evaluation of hybrid methods for matching biomedical terminologies: mapping the gene ontology to the UMLS. Stud. Health Technol. Inform. 95, 62-67.
[13] Unified Medical Language System (UMLS). (http://www.nlm.nih.gov/research/umls/metaa1.html).
[14] Berman, J. J. (2004). Resources for comparing the speed and performance of medical autocoders. BMC Med. Inform. Decis. Mak. 4, 8.
[15] Berman, J. J. (2004). Doublet method for very fast autocoding. BMC Med. Inform. Decis. Mak. 4, 16.
[16] Association for Pathology Informatics. (http://www.pathologyinformatics.org/informatics_r.htm).
[17] Comprehensive Perl Archive Network (CPAN). (http://www.cpan.org).
[18] ActiveState. (http://www.activestate.com).
[19] Powsner, S. M., Costa, J. and Homer, R. J. (2000). Clinicians are from Mars and pathologists are from Venus. Arch. Pathol. Lab. Med. 124, 1040-1046.