BMC Bioinformatics

RESEARCH

Open Access

G-Bean: an ontology-graph based web tool for biomedical literature retrieval

James Z Wang[1*], Yuanyuan Zhang[1], Liang Dong[2], Lin Li[3], Pradip K Srimani[1], Philip S Yu[4]

From IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2013)
Shanghai, China. 18-21 December 2013

Abstract

Background: Currently, most people use NCBI's PubMed to search the MEDLINE database, an important bibliographical information source for life science and biomedical information. However, PubMed has some drawbacks that make it difficult to find relevant publications pertaining to users' individual intentions, especially for non-expert users. To ameliorate the disadvantages of PubMed, we developed G-Bean, a **g**raph based **b**iomedical s**ea**rch e**n**gine, to search biomedical articles in MEDLINE database more efficiently.

Methods: G-Bean addresses PubMed's limitations with three innovations: (1) *Parallel document index creation*: a multithreaded index creation strategy is employed to generate the document index for G-Bean in parallel; (2) *Ontology-graph based query expansion*: an ontology graph is constructed by merging four major UMLS (Version 2013AA) vocabularies, MeSH, SNOMEDCT, CSP and AOD, to cover all concepts in National Library of Medicine (NLM) database; a Personalized PageRank algorithm is used to compute concept relevance in this ontology graph and the Term Frequency - Inverse Document Frequency (*TF-IDF*) weighting scheme is used to re-rank the concepts. The top 500 ranked concepts are selected for expanding the initial query to retrieve more accurate and relevant information; (3) *Retrieval and re-ranking of documents based on user's search intention*: after the user selects any article from the existing search results, G-Bean analyzes user's selections to determine his/her true search intention and then uses more relevant and more specific terms to retrieve additional related articles. The new articles are presented to the user in the order of their relevance to the already selected articles.

Results: Performance evaluation with 106 OHSUMED benchmark queries shows that G-Bean returns more relevant results than PubMed does when using these queries to search the MEDLINE database. PubMed could not even return any search result for some OHSUMED queries because it failed to form the appropriate Boolean query statement automatically from the natural language query strings. G-Bean is available at http://bioinformatics. clemson.edu/G-Bean/index.php.

Conclusions: G-Bean addresses PubMed's limitations with ontology-graph based query expansion, automatic document indexing, and user search intention discovery. It shows significant advantages in finding relevant articles from the MEDLINE database to meet the information need of the user.

Background

Currently, one of the most important bibliographical information sources for life science and biomedical research is MEDLINE database [1]. Building a Web-based tool to find relevant biomedical literature in MEDLINE database in response to a query remains a challenge due to the increase in volume and diversity of topics of biomedical literatures [2]. The primary portal to search the MEDLINE database is PubMed [3] by National Center for Biotechnology Information (NCBI). However, finding relevant publications with PubMed pertaining to users' individual interests is still daunting, especially for non-expert users. Due to the difficulty of forming appropriate query statements, only experienced users such as librarians [4] can obtain accurate search results using PubMed interface. It is widely reported that less-experienced users, including those who regularly use the PubMed system, do

* Correspondence: jzwang@clemson.edu
[1]School of Computing, Clemson University, Clemson, SC, 29634, USA
Full list of author information is available at the end of the article

not utilize it as effectively as experienced users [5-7]. Those less-experienced users either fail to employ the most relevant context-sensitive keywords or fail to effectively formulate query expressions using Boolean logic [8,9]. It has been reported [8] that a novice user (e.g. a third year medical student) requires an average of fourteen separate queries to get the desired information.

In addition, PubMed does not always return the most relevant articles for user queries. PubMed's underperformance in biomedical information retrieval is partly due to the fact that it uses only a very small subset of the Medical Subject Headings (MeSH) [10] to index the biomedical articles. MeSH (ver. 2013) consists of 26,853 descriptors, 83 qualifiers, over 213K assisting entry terms, and over 214K supplementary concept records. However, PubMed uses only the descriptors and qualifiers for indexing purposes. It means only 5.9% of all concepts in MeSH are used for the document indexing. While there are 2.9 million biomedical concepts enumerated in UMLS Metathesaurus 2013AA [11], the PubMed index utilizes less than 1% of this available vocabulary. Many studies, such as Hersh [4,12,13], Yoo [14] and Abdou [15], have attempted to address PubMed's low vocabulary coverage problem by expanding user queries with more concepts in MeSH ontology. However, it has been observed that these query expansion approaches [14,15] offer no significant advantages over the free-text based search methods; missing concepts and incomplete synonym sets (due to the use of only MeSH ontology) were found to be the major causes of the inadequacy of existing query expansion schemes.

To address these issues, we develop a new Web-based literature retrieval tool, G-Bean (a **g**raph based **b**iomedical s**ea**rch e**n**gine), to query documents in MEDLINE database. The major contributions of G-Bean are three folds: 1) it uses a multithreaded parallel algorithm to automatically generate the document index to address the inefficiency of the PubMed's manual indexing process; this automated index generation scheme allows incremental index update for timely index maintenance; 2) it merges multiple biomedical ontologies into a single ontology graph and uses all concepts in this ontology graph to index documents, ameliorating PubMed's low concept coverage problem of using only MeSH terms for indexing; 3) it automatically retrieves additional relevant articles based on user's current selection and ranks them according to their semantic similarities with all articles selected so far while PubMed can recommend a list of articles matching the same keywords with only the current viewing article [16].

Methods
Ontology-graph construction
In G-Bean, we employ a subset of UMLS Metathesaurus as the knowledge source to construct an ontology graph.

The Metathesaurus of UMLS is the largest vocabulary database that contains information about biomedical and health related concepts and inter-relationships among concepts [17]. Each biomedical concept in the Metathesaurus is a grouping of synonymous terms and is identified by a distinct eight character alphanumeric string, called Concept Unique Identifier (CUI). The CUI is linked to a set of lexical variants strings, which is an alternative way to represent the concept. The MRCONSO table contains information of these CUIs to resolve synonymy problems that may arise in organizing medical text. Information includes concept-names, spelling variations, and acronyms, etc. The inter-concept relationships, such as parent/child, and immediate siblings, are stored in the MRREL table. Our ontology graph is automatically constructed using the information from MRCONSO and MRREL tables where a vertex represents a concept and an edge represents an inter-concept relationship.

Since four ontologies in UMLS, MeSH, SNOMEDCT, CSP and AOD cover all senses of the target words in NLM database [18], we obtain these four ontologies from UMLS 2013AA [11] to construct the ontology graph with the assistance of UMLS MetamorphoSys, a multi-platform UMLS installation tool for UMLS Knowledge Sources [19].

Parallel index creation for MEDLINE database
We first download the 2014 MEDLINE/PubMed Baseline database which contains 22,376K records from NLM [20] and then adapt Lucene Java search library (version 4.5.1) to create index for MEDLINE documents on the G-Bean server [21]. Each MEDLINE citation has a unique PubMed identifier called PMID. Since MEDLINE citation records do not contain full text articles, only the title and abstract of the documents are processed and indexed. A modified Lucene standard analyzer with an enhanced MIT stop-list and the Porter stemmer is used to analyze (process special characters), tokenize (break into words), stem (get base of word) and index MEDLINE document's title and abstract respectively. Moreover, MetaMap 2013 is employed to map the title and the abstract into a set of associated CUIs which are then indexed with our multithreaded indexing process.

The size of the 2014 MEDLINE/PubMed Baseline database is over 160GB. Building an index for this large dataset is challenging with Lucene library. It takes more than 10 days to generate the entire index for all MEDLINE documents using a computer with system parameters shown in Table 1.

To speed up the index creation, we modify Lucene Java search library to make it multithread capable. We use a threadpool [22] to submit multiple tasks to the multi-core computer for MEDLINE index creation.

Table 1 Physical configuration of the computer used to create index of MEDLINE documents

CPU	Intel Core i7-2600 CPU @ 3.40GHz
The number of cores	4
Memory	8GB
System type	X86_64

The thread pool is created by using ThreadPoolExecutor, as shown below:

> new ThreadPoolExecutor(int corePoolSize,
>   int maximumPoolSize,
>   long keepAliveTime,
>   TimeUnit unit,
>   BlockingQueue<Runnable> workQueue,
>   RejectedExecutionHandler handler)

where *corePoolSize* is the number of threads in the core pool, *maximumPoolSize* indicates the maximum number of threads allowed in the thread pool, *keepAliveTime* (i.e., thread keep-alive time) is the amount of time that waiting threads in excess of the core pool size may remain idle before being terminated, *unit* identifies the time unit for the *keepAliveTime* argument, *workQueue* is the queue to hold tasks before they are executed, *handler* demonstrates the handler that blocks the execution when maximum number of threads has been reached or queue capacities are exceeded. Since index creation is mostly a CPU-bound task as it does not involve much of I/O operations, the *corePoolSize* is set close to the number of CPU cores for our index creation.

The MEDLINE dataset contains 746 compressed files. During index creation, the RAM size required by Lucene is determined by the buffer size used by IndexWriter. To maximize the throughput of index creation, we should set the IndexWriter buffer as large as possible. Ideally, we should create the index for all 746 files at once. However, due to the memory space limitation in the computer we used, we divided the 746 files into 8 groups and created one index for each group. Then we use IndexWriter's addIndexes method to merge these 8 partial indices into the final index for the entire 746 files. With this approach, the total time for creating the MEDLINE index is:

$$T_p = \sum_{i=1}^{8} T_i + T_{merge} \tag{1}$$

where T_i is the time to create the partial index for files in group i, T_{merge} is the time to merge the 8 partial indices into the final index.

Evaluation of multi-thread based parallel index creation
To evaluate how the multithreaded parallel index creation approach speeds up the creation of MEDLINE

index, we compare the total time for creating the MEDLINE index using this new approach with that using the original Lucene library on the same computer as shown in Table 1. Experiments show that the best performance is achieved using *ArrayBlockingQueue* with the following settings:

$$n_{cps} = 4, n_{mp_s} = 5, n_{queue} = 18$$

where n_{cps} is the value of *corePoolSize*, n_{mps} is the value of *maximumPoolSize* and n_{queue} is the size of *ArrayBlockingQueue*.

We run our experiment three times and obtain the average time to create the MEDLINE index using the multi-threaded approach and the original Lucene library respectively. The average time for creating the MEDLINE index using our multithread approach is 250,917 seconds (less than 3 days), while the average time used for creating the MEDLINE index with the original Lucene library is 913,340 seconds (more than 10 days); the multithreaded index creation achieved a speed-up of 3.64 over the original Lucene library on the same computer with 4 CPU cores. The resulting index of the MEDLINE database occupies 30.4 GB disk space. Since our proposed method computes the indices for 8 groups of files separately and merge the partial indices into the final index, we can also update the index weekly through merging the index of the newly posted documents with the existing index. The newly posted documents can be downloaded from MEDLINE database [23].

Ontology-graph based query expansion scheme
Query expansion is widely used to reconstruct a seed query by adding extra related words to the input query with the purpose of matching additional related documents. It is helpful to retrieve potential relevant documents not indicated by initial query [24-27]. After getting the input query from the user, G-Bean expands the query with additional related words retrieved from the constructed ontology graph. The query expansion process first uses MetaMap 2013 [28] to map the input query to concepts in UMLS to find the CUIs representing the input text. These mapped CUIs are marked as Original CUIs and are used as the initial teleportation probability vector for the Personalized PageRank algorithm, which is applied over the ontology graph to calculate the Personalized PageRank Values (PPV) for concepts in the ontology graph [29-31]. The PPV represents the relevance of a concept to the query. We call these CUIs labeled with their PPVs as PPV CUIs. The top 500 ranked PPV CUIs are selected as the concept candidates for query expansion.

Because the original CUIs are used as the initial teleportation probability vector in PPV computation, these CUIs will always have high PPV scores and will always

be ranked at the top of the PPV CUI list. Thus, the direct use of PPV CUIs for query expansion does not make any significant difference from simply using the Original CUIs. Furthermore, Personalized PageRank algorithm has a tendency of ranking the general concepts (frequently occurred) higher than more specific concepts since there are more links to general concepts. Thus, simply selecting the top concept candidates from the PPV CUI list for query expansion might greatly decrease the query accuracy since more general terms are included in the expanded query. To alleviate this problem, we employ a *TF-IDF* weighting scheme to re-rank the PPV CUIs. The OHSUMED documents, a clinically-oriented MEDLINE subset which consists of 348,566 documents covering references from 270 medical journals, are used to build the *IDF* repository to estimate the popularity of the PPV CUI in all OHSUMED documents. We calculate a weight value for the concept ranked #i among the PPV CUI list using formulas shown in (2) and (3):

$$idf_i = \max \left\{ 0, \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right) \right\} \quad (2)$$

$$w_i = p_i^{\gamma} \cdot idf_i \quad (3)$$

where w_i is the weight, p_i is the L_1-normalized PPV score for this concept (the calculation of PPV score for concept is discussed in [32]), $\gamma \in [0,1]$ is a tuning parameter used to increase the p_i^{γ} by decreasing γ, idf_i represents the inverse document frequency of this concept, N is the total number of documents in *IDF* repository, and n_i specifies the number of documents in *IDF* repository that contain this concept.

The PPV CUIs are re-ranked by their weights and top ranked ones are used to expand the query. Extensive experimental results show that the query expansion scheme of G-Bean outperforms the popular Lucene approach by 22% while other existing query expansion approaches are unable to beat the free-text based Lucene strategy [32].

Document retrieval
When a query is entered, query text is analyzed by the same Lucene analyzer used to create the index, with an enhanced MIT stop-list and the Porter stemmer to extract query terms. MetaMap 2013 is used to map the query text into Metathesaurus CUIs. The query is expanded with the ontology-graph based query expansion scheme discussed in the previous section and the expanded query is submitted to G-Bean for searching the relevant documents.

After the user reviews a returned article, he/she can indicate if he/she is interested in the article. G-Bean can form a new query using the key concepts automatically obtained from all articles that are interested by the user

and retrieve a list of new articles that are relevant to all articles selected by the user. Therefore, the user does not need to browse through the long list of initial search results to find new articles related to their interested articles. PubMed can also display a list of recommended articles when a user is viewing one particular article. However, PubMed can recommend the articles only related to the current viewing article based on keywords matching. Thus, the recommendation may not accurately reflect the user's true search intention. In addition, keywords matching may return inaccurate results due to polysemy problem of natural language; it may miss some relevant articles due to the synonymy problem of natural language. G-Bean utilizes the ontology-graph based query expansion to minimize these problems. With more articles selected, G-Bean can accurately determine the user's true search intention by analyzing the articles that the user is interested in and provides better recommendation, especially for interdisciplinary research articles. The major steps of a G-Bean search are shown in Figure 1.

G-Bean implementation
We have implemented and published G-Bean as a Web-based application which accepts any biomedical related user query and returns related articles in MEDLINE database. We use the Client-Server architecture powered by Java Servlet Pages (JSP) to implement the G-Bean system since the Java version of Lucene library was used to index the MEDLINE documents. The communication between the client and the server follows the HyperText Transfer Protocol (HTTP). The architecture of G-Bean consists of client-side and server-side components, as shown in Figure 2.

Client-side implementation
A G-Bean client is developed as a Web application. The JSP script collects the query from the user, dispatches it to the server and displays the search results. The user interface of G-Bean is illustrated in Figure 3. Search results are presented in three areas under the search bar: the left area lists the articles returned by user's initial query; the top right area lists the articles which the user is interested in and the bottom right area lists the articles related to all articles the user is interested in.

Asynchronous **J**ava**S**cript **A**nd **X**ML (Ajax) technique is employed to implement the following functions due to its ability to make partial page updates without reloading the whole page. *1) Document retrieval:* G-Bean provides an easy-to-use interface for user to retrieve documents related to the query from the MEDLINE database. Clicking the "Search" button triggers the server to retrieve articles related to the input query; *2) Document ranking:* the returned articles in the left area are ranked by relevance to the query by default. G-Bean allows the users to rank the returned articles by date, author name and title
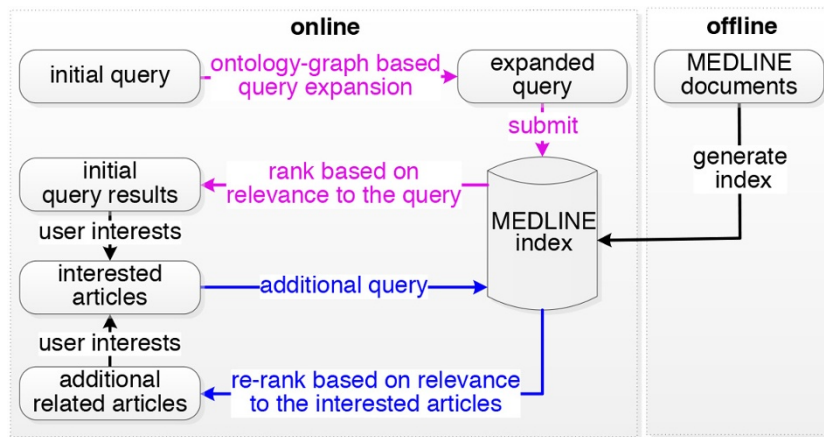
Figure 1 Major steps of search process in G-Bean.

as well. As shown in Figure 3, the user can select the ranking criterion in the drop-down list A to rank the returned articles according to his/her need; 3) *Document preview:* for a particular article, clicking the "Abstract" link (B in Figure 3) allows the user to preview the abstract; 4) *PubMed article retrieval:* G-Bean provides the PubMed link for each returned article, which allows the user to retrieve information about the article in PubMed. Clicking the "PubMed" link (C in Figure 3) opens a new window to display the PubMed record for this article; 5) *User intention discovery:* G-Bean allows the user to select articles of interest so that it can capture the user's search intention and to retrieve a list of articles relevant to all interested articles. If the user is interested in a returned article after viewing it, he/she can click the "Like" link (D in Figure 3) to add this article to the article list of interest (F in Figure 3); if a user changes his/her mind and wants to remove an article

from the list F, he/she can click the "Remove" link (E in Figure 3); 6) *Additional related articles update:* as long as the articles in list F are updated, the server retrieves articles related to those articles. The newly retrieved articles are presented in the bottom right area (G in Figure 3). These articles are ranked by the relevance to all articles in list F by default. Users can select to rank the newly retrieved articles by date, author name and title as well.

Server-side implementation

G-Bean uses Server Applet (Servlet) to receive and respond to requests from clients via HTTP; Apache Tomcat is used as the Servlet container to manage the Servlet. The data flow at the server-side is shown in Figure 4. The document index is created offline by our proposed multi-thread process. The online part of the server side functions includes query expansion, key concept extraction, and search results ranking.
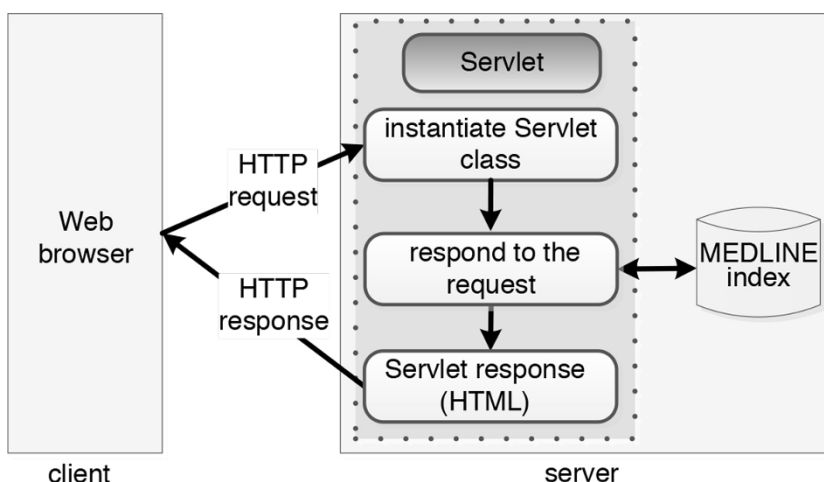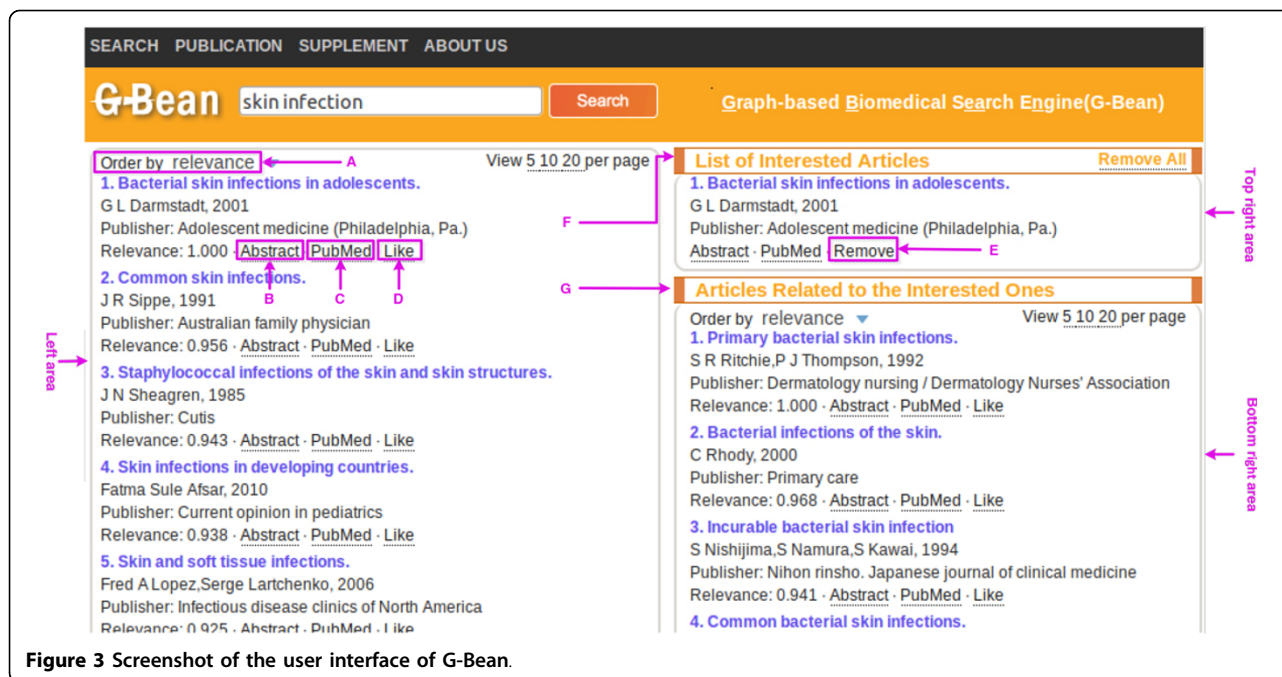

Figure 2 The architecture of G-Bean.

Figure 3 Screenshot of the user interface of G-Bean.

Results

To evaluate G-Bean's search performance, we conducted a subjective evaluation using the 106 benchmark queries from the OHSUMED dataset, which is generated by clinicians in course of their patient care. The 106 queries consist of patient information (a brief statement about the patient) and information need (a clinician's information request statement for the patient) fields [33].

We invited 20 graduate students in Clemson University to use these 106 benchmark queries to search the MEDLINE citations through G-Bean and PubMed respectively. The returned results on both search engines are set to be ranked by their relevance to the query. The students carefully examined the results returned by both search engines for each query, and decided independently which search engine produced more relevant search results. For a given query, they were asked to choose one of the following three answers after carefully reviewing the search results:

(a) G-Bean returns better search results than PubMed does;
(b) PubMed returns better search results than G-Bean does;
(c) G-Bean and PubMed return similar search results.

After collecting all answers from the students, we summarize the subjective search performance comparison between G-Bean and PubMed for each query into 5 categories:

- G-Bean and PubMed return similar search results;

- G-Bean is definitely better than PubMed;
- G-Bean is marginally better than PubMed;
- PubMed is definitely better than G-Bean;
- PubMed is marginally better than G-Bean.

Given a query, let n_a, n_b, n_c denote the number of students who chose (a), (b), and (c) respectively. We consider G-Bean and PubMed return similar search results iff $n_c \geq 10$ or $n_a = n_b$. Otherwise, we deem:

- G-Bean is definitely better than PubMed iff $n_b = 0$ or $(n_a - n_b)/n_b \geq 25\%$;
- G-Bean is marginally better than PubMed iff $n_b > 0$ and $0 < (n_a - n_b)/n_b < 25\%$;
- PubMed is definitely better than G-Bean iff $n_a = 0$ or $(n_b - n_a)/n_a \geq 25\%$;
- PubMed is marginally better than G-Bean iff $n_a > 0$ and $0 < (n_b - n_a)/n_a < 25\%$.

Our summary, as shown in Table 2, indicates that G-Bean returned definitely better search results in 67 of these benchmark queries and marginally better search results in 12 of these benchmark queries, while PubMed retuned definitely better results in only 7 of these queries and marginally better results in 1 of these queries. Overall, G-Bean returned better search results in 79 of these benchmark queries while PubMed returned better search results in only 8 of these benchmark queries. For the remaining 19 queries, these two search engines returned similar search results. This subjective evaluation confirms the efficiency of G-Bean search engine on biomedical
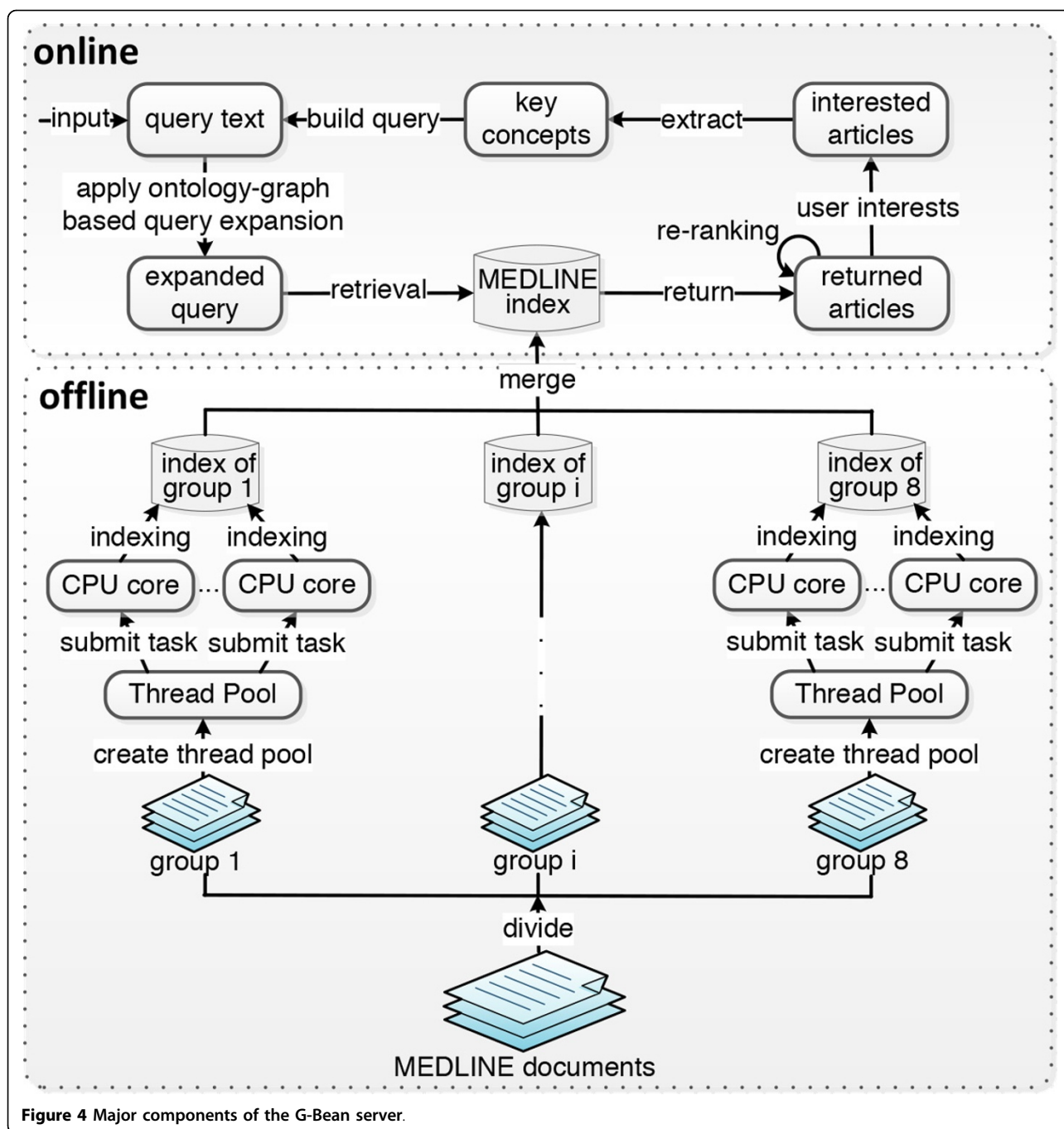
Figure 4 Major components of the G-Bean server.

Table 2 Performance comparison between G-Bean and PubMed using the OHSUMED 106 queries

Number of queries that G-Bean returned definitely better results	67
Number of queries that G-Bean returned marginally better results	12
Number of queries that PubMed returned definitely better results	7
Number of queries that PubMed returned marginally better results	1
Number of queries that the two search engines returned similar results	19

information retrieval. The details of the subjective evaluation experiment are available at http://bioir.cs.clemson. edu:8080/BioIRWeb/supplement.jsp.

It is worth-noting that no student could find any relevant article using PubMed for some queries such as queries #17, #52, and #95. For some other queries, such as queries #23, #49, #71 and #89, PubMed only returned one result in each case. Pursuing further investigation, we found that PubMed assumed "AND" operators for

keywords in a query string. For instance, in query #17: "Rh isoimmunization, review topics", PubMed obtained four keywords, *Rh, isoimmunization, review*, and *topics*. It assumed "AND" operation on these keywords to form the query for searching the MEDLINE database, i.e., it tried to retrieve articles containing all these four keywords. As a result, PubMed returned no result for OHSUMED query #17 as shown in Table 3 because there is no article in MEDLINE database containing all these four keywords. Unfortunately, none of the graduate students found this problem in their evaluation of PubMed search interface, nor did they figure out how to get a better search result using PubMed. Actually, if we take out the keyword "topics" from the query #17 and submit it to PubMed, it returns relevant articles as shown in Table 4. For an experienced user, it is not very hard to form a proper query string (with some Boolean operators) to search the intended articles. However, for a novice user, such as a graduate student, it is frustrating when the query returns no search results. In an extreme situation, if a user happens to input a keyword not in any of the articles, e.g., a typo, no result will be returned. In addition, PubMed uses MeSH to index documents. If a query contains no MeSH terms,

PubMed may either return no search results or return irrelevant results after a long period of search. On the other hand, as shown in Table 3, G-Bean returned articles closely related to the query in most of the cases. We note that these search results are obtained in January, 2014. Since MEDLINE database updates periodically, search results obtained later may be different from those reported in this paper.

Conclusions

In this paper, we develop G-Bean for biomedical information retrieval from MEDLINE database. Four major ontologies, MeSH, SNOMEDCT, CSP and AOD, which cover all concepts in NLM database, are used to build an ontology graph. An ontology graph based query expansion scheme is used to expand the input query with additional more specific query terms to retrieve relevant articles more accurately. To address the weakness of manual indexing mechanism used in PubMed, we use a multithreaded parallel program to speed up the index creation so that we can timely update the document index for information retrieval. By discovering the user's true search intention based on articles he/she is interested in, G-Bean shows significant advantage in retrieving relevant articles compared to PubMed's search interface. Our performance study shows that search results returned by G-Bean are more relevant than those returned by PubMed using the 106 benchmark queries from OHSUMED dataset.

Table 3 Top 5 returned results for query #17 "Rh isoimmunization, review topics" in G-Bean and PubMed

Rank	G-Bean		PubMed	
	Title	PMID	Title	PMID
1	A review of 58 Rh-isoimmunized cases.	3021092	No item returned.	
2	Suppression of Rh isoimmunization. A review.	101918		
3	Isoimmunization in pregnancy.	15145363		
4	Rhesus isoimmunization in twin gestation.	6433714		
5	Isoimmunization with anti-U antibody.	6716502		

Table 4 Top 5 returned results for query "Rh isoimmunisation, review" in PubMed

Rank	PubMed	
	Title	PMID
1	Postpartum Rh immunoprophylaxis	23168770
2	Performance in appropriate Rh testing and treatment with Rh immunoglobulin in the emergency department	22153971
3	Diagnostic laboratory technologies for the fetus and neonate with isoimmunization	21641488
4	Management and prevention of red cell alloimmunization in pregnancy: a systematic review	23090532
5	Prediction of the rate of decline in fetal hemoglobin levels between first and second transfusions in red cell alloimmune disease	22949399

Conference on Bioinformatics and Biomedicine (BIBM 2013): Bioinformatics. The full contents of the supplement are available online at http://www.biomedcentral.com/bmcbioinformatics/supplements/15/S12.

Authors' details
[1]School of Computing, Clemson University, Clemson, SC, 29634, USA. [2]Barnes and Noble, LLC, New York City, NY, 10007, USA. [3]Department of Computer Science & Software Engineering, College of Science and Engineering, Seattle University, Seattle, WA, 98122, USA. [4]Department of Computer Science, University of Illinois at Chicago, Chicago, IL, 60607, USA.

References
1. MEDLINE. [http://www.nlm.nih.gov/bsd/pmresources.html].
2. Islamaj DI, Murray GC, Névéol A, Lu Z: **Understanding PubMed® user search behaviour through log analysis.** *Database (Oxford)* 2009, **2009**: bap018.
3. PubMed. [http://www.ncbi.nlm.nih.gov/pubmed].
4. Hersh WR: **Information retrieval: a health and biomedical perspective.** New York, NY: Springer; 2009.
5. Bernstam E: **MedlineQBE (Query-by-Example).** In *Proceedings of Amia Symposium* Suzanne Bakken 2001, 47-51.
6. Haynes RB, McKibbon KA, Walker CJ, Ryan N, Fitzgerald D, Ramsden MF: **Online access to MEDLINE in clinical settings: A study of use and usefulness.** *Annals of Internal Medicine* 1990, **112(1)**:78-84.
7. McKibbon AK, Walker-Dilks CJ: **The qualify and impact of MEDLINE searches performed by end users.** *Health Libraries Review* 1995, **12(3)**:191-200.
8. Wildemuth BM, Moore ME: **End-user search behaviours and their relationship to search effectiveness.** *Bulletin of the Medical Library Association* 1995, **83(3)**:294-304.
9. Bradigan PS, Mularski CA: **End-user searching in a medical school curriculum: an evaluated modular approach.** *Bulletin of the Medical Library Association* 1989, **77(4)**:348-356.
10. Lipscomb CE: **Medical subject headings (MeSH).** *Bulletin of the Medical Library Association* 2000, **88(3)**:265-266.
11. Metathesaurus 2013AA. [http://www.nlm.nih.gov/pubs/techbull/mj13/mj13_umls_2013aa_release.html].
12. Hersh WR, Buckley C, Leone TJ, Hickam DH: **OHSUMED: an interactive retrieval evaluation and new large test collection for research.** In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: 3-6 July 1994.* Dublin, Ireland;W. Bruce Croft, C. J. van Rijsbergen 1994:192-201.
13. Hersh WR: **Report on the TREC 2004 genomics track.** *ACM SIGIR Forum* 2005, **39(1)**:21-24.
14. Yoo S, Choi J: **Improving MEDLINE document retrieval using automatic query expansion.** In *Proceedings of Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries: 10-13 December 2007* Edited by Dion Hoe-Lian Goh, Tru Hoang Cao, Ingeborg Torvik Sølvberg, Edie Rasmussen. Hanoi, Vietnam **4822**:241-249.
15. Abdou S, Savoy J, Ruch P: **Evaluation of stemming, query expansion and manual indexing approaches for the genomic task.** In *Proceedings of the Fourteenth Text Retrieval Conference: 15-18 November 2005.* Gaithersburg, Maryland;E. M. Voorhees, Lori P. Buckland 2005:863-871.
16. Zhang Y, Dong L, Li L, Srimani PK, Yu PS, Wang JZ: **G-Bean: an Ontology-graph based Web Tool for Biomedical Literature Retrieval.** In *IEEE International Conference on Bioinformatics and Biomedicine: 18-21 December 2013.* Shanghai, China;Sunghoon Kim, Guo-Zheng Li, Habtom Ressom 2013:623.
17. Humphreys BL, Lindberg DA, Schoolman HM, Barnett GO: **The unified medical language system: an informatics research collaboration.** *Journal of the American Medical Informatics Association* 1998, **5(1)**:1-11.
18. Chvatal V: **A greedy heuristic for the set-coving problem.** *Mathematics of Operations Research* 1979, **4(3)**:233-235.
19. UMLS MetamorphoSys. [http://www.nlm.nih.gov/pubs/factsheets/umlsmetamorph.html].
20. **2014 MEDLINE®/PubMed® Data Files.** [http://www.nlm.nih.gov/bsd/licensee/access/medline_pubmed.html#baseline].
21. McCandless M, Hatcher E, Gospodnetić O: *Lucene in Action* Greenwich, CT: Manning Publication; 2010.
22. Pyarali I, Spivak M, Schmidt R: **Evaluating and optimizing thread pool strategies for real-time CORBA.** In *Proceedings of the 2001 ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems: 18-18 June 2001.* Snowbird, UT, USA;Rastislav Bodik, Vugranam C. Sreedhar 2001:214-222.
23. MEDLINE Update Files. [http://www.nlm.nih.gov/bsd/licensee/access/medline_pubmed.html#update].
24. Voorhees EM: **Query expansion using lexical-semantic relations.** In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and development in Information Retrieval: 3-6 July 1994.* Dublin, Ireland;Bruce W. Croft, C. J. van Rijsbergen 1994:61-69.
25. Lu K, Mu X: **Query expansion using UMLS tools for health information retrieval.** In *Proceedings of the 2009 Annual Meeting of the American Society for Information Science and Technology: 6-11 November 2009* Andrew Grove. Vancouver, British Columbia, Canada **46(1)**:1-16.
26. Jalali V, Borujerdi M: **Information retrieval with concept-based pseudo-relevance feedback in MEDLINE.** *Knowledge and Information Systems* 2011, **29**:237-248.
27. Cao G, Nie J, Gao J, Robertson S: **Selecting good expansion terms for pseudo-relevance feedback.** In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: 20-24 July 2008.* Singapore;Syung Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani 2008:243-250.
28. Aronson AR, Lang FM: **An overview of MetaMap: historical perspective and recent advances.** *Journal of the American Medical Informatics Association* 2010, **17(3)**:229-236.
29. Haveliwala TH: **Topic-Sensitive PageRank.** In *Proceedings of the 11th International Conference on World Wide Web: 7-11 May 2002.* Hawaii, USA; David Lassner, Dave De Roure, Arun Iyengar 2002:517-526.
30. Haveliwala TH, Kamvar SD, Jeh G: **An analytical comparison of approaches to personalizing PageRank.** *Stanford University Technical Report* Stanford; 2003.
31. Agirre E, Soroa A: **Personalizing PageRank for word sense disambiguation.** In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: 30 March-3 April 2009.* Athens, Greece;Claire Gardent, Joakim Nivre 2009:33-41.
32. Dong L, Srimani PK, Wang JZ: **Ontology graph based query expansion for biomedical information retrieval.** In *IEEE International Conference on Bioinformatics and Biomedicine: 12-15 November 2011.* Atlanta, GA;Shinichi Morishita, Fangxiang Wu, Mohammed Zaki 2011:488-493.
33. OHSUMED queries. [http://ir.ohsu.edu/ohsumed/ohsumed.html].