The VLDB Journal (2009) 18:277–301
DOI 10.1007/s00778-008-0099-9

REGULAR PAPER

Context-based literature digital collection search

Nattakarn Ratprasartporn · Jonathan Po · Ali Cakmak · Sulieman Bani-Ahmad · Gultekin Ozsoyoglu

Received: 27 June 2007 / Revised: 18 February 2008 / Accepted: 22 February 2008 / Published online: 26 March 2008

Abstract We identify two issues with searching *literature digital collections within digital libraries*: (a) there are no effective paper-scoring and ranking mechanisms. Without a scoring and ranking system, users are often forced to scan a large and diverse set of publications listed as search results and potentially miss the important ones. (b) Topic diffusion is a common problem: publications returned by a keyword-based search query often fall into multiple topic areas, not all of which are of interest to users. This paper proposes a new *literature digital collection* search paradigm that effectively ranks search outputs, while controlling the diversity of keyword-based search query output topics. Our approach is as follows. First, during pre-querying, publications are assigned into pre-specified ontology-based contexts, and query-independent context scores are attached to papers with respect to the assigned contexts. When a query is posed, relevant contexts are selected, search is performed within the selected contexts, context scores of publications are revised into relevancy scores with respect to the query at hand and the context that they are in, and query outputs are ranked within each relevant context. This way, we (1) minimize query output topic diversity, (2) reduce query output size, (3) decrease user time spent scanning query results, and (4) increase query output ranking accuracy. Using genomics-oriented PubMed publications as the testbed and Gene Ontology terms as contexts, our experiments indicate that the proposed context-based search approach produces search results with up to 50% higher precision, and reduces the query output size by up to 70%.

Keywords Context-based search · Digital collections · Ontology · Context score · Ranking

N. Ratprasartporn (✉) · J. Po · A. Cakmak · S. Bani-Ahmad · G. Ozsoyoglu
Department of Electrical Engineering and Computer Science, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106, USA
e-mail: nattakarn@case.edu

J. Po
e-mail: jlp25@case.edu

A. Cakmak
e-mail: cakmak@case.edu

S. Bani-Ahmad
e-mail: sulieman@case.edu

G. Ozsoyoglu
e-mail: tekin@case.edu

1 Introduction

At the present time, literature digital collection search queries have two problems. First, ranking mechanisms/functions for searched and located literature publications are either ineffective or do not exist, forcing users to scan potentially large numbers of publications and possibly missing important ones. As examples, well-known literature digital collection search portals, such as the ACM Portal [32] and Google Scholar [11], use only simple text-based and/or citation-based scores to rank search results, and rankings are hardly useful. PubMed [1], which contains more than 14 million life sciences publications, lacks a paper-scoring system to rank papers satisfying a keyword search. PubMed simply lists search results in descending order of their PubMed ids or publication years. Second, topics of publications returned as a response to a keyword-based publication search query are often diverse, and returned publications routinely fall into multiple topics, leading to the problem of *topic diffusion* across search results. Clearly, some of these topics may not be of interest to users.

In order to (i) effectively rank query output publications of literature digital collection keyword-based search queries, and (ii) provide controlled ways of eliminating query output topic diversity, we propose a new literature digital collection searching paradigm, called *Context-Based Search* (CBS) approach, as follows:

1. We perform two query-independent pre-processing steps before any query session starts: assign publications into pre-specified and possibly multiple ontology-based contexts; and compute *context* (*importance*) *scores* for papers. Therefore, each context contains two types of information: (i) the *paper set* of the context and (ii) the context score of each paper.
2. Then, at search time, we perform the following steps.
   (a) Select search contexts automatically (or manually, by the user)
   (b) Perform keyword-based search within the selected contexts, and
   (c) Within each context, compute relevancy scores of located publications, re-rank search results, and return the located publications

With the CBS approach, (i) search input includes only papers residing in the selected contexts as opposed to all papers (ii) search output is enhanced by a highly useful context-based paper classification, (iii) topic diffusion across search results is controlled, and (iv) query output sizes are reduced to include only search results in the contexts of interest.

Since the CBS approach performs a search within selected contexts, some important results might be missing if they are not in the selected contexts. As an alternative, step 2 of the CBS approach is modified to include all search results (*CBS_all*) as follows:

(a) Select search contexts automatically (or manually, by the user)
(b) Perform keyword-based search across all papers to select the publications to be returned
(c) For the returned papers that reside in the selected contexts, compute relevancy scores of these papers in each context. For the papers that do not reside in any selected contexts, group those papers together into a "remainder context"
(d) Re-rank search results and return the located publications

The CBS (and CBS_all) approach can be characterized as applying the *Context-Selection-First* strategy (i.e., step 2(a)). We compare the CBS approach with an alternative approach, called *Search-and-Distribute to Contexts*(SDC) approach:

1. Perform step 1 of the CBS approach
2. Perform keyword-based search across all the papers to select the publications to be returned, and to select the involved contexts (based on whether or not they contain a query output paper)
3. Re-rank the selected publications within each located context

In comparison with the current practice, the CBS approach ranks papers based on the contexts of interest, increasing the accuracy and consistency of the ranking. In comparison with the SDC approach, the emphasis of the CBS approach is on locating interesting contexts first, as opposed to presenting all contexts that the results are in. The SDC approach is comparable to existing systems that classify or cluster search results to all possible relevant contexts [15,26,48–50] (see Sect. 3 for more details).

Finally, adapting an approach from [22], we also add user interaction to context-based search. After each query execution, the context-based search engine returns a hierarchical view of the involved contexts and per-context search results as separately ranked lists of papers. After viewing the results, the user can drill down/up in the context hierarchy, select new contexts for the next query, and resubmit the revised query.

We have evaluated the CBS and the SDC approaches using genomics-related PubMed publications as the test literature digital collection instance. To define the contexts, or, more correctly, *context hierarchies*, one can use two alternatives.

- Use an existing (in our test case, biomedical) ontology for PubMed-related contexts, or
- Automatically locate a context hierarchy from the existing publications of a literature digital library

In this paper, we have chosen the first approach. At the present time, within the biomedical domain, there are a large (>40) number of ontologies, defined and curated by domain experts and available online—to reflect the needs of different Life Sciences communities [7]. Within the genomics domain, Gene Ontology (GO) [2] is a well-known and highly useful biomedical ontology, used for annotating genes and gene products, and for various genomics-related data mining tasks. GO consists of three separate controlled vocabularies describing gene and gene product functionalities, one of which is cellular activities. To evaluate our context-based approach, we select the GO hierarchy as our context hierarchy. In other words, a context in our environment refers to a GO term, and a given context paper set is pre-located based on the semantic properties of the GO term.

Contributions of this paper are:

- We propose query-independent algorithms and techniques to (a) automatically populate the paper set of a GO-specific context, and (b) compute context scores of papers.
- To apply the context-based search to other publication domains and ontologies, we present ways of generalizing our approach to non-domain-specific contexts. In other words, we provide algorithms that do not utilize specific semantic properties of GO and PubMed.
- To assist users in context selections, we present non-domain-specific approaches to automatically select search contexts for a given query.
- After selecting contexts of interest, we present alternatives to perform context-based search and to rank search results within the contexts.
- While we recommend the context-based search result format (i.e., search results are presented as groups within their corresponding contexts), we also provide an algorithm to merge search results from multiple contexts into a single result set.
- Using a large set of different keyword-based query types, we evaluate the accuracy of the context-based approach via recall and precision analysis. To calculate recall and precision in an automated manner, we present an approach to locate the *A(rtificially)C(onstructed)- answer set* for a given query without any human expert help. To ensure the accuracy of the AC-answer set approach, we present a manual verification of its correctness.

For evaluation, a digital collection database is populated with 72,027 genomics-based PubMed papers [8], and the papers are assigned to one or more GO terms. Keyword-based searches are performed within automatically selected contexts, and context-based search (CBS and CBS_all) results are compared to (i) the SDC results, and (ii) the current PubMed keyword-based search (PM) results. Experimental results show the following:

- The CBS approach reduces the query output size by up to 70% as compared to the PM approach. Compared with the SDC approach, the CBS approach reduces the search output size by up to 30%.
- The number of contexts returned from the SDC approach (i.e., contexts that all of the search results reside) is very large. Therefore, it is not practical for the user to navigate through search results using this large set of contexts. On the other hand, The CBS approach selects only contexts that are relevant to the query; thus, the number of contexts involved in the CBS approach is approximately 10 times less than the SDC approach.
- In addition to reducing search result diversity and size, the CBS approach produces accurate results. This is signified by the fact that the CBS approach produces search results with comparable recall when considering a large set of search results and up to 50% higher precision for high ranking papers, as compared with the PM approach. Compared with the SDC approach, recall and precision are not significantly different. However, as mentioned before, the number of search results and involved contexts are significantly reduced.

Section 2 is an overview of our context-based search approach. Section 3 summarizes and compares our approach with the related work. Section 4 presents techniques to populate contexts with papers. In section 5, we describe methods to automatically select search contexts for a given query term. Section 6 explains alternatives to search and rank search results within contexts. Section 7 presents ways of merging results from multiple contexts. Sections 8 and 9 present the experimental setup and experimental results, respectively. Section 10 concludes.

2 Overview

The following sequence of algorithms is used to perform our proposed context-based search approach:
(1) *Populate_Contexts*: Figure 1a, b presents two algorithms, namely *text-based* and *pattern-matching-based* approaches, which are employed to locate the paper set of a context (the details are in Sect. 4).
Note that *Populate_Contexts* is pre-executed and not dependent on queries.
(2) *Evaluate_Query*
(2.1) Select_Query-Contexts: Keywords specified in a keyword-based search query constitute the search terms. Using the search terms, we first select the contexts to search for. Figure 2a, b summarize two algorithms, namely, *text-based* and *context-keyword-based* query context selection algorithms (the details are in Sect. 5).
(2.2) Perform_Search_per_Selected_Context: The search within each context is performed using a text-based similarity measure between the given query term and papers in the selected query context. And, publication results are ranked separately within each context using their relevancy (scores) to the query for each context. The relevancy score of a paper in a context is defined as a combination of the paper-to-query matching score and the pre-computed context score of the paper. Figure 3 presents two algorithms (CBS and CBS_all) to perform search, and rank search results within contexts (the details are in Sect. 6)
(2.3) Merge_Query_Results: When multiple contexts are selected for search, results are displayed separately under different contexts. In the case that the user wants to merge these results, a merging function which assigns only one aggregate score to each paper is presented. The new aggregate score of

Algorithm Text-based Paper-Context Assignment
Input: t: similarity threshold
Output: context[]: an array of assigned context-papers
for each context c_i **do begin**
  select representative paper(s) that best characterizes c_i;
  for each publication p in the database **do**
    for each representative paper r **do**
      if sim(r, p) $\geq$ t **then** add p to context[i];
endfor

(a)

Algorithm Pattern-Matching-based Paper-Context Assignment
Input: t: minimum context-engagement threshold
Output: context[]: an array of assigned context-papers
for each context c_i **do**
  construct patterns from c_i's training papers;
  for each paper p in our database **do begin**
    for each pattern pt **do**
      if p contains pt **then**
        p.score(c_i) = p.score(c_i) + pt.score;
    if p.score(c_i) $\geq$ t **then**
      add p to context[i];
endfor
endfor

(b)

Fig. 1 Algorithms: **a** Text-based and **b** Pattern-Matching-based Paper-Context assignments

Algorithm Select_Text-based_Query_Context
Input:
  q: query term (possibly multiple words)
  t: similarity threshold
  //t is selected by the CBS system.
  $w_{centroid}$, $w_{threshold}$: similarity weights
Output:
  A set of query contexts for query q
for each context c_i **do begin**
  //similarity between the centroid of c_i and q
  compute $sim_{centroid}(c_i, q)$
  //similarity between the context term and q
  compute $sim_{ContextTerm}(c_i, q)$
  if $\left(\begin{array}{l} w_{centroid} \cdot Sim(C_{centroid}, q) \\ +w_{GOterm} \cdot Sim(C_{GOterm}, q) \end{array} \right) \geq t$
  then
    add c_i to query_context set;
endfor

(a)

Algorithm Select_Context-Keyword-based_Query_Context
Input:
  context_keyword[c_i]: a list of keywords representing context c_i, $1 \leq i \leq m$
  k: an occurrences threshold
  //k is selected by the CBS system.
  q: query term (possibly multiple words)
Output: A set of query contexts for query q
for each context c_i **do**
  if k% of words in q appear in context_keyword[c_i] **then**
    add c_i to query_context set;

(b)

Fig. 2 Algorithms: **a** Text-based and **b** Context-Keyword-based query context selection

a paper is computed using (a) the relevancy score of the paper to the query in each context, and (b) the similarity between each context and the query. Figure 4 presents the algorithm used to combine search results from different contexts (the details are in Sect. 7).

The context-based search paradigm is evaluated using recall and precision analysis of multiple queries. To compute recall and precision in an automated manner, we employ the *A(rtificially)C(onstructed)-answer set* of a given query as follows. A standard keyword-based search with a high threshold is used to find an initial answer set, which is then enlarged iteratively using text-based (the first approach in Sect. 8.4.1) and citation-based (the second approach in Sect. 8.4.1) expansions. In the text-based expansion, papers that are sufficiently similar to a paper in the initial set are added to the AC-answer set. For the citation-based expansion, papers in the citation path of length at most k from a paper in the initial set and with high citation scores are included in the AC-answer set. We verified the correctness of using the AC-answer set as the true answer set manually, and have found it to be at least 95% accurate (Sect. 8.4.2).

Multiple search terms are selected to be used as query keywords in keyword-based queries (Sect. 8.3). We evaluate our approach based on the average and the median recall and precision scores of search queries, and evaluate our context-based approach and compare with existing approaches (Sect. 9).

3 Related work

Currently, there are many literature search systems available online, e.g., highly popular sites such as *Citeseer* [10] and *Google Scholar* [11], professional society-related sites such as *IEEE Xplore* [12], research prototype sites such as our *Case Explorer* [13,14]. These systems rank search results based on the relevancy to the query term and/or the importance of the papers, and do not use contexts to organize search results. Similarly, Chen et al. [66] present a ranking approach

Fig. 3 Algorithm: **a** CBS search and **b** CBS_all search

Algorithm Search_CBS
Input:
q: query term
query_contexts: list of selected query contexts
Output:
A set of search results within selected query contexts
for each paper p in each context c in query_contexts **do begin**
Compute sim(p, q) as a text-based similarity between p and q;
Compute relevancy_score(p, q, c) as a combination of sim(p, q) and the context score of p in c;
end for
Rank papers in each context in descending order of their relevancy_scores and return;

(a)

Algorithm Search_CBS_All
Input:
q: query term
query_contexts: list of selected query contexts
Output:
A set of search results within selected query contexts
for each paper p in the database **do begin**
Compute sim(p, q) as a text-based similarity between p and q;
for each paper p in each context c in query_contexts **do begin**
Compute relevancy_score(p, q, c) as a combination of sim(p, q) and the context score of p in c;
for each paper p not in query_contexts **do begin**
Assign p to a remainder context;
Use sim(p, q) as relevancy_score(p, q, remainder_context)
end for
Rank papers in each context in descending order of their relevancy_scores and return;

(b)

Fig. 4 Algorithm: Merge Query Results

Algorithm Merge_Results
Input:
q: query term
context_relevancy: list of similarity scores of each context to the query term
paper_context_relevancy: list of relevancy scores of each paper in each context to the query term
Output:
An array of merged results with new relevancy scores
for each paper p in the output of q **do begin**
for each selected context c_i where paper p resides **do begin**
Compute new_relevancy_score(p) as a combination of context_relevancy(c_i, q) and paper_context_relevancy(p, c_i, q);
Add p and new_relevancy_score(p) to the merged_result;
endfor
sort merged_results and return;

with utilization of contexts. Their method, first, separately searches distinct fields (e.g., title, abstract, authors, and publication venue) of publications and then, uses an artificial intelligence method (neural networks) to aggregate the computed similarity scores of all fields to user query.

In one contextual web search approach, a context is captured around the user-highlighted text, and augmented queries are created from the selected context words [24,25]. This approach is similar to our context-based search approach in the sense that users can specify contexts of interests before viewing search results. The main differences are that the contexts of this approach come from documents as opposed to a pre-defined ontology-based hierarchy, and no structural and hierarchical information are used.

Another technique, called TileBars [38,39], lets the user enter a query in a faceted format (i.e., each line represents each topic) and provides graphical bar in order to show the degree of match for each facet. TileBars illustrate which parts of each document contain which topic by dividing the bar into columns, where each column refers to a part in the document. The darkness of the square indicates the number of times the topic occurs in the part of the document. With this approach, the user can easily see the relevancy of the document to each specified topics. However, search results are shown as one list and no categorization of search results is provided.

A number of categorization techniques have been proposed to make search results more understandable. Two widely-used categorization techniques are document clustering and document classification. Document clustering creates categories (or contexts) by grouping similar documents together while document classification assigns documents to a set of predefined categories [40].

Document clustering can also be further classified as flat clustering and hierarchical clustering [26]. For flat clustering, Scatter/Gather [41] was one of the first clustering systems on top of the information retrieval engine. Scatter/Gather groups documents based on the similarities in their contents, where a list of commonly occurring words in the cluster is used to represent the cluster. Grouper [42] uses Suffix Tree Clustering (STC) that identifies sets of documents sharing common phrases. Lingo [43] uses singular value decomposition (SVD) to find meaningful labels for the clusters. Lingo 3G is an implementation of the Lingo project [59]. Findex [40] seeks for the most frequent words or phrases among search results and use them to define categories, which are displayed in a separate list beside the results. Zeng et al. [44] use a supervised learning method to improve the performance of search result grouping. All possible phrases are extracted from the titles and web-snippets, and a score of each phrase is computed using a regression model learned from previous training data. Phrases with high scores are used as cluster names. Desai and Spink [67] propose a clustering scheme to group documents by relevance. The motivation behind this study is that search query results (i.e. identified relevant documents) are usually topically diffused. And, the most highly relevant results may not always be listed at the top of the ranked list and partially relevant results can be scattered throughout the set. A major drawback of this study is that the clustering step is performed online, which negatively affects the scalability of a digital collection search engine. In contrast, our approach identifies clusters (or contexts) offline.

Another type of document clustering is hierarchical clustering, which automatically derives a context hierarchy from search results. SHOC [45] uses Suffix Array for extracting sentences and organizes context hierarchy using SVD. DisCover algorithm [46] identifies at each level of the hierarchy topics that maximize the coverage while maintaining the distinctiveness of the topics. CIIRarchies [47] builds statistical models of language to identify cluster terms in a document set and construct the hierarchy using a recursive algorithm. Several commercial systems, e.g., Vivisimo [48], Clusty [49], Mooter [50], and iBoogie [60], also automatically create hierarchical clusters of search results. Vivisimo is one of the best web-clustering systems; however, little descriptive information has been provided about this commercial software. Snaket [26] is an open-source system in the literature that achieves efficiency and efficacy performance close to Vivisimo. Snaket uses two knowledge bases to select and rank gapped sentences of variable length that are used as cluster labels, and uses a bottom-up hierarchical clustering algorithm to construct a folder hierarchy.

In addition to text clustering, Cha-Cha [51] and AMIT [52] use hyperlinks to create contexts. Cha-Cha creates a context hierarchy of search results by computing the shortest hyperlink path from a root page to every web page. AMIT shows all outgoing links from a starting root node.

Although clusters (or contexts) created from the clustering techniques are closely related to search results, automatically-constructed contexts are not as meaningful as well-defined human-generated context hierarchy [15,53]. Another drawback of the automatically-constructed contexts is that the user cannot select contexts of interest before viewing search results or modify search results beyond the constructed contexts.

Our context-based search approach is closely related to several existing information retrieval systems that utilize document classification techniques to improve the search experience. DynaCat [53] aims to answer the questions in the medical domain. A set of query types that cover typical kinds of medical queries are created. For each query type, category criteria are defined. DynaCat first maps the user's question to its manually-defined query types. Then, the returned documents are organized by comparing keywords (or index terms) of each document to the pre-defined category criteria. The hierarchy is formed based on the position of the categories in Medical Subject Headings (MeSH) [34] hierarchy. Compared to our context-based search approach, DynaCat is more specific in the sense that queries must be in the form of questions as opposed to general query terms. Moreover, DynaCat requires human-generated keywords for each paper, which are not always available.

Another IR system, Textpresso [54], utilizes 30 very shallow pre-defined categories, where parts of the categories are GO terms. When the user of Textpresso defines query keyword, he/she can select categories to make the search more specific. Search results that contain a particular word or phrase defined as a member of the selected categories are then returned. GOPubMed [15] utilizes the Gene Ontology hierarchy as a vehicle to navigate through search results. GoPubMed queries are first submitted to PubMed, and the corresponding PubMed paper "abstracts" are retrieved and categorized by Gene Ontology terms. GOPubMed categorization fully relies on the existence of GO term words in the abstracts. Additionally, GoPubMed does not rank results or provide importance scores for papers. The advantage of Textpresso over GOPubMed is that Textpresso users can define contexts of interest before viewing search results; therefore, search results are reduced to include only papers in the selected contexts. The superiority of GOPubMed is the use of full GO hierarchy as a tool to classify search results. Textpresso returns only one ranked list of relevant abstracts, and the ontology is not used to categorize search results. Our context-based search approach combines the advantages of both systems in that the user of our system can manually select search contexts or let our system automatically select the contexts that are relevant to query keywords. Then, the user of our system has options to view all search results as one

list or view only ranked search results in each individual context. Since the number of contexts involved with all search results can be very large (see Sect. 9.6), our context-based approach is superior to GOPubMed because we present only those contexts of interest, as opposed to all possible relevant contexts. Another problem of Textpresso and GOPubMed is that they find the context terms directly in the text during the classification process. As observed by the *PubMed Abstracts FullText Search Tool* [13], only 78% of the 14 million PubMed abstracts contain words occurring in a GO term (i.e., approximately 3 million PubMed papers have no GO term association). Our approach uses more sophisticated algorithms that allow the assignment of a relevant paper to a context even when the paper does not contain the context term (see Sect. 4).

Castells et al. [55] proposes the use of ontology to improve the accuracy of search results. Specifically, a document is annotated to an ontology concept if there are occurrences of the concept labels in the document. Scores of documents in contexts are assigned during the annotation using Term Frequency-Inverse Document Frequency (TF-IDF) of the context labels. An ontology-based query (i.e., RDQL [56]) is required as an input of the search. Conditions in the query are mapped to the ontology, and documents that are annotated with the matched ontology concepts are ranked and returned. Similar to our context-based approach, the assignment of documents with scores to contexts are done as a pre-processing step. However, unlike Castells' approach, the score of a paper in a context from our approach is not just a TF-IDF score of the context term (see Sect. 4.3). Moreover, we use regular keywords as an input of the search as opposed to a specific RDQL query. And, Castells' approach uses only an exact match when mapping the query to the contexts, while our approach utilizes more information (e.g., terms in the context's centroid, see Sect. 5) to rank and to increase the number of matched contexts. Another difference between our approach and Castells' approach is that only one ranked list of search results are allowed in Castells' approach, while our approach allows both merged and per-context results.

PageRank [5,6,20] and HITS [6] can be used for citation-based paper score computations. PageRank recursively determines the importance of a webpage (document) by the number of links (citations) to it and the ranks of the linking URLs (citing papers). Hyperlink Induced Topic Search (HITS) is based on two types of special documents: authorities and hubs. Authorities contain definitive high-quality information. Hubs are documents that link to authorities. The paper's HITS score is recursively determined. Unlike PageRank, HITS scores are query-dependent. Similar to the context score computation, Topic Sensitive PageRank [21] creates 16 topic-sensitive PageRank vectors with each vector biased by URLs in the top level of the Open Directory Project (ODP [33]). However, the hierarchical structure of the ODP is not considered during score computation.

Besides improving search relevance, the notion of contexts can also be applied in other applications. For example, in information theory, *information content* [19] of a context is computed by counting the number of objects (documents) assigned to the context (concept). When a context is mapped to a large set of objects, it is more general and less informative. Pedersen et al. [61] adapt both path length and information content approaches to measure the similarity and relatedness between concepts in the biomedical domain. The information content method is successfully applied to measure semantic similarity of two proteins based on their Gene Ontology annotations [62]. Maguitman et al. [63] utilize the information content technique to compute the semantic similarity between two documents that are stored in the graph-based structure (context) of the Open Directory Project ontology [33]. The notion of contexts is also successfully used in searching literature digital collections to find related publications of a given publication [64]. In this approach, the relatedness between papers is computed using the relatedness between contexts that the papers reside and the relatedness between the contexts and the papers.

4 Classifying papers with scores to contexts

This section presents approaches to assign papers to their relevant contexts, and to compute context scores of papers. Obviously, the manual categorization (e.g., the Open Directory Project [33] or Yahoo! Directory [57]) is highly accurate. However, manual assignment is not always available, and very time-consuming. Several IR systems [15,54,55] have used automatic approaches to classify documents to pre-defined contexts as discussed in Sect. 3.

In this section, we presents two approaches, namely *text-based* and *pattern-extraction-based* approaches, to automatically locate papers of contexts, called *p(aper)-clusters*.

The first approach uses text-based similarity measures, such as cosine similarity of the vector space model [9], to locate papers that are sufficiently similar to a given context. The text-based approach presented in this paper is adapted from the nearest neighbor learners technique [3], which classifies a paper to a context if the paper and the context's training paper(s) or the context's (semantic) centroid are sufficiently similar. The second approach constructs patterns from a context's training data set and uses those patterns to locate the p-cluster of the context. To provide a comparative assessment of papers in a context, we present three different context score functions. Finally, we generalize the approaches to non-domain-specific contexts.

4.1 Text-based measures for locating p-clusters

One way to assign papers to a context is to include only those papers with occurrences of the context term or with sufficiently high text-based similarity scores to the context term (or its synonyms). However, since a context is represented by a short phrase, which is much shorter than papers, incorrect assignments may occur. Instead of using the context term, our approach is first to select a *representative paper* or a set of representative papers that characterizes the context. Then, text-based similarity measures are applied to locate papers that are sufficiently similar to the representative paper(s) of the context. However, the representative paper might be long, and may contain paragraphs not relevant to the context. Our approach is to extract *significant paragraphs* of the representative paper and construct a "new revised representative paper" of the context by removing all the insignificant paragraphs.

After assigning papers to contexts, some papers in the database might not be associated with any context. These papers are assigned to contexts using an approach similar to the K-Means algorithm [3], i.e., contexts whose centroids are "sufficiently similar" to the remaining paper p are assigned as the contexts of p.

In a context hierarchy in general, descendant context terms are more specific than their ancestor context terms. Hence, p-clusters of descendant contexts are also relevant to ancestor contexts. Therefore, after the paper assignment to a context c, by default, all papers in the p-clusters of descendant contexts of c are also included in the p-cluster of c.

4.1.1 Selecting representative paper

A large number of genes have been annotated with GO terms by domain experts. And, for each annotation, an *evidence paper* and its corresponding *evidence code* are specified as a *support for the annotation*. We refer to all the evidence papers that are used to annotate different genes with a given GO term as the *evidence papers* (or, *base papers*) *of the GO term* itself. We then select a *representative paper* for a context from the *evidence papers* of that context. Then, papers in the database that are sufficiently similar to the representative paper of the context are assigned to the context's p-cluster.

For GO contexts with no base papers, the representative paper for the GO context is located as follows. GO terms at the same hierarchical path of the GO hierarchy are related to each other (mostly through the is-a relationship), and, thus, the base papers of the ancestor/descendant contexts of a given context are also relevant to that context. Therefore, for a GO context with no base papers, the base papers of its children and its parent contexts are selected. If no such papers exist, we move up and down one step in the hierarchy until we find base papers.

Among the base papers of a context, one paper is chosen to be a representative paper of the context as follows. First, we choose only base papers with the most reliable evidence codes of TAS (Traceable Author Statement) or IDA (Inferred from Direct Assay) [18]. If no such paper exists, all evidence papers are used. Afterwards, we use the context term *and* its synonyms to locate the representative paper. Intuitively, a paper whose text contains a large number of occurrences of the context term and its synonyms is highly relevant to the context. After counting the occurrences of the GO term and its synonyms in each base paper, our approach selects the paper with the highest *occurrences score*, which is defined as

$$Occurrences\_Score = \frac{n}{N} \tag{4.1}$$

where n is the number of times the context term appears in the paper, and N is the length of a vector representing the paper.

If there are no base papers containing all the context term words (perhaps because some terms are very long) then new shorter term(s) are constructed from the original term by

(i) Splitting the term words before and after a comma, conjunction, and preposition, and/or
(ii) Removing phrases within parentheses.

Example The term "signal transduction during filamentous growth" is split into two terms "signal transduction" and "filamentous growth".

With these new terms, *Occurrences_score* becomes

$$Occurrences\_Score = \frac{\sum_i n_i}{N} \tag{4.2}$$

where i represents each new term.

The representative paper has the highest *Occurrences_Score* among base papers, and thus best describes the context. However, it is possible to have a case where there are only few PubMed papers that are similar to the representative paper. To increase the number of papers in each context, base papers with k highest *Occurrences_Score* values are selected as the new set of representative papers.

4.1.2 Selecting significant paragraphs

Using a representative paper to locate the context's *p-cluster* may not always work: *Occurrences_Score* of the representative paper may be the highest among the base papers, but only a few paragraphs may be relevant to the context. An alternative approach is to extract and utilize only *significant paragraph*s as follows. First, context term and its synonyms are split to a set of significant words. After removing stopwords and frequent words, representative paper paragraphs with a significant word are chosen as significant paragraphs.

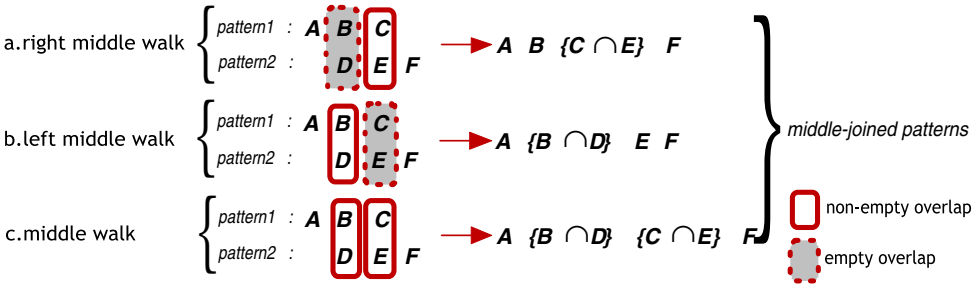

Fig. 5 Middle-joined pattern construction

Finally, significant paragraphs are combined together to form a "new" shorter representative paper. Papers in the database that are sufficiently similar to this new representative paper are then included in the p-cluster of the context.

Selecting significant paragraphs is completely automated. It is also domain independent since this task utilizes only concept names and synonyms (if available), and concept names are essential pieces of information that each ontology (or concept taxonomy) must retain.

4.2 Pattern extraction for locating p-clusters

This section presents a pattern extraction technique that constructs patterns from a context's training data set (i.e., base papers for GO-specific contexts). The constructed patterns are then used to assign papers to contexts. *Significant terms* (*phrases*), which are terms related to a context, are constructed from two sources: (i) words in the context term, and (ii) *frequent terms* (*phrases*) located in the training papers. During the frequent phrase construction, significant terms from each source are combined using a procedure similar to the apriori algorithm [16]. In order for a frequent phrase to be significant, the candidate phrase must have enough support, i.e., the ratio of training papers containing the phrase to the whole training set size. This procedure is repeated to construct larger frequent phrases until no more new phrases can be created.

Patterns are constructed from significant terms as follows. A pattern of a term consists of three tuples [4]: <Left> <Middle> <Right>, where each tuple is a set of words. Significant words (i.e., words in the significant terms) appearing in the training data are assigned to <Middle> tuple. Words surrounding the significant words are assigned to <Left> and <Right> tuples. The number of words for <Left> and <Right> tuples are determined by a window size.

Based on patterns constructed above, *extended* patterns are constructed by virtually walking from one pattern to another. Depending on the type of the walk, two different extended patterns are built: (i) *side-joined*, and (ii) *middle-joined* patterns.

A *side-joined* pattern is created when there is an overlap between the left tuple of one pattern and the right tuple of another pattern. E.g., if P_1 = <A><B><C> and P_2 = <C><D><E>, then the side-joined pattern P_3 =<A><B><C><D><E> is constructed.

A *middle-joined* pattern is created when there is an overlap between the middle tuple of one pattern and the left or right tuple of another pattern. Middle-joined pattern construction is illustrated in Fig. 5.

After constructing the context's patterns, papers containing pattern occurrences are added to the context's p-cluster. For contexts with no or few training papers, we use the context hierarchy. Descendant context's papers are included with the ancestor context. If the context is still empty, then the closest ancestor's paper set is assigned to the context. Since the ancestor of a context term is less *informative*, assigning papers from an ancestor context to one of its descendant contexts introduces a *decay of informativeness* for the context term. Informativeness of a context is approximated by its information content, $I(C)$, which is defined as [19]

$$I(C) = \log\left(\frac{1}{p(C)}\right) \tag{4.3}$$

where $p(C)$, the "relative size" of C in the context, is approximated by:

$$p(C) = \frac{(\#\text{ of C's Descendants})}{(\text{total \# of context terms})} \tag{4.4}$$

In order to quantify the *rate of decay*, we compare the information content, $I(C)$, of the descendant term (C_{desc})to its ancestor (C_{ancs}) and adjust the papers' scores. *Rate of decay* is defined as

$$\text{RateOfDecay}(C_{\text{ancs}}, C_{\text{desc}}) = \frac{I(C_{\text{ancs}})}{I(C_{\text{desc}})} \tag{4.5}$$

4.3 Assigning context scores to context papers

Next we discuss two different score functions to compute the *context score* of paper p in context c [23]:
Text-based context score function: The text-based context score of a paper in each context is computed using text-based similarity measures based on the Term Frequency-Inverse Document Frequency (TF-IDF) model [9]. In each context c, a paper p's context score is defined as the text-based similarity score between c's representative paper and p. In other

words, papers in c that are highly similar to the representative paper of c receive high context scores.

As described in Sect. 4.1.1, a set of representative papers may be used to locate the p-cluster of c. In that case, the context score of p in c is computed as

$$\text{Score}(p) = \max_{p_r \in R(c)} (\text{sim}(p_r, p)) \tag{4.6}$$

where $R(c)$ is the set of representative papers of c, and $\text{sim}(p_r, p)$ is the text-based similarity between p and p_r, one of the representative papers of c.

Pattern-based context score function: After constructing patterns of c_i as described in Sect. 4.2, the context score is assigned using (1) the confidence that a pattern represents c_i, and (2) the matching strength between p and the pattern. That is

$$\text{Score}(p) = \sum_{pt \in \text{Ptr}(P)} \text{Score}(pt) * M(p, pt) \tag{4.7}$$

where $\text{Ptr}(p)$ is the set of patterns that match to paper p, $\text{Score}(pt)$ is the score of pattern pt, and $M(p, pt)$ is the matching strength of pattern pt in paper p.

$M(p, pt)$ is influenced by the (1) paper paragraph containing the pattern match, and (2) similarity between the pattern (i.e., pt) and the matching phrase in p.

$\text{Score}(pt)$ is computed based on the following middle tuple properties [4,23]:

- Middle tuples consisting of only frequent terms, only the context term, or both frequent and context terms, receive the high, higher, and the highest scores, respectively.
- Context term words with higher *selectivity* receive a higher score. Selectivity describes the word's occurrence frequency among all context terms.
- A pattern's score is inversely proportional to the middle tuple frequency among all the database papers.
- Higher scores are assigned to patterns whose middle tuples are frequent in the context's training papers.

Since contexts are represented hierarchically, a paper p can reside in both context c_i and c_i's descendant contexts. Compared to c_i, c_i's descendant contexts are more specific, and the descendant contexts' paper sets are less diverse. Hence, a high context score for p in c_i's descendant contexts means that p is highly relevant to c_i. Therefore, regardless of context score function choice, a final score computation step takes place as follows. Let p reside in context c_i with score s_i, and descendant contexts $c_k \ldots c_n$ of c_i with scores $s_k, \ldots, s_n$. Then p's score in context c_i is modified as $\max(s_j)$, $j \in \{i, k, \ldots, n\}$.

In the experimental results sections, we use (1) the text-based score function when papers are assigned to contexts using the text-based approach (as described in Sect. 4.1), and (2) the pattern-based score function when the paper-context assignment is done using the pattern-extraction-based approach (as described in Sect. 4.2). In addition to text- and pattern-based score functions presented in this section, we proposed another score function, called citation-based score function, in our previous work [23]. The citation-based score is computed using a variation of the PageRank algorithm [5,6,20]. However, we have decided not to present the results involving the citation-based score function because it is less accurate than the text-based and the pattern-based score functions in a context-based environment [23]. More specifically, citation-based scores give good results in terms of accuracy and score distribution for the upper level (i.e., more general) contexts. However, as we drill down in the context hierarchy, the number of papers and citations within the contexts are reduced. Therefore, papers of these contexts cite or are cited by large numbers of papers outside the contexts. This causes the citation graphs to be sparse within those contexts, which negatively affects the accuracy and score distribution of the citation-based score function. Another possible reason is that citing and cited papers may not be topically related to each other. Therefore, citations in a context may not always indicate that citing/cited papers are important with respect to the context [23].

4.4 Generalizing the approach: removing GO- and PubMed-related specifics

Approaches presented in Sects. 4.1–4.3 use the GO-specific notion of evidence papers in choosing the representative paper of a context and the training data set for patterns. To generalize the overall approach, we present a non-domain-specific method to define: (a) a *representative paper* for the text-based approach (Sect. 4.1.1), and (b) a *training data set* for the pattern-based approach (Sect. 4.2).

Representative paper for text-based approach

Context C's representative paper is identified as follows:

(i) Send C's (and C's descendant contexts') terms and synonyms to text-based search engine(s), and retrieve a set of papers, called S, with high similarity scores. If the number of papers in S is too small, C's parent context terms and synonyms are used.

(ii) For each paper in S, construct a corresponding Term Frequency-Inverse Document Frequency (TF/IDF) [9] vector. When step (i) includes parent contexts, a weight is assigned to reduce the TF/IDF scores in the vector representing each paper from each parent context because a parent context is more general than the context itself. The weight is defined as the *Rate of decay* of the parent context when compared to C (see Eq. (4.5) for the definition of the *Rate of decay*).

(iii) Compute the centroid of all retrieved papers, and use the highest k terms as a (virtual) representative paper

of C. The value of k is defined as the average number of terms in the papers in S.

The length of the context term, which is much shorter than the papers, may negatively affect the text-based similarity. We reduce the effect of short context terms by setting a high similarity score in step (i). While not all papers relevant to the context receive high text-based similarity scores, a high scoring paper (i.e., a paper with a large number of occurrences of the context term words) is considered to be related to the context.

By adjusting the TF/IDF scores of the parent contexts in step (ii), we ensure that information from more general contexts (i.e., the parent contexts) are less important when constructing the (virtual) representative paper of the context in step (iii).

Training papers for pattern-based approach
Frequent papers in S, where S is computed as in step (i) above, are used as context C's training papers. A paper p's frequency, $F(p)$, is defined as

$$F(p) = \frac{\#\text{ contexts containing } p}{\#\text{ contexts}} \tag{4.8}$$

where *contexts* refer to C and its descendant contexts, and each context contains a subset of S's papers returned from step (i) above, $p \in S$. Only papers with high frequency are included in C's training data set.

When C is at the upper level in the context hierarchy, C is a general term and the training data set of C should cover broad areas related to C (i.e., a number of C's descendants). However, If C has no or few descendant contexts (i.e., C is a leaf context or a context at a high level in the context hierarchy), C's term is usually very specific. In this case, all of the papers in S are considered relevant to C and included in the training paper set of C. We use the relative size of C (as defined in Eq. (4.4)) to define the specificity of C, and we apply Eq. (4.8) only when C is not specific. Otherwise, we include all papers.

After locating the training papers of C, the patterns of C are constructed by using C's term and C's training papers (as described in Sect. 4.2).

5 Selecting contexts for keyword search

A context-based search query term is any set of keywords. After mapping a given query to a set of *query contexts*, we perform the search and rank search results within these contexts. Note that users may also manually provide a set of query contexts; however, it may not be easy for the users to select query contexts when the number of available contexts are very large (e.g., approximately 20,000 for GO contexts). To guide users in context selection, next we present two approaches to select query contexts automatically for a given query term. A hierarchical view of the automatically selected query contexts are then shown to the users. After viewing the query contexts, the users can drill down or up in the context hierarchy and manually modify query contexts.

5.1 Text-based similarity between context and search term

Our first automated context selection approach uses a text-based similarity measure. Intuitively, a context whose term or p-cluster is very similar to the search term should be included in the query contexts. Therefore, each context is represented by three components: the context term, its synonyms, and the centroid of the context's p-cluster. In this approach, query contexts refer to contexts whose components are sufficiently similar to the query term. We compute text-based similarity between (1) each context's centroid and a query term, and (2) each context term plus its synonyms and a query term. Then, both scores are combined as a text-based similarity score of context C and query q as follows:

$$\begin{aligned}&\text{Sim}_{\text{Centroid\_ContextTerm}}(C, q)\\ &\quad = w_{\text{centroid}} \cdot \text{Sim}(C_{\text{centroid}}, q)\\ &\qquad + w_{\text{ContextTerm}} \cdot \text{Sim}(C_{\text{ContextTerm}}, q)\end{aligned} \tag{5.1}$$

where w_{centroid} and $w_{\text{ContextTerm}}$ are the centroid and context term weights, respectively, we select $w_{\text{centroid}} > w_{\text{ContextTerm}}$ since the centroid is longer than the context term, which increases the chance of the match to q, $w_{\text{centroid}} + w_{\text{ContextTerm}} = 1$ (see Sect. 9.2 for the evaluation), Sim refers to the cosine similarity, C_{centroid} is the centroid of C's p-cluster, and $C_{\text{ContextTerm}}$ is the context term and its synonyms. Each context term can have multiple synonyms, and high similarity between a synonym and q means that the context is highly related to q. $\text{Sim}(C_{\text{ContextTerm}}, q)$ is computed using both the context term and its synonyms as

$$\text{Sim}(C_{\text{ContextTerm}}, q) = \text{Max}_i(w_i \cdot \text{Sim}(t_i, q)) \tag{5.2}$$

where t_i refers to the context term or each of its synonyms, w_i is the weight of t_i. For GO-specific contexts, GO website defines multiple types of synonyms: "*Exact*", "*Narrow*", "*Broad*", and "*Related*" [2]. For example, the GO term "maltose catabolic process" has five synonyms: four "exact" synonyms include "malt sugar catabolic process", "malt sugar catabolism", "maltose breakdown" and "maltose degradation", and one "narrow" synonym, "maltose hydrolysis". Not all types of synonyms are equally important. Thus, a weight w_i is assigned to each synonym type. For experimental evaluation, the following weights are found to perform the best.

- $w_i = 1$ when t_i is a GO term or *Exact* synonym
- $w_i = 0.9$ when t_i is a *Broad* or *Narrow* synonym

- $w_i = 0.8$ when t_i is a *Related* synonym.

For non-GO contexts, we assign $w_i = 1$.

Finally, given a query term q, those contexts with sufficiently high $\text{Sim}_{\text{Centroid\_ContextTerm}}$ scores (see Eq. (5.1)) (i.e., higher than a threshold t) are selected to be the query contexts of q. In the experiments, $t = 0.05$ is found to be the best value.

5.2 Occurrences in context keywords

Our second automated context selection approach utilizes *context keywords*. After defining *context keywords* for each context, we choose those contexts whose keywords occur frequently in the search term.

The motivation for our approach comes from web computing. In Hyperlink-Induced Topic Search (HITS) [6], a *root set* of documents is obtained after sending a query to a text-based information retrieval system. Although the query words rarely occur near good links (URLs) in many web pages, the centroid of the root set features terms near good links with large weights [3]. Hence, the largest components of the root set centroid vectors are extremely intuitive. Considering papers in each context as a root set of the context, terms with the largest TF-IDF [9] values in the root set centroid vector are then used as the context keywords. Steps to compute the context keywords are: (1) compute context centroid; (2) select top-k words from the centroid to be context keywords; and (3) add every word of the context term (and its synonyms) to the context keywords.

Next, for each query term, each word in the term is stemmed, stopwords are removed, and contexts containing more than k% of the "query words" in "context keywords" are selected. Finally, those contexts with sufficiently high similarity scores (i.e., higher than a threshold t) between context C and query q , namely, $\text{Sim}_{\text{keywords}}(C, q)$, are selected, where

$$\text{Sim}_{\text{keywords}}(C, q) = \frac{n}{N} \tag{5.3}$$

n is the number of "query words" that appear in "context keywords", and N is the number of "query words". For experimental evaluation, threshold $t = 0.3$ is found to be the best value.

6 Search and rank search results within query contexts

As discussed in the introduction, there are two alternatives to perform context-based search after selecting query contexts. The first alternative (*CBS*) performs search within the selected query contexts. In other words, only papers that reside in the query contexts are involved in the search. Then, in each context, search results returned from the keyword-based search are ranked by their *relevancy scores* with respect to the context and the query term. The relevancy score of paper p to query q in context c_i is computed as

$$R(p, q, c_i) = w_{\text{context}} \cdot \text{Context\_Score}(p, c_i) + w_{\text{matching}} \cdot \text{Text\_Matching\_Score}(p, q) \tag{6.1}$$

where context_score(p, c_i) is the context score of p in context c_i (see Sect. 4.3 for the formula), text_matching_score(p, q) computes the similarity (e.g., cosine similarity [9]) between p and q, and w_{context} and w_{matching} are weights of the context score and the text matching score, respectively. $w_{\text{context}} + w_{\text{matching}} = 1$. By default, we define $w_{\text{matching}} > w_{\text{context}}$ (i.e., we used $w_{\text{matching}} = 0.8$ and $w_{\text{context}} = 0.2$ in the experiments). In this definition, the text-matching scores between the query keyword and the papers are considered more important than the context scores of the papers. However, the weights can be adjusted based on users' preference. For example, if the user wants to increase the significance of the contexts, w_{matching} will be reduced while w_{context} will be increased, and search results within the contexts will be ranked with respect to the new weights.

While the CBS approach reduces search input size and returns only papers in the contexts of interest, some important papers may be missing if they are not in the selected query contexts. The second alternative, called *CBS_all*, performs a keyword search across all papers (as opposed to papers within query contexts). Then, search results are grouped and ranked with respect to the query contexts using equation 6.1. Papers that do not reside in any query context are grouped together in a "remainder context". Papers in the remainder context are ranked with respect to their text-matching scores to the query keyword. Compared to the CBS approach, the CBS_all approach includes all papers returned from the keyword-based search; therefore, recall (see section 8.2 for the definition of recall) is preserved. However, the CBS approach searches and returns only papers within selected contexts; thus, search input and output sizes are reduced.

In order to better suit users' needs, users can decide between CBS and CBS_all. However, users are not required to know the details of both processes. Instead, users are asked if they want to (1) perform a search within a set of papers that are related to their queries and selected contexts (i.e., lower recall, faster response time, and smaller output size), or (2) perform a search across all papers (i.e., higher recall, slower response time, and larger output size). If a user chooses option (1), CBS is performed; otherwise, CBS_all is employed.

7 Merging query results from multiple contexts

As stated in the introduction, added value exists in listing separately (1) search results individually for each query

context, and (2) different relevancy scores for the same paper in different contexts. With multiple query results, users benefit from viewing search results in each context separately as well as a follow-up interactive drill down/up in the context hierarchy to revise query contexts for the next search. Having said this, users may want to view a single result set independent of the individual searched contexts. To effectively rank search results for the latter case, scores of a paper residing in multiple contexts need to be merged into a final score, which is discussed next.

When appearing in multiple contexts, paper p's overall relevancy score $R(p, q)$ to the query q is computed using (1) the relevancy score of p to q in each context, and (2) the relevancy of each context containing p to q, as follows:

$$R(p,q) = \frac{\sum_{i=1}^{n_p}(w_{\text{PaperRelevancy}} \cdot R_1(p,q,c_i) + w_{\text{context}} \cdot R_2(c_i,q))}{n_p} \tag{7.1}$$

where $R_1(p, q, c_i)$ is the relevancy score of p to q in the context c_i (Eq. (6.1)), $R_2(c_i, q)$ is the relevancy score of the context c_i to the query q (see below), n_p is the number of contexts that contain p, $w_{\text{PaperRelevancy}}$ and w_{context} are the weights of R_1 and R_2, respectively, $w_{\text{PaperRelevancy}} + w_{\text{context}} = 1$. We define $w_{\text{PaperRelevancy}} > w_{\text{context}}$ (i.e., we used $w_{\text{PaperRelevancy}} = 0.6$ and $w_{\text{context}} = 0.4$ in the experiments). Although we selected fixed weight values in the experiment, we found that changing weight values does not significantly change the accuracy of the search (see Sect. 9.2 for more details).

In Sect. 5, query contexts are selected based on the relevancy between the contexts and the query term. For automatically-selected query contexts, Eqs. (5.1) and (5.3) are used to estimate $R_2(c, q)$. As described in Sect. 5, users may also manually modify selected query contexts. In such cases, sometimes, query-context relevancy scores (from Eqs. (5.1) or (5.3)) of the manually-selected contexts may be low. As an alternative, users are allowed to define the importance of the manually-selected contexts as high (score = 1), medium (score = 0.5), or low (score = 0.1). For the CBS_all approach, a remainder context relevancy score is 0. Therefore, papers in the remainder context are usually ranked lower than papers in the query contexts.

Finally, each returned paper p is assigned a single relevancy score, $R(p, q)$. The paper results are then sorted by these scores and returned to the user.

8 Experimental setup

We downloaded, parsed, and populated our database with information from 72,027 full-text PubMed papers. All selected papers came from the genomics area, which constitutes a "semantically related" subset of PubMed papers related to GO [8].

8.1 Locating P-clusters of contexts

In Sects. 4.1 and 4.2, we presented two approaches to create p-clusters of contexts, namely, text-based and pattern-extraction-based approaches. These two approaches were utilized to construct two p-cluster sets for our experimental evaluation: *text-based p-cluster set*, which was created using the text-based approach (as discussed in Sect. 4.1) and *pattern-based p-cluster set*, which was created using the pattern-extraction-based approach (as described in Sect. 4.2).

The context-paper statistics of p-cluster sets 1 and 2 are presented in Table 1.

Observation: Pattern-based p-cluster set yields the higher number of papers per context than text-based p-cluster set.

Therefore, pattern-based p-cluster set requires higher storage space. Moreover, pattern-based p-cluster creation is more expensive than text-based p-cluster creation. More specifically, the pattern-based approach first creates multiple patterns for each context. Then, occurrences of these patterns in database papers are found. Therefore, the cost of the pattern-based approach is $p*c*n$, where p is the number of patterns in a context, c is the number of contexts, and n is the number of papers in the database.

For the text-based approach, we compute the text-based similarity between the representative papers of each context and the papers in the database. Thus, the cost of text-based p-cluster creation is $r*c*n$, where r is the number of representative papers in the context (note that r is usually a small number, close to 1), c is the number of contexts, and n is the number of papers in the database. As compared with the pattern-based approach, p, the number of patterns, is usually much larger than r (i.e., p is usually larger than 10, on the average).

8.2 Accuracy evaluation

To evaluate the accuracy of the context-based search approach, recall and precision scores of selected queries are used. Given a search term t as a query, its recall and precision

Table 1 Papers per context information

Per context info	p-Cluster set	
	Text	Pattern
Max no. of papers	55218	56750
Mean no. of papers	230.83	796
Median no. of papers	55	182

are defined as

$$\text{Recall}_t = \frac{|S_t \cap R_t|}{|R_t|} \tag{8.1}$$

$$\text{Precision}_t = \frac{|S_t \cap R_t|}{|S_t|} \tag{8.2}$$

where S_t is the search result set for query term t, and R_t is the correct answer set for t. In addition to recall and precision, we used the harmonic mean F1, which combines recall and precision. F1 is defined as

$$\text{F1}_t = \frac{2}{\frac{1}{\text{Recall}_t} + \frac{1}{\text{Precision}_t}} \tag{8.3}$$

8.3 Selecting search terms

We used two search term sets in the experiments: *GO-related* set and *MeSH* set. Both search term sets are elaborated as follows.

Since the subset of PubMed papers that are in our database and used in the experiments were chosen from PubMed journals and are "semantically related" to GO, we first selected search terms to be somewhat related to the GO terms. We call this search term set, *GO-related* set.

The GO site contains manual expert-based mappings of non-GO ("external") concepts from external classification systems, (e.g., SWISS-PROT keywords [30], TIGR roles [31], etc.) to equivalent GO terms [2]. Search terms in the GO-related set are selected from the external concepts as follows. Approximately 400 search terms are selected randomly. To ensure that the search terms used in the experiments are relevant to some papers in our database, all the selected terms are submitted to a text-based information retrieval system, and approximately 120 terms that contain at least one paper with matching scores above a threshold are chosen. Examples of the selected terms are *Formate dehydrogenase, Histone deacetylase* and *Innate immunity*.

Although search terms in the GO-related set are well-suited to our choice of papers and context hierarchy, there is no gold standard to define the correct result set (R_t of Eqs. (8.1) and (8.2)) for each search term in this test set (see Sect. 8.4 for details of GO-related set evaluation). Therefore, we created another test set, called *MeSH* set, whose correct answer set of each search term has been chosen by domain experts.

For the MeSH set, we selected randomly approximately 170 Medical Subject Headings (MeSH) terms [34] to be used as search terms in the experiments. In PubMed services [1], skilled subject analysts have assigned relevant MeSH terms to each PubMed paper. In other words, if a MeSH term is annotated to a paper, that paper is known to be relevant to the term by domain experts. Therefore, one can use the set of annotated papers as the "correct search result set" when the search term is the MeSH term itself.

By using a MeSH term (t) as a search term, we define R_t (or the correct answer set of the search for term t) of Eqs. (8.1) and (8.2) as a set of papers that are annotated with the term t.

Examples of the selected search terms are *Acute-Phase Reaction*, *Bone Morphogenetic Proteins*, *Fibroblast Growth Factors*, *Histone Deacetylases*, and *Phagocytosis*.

8.4 Finding AC-answer set of a query without expert help

To evaluate and compare different approaches for keyword-based querying, clearly, the best approach is to obtain true answer sets of queries manually via domain expert judgments. However, such an approach is not always available and precludes using large numbers of queries to evaluate the overall methodology. Thus, we developed an approach to find the A(rtificially)C(onstructed)-answer set of a query automatically. The AC-answer set is used to evaluate queries with no human judgments of their search results (i.e., the GO-related set, see Sect. 8.3). Through domain expert evaluations of a small number of queries, we refined the AC-answer set creation process, and manually verified its correctness (see Sect. 8.4.2). The AC-answer set is then used extensively in the experiments to evaluate a large number of search query recall and precision scores for the GO-related search term set.

8.4.1 AC-answer set construction

To construct an AC-answer set, we use an approach similar to the pearl-growing search strategy [35,36]. That is, we locate a highly-relevant paper set for a given query and expand it iteratively through a highly compute-intensive expansion process. Given a keyword query q, the database is queried for papers using a text-based similarity measure, and papers with similarity scores above a threshold t are included in the initial answer set S_1. By utilizing a high value of t, we ensure that *papers in* S_1 are highly relevant to the query term. After initial construction, we expand S_1 by using text- and citation-based approaches.

Text-based expansion This approach uses the text-based similarity measure to locate additional papers. Since a paper in S_1 is highly relevant to the keyword query, papers that are highly similar to the paper in S_1 are potentially relevant to the query. Thus, papers with high similarity scores to S_1's centroid or a paper in S_1 are added to the AC-answer set.

Citation-based expansion This approach expands the AC-answer set with citations of a paper in S_1. Since a paper usually cites or is cited by other papers that are relevant to it, citations of a paper in S_1 are potentially relevant to the

query term. There are two approaches involving the citation-based expansion: Radius-k expansion and Citation-similarity-based expansion. Each approach is elaborated as follows.

Radius-k expansion Given a paper p, papers in the *radius-k* of p are defined as papers on a citation path of length at most k starting or ending at p. We first start with $k = 1$, i.e., the initial possible answer set (E) includes papers that cite or are cited by a paper in S_1. The following criteria determine a paper in E's relevancy to the query:

(i) A paper that cites or is cited by a large number of papers in S_1 is relevant to the query
(ii) A paper that cites or is cited by most of the papers in our database is less relevant to the query term.

Based on the above criteria, a *citation-based expanding score*, which is used to filter out irrelevant publications from E, is computed as follows:

$$\text{Expanding\_Score}(p) = w_1 \cdot \text{CBC\_Score}(p) + w_2 \cdot \text{Expected\_Relevancy}(p) \tag{8.4}$$

where p is a paper in E, w_1 and w_2 are weights of CBC_Score (citation-based connectivity score) and Expected_Relevance, respectively, and $w_1 + w_2 = 1$,
CBC_Score(p) is computed as

$$\text{CBC\_Score}(p) = \frac{CC(p, S_1)}{|S_1|} \tag{8.5}$$

where $CC(p, S_1)$ is the number of papers in S_1 that cite or are cited by p.
Expected_Relevancy(p) is computed as

$$\text{Expected\_Relevancy} = \frac{CC(p, S_1)}{CC(p, S_{\text{all}})} \tag{8.6}$$

where $CC(p, S_1)$ is the number of papers in S_1 that cite or are cited by p, and $CC(p, S_{all})$ is the number of papers in the database that cite or are cited by p.

CBC_Score satisfies the first criterion (i) above, which increases the score of a paper that cites or is cited by a large number of papers in S_1. Expected_Relevancy satisfies the second criterion (ii) above, which decreases the score of a paper that cites or is cited by a large number of papers in the database.

Finally, only papers in E with high Expanding_Score values (Eq. (8.4)) are added to the AC-answer set.

To further increase the size of the AC-answer set, we expand S_1 by a path of length 2 (i.e., radius-2 expansion). However, in this case, the number of citations may become large, and many citations may not be relevant to the query term. To eliminate such citations, we introduce a paper set S_2 to filter out irrelevant citations. S_2 is a set of papers returned from existing keyword-based search engines. In the case of PubMed papers, we utilize Entrez Programming Utilities' web service [17] to retrieve S_2 from the given query term. Then, only those papers in a citation path of length 2 and appearing in S_2 are added to the AC-answer set. Although S_2 may contain some papers that are not highly relevant to the query term, and citation paths of length longer than one usually lose context, the appearance of a paper p in $S_2$and a citation path of length 2 significantly increases $p's$ potential to be in the true answer of the search query.

Citation-similarity-based expansion Citation similarity [27] is computed using co-citation [28] and bibliographic coupling [29]. Bibliographic coupling gives a high similarity score to a pair of papers (p1, p2) with large numbers of common citations. Co-citation gives a high similarity score to a pair of papers (p1, p2) if the number of papers that cite both p1 and p2 is large. In this approach, publications in the database with high citation similarity scores to a publication in S_1 are added to the AC-answer set. Citation similarity [27] is computed as follows:

$$\text{Sim}_{\text{Citation}}(\text{p1,p2}) = \text{BibWeight} * \text{Sim}_{\text{bib}}(\text{p1,p2}) + (\text{1- BibWeight}) * \text{Sim}_{\text{coc}}(\text{p1,p2}) \tag{8.7}$$

where p_1 and p_2 are publications, $p_1 \in S_1$, $p2 \notin S_1$, Sim_{bib} is the bibliographic coupling score, Sim_{coc} is the co-citation score, BibWeight is the bibliographic coupling weight, CocWeight = 1-BibWeight is the cocitation weight, and $0 \leq \text{BibWeight} \leq 1$. Sim_{bib} is defined as

$$\text{Sim}_{\text{bib}}(\text{P}_1, \text{P}_2) = \frac{(\#\text{common citations between P}_1 \text{ and P}_2)}{\text{MaxB}} \tag{8.8}$$

where MaxB is the maximum number of common citations between any pair of papers in the database.

Sim_{coc} is defined as:

$$\text{Sim}_{\text{coc}}(\text{P}_1, \text{P}_2) = \frac{(\#\text{papers that co-cite P}_1 \text{ and P}_2)}{\text{MaxC}} \tag{8.9}$$

where MaxC is the maximum number of papers that co-cite any pair of papers in the database.

8.4.2 AC-answer set verification

We manually verified the AC-answer set accuracy in terms of precision (see Sect. 8.2 for the definition of the recall and precision). Recall was not used for evaluation because it is not feasible to scan through all the papers in the database ($\sim$ 72,000 papers) to find all the correct answers of a given query term.

We randomly chose 10 search terms from the GO-related search term set (see Sect. 8.3) as a test set. We constructed the AC-answer set for each search term in the test set. Since the number of papers in the AC-answer set can be large

(>100), we randomly selected 20 papers in each AC-answer set for evaluation, i.e., approximately 200 full-text publications were involved in the manual evaluation.

Using the above criteria to select the test instance, the AC-answer sets are found to be at least 95% accurate. From our manual evaluation, the "noise" of the expansion (i.e., changes in the accuracy due to the expansion) depends on the choice of the initial set (S_1). More specifically, when S_1 includes only papers with high similarity scores to the query (i.e., the threshold for S_1 is high), papers retrieved after the expansion steps are at least 95% accurate. When relaxing the threshold for S_1, accuracy is reduced both in the initial set S_1 and in the paper set from the expansion step. Also, all of the expansion steps from the high-threshold S_1 produce results with higher accuracy than S_1 itself with lower threshold. We also evaluate the effect of the 5% AC-answer set error in Sect. 9.3.

One may reason that, since in our experimental evaluations of the context-based approach, we use the AC-answer set, why not use it directly as the search output? This is neither feasible, nor as informative as the CBS approach: first, the context-based search framework presented in this paper controls paper topic diversity tightly and in an explicit manner, and reduces search output size. Therefore, the overall context-based search approach is much more informative and richer than the AC-answer set. Second, the AC-answer set is computationally very expensive. More specifically, the expansion steps from the initial answer set involve a large number of computations including (1) the paper similarity measures between each paper in the initial set and the papers in the database, and (2) the citation score computation for each citation from the initial set. Thus, the AC-answer set, while very useful for evaluation purposes, is not practical to use as a search technique for literature digital collection search engines.

9 Experimental results

In this section, we compare recall, precision, and/or harmonic mean of recall and precision (F1) of selected search terms from the GO-related and MeSH search term sets (as described in Sect. 8.3) when performing different context-based search approaches.

Given a keyword query q, steps to perform CBS and CBS_all query searches in the experiments are as follows: (1) select contexts automatically from the search term; (2) search within selected contexts (CBS) or search across all papers (CBS_all); (3) rank search results within the selected contexts; and (4) merge search results from different contexts into a single result set. In most of the experiments, we used only merged search results since the accuracy before and after merging results are found not to be statically significantly different. More specifically, from an analysis of variance (ANOVA) of the harmonic mean of recall and precision (F1) before and after merging search results with scores above different cutoff thresholds, the analysis shows $p > 0.05$ for every threshold value.

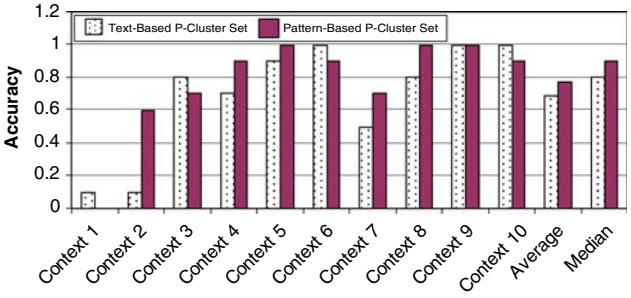

Fig. 6 Accuracy of context-paper classification approaches (direct evaluation)

We first evaluate the accuracy of text- and pattern-matching-based approaches that we used to classify papers to contexts. Then, we test the accuracy difference when changing weights of the formulas proposed in this paper. Next, we measure the effect of 5% incorrect AC-answer set. We then measure the goodness of the query context selection approaches presented in Sect. 5. Finally, we compare our context-based search approach with search results from PubMed and an alternative search-and-distribute to contexts (SDC) approach.

9.1 Comparison between text- and pattern-based p-cluster sets

In Sect. 4, we proposed two approaches to classify papers to their relevant contexts, namely, text-based and pattern-extraction-based approaches. In this section, we compare the accuracy of the text-based p-cluster set to the pattern-based p-cluster set using (1) direct evaluation on a small number of papers and contexts, and (2) precision versus recall curves using search terms in the GO-related set and the MeSH set (see Sect. 8.3).

In the direct evaluation, we randomly selected 10 contexts (GO terms). For each selected context, we randomly selected 10 papers from the text-based p-cluster set and 10 papers from the pattern-based p-cluster set (i.e., the evaluation covers approximately 200 papers). In each context, each paper was evaluated by an expert as being relevant or irrelevant to the context. We compute the accuracy of each context as follows:

$$\text{Accuracy}(c_i) = \frac{(\#\text{ papers being judged as relevant})}{10} \tag{9.1}$$

Figure 6 presents the results of the direct evaluation.

Observations

- Out of 10 contexts, 7 (text-based p-cluster set) and 8 (pattern-based p-cluster set) contexts receive high accuracy (> 0.7).
- On the average, the pattern-based approach yields slightly better accuracy than the text-based approach.

The reason that p-clusters of some contexts are not highly accurate is that we utilized the Porter stemming algorithm [37] to reduce each term in the papers and contexts to its linguistic root form, and some domain-specific terms were incorrectly reduced to more general terms. Therefore, incorrect classifications do occur. For example, the GO term representing context 1 is "Activin Complex", and the Porter stemming algorithm reduces the term "activin", a polypeptide growth factor, to "active". This problem would probably occur in other domains when reduced forms of both common adjectives/nouns (e.g., active) and specific terms (e.g., activin) are the same. However, the negative effect of this problem would be minimal for domains whose terms are highly scientific names, such as chemical metabolite names in ChEBI ontology [65], a hierarchically organized dictionary of molecular entity names. As an example, applying Porter Stemming on "glucose" or "zinc" would not have negative effects in terms of turning a specific term into a general term.

For the second evaluation that involves all the papers in all contexts, Figs. 7 and 8 illustrate average precision versus recall values when performing context-based searches (i.e., CBS and CBS_all, see Sect. 6) using search terms in the GO-related set and the MeSH set, respectively.

Observations

- For the CBS_all approach, text- and pattern-based approaches produce comparable accuracy.
- For the CBS approach, pattern-based p-cluster set yields up to 15% higher precision at moderate recall values.

With respect to the accuracy, pattern-based p-cluster set is better than text-based p-cluster set although the accuracy difference is not highly significant. Thus, experiments in the remaining sections were performed using only the pattern-based p-cluster set. However, as mentioned in Sect. 8.1, the pattern-based p-cluster creation is more expensive than the text-based p-cluster creation. Therefore, the text-based approach can be used as an alternative.

9.2 Effect of varying weights

Weights and formula thresholds presented in this paper are chosen by evaluating multiple CBS query result sets. And, the best weights are used in the experiments. For example, steps to select the weights are as follows:

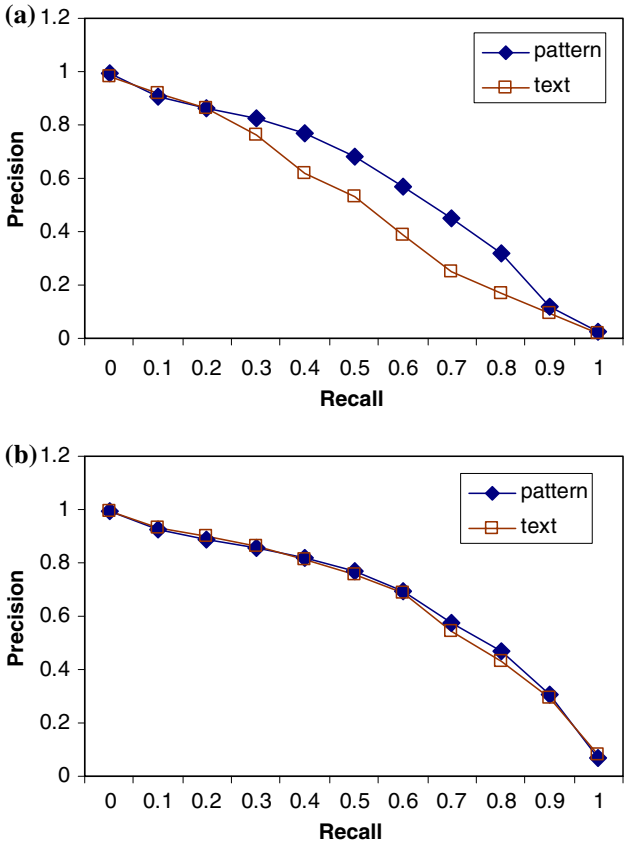

Fig. 7 Average precision versus recall curve for GO-related search term set: **a** CBS and **b** CBS_all

(i) Perform CBS searches using query terms selected in Sect. 8.3. Other weights are fixed while weight w_i varies. For example, we tested $[w_{\text{centroid}}, w_{\text{ContextTerm}}]$ of Eq. (5.1) with the values of $[x, y]$ where $x \geq 0.6$ and $y \leq 0.4$.
(ii) Compute the average recall and precision scores for each weight w_i.
(iii) Select w_i that yields the best results, i.e., both recall and precision scores are high on average for the selected w_i.

Although we selected the best weights for the experiments, we found that the results are not sensitive to the weights for most cases, i.e., the accuracy differences when changing weight values are not statistically significant (i.e., $p > 0.05$). Table 2 shows an analysis of variance (ANOVA) of the harmonic mean (F1) of recall and precision of search results at top 75, 50, and 10% of the CBS approach when using five different values of $[w_{\text{centroid}}, w_{\text{ContextTerm}}]$ of Eq. (5.1) and $[w_{\text{PaperRelevancy}}, w_{\text{Context}}]$ of Eq. (7.1). Note that we show

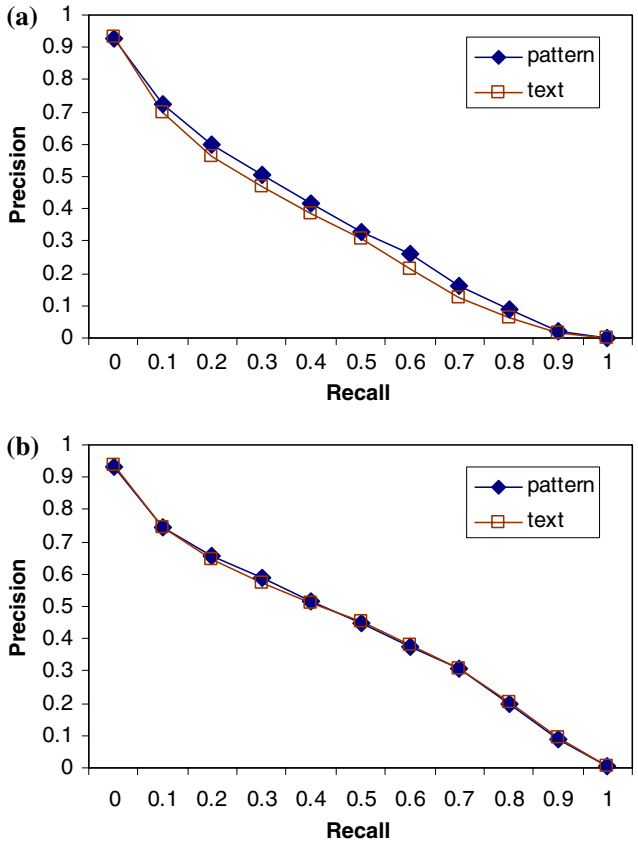

Fig. 8 Average precision versus recall curve for MeSH search term set: **a** CBS and **b** CBS_all

only the results from the MeSH search term set since the results from the GO-related search term set are similar.

9.3 Effect of incorrect AC-answer set

As evaluated in Sect. 8.4.2, the AC-answer set of query term t may contain up to 5% incorrect query results. Therefore, the best case (i.e., the most accurate recall and precision computations) occurs when the AC-answer set of t is 100% correct. For the best case, Eqs. (8.1) and (8.2) are used to measure the accuracy. The worst case recall and precision occur when (1) the AC-answer set of t contains 5% incorrect results, and (2) all the incorrect results are included in the search results of t. In other words, all of the 5% incorrect search results are misjudged as being correct. The worst-case recall and precision

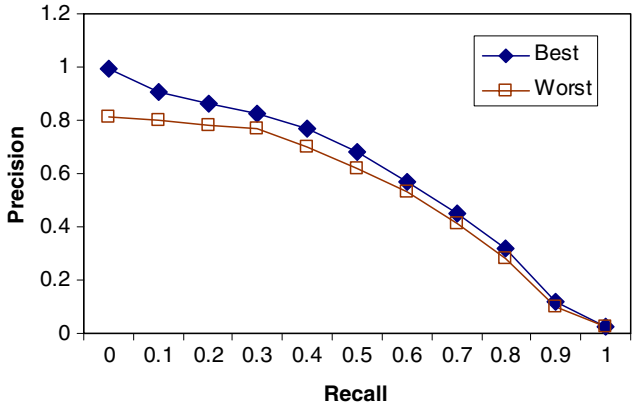

Fig. 9 Average precision versus recall when including 5% incorrect AC-Answer set results

are defined as

$$\text{Recall}_t = \frac{|S_t \cap R_t| - (|R_t| * 0.05)}{|R_t| - (|R_t| * 0.05)} \tag{9.2}$$

$$\text{Precision}_t = \frac{|S_t \cap R_t| - (|R_t| * 0.05)}{|S_t|} \tag{9.3}$$

where S_t is the search result set for term t, and R_t is the AC-answer set of t.

To incorporate the effect of these incorrect search results, we compare the best case with the worst case recall and precision scores. Figure 9 illustrates the results of the CBS approach.

Observation
When search results include 5% incorrect AC-answer set results, the accuracy of search results decreases 5%, on the average, and 18% at the maximum (only at the lowest recall value).

From the experimental results, the worst case and the best case accuracy are not significantly different. Therefore, we use only the best case scores to evaluate the accuracy of search results in the experiments.

9.4 Comparing text-based and context-keyword-based query context selections

In this experiment, we compare two different approaches, namely, *text-based* approach (as described in Sect. 5.1) and *context-keyword-based* approach (as described in Sect. 5.2)

Table 2 ANOVA analysis of varying weights

	Top-75%	Top-50%	Top-10%
$[w_{\text{centroid}}, w_{\text{ContextTerm}}]$	$F(4, 850) = 0.025$, $p = 0.99$	$F = 0.057$, $p = 0.98$	$F = 1.04$, $p = 0.38$
$[w_{\text{PaperRelevancy}}, w_{\text{Context}}]$	$F(4, 850) = 0.001$, $p = 0.99$	$F = 0.004$, $p = 0.99$	$F = 0.004$, $p = 0.99$

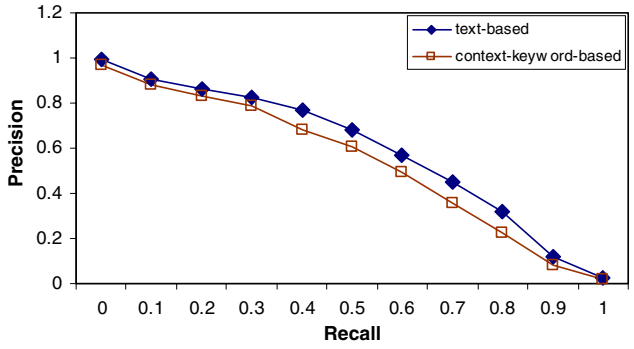

Fig. 10 Average precision versus recall curve for GO-related search term set

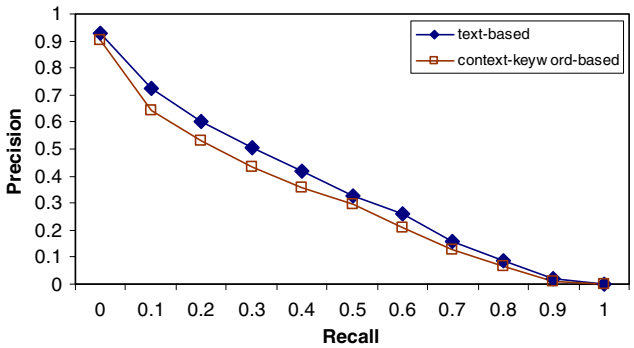

Fig. 11 Average precision versus recall curve for MeSH search term set

for query context selection. The following criteria determine the goodness of query contexts of query term t.

(i) Recall and precision of search results are high.
(ii) P-clusters of the query contexts subsume all correct search results of t. In other words, we are able to find all the correct search results by searching within the query contexts.
(iii) The set of selected query contexts is minimal, i.e., the selected set contains only contexts where the correct paper results of t reside in.
(iv) The number of unique papers in the query contexts' p-clusters is much smaller than the number of all papers in the database, i.e., search input size is reduced.

Figures 10 and 11 illustrate average precision versus recall curves of the CBS search results.

Observations

Text-based context selection approach possesses approximately 5% higher accuracy than context-keyword-based context selection approach.

The following steps are used to evaluate criteria (ii), (iii), and (iv).

For each search term t,

(a) Apply each query context selection approach and retrieve a set of query contexts S_{Qt}
(b) Retrieve all the unique papers of all the contexts in S_{Qt}, called P_{Qt}
(c) Compute the *Subsumption Score* of P_{Qt}, which is defined as:

$$\text{Subsumption\_Score}(P_{Qt}) = \frac{|P_{Qt} \cap R_t|}{|R_t|} \tag{9.4}$$

where R_t is the AC-answer set (GO-related search term set) or the correct answer set (MeSH search term set) of t. Criterion (ii) is satisfied if the subsumption score approaches 1, i.e., P_{Qt} includes most of the papers in the correct answer set of t.

(d) Compute the *Minimality Score* of S_{Qt}, which is defined as

$$\text{Minimality\_Score}(S_{Qt}) = \frac{|S_{Qt} \cap S_{Ct}|}{|S_{Qt}|} \tag{9.5}$$

where S_{Ct} is a set of all contexts that contain a paper in the correct answer set of t. Criterion (iii) is satisfied if the miminality score gets closer to 1.

(e) Compute the *Ratio Score* of P_{Qt}, which is defined as

$$\text{Ratio\_Score}(P_{Qt}) = \frac{|P_{Qt}|}{|P_{All}|} \tag{9.6}$$

where $|P_{All}|$ is the number of papers in the database (i.e., 72,027). Criterion (iv) is satisfied when the ratio is close to 0. As explained in criterion (iv), we expect the size of P_{Qt} to be small compared to all the papers in the database.

Tables 3 and 4 show the results of criteria (ii)–(iv) evaluation.

Observation

(1) In terms of correct result subsumption, the text-based approach is up to 8% better than the context-keyword-based approach.
(2) The minimality score of the text-based approach is up to 10% higher than the context-keyword-based approach.
(3) The search input size of the text-based approach is up to 5% larger than the context-keyword-based approach. However, both approaches reduce search input size by at least 50%.

From experimental results, the text-based approach satisfies criteria (i)–(iii) slightly better (<10%) than the context-keyword-based approach, but the context-keyword-based approach possesses slightly better (<5%) criterion (iv). Therefore, we conclude that the text-based approach is overall slightly better than the context-keyword-based approach.

Table 3 Statistics of query context selection approaches (GO-related search term set)

	Query context selection approach	
	Text-based	Context-keyword-based
Mean subsumption score	0.85	0.82
Median subsumption score	0.91	0.88
Mean minimality score	0.53	0.52
Median minimality score	0.51	0.47
Mean ratio score	0.46	0.44
Median ratio score	0.51	0.46

Table 4 Statistics of query context selection approaches (MeSH search term set)

	Query context selection approach	
	Text-based	Context-keyword-based
Mean subsumption score	0.72	0.66
Median subsumption score	0.81	0.73
Mean minimality score	0.71	0.66
Median minimality score	0.8	0.70
Mean ratio score	0.41	0.39
Median ratio score	0.4	0.39

Therefore, all of the experiments were performed using only the text-based approach.

9.5 Comparing context-based results against PubMed results

Using queries from the GO-related search term set, we compare recall and precision scores from the CBS approach to PubMed's general keyword-based search. Papers that are in PubMed search results but not in our database are filtered out before evaluations. Note that, queries (MeSH terms) from MeSH set are not used because we collected the correct answers of this set from PubMed (as discussed in Sect. 8.3); thus, PubMed will return all papers that are marked up with the MeSH term as search results when we use that MeSH term as the search term.

While this experiment's CBS results include only papers with scores above a certain threshold t, PubMed search results include all papers since PubMed lacks a scoring function. We selected three sets of cutoff thresholds. The first set ($t \leq 0.1$, or low threshold values) contains a large number of results (i.e., at least 30% of all results). With the first set, we expect high recall. The second set ($0.15 \leq t \leq 0.3$, or moderate threshold values) contains a moderate number of search results. We expect high F1 for the second set. The last set ($t \geq 0.35$) contains high ranking results, and we expect high precision for this set. Table 5 shows the average number of papers from PubMed, CBS, and CBS_all approaches. Figure 12 compares the average recall and precision scores of PubMed, CBS, and CBS_all approaches. Figure 13 illustrates the average F1 scores of the three approaches.

Table 5 The average number of papers returned from PubMed, CBS, and CBS_all

Threshold	CBS	CBS_all	PubMed
0.05	2689	2892	1466
0.1	796	893	
0.15	230	288	
0.2	117	160	
0.25	66	100	
0.3	44	71	
0.35	31	53.86	
0.4	23	41	
0.45	18	32	
0.5	14	30	

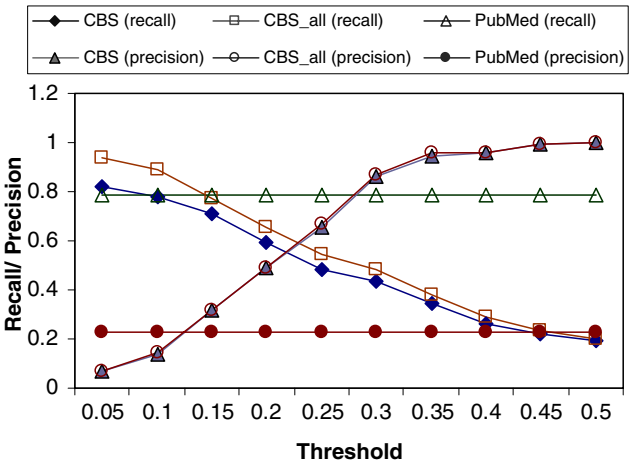

Fig. 12 Average recall and precision of PubMed search, CBS, and CBS_all

Observations

(1) At $t > 0.15$, PubMed recall is higher than the context-based (CBS and CBS_all) recall. This is due to PubMed searching and returning more papers on average than the context-based search approaches.
(2) The context-based approaches produce approximately 50% higher precision at high thresholds and approximately 20% higher precision at moderate thresholds.
(3) At moderate thresholds, the context-based approaches yield approximately 25% higher F1 scores than PubMed. Moreover, from Table 5, the number of context-based search results at moderate thresholds are much smaller (approximately 10 times) than PubMed search results.

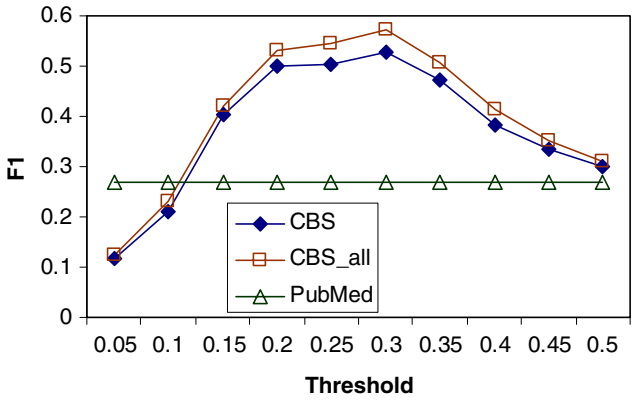

Fig. 13 Average F1 scores of PubMed search, CBS, and CBS_all

Since most web search engine users stop looking at search results after the second page [3], high recall for a large number of search results is less significant than high precision for high-ranking results and high F1 scores for a reasonable number of search results. Similarly, for a large literature digital collection (e.g., PubMed), achieving high precision and high F1 scores is more important than achieving high recall. Experimental results in this section show that the context-based search approach effectively ranks search results while decreasing the number of papers returned when using cutoff thresholds.

9.6 Comparing CBS results against alternative context-based results

This section compares search results from our context-based approach to an alternative approach, called search-and-distribute to contexts (SDC) approach. Steps to perform the SDC approach are as follows:

1. Perform pre-processing steps to classify papers with scores to contexts (as discussed in Sect. 4)
2. Search across all papers to retrieve search results
3. Retrieve involved contexts, which are selected as all contexts that the paper results reside in
4. Rank paper results within the contexts using the formula presented in Sect. 6 and return

The difference between the CBS and the SDC approach is that the SDC approach organizes search results into contexts based on whether or not the results reside in the contexts, while the CBS approach selects relevant contexts based on the relevancy to the query. Moreover, the CBS approach allows users to modify query contexts before viewing search results. As mentioned in Sect. 3, several existing context-based information retrieval systems (e.g., [15,26,48–50]) categorize search results into all possible contexts, and, these systems are comparable to the SDC approach. However, these systems either are implemented on different data sets (i.e., web documents) [26,48–50] or do not provide scores to rank search results [15]. Therefore, we used the SDC approach to compare with our context-based approach in this section. Since the merging algorithm (as described in Sect. 7) cannot be applied directly to the SDC approach, we use non-merged search results for the experiment in this section for both CBS and SDC approaches. Tables 6 and 7 summarize the statistics on the average number of involved contexts and the average number of search results, respectively. Figures 14 and 15 illustrate the average recall and precision scores of both approaches. Note that the recall and precision charts of both GO-related and MeSH search term sets are similar; therefore, we show in Figs. 14 and 15 only the results from MeSH search term set.

Observation

(1) From Table 6, the number of contexts returned from the SDC approach (i.e., contexts that all search results reside) is very large. Therefore, it is not practical for the user to navigate through search results using this large set of contexts. On the other hand, our automatic context selection technique ranks contexts based on the relevancy to the query (as discussed in Sect. 5), and thus significantly reduces the number of involved contexts.

Table 6 The average number of involved contexts for the CBS and SCD approaches

Threshold	GO-related set		Mesh set	
	CBS	SDC	CBS	SDC
Avg. no. contexts	214	2225	162	3979

The number of query contexts of the CBS and CBS_all are the same

Table 7 The average number of search results for the CBS and SDC approaches with different threshold (t)

Threshold	GO-related set			Mesh set		
	CBS	CBS_all	SDC	CBS	CBS_all	SDC
0.05	1517	1726	2505	939	998	1568
0.1	394	499	633	190	209	283
0.15	168	243	265	76	86	93
0.2	90	140	150	48	54	56
0.25	59	98	102	37	41	41
0.3	42	72	74	29	32	32
0.35	31	55	57	24	26	27
0.4	24	44	47	20	21	22
0.45	18	36	38	17	18	18
0.5	15	30	32	14	15	15

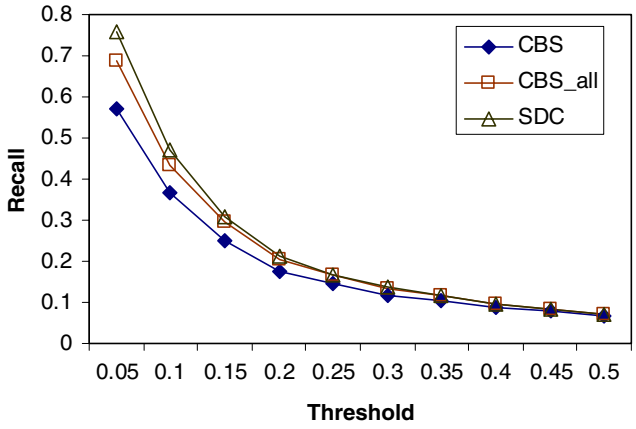

Fig. 14 Average recall of CBS, CBS_all, and SDC

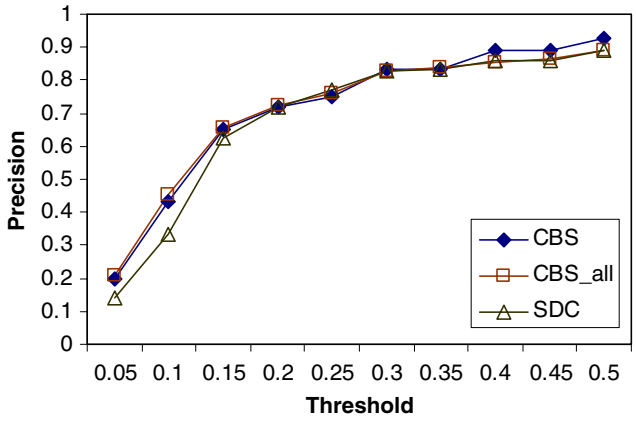

Fig. 15 Average precision of CBS, CBS_all, and SDC

(2) On the average, the number of papers returned from the SDC approach is 30% higher than the CBS approach and 10% higher than the CBS_all approach.
(3) At low thresholds, the SDC approach produces approximately 10% higher recall than the CBS approach. At moderate and high thresholds, all approaches provide comparable recall.
(4) At low threshold, the CBS and CBS_all approaches produce up to 10% higher precision than the SDC approach. At moderate threshold, all approaches provide comparable precision. At high threshold the CBS approach yields approximately 5% higher precision than the SDC approach.

Although recall and precision of the CBS (and CBS_all) and the SDC approaches are not significantly different, the CBS approach is superior to the SDC approach in the sense that it significantly reduces the number of papers returned as well as the number of involved contexts to include only contexts that are relevant to the query and are of interest to the users.

10 Conclusions and future work

At the present time, a major problem in searching literature digital collections within digital libraries is the lack of effective paper scoring and ranking systems. For a keyword-based search, a number of returned papers can be very large. Search results may also contain various topics, not all of which are of interest to users. In order to solve the above-summarized problems, we proposed a context-based searching paradigm for literature digital collections. In our approach, a context defines one or more paper topics, and contexts are related to each other through a context hierarchy.

In our approach, we use well-defined ontology terms as contexts, and papers are classified into contexts through a pre-processing step using text- or pattern-extraction-based techniques. Context scores are also assigned to papers within each context, where high context scores mean papers are highly relevant to a given context. After a user specifies a query term, we present to the user a set of contexts that are relevant to the query. Then, the user can manually modify the set of selected contexts. Search is performed within the selected contexts, and search results are ranked and returned to the user based on the matching strength to the query and their context scores.

Our context-based approach improves various shortcomings of the present search methods as follows:

- Since papers are classified to relevant contexts through a pre-processing step, the complete paper-context information is available before performing a search. After users define search queries, the queries are automatically matched against context information, and only relevant contexts are presented to the users. With this information, the users can define a scope on contexts of interest before viewing search results. Thus, the selected contexts are highly meaningful since they are (1) relevant to the queries and (2) interesting to the users. In contrast to many existing categorization techniques, context-based search results are grouped within only interesting contexts as opposed to a large number of all possible various topics (contexts). This solves the topic diffusion problem across search results.
- Using recall and precision analysis, we evaluated our approach and compared it to other approaches. Experimental results demonstrate that our approach produces comparable recall for a large set of search results and higher precision for high ranking papers. Moreover, the number of search results and contexts returned are much smaller in our approach. For any keyword-query executed on a search engine, most users view only top results. Therefore, high precision for high ranking results is crucial.

- Our context-based approach is general and can be applied to other domains.
  - Context-based search engine allows any set of keywords as opposed to some systems that allow only specific types of keywords (see Sect. 3 for details).
  - Although we initially group search results within the contexts that they belong to, if the user likes the traditional approach and wants to view only single ranked list of search results, we provide an approach to merge relevancy scores from different contexts into one final score, and use the new scores to rank search results.
  - We present ways to generalize paper classification techniques to non-domain-specific methods that do not utilize any specific properties of the associated ontology. Thus, our approach can be applied to any sets of papers and contexts.

All in all, the main assets of the context-based approach when compared to existing approaches are the accuracy improvement, the topic diffusion and output size reduction, and the generality of the approach.

Although we present a complete context-based search framework, there is still room for further improvement. First, as discussed in Sect. 9.1, p-clusters of some contexts are not highly accurate because of incorrect term stemming for domain-specific terms. The simple stemming algorithm that we used needs to be replaced by a more complex and specific algorithm that differentiates domain-specific terms from general terms.

Second, in this paper, we have evaluated the context-based search approach on genomics-based PubMed publications and Gene Ontology hierarchy. The context-based search approach can also be applied to other publication domains using other domain-specific ontologies as contexts. For example, computer science papers can be used as a testbed, and a possible computer science-related context hierarchy is the ACM Computing Classification System [58]. By changing publication and context domains, there are two ways to assign papers to contexts: (1) using semantic properties of the ontology, or (2) using our generalized approach (as discussed in Sect. 4.4). Other steps of our approach, i.e., selecting query contexts, ranking search results, and merging search results, can be applied directly to other domains.

Another future research direction is to investigate the effectiveness of other variations on context score computations (see Sect. 4.3 and [23]). For example, the text-based context score of a paper in a context may be modified as the similarity between the paper and the centroid of all of the documents in the context. For the citation-based context score function, instead of omitting citation relationships from different contexts during context score computations, we can assign weights to these relationships. In other words, citation relationships from other contexts can boost context scores of the papers in a given context [23].

Yet another direction is to improve the automatic context selection techniques (as described in section 5) by using a learning mechanism. When a user performs a context-based search, he/she first enters a search term. Then our system provides a list of potentially relevant contexts, which the user further modifies. If we cache the user-selected contexts of a given search query, those contexts can be potentially highly relevant to the query. Since the number of contexts of a given query can be quite large, which contexts to keep also needs to be investigated.

Glossary

AC-answer set	Artificially constructed answer set of a query. The AC-answer set is used as a correct answer set of a query when the actual correct answer set is not available
CBS	Context-based search approach which involves the following steps: (1) Classify papers to contexts and assign context scores to papers (2) Select query contexts to search for (3) Search within the selected contexts (4) Rank search results within the contexts and/or merge search results into one list
CBS_all	Context-based search approach that involves all papers, as opposed to papers in selected contexts. Steps to perform the CBS_all are as follows: (1) Classify papers to contexts and assign context scores to papers (2) Select interesting contexts (3) Search across all papers to retrieve search results (4) Rank search results within the contexts and/or merge search results into one list. For search results that are in the selected contexts, rank them with respect to the contexts. Search results that are not in the selected contexts are assigned to a remainder context
Context score of a paper	A context score of a paper within a context indicates the level of relevancy (importance) of the paper

	with respect to the context. In each context, papers with high context scores are highly relevant to the context
GO	Gene ontology [2]. GO provides controlled vocabularies that describe gene and gene products in terms of their biological processes, cellular components, and molecular functions
MeSH	Medical subject headings [34], the national library of Medicine's controlled vocabulary thesaurus
Paper set of a context	See p-cluster set of a context
p-Cluster set of a context	A p-cluster set of a context is a set of papers that are classified to the context because they are relevant to the context
PubMed	PubMed is a literature digital library containing more than 14 million biomedical publications
Query contexts	Query contexts are contexts that are selected automatically or manually by users as being relevant to the query. Search results are grouped within the query contexts to reduce the topic diffusion problem across search results
SDC	An alternative context-based search approach, which involves the following steps: (1) Classify papers to contexts and assign context scores to papers (2) Search across all papers to retrieve search results (3) Select all contexts that contain the query paper results (4) Rank search results within the contexts
Topic diffusion	Topic diffusion across search results means that publications returned by a keyword-based search query often fall into multiple topic areas, not all of which are of interest to users

References

1. PubMed, http://www.ncbi.nlm.nih.gov/entrez/query.fcgi
2. Gene Ontology, http://www.geneontology.org
3. Chakrabarti, S.: Mining the Web, Discovering Knowledge from Hypertext Data. Morgan-Kaufmann, Los Altos, CA (2003)
4. Cakmak, A., Ozsoyoglu, G.: Annotating genes using textual patterns. PSB (2007)
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems (1998)
6. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: ACM-SIAM Symp. on Discr Alg. (1998)
7. Ontology Lookup Service, http://www.ebi.ac.uk/ontology-lookup
8. Po, J.: Context-based search in literature digital libraries. MS Thesis, CWRU (2006)
9. Salton, G.: Automatic Text Processing. Addison-Wesley, Reading, MA (1989)
10. CiteSeer literature search system, http://citeseer.ist.psu.edu/cs
11. Google Scholar, http://scholar.google.com/scholar/about.html
12. IEEE Xplore, http://www.ieee.org/ieeexplore
13. CaseExplorer, http://nashua.case.edu/anthexpl
14. Chmura, J., Ratprasartporn, N., Ozsoyoglu, G.: Scalability of databases for digital libraries. ICADL pp. 435–445 (2005)
15. Delfs, R., Doms, A., Kozlenkov, A., Schroeder, M.: GoPubMed: ontology-based literature search applied to Gene Ontology and PubMed. In: German Conference on Bioinformatics (2004)
16. Agrawal, R., Ramakrishnan S.: Fast algorithms for mining association rules. VLDB (1994)
17. ESearch Entrez Utility, http://eutils.ncbi.nlm.nih.gov/entrez/query/static/esearch_help.html
18. GO Evidence Code Hierarchy, http://www.geneontology.org/GO.evidence.shtml#hier
19. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. IJCAI (1995)
20. Cakmak, A.: HITS- and PageRank-based importance score computations for ACM anthology papers. Technical Report, CWRU (2003)
21. Haveliwala, T.: Topic-sensitive PageRank. WWW (2002)
22. Aussenac-Gilles, N., Mothe, J.: Ontologies as background knowledge to explore document collections. RIAO (2004)
23. Ratprasartporn, N., Bani-Ahmad, S., Cakmak, A., Po, J., Ozsoyoglu, G.: Evaluating utility of different score functions in a context-based environment. In: DBRank Workshop – in Conjunction with ICDE 2007
24. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: the concept revisited. WWW (2001)
25. Kraft, R., Chang, C.C., Maghoul, F., Kumar, R.: Searching with context. WWW (2006)
26. Ferragina, P., Gulli, A.: A personalized search engine based on web-snippet hierarchical clustering. WWW (2005)
27. Al-Hamdani, A.: Querying web resources with metadata in a database. PHD Dissertation, CWRU (2004)
28. Small, H.: Co-citation in the scientific literature: a new measure of the relationship between two documents. J. Am. Soc. Informat. Sci. **24**(4), 28–31 (1973)
29. Kessler, M.M.: Bibliographic coupling between scientific papers. Am. Documentat. **14**, 10–25 (1963)
30. SWISS-Prot Keywords, http://www.expasy.org/cgi-bin/keywlist.pl
31. The Institute of Genomic Research (TIGR), http://www.tigr.org/
32. ACM Digital Library, http://www.acm.org/dl
33. Open Directory Project, www.dmoz.org
34. Medical Subject Heading (MeSH), http://www.nlm.nih.gov/mesh/
35. Hawkins, D.T., Wagers, R.: Online bibliographic search strategy development. Online, May 1982
36. Schlosser, R.W., Wendt, O., Bhavnani, S., Nail-Chiwetalu, B.: Use of information-seeking strategies for developing systematic reviews and engaging in evidence-based practice: the application of traditional and comprehensive pearl growing. A Review. Int. J. Language Commun. Disorders **41**(5), 567–582 (2006)
37. Porter, M.F.: An algorithm for suffix stripping. Program **12**(3), 130–137 (1980)

38. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley, Reading, MA (1999)
39. Hearst, M.A.: TileBars: visualization of term distribution information in full text information access. In: Proc. of the ACM SIGCHI conference on human factor in computing systems, pp. 59–66 (1995)
40. Kaki, M.: Findex: search results categories help users when document ranking fails. In: Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems (2005)
41. Hearst, M.A., Pedersen, J.O.: Reexamining the cluster hypothesis: scatter/gather on retrieval results. SIGIR (1996)
42. Zamir, O., Etzioni, O.: Grouper: a dynamic clustering interface to web search results. WWW (1999)
43. Osinski, S., Weiss, D.: Conceptual clustering using lingo algorithm: evaluation on open directory project data. In: Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'04 Conference, Zakopane, Poland, pp. 359–368, (2004)
44. Zeng, H., He, Q., Chen, Z., Ma, W.: learning to cluster web search results. SIGIR (2004)
45. Zhang, D., Yong, Y.: Semantic, hierarchical, online clustering of web search results. In: Proceedings of the 6th Asia Pacific Web Conference (APWEB), Hangzhou, China, April 2004
46. Kummamuru, K., Lotlikar, R., Roy, S., Singal, K., Krishnapuram, R.: A Hierarchical Monothetic Document Clustering Algorithm for Summarization and Browsing Search Results. WWW (2004)
47. Lawrie, D.J., Croft, W.B.: Generating hierarchical summaries for web searches. SIGIR (2003)
48. Vivisimo, http://vivisimo.com/
49. Clusty, http://clusty.com/
50. Mooter, http://www.mooter.com/
51. Chen, M., Hearst, M.A.: Presenting web site search results in contexts: a demonstration. SIGIR (1998)
52. Wittenburg, K., Sigman, E.: Integration of browsing, searching, and filtering in an applet for web information access. In: Proceedings of the ACM Conference on Human Factors in Computing systems, Late Breaking Track (1997)
53. Pratt, W., Hearst, M.A., Fagan, L.M.: A knowledge-based approach to organizing retrieved documents. AAAI (1999)
54. Muller, H.M., Kenny, E.E., Sternberg, P.W.: Textpresso: an ontology-based information retrieval and extraction system for biological literature. PLoS Biol. **2** (2003)
55. Castells, P., Fernandez, M., Vallet, D.: An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. IEEE Trans. Knowl. Data Eng. **19**(2) (2007)
56. RDQL – A Query Language for RDF, http://www.w3.org/Submission/RDQL/
57. Yahoo! Directory, http://dir.yahoo.com/
58. ACM Computing Classification Systems, http://acm.org/class
59. LINGO 3G, http://company.carrot-search.com/lingo-applications.html
60. iBoogie, http://www.iboogie.com/Text/about.asp
61. Pedersen, T., Pakhomov, S., Patwardhan, S., Chute, C.: Measures of semantic similarity and relatedness in the biomedical domain. J. Biomed. Informat. (2006)
62. Lord, P.W., Stevens, R.D., Brass, A., Goble, C.A.: Investigating semantic similarity measures across the gene ontology: the relationship between sequence and annotation. Bioinformatics **19**(10) (2003)
63. Maguitman, A.G., Menczer, F., Roinestad, H., Vespignani, A.: Algorithmic detection of semantic similarity. WWW (2005)
64. Ratprasartporn, N., Ozsoyoglu, G.: Finding related papers in literature digital libraries. In: 11th European Conference on Research and Advanced Technology for Digital Libraries (ECDL) (2007)
65. ChEBI, http://www.ebi.ac.uk/chebi/
66. Chen, Y.-L., Wei, J.-J., Wu, S.-Y., Hu, Y.-H.: A similarity-based method for retrieving documents from the SCI/SSCI database. J. Informat. Sci. **32**(5), 449–464 (2006)
67. Desai, M., Spink, A.: An algorithm to cluster documents based on relevance. Int. J. Informat. Process. Manage. **41**(September), 1035–1049 (2005)