Bulletin of the Technical Committee on

Data Engineering

December 2005 Vol. 28 No. 4 **IEEE Computer Society**

Letters

Special Issue on Searching and Mining Literature Digital Libraries

Conference and Journal Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

There are two Data Engineering Bulletin web sites: http://www.research.microsoft.com/research/db/debull and http://sites.computer.org/debull/.
The TC on Data Engineering web page is http://www.ipsi.fraunhofer.de/tcde/.

Letter from the Editor-in-Chief

The Data Engineering Conference ICDE'06

The next International Conference on Data Engineering (ICDE'06) will be held in Atlanta in April, 2006. This conference is the flagship conference of the IEEE Technical Committee on Data Engineering. Atlanta is a great venue, and April is a wonderful time to visit the city, with balmy weather and with magnolias and peach trees in bloom. This year's conference is a very selective conference with high quality papers. Additional information about the conference is available on the conference web site, `http://icde06.cc.gatech.edu/`, including the technical program.

About the Bulletin

I have made a minor change in how the Bulletin is available. Starting with the current issue, the individual papers accessed via the Bulletin web sites will be in PDF, not postscript. My primary reason for doing this is because PDF files are smaller than PS files, and hence download more quickly. I invite your comments on this change. My intent is to convert, over time, the individual papers of past issues into PDF as well. So if you do not like this turn of events, please send me email at `lomet@microsoft.com` telling me why this is a bad idea.

The Current Issue

In the database world, we extract information via very precise query languages. Moving to the world of documents has required our community to master and hopefully enhance in our own way, the technology of the information retrieval community. The world is clearly moving to putting everything online in the hopes that we will learn how to exploit this as an invaluable resource for much of what we do, surely professionally, and perhaps personally as well. We, the database community, should be able to help.

One interesting manifestation of this move toward putting things online is the rapid growth of literature digital libraries, both in professional domains and more generally. This is happening now, as anyone who has consulted DBLP will be aware. But this area also has many challenges. Perhaps we would like to know which papers have been published, to choose an area “at random”, in “application recovery”. This is a complicated query, much more like an IR query than a SQL query, but it requires more than simply key word search, even when augmented with web link analysis.

It is the desire to extract and exploit information such as the above that makes the current Bulletin issue so important, interesting, and timely. Gultekin Ozsoyoglu has worked in this area himself. So he brings to his editorial duties as a special issue editor, knowledge both of the field and of its research participants. The names of the authors may be less familiar to you than is normally the case with Bulletin authors. But this is an opportunity for readers to very rapidly get the feel for the exciting things that are happening with literature digital libraries and how they might be exploited. I want to thank Tekin for his fine job with the current issue. He has assembled an excellent overview of the current state-of-the-art in this increasingly important area.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

Literature digital libraries, now an indispensable part of research and education worldwide, are increasing in size at very high numbers. As an example, PubMed, a literature digital library for biomedical sciences, currently contains 15 million papers, and is increasing at a rate of 400,000 papers every year. This issue of Data Engineering Bulletin is on the critical areas of searching, mining, querying, and information extraction from literature digital libraries.

In ”Scaling Information Extraction to Large Document Collections”, Eugene Agichtein classifies and reviews four approaches for scalable information extraction from large document collections, namely, scanning large document collections, exploiting general-purpose search engines, employing specialized indexes and search engines, and using parallelization and distributed processing. Algorithmic approaches trade off information extraction accuracy and completeness for speed. A promising approach is to store semantically annotated documents in semi-structured form.

Text analysis engines are different than search engines in that they allow for queries with words and entities such as punctuation, tags, etc. as well as returning results of different types, e.g., sections and phrases of documents. In ”Fast and Furious Text Mining”, Joel D. Martin describes and briefly evaluates the performance of a text analysis engine called TLM (”Text and Language Mining”) with a highly expressive query language. TLM is part of an integrated suite of tools called LitMiner.

Example-based publication searching is becoming common place in digital libraries, which essentially requires the evaluation of a publication similarity measure. In ”Evaluating Publication Similarity Measures”, Sulieman Bani-Ahmad, Ali Cakmak, Gultekin Ozsoyoglu and Abdullah Al-Hamdani classify the existing publication similarity measures as text-based (from Information Retrieval) and citation-based employing bibliographic coupling and/or co-citation, and extend and evaluate a number of publication similarity measures in terms of accuracy, separability, and independence.

Current search engines are known to perform poorly for a number of ”hard” queries. In ”Hard Queries can be Addressed with Query Splitting Plus Stepping Stones and Pathways”, Xiaoyan Yu, Fernando Das-Neves, and Edward A. Fox propose an approach based on ”Stepping Stones and Pathways” and query splitting, and find the approach feasible and promising.

In ”Ten-Year Cross-Disciplinary Comparison of the Growth of Open Access and How it Increases Research Citation Impact”, Chawki Hajjem, Stevan Harnad, Yves Gingras report that openly accessible (OA) articles from ten disciplines are cited more than those that are not. Their results indicate that the overall percentage of OA articles varies from 5% to 16%, and OA articles have from 25% to 250% more citations as compared to non-OA articles.

Finally, in ”A System of User-Guided Biological Literature Search Engine”, Meng Hu and Jiong Yang propose and briefly evaluate a new digital library search paradigm based on iterative clustering and user feedback.

I hope that you will find this issue useful and informative. My special thanks to all the authors for their contributions to this special issue of the Bulletin.

Gultekin Ozsoyoglu
Case Western Reserve University
Cleveland, Ohio, USA

Scaling Information Extraction to Large Document Collections

Eugene Agichtein
Microsoft Research
eugeneag@microsoft.com

Abstract

Information extraction and text mining applications are just beginning to tap the immense amounts of valuable textual information available online. In order to extract information from millions, and in some cases, billions of documents, different solutions to scalability emerged. We review key approaches for scaling up information extraction, including using general-purpose search engines as well as indexing techniques specialized for information extraction applications. Scalable information extraction is an active area of research, and we highlight some of the opportunities and challenges in this area that are relevant to the database community.

1 Overview

Text documents convey valuable *structured* information. For example, medical literature contains information about new treatments for diseases. Similarly, news archives contain information useful to analysts tracking financial transactions, or to government agencies that monitor infectious disease outbreaks. All this information could be managed and queried more easily if represented in a structured form. This task is typically called *information extraction*. More specifically, information extraction systems can identify particular types of entities (e.g., person names, locations, organizations, or even drug and disease names) and relationships between entities (e.g., employees of organizations or adverse interactions between medical drugs) in natural language text. In this paper we focus on *entity extraction* (NER) and *event or relation extraction* (RE). Once created, the structured representation of entities or relations can be used to answer specific questions quickly and precisely by retrieving answers instead of complete documents, for sophisticated query processing, data integration, and data mining. Managing text is an increasingly important use of relational database management systems [9], and information extraction can be a key technology for this effort.

We focus on extracting information from large document collections (e.g., newspaper archives, web snapshots, biomedical literature archives). This setting is particularly important as information extraction is most useful when the collections are too large to process manually. Additionally, as we will describe, some extraction systems perform best precisely when the collection sizes are large (e.g., [1, 25]). Hence, for usefulness and even accuracy, scaling information extraction to large document collections is crucial. The document collection sizes we consider range from a few hundred thousand documents (e.g., Newspaper archives) to millions of documents (e.g., PubMed and other "hidden web" databases) to tens or hundreds of millions of documents (e.g., Web snapshots, focused web crawls). We provide a brief overview of information extraction process in Section 2.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Unfortunately, extracting the entities and relationships from a document is computationally expensive. Even simple information extraction tasks can require days or weeks of running time to process a large collection. For example, Ravichandran et al. [27] estimate that to just perform part-of-speech tagging (a common pre-processing step for information extraction) over a terabyte of text (between 50 and 100 million documents) required 125 days on a 2.5GHz PC, and a shallow syntactic parse required 10 machine-years. Clearly, this is not feasible for large document collections[1]. To scale up information extraction to large collections, four main approaches have been used:

- Scanning the collection using simplified and efficient rules: In this case, every document is processed using patterns and rules highly optimized for speed. In this model the complete scanning process is repeated for each new task (Section 3).
- Exploiting general-purpose search engines: To avoid scanning all documents in a collection, some systems use generic search engines to zoom in on relevant documents (Section 4).
- Using specialized indexes and custom search engines: A special-purpose search engine can index and query annotations useful for a predefined family of information extraction tasks. In some cases this may allow doing extraction over the index only, for dramatic efficiency gains (Section 5).
- Distributed processing: We briefly describe representative distributed data mining solutions that could be applied for scalable text mining and information extraction (Section 6).

Some of the efficiency approaches can degrade extraction completeness and accuracy, as well as generality and applicability of the resulting solutions. We discuss these challenges and promising research directions in Section 7, which concludes the paper.

2 Background: Information Extraction

The general information extraction process is outlined in Figure 1 (adapted from [15]). In general, a document is broken up into chunks (e.g., sentences or paragraphs), and rules or patterns applied to identify entities. For the NER task, systems usually scan each document for textual "clues" indicating presence of a useful entity. Most common clues are the text surrounding the entity and the text of entity itself, as well as part-of-speech tags and word classes if available. Then, for the RE task, scenario-level extraction patterns are applied to infer relationships between the extracted entities (See [15] for natural language processing-focused overview). Some systems can use statistics collected over the whole collection to assign confidence scores to extracted objects. Either after or during the extraction, information can be merged for multiple occurrences of the same object (and different objects with shared attribute values can be disambiguated). These postprocessing steps are relatively fast compared to the actual information extraction process, and are beyond the scope of this paper. Note that entities can be extracted independently of the relation, so that entity annotations can be shared across multiple relation extraction tasks.

The different stages in the extraction process have varying computational requirements. Most probabilistic parsers or taggers use a form of Viterbi algorithm for decoding the most likely sequence of tags (e.g., [22]), which have linear complexity with respect to sequence length and corpus size, but with widely varying constants. Pattern-based extraction systems (e.g., [1]) apply each pattern to each candidate passage in a document, resulting in complexity linear with the size the collection and the number of patterns used (which can be large for partially supervised and unsupervised extraction systems). Complexity of rule-based extraction systems is difficult to estimate, but is consistently reported to be high, as it usually takes seconds to process a medium-size document (3K), resulting in estimates of years [27] required to process large document collections.

[1] Most preprocessing steps only need to be run once if we store the annotated text. Also, the preprocessing step is inherently parallelizable. We discuss these issues in subsequent sections.

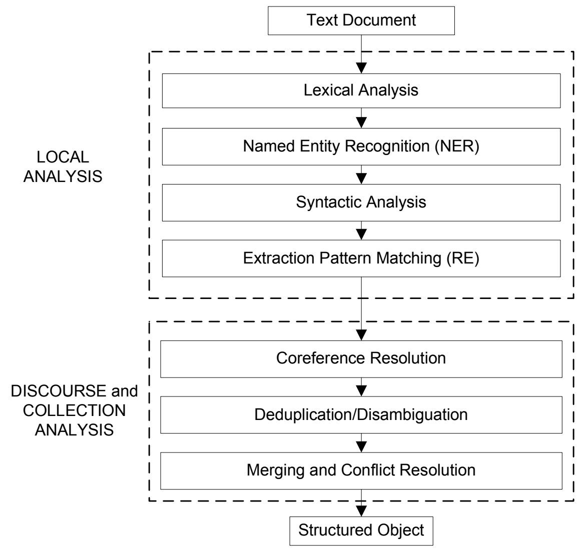


Figure 1: Typical stages in the information extraction process.

3 Scanning Large Document Collections

A traditional information extraction approach is to scan every document in a given collection, possibly using various forms of filtering to discard documents (or passages) as early as possible in the process. One approach is to use a classifier or hand-crafted patterns. Only the documents that match these (presumably "cheap") filters are processed further. For example, a system for extracting information about disease outbreak events [16] uses hand-crafted regular expressions to select documents to process further with the full features extraction system. These filtering patterns are usually designed to have high recall (i.e., not to discard useful documents) while ignoring a large fraction of the non-useful documents. In some settings (e.g., focused crawling), it is possible to discard documents without processing the document text (e.g., by applying rules to the document URLs or links pointing at the document) [5, 8]. Efficient text filtering (e.g., by using optimized regular expression matching and even specialized hardware solutions) were reported for text filtering as early as 1993 [24], and could be naturally adapted to work with information extraction.

A different approach is to use only extremely simple, "cheap" extraction patterns, and apply them to *every* document in the collection [25]. This relies on the assumption that information in large text collections appears *redundantly*, and at least some of the occurrences of a desired entity or relationship will match one of the simple patterns. The authors describe experiments with extracting pairs of noun phrases for the *is-a* relations (e.g., ⟨"MCI WorldCom", "phone company"⟩). The system uses 15 simple lexical and part-of-speech patterns, followed by a more expensive machine learning-based postprocessing step. The authors report requiring 10 days to process a 15GB document collection (approximately 5 million documents) using this implementation, which is still an order of magnitude slower than part-of-speech tagging. Interestingly, the reported accuracy of the simple lexical pattern-based system is comparable to the accuracy of the much slower approach requiring full syntactic parsing of each sentence.

The created annotations can be stored and re-used for all future extraction tasks that require such information (e.g., locations of the named entities in the documents to be used for the relation extraction task). Hence, the initial pre-processing effort would amortize if the annotations are general enough. Another example of such preprocessing is indexing the words and the documents in which they occur, as typically done by general-purpose text search engines. Next we describe two scalable information extraction architectures that make use of such indexing.

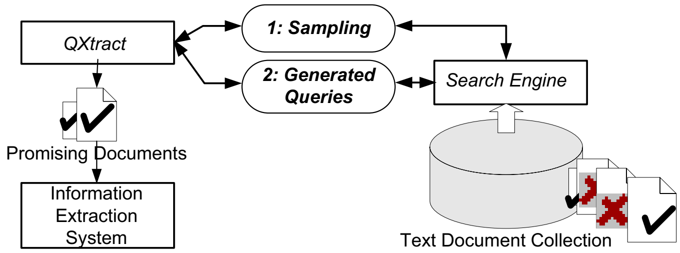


Figure 2: Querying generic search engines for scalable information extraction.

4 Exploiting General-Purpose Search Engines

Often, only a small fraction of the documents contain information that is relevant to the extraction task. Hence it is not necessary for extraction completeness –or desirable from an efficiency viewpoint– to run the information extraction system over every database document. Furthermore, if a document collection is the set of all web pages indexed by a search engine such as Google, then it is virtually impossible to extract information from every page. For these reasons, an intuitive approach is to zoom in on the promising documents, while ignoring the rest. This approach was introduced in the *QXtract* system [2] for efficiently extracting relations from large document collections.

The general *QXtract* architecture is outlined in Figure 2. Starting with a set of user-provided seed tuples for the target relation, *QXtract* retrieves a small sample of documents, likely to be useful to the extraction system, as well as other randomly chosen documents, likely to be useless to the extraction system. The information extraction system is run over this sample set, producing as output a set of extracted tuples and the identifiers of useful documents. The documents in the sample are thus labeled automatically as either positive or negative examples, where the positive examples represent the documents in the sample from which the information extraction system was able to produce tuples. These examples allow *QXtract* to derive queries targeted to match –and retrieve– documents similar to the positive examples. These queries are used to retrieve a set of promising documents from the database, to be returned as *QXtract*'s output and finally processed by the information extraction system. The performance improvement can be substantial: *QXtract* allows a state-of-the-art information extraction system to extract 48% of the tuples in the target relation when retrieving only 5% of the documents in the collection, for an order of magnitude increase in efficiency at the expense of extraction completeness. The *QXtract* approach is general in that any information extraction system could be plugged and use *QXtract* as an interface to large collections, hidden web databases, or, in principle, the web at large.

More recently, Etzioni et. al introduced the KnowItAll system [14] for extracting concepts and relationships from the web (e.g., the "is-a" relationship between noun phrases). KnowItAll uses a set of predefined generic extraction rules (e.g., "NP1 such as NP2", where NP stands for noun phrase, indicating that a string tagged as NP2 in a document is an instance of a class named in NP1.). To retrieve candidate documents, KnowItAll automatically generates queries by instantiating the general patterns with the target class (e.g., for the "cities" class, a query would be "cities such as") and submits these to a generic web search engine such as Google. The returned documents are retrieved, parsed with part-of-speech tagger, and patterns applied following the general information extraction framework of Section 2. As an interesting use of web search engines, KnowItAll estimates the confidence of the extracted values by using web co-occurrence statistics via Google hit counts. Specifically, KnowItAll uses a form of pointwise mutual information (PMI) between words and phrases estimated similarly to Turneys PMI-IR algorithm [32]. PMI-IR estimates mutual information between the class name (e.g., "cities") and a proposed city instance (e.g., "Seattle") by computing web hit counts of each phrase individually, as well as the number of pages containing the phrase "cities such as Seattle". Hence, for each candidate concept or relation tuple, KnowItAll would issue at least three distinct web search queries (first to retrieve a document, and then two more queries to compute the PMI-IR measure).

In addition to improving the efficiency of extraction, a system that queries a generic search interface might be adapted to extract relations from "hidden-web" databases only accessible via generic search interfaces [4, 18], allowing a system to process relevant documents not otherwise reachable via crawling or scanning mechanisms.

While clearly more feasible than processing every document in the collection, both *QXtract* and KnowItAll can still require days (or even weeks) to extract a large fraction of all relation tuples or concepts hidden in the collection documents. This limitation is addressed by more recent systems in the KnowItAll family, as discussed in the next section. Another shortcoming of both systems is retrieving thousands of results for each query (a functionality rarely supported by generic search engines). By removing reliance on *generic* web search engines and incorporating extraction-specific features at *index time*, it is possible to dramatically increase information extraction efficiency and scalability, as we describe next.

5 Using Specialized Indexes and Search Engines

General-purpose search engines are designed for short keyword queries and for retrieving relatively few results per query. In contrast, information extraction systems can submit sophisticated and specific queries and request many or all query results. To better support information extraction, Cafarella et al. [7] introduced the Bindings Engine (BE), which supports queries containing typed variables and some linguistic functions. For example, in response to the query "Mayors such as ProperNoun(Head(NP))", BE would return a list of proper nouns that appear in that context. To accomplish this, BE indexes the *neighborhood* of words (Figure 3 adapted from Cafarella et al. [7]).

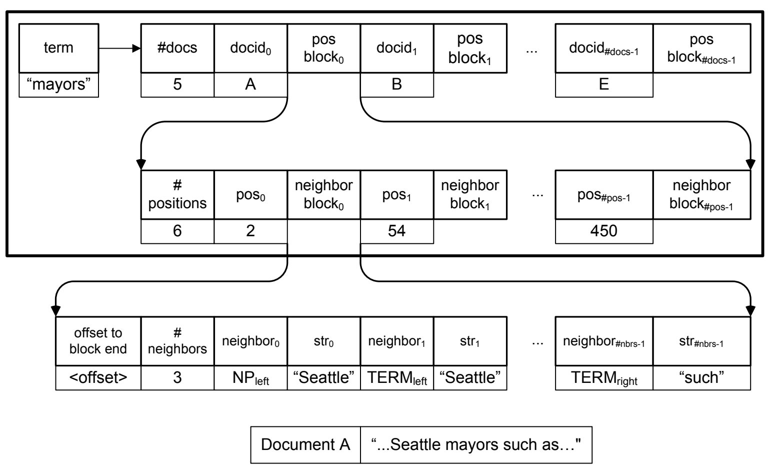


Figure 3: The BE engine neighborhood index.

The neighborhood index is similar to the inverted list index [31], but for each posting BE stores up to K words immediately to the left and to the right of each term. Additionally, BE stores all part-of-speech labels for each term (and, in principle, any other available semantic information) computed at index time. By using this expanded index, a query such as "mayors such as", which might be issued by a class extraction system for extracting names of all mayors, will retrieve the postings list for the word "mayors" and then scan the list returning all proper noun phrases that are preceded by the "such as" string. BE is well suited to extraction patterns using exact phrases (e.g., DIPRE [5] and KnowItAll [14]). As reported by Cafarella et al. in [6], the KnowItNow information extraction system and other systems in the KnowItAll family[2] use the BE search engine to quickly extract information from an indexed web snapshot of 90 million documents.

[2] Available at http://www.cs.washington.edu/research/knowitall/.

A related approach has been used for extraction-based Question Answering (notably by Prager et al. [26] and Chu et al. [10]), where a system retrieves short *answers* to natural language questions extracted at query time from the text documents in the collection. During an indexing pass over the collection, the entities predicted to be potential answers to questions are extracted and stored, and at query time only the documents (or passages) containing an entity of appropriate type (e.g., person name) are retrieved for extracting candidate answers. An intriguing new search engine was recently demonstrated by Resnik et al. [28] for indexing and searching linguistic (e.g., syntactic) structures[3] but it has not yet been formally evaluated for relation extraction or question answering tasks.

Unfortunately, word neighborhood indexing may not be directly amenable for extraction patterns without lexical items (e.g., patterns such as "Adjective ProperNoun(Head(NP))"), for patterns with only frequent words in patterns (e.g., "⟨Organization⟩ in ⟨Location⟩" [1]) or for probabilistic extraction models (e.g., HMMs [23] or CRFs [29]). Furthermore, extractors that rely on web page structures such as HTML lists (e.g., [11, 14]) still have to retrieve the complete document and apply extractors as the original *QXtract* or KnowItAll system would.

More generally, annotations such as part-of-speech tags and sentence boundaries can be viewed as adding partial structure to the text documents, which can then be represented in a semi-structured form (e.g., in XML format), and indexed for fast querying (e.g., [20]). Preliminary question answering results over annotated and indexed XML documents [21] indicate that with a rich schema and carefully constructed XPath queries it may be possible to represent question answering and information extraction as a retrieval task. We explore this idea further in Section 7.

6 Distributed Processing

So far we focused on algorithmic techniques for scaling up information extraction. Parallelization and distributed processing are attractive alternatives for processing extremely large collections, such as the billions of documents on the web. Information extraction is particularly amenable to parallelization, as the main information extraction steps, (e.g., part-of-speech tagging and shallow syntactic parsing) operate over each document independently (e.g., [13]). Hence, most parallel data mining and distributed processing architectures (e.g., Google's MapReduce [12]) might be easily adapted for information extraction over large collections.

Extracting information is only one of the steps in large scale web mining and extraction. Discovering useful document sources [3, 19], crawling (retrieving documents), extracting and indexing relevant document features, and other tasks are all required for a complete, enterprise-scale systems. IBM's WebFountain [13, 17], an influential end-to-end system, puts these steps together for information extraction and text mining from the web. WebFountain retrieves, processes, extracts and indexes information from billions of documents on the web and in local collections. The WebFountain approach includes both algorithmic and hardware solutions, and uses a heavily distributed architecture with clusters of nodes devoted to crawling, extracting and indexing web page content. WebFountain is a blackboard architecture that allows multiple *annotators* (i.e., extraction systems) to store tags (e.g., named entities) or any other annotations with each document for further processing. Unfortunately, a distributed architecture with hundreds of machines (WebFountain) or thousands of machines (Google's Map/Reduce) requires significant resources to create and maintain, which limits the applicability of this approach. As we have shown previously, it is possible to perform scalable information extraction even with modest hardware resources.

[3] Available at http://lse.umiacs.umd.edu:8080/.

7 Opportunities and Challenges

We described four general approaches for scaling information extraction to large document collections. A truism stating that "there is no free lunch" applies. Current algorithmic techniques either trade off information extraction accuracy and completeness for speed (e.g., Sections 3 and 4), or impose restrictions on the types of extraction patterns supported (Section 5). Hence, choosing the appropriate approach is heavily dependent on the application and use requirements.

One promising general approach that we mentioned earlier is to store the semantically annotated documents (e.g., with part-of-speech or named entity tags) in semi-structured form (e.g., in XML). The annotated documents could be indexed to speed up future information extraction runs. While many indexing and querying methods for semi-structured data (e.g. [20]) have been developed in different contexts, these techniques have not been adequately explored for information extraction and are a promising direction for research.

A dimension of information extraction scalability not addressed in this survey is a trade-off between domain independence and extraction accuracy. While named entity extraction technology is relatively mature and is generally accurate for common entity types (e.g., person and location names), domain-independent relation and event extraction techniques are still error-prone, and are an active area of natural language processing and text mining research. One interesting research direction is to apply probabilistic query processing techniques (reviewed in [30]) to derive usable query answers from the noisy information extracted from text.

As we discussed, redundancy and variability in large document collections can mitigate the inherent difficulty in interpreting natural language text. By operating over large collections, information extraction systems can significantly improve both accuracy and coverage of the extracted information. For this, efficient techniques for extracting information from such large document collections are crucial, and would greatly enhance our ability to manage and exploit the available textual information.

References

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL 2000)*, 2000.

[2] Eugene Agichtein and Luis Gravano. Querying text databases for efficient information extraction. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE 2003)*, 2003.

[3] Abdullah Al-Hamdani and Gultekin Ozsoyoglu. Selecting topics for web resource discovery: Efficiency issues in a database approach. In *Proceedings of the DEXA Conference*, 2003.

[4] BrightPlanet.com LLC. The Deep Web: Surfacing hidden value. Available at http://www.completeplanet.com/Tutorials/DeepWeb/index.asp, July 2000.

[5] Sergey Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the First International Workshop on the Web and Databases, WebDB 1998*, 1998.

[6] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. KnowItNow: Fast, scalable information extraction from the web. In *Conference on Human Language Technologies (HLT/EMNLP)*, 2005.

[7] Michael J. Cafarella and Oren Etzioni. A search engine for natural language applications. In *Proceedings of the World Wide Web Conference (WWW)*, 2005.

[8] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640, May 1999.

[9] Surajit Chaudhuri, Raghu Ramakrishnan, and Gerhard Weikum. Integrating db and ir technologies: What is the sound of one hand clapping? In *Second Biennial Conference on Innovative Data Systems Research*, 2005.

[10] Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, Abraham Ittycheria, and Sasha Blair-Goldensohn. IBM's PIQUANT II in TREC 2004. In *13th Text Retrieval Conference (TREC)*, 2004.

[11] William W Cohen, Matthew Hurst, and Lee S Jensen. A flexible learning system for wrapping tables and lists in html documents. In *Proceedings of the World Wide Web Conference (WWW)*, 2002.

[12] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[13] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Sridhar Rajagopalan Tapas Kanungo, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and SemSeeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the World Wide Web Conference (WWW)*, 2003.

[14] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S.Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 2005.

[15] Ralph Grishman. Information extraction: Techniques and challenges. In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, International Summer School, (SCIE-97)*, pages 10–27, 1997.

[16] Ralph Grishman, Silja Huttunen, and Roman Yangarber. Information extraction for enhanced access to disease outbreak reports. *Journal of Biomedical Informatics*, 35(4):236–246, August 2002.

[17] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a WebFountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 2004.

[18] Panagiotis G. Ipeirotis and Luis Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB)*, 2002.

[19] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001.

[20] Quanzhong Li and Bongki Moon. Indexing and querying xml data for regular path expressions. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, 2001.

[21] Ken C. Litkowski. Question answering using xml- tagged documents. In *The Eleventh Text REtrieval Conference (TREC)*, 2002.

[22] Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[23] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning*, 2000.

[24] M. Mettler. TREC-II routing experiments with the TRW/Paracel Fast Data Finder. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, 1993.

[25] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *Conference on Computational Linguistics (COLING)*, 2004.

[26] John Prager, Eric Brown, and Anni Coden. Question-answering by predictive annotation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2000.

[27] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. The terascale challenge. In *KDD Workshop on Mining for and from the Semantic Web*, 2004.

[28] Philip Resnik and Aaron Elkiss. The linguist's search engine: An overview (demonstration). In *ACL*, 2005.

[29] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, 2005.

[30] Dan Suciu and Nilesh Dalvi. Foundations of probabilistic query answering. Tutorial at the *ACM SIGMOD International Conference on Management of Data*, 2005.

[31] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, December 2001.

[32] Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *European Conference on Machine Learning (ECML)*, 2001.

Fast and Furious Text Mining

Joel D. Martin
National Research Council, Canada
joel.martin@nrc.gc.ca

Abstract

Text mining studies in biology are often limited to thousands instead of millions of Medline records or are very slow. However, with a modified search engine, many common text mining tasks can be done rapidly. In fact, some information extraction and text categorization tasks can be achieved in seconds or minutes even across tens of gigabytes of (previously indexed) text. In this paper, we present TLM, an efficient implementation of a text analysis engine that uses a highly expressive query language. With this language, users can create queries that quickly accomplish what previously required several different custom-built systems to achieve.

1 Introduction

Text mining is our only hope to find all the literature references to specific facts, such as gene or protein interactions. At present, it is still a hope and not fully a reality. Most text mining tools work for a small number of abstracts, or more rarely full-text articles (e.g., [1]). Some do work for millions of articles but are relatively slow (e.g., hours to days, [6]). Still other approaches have been designed to process millions of articles quickly, but they can apparently lose considerable accuracy compared to slower methods (e.g., [11]).

The challenge then is to build tools that permit a wide variety of very rapid text mining across millions of documents. This challenge is even more relevant when we consider that the next generation of text mining tools will be expected to handle terabytes of full-text articles, not just gigabytes of abstracts. If we cannot rapidly mine the text of Medline, how can we hope to handle the full articles?

Below, we describe a text analysis engine called TLM (Text and Language Mining) with a highly expressive query language. TLM is a principle component of our integrated suite of tools called LitMiner ([9]). TLM permits queries that can quickly accomplish what previously required several different custom-built systems to achieve.

2 TLM: A Text Analysis Engine

How is text analysis different from search? On a search engine, users compose words into queries and expect lists of documents in return. That is an important capability and many other tasks are made possible by search engines. However, our text mining tasks often require a little more and would be easier with a slightly different engine.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

A text analysis engine would let us compose queries out of words and other entities such as punctuation, tags, part of speech, phrases, sentences, etc. For example, we might want all the occurrences of **`THA`** in parentheses: **`'X(' THA ')X'`** (the *X* is a wildcard character for punctuation). Instead, we might want to refer to any occurrence of any verb: **`is <Verb> by`**, or might want to find sentences that contain two or more genes, **`<Sentence> > (<Gene> .. <Gene>)`**.

Second, a text analysis engine would return results of different types, including documents, but also sections, sentences, phrases, terms, and words. For example, in asking for sentences that contain two genes, we may not care which documents they come from. We just want the statements themselves so we can see which genes are said to inhibit each other. In asking for three words before THA, **`* * * 'X(' THA ')X'`**, we do not want to see the matching documents. We want to see the possible expansions for the acronym THA, such as Total Hip Arthroplasty(374), tetrahydroacridine(25), or Tokai High Avoider(4).[1]

Third, a text analysis engine should permit rapid statistical analysis of the text pieces that are returned. There is a wide range of possible analyses, including simple frequencies in documents or sentences and ranging to more complex distributions.

2.1 Engine Design

TLM is a relatively mature implementation of a text search and analysis engine. Figure 1 shows one client graphical user interface (GUI) that is connected to a remote installation of TLM. The figure illustrates the query **`* * * 'X(' THA ')X'`** and shows its output. TLM has many added conveniences for users and has been optimized for many types of search, but its operation can be summarized by five basic ideas that are outlined below. Although none of these ideas is completely new, some aspects are unusual or unique when compared to search engines. Furthermore, the combination of the five ideas is new. That combination is essential for supporting the above definition of text analysis.

The first and most fundamental idea behind TLM's functionality is borrowed directly from search engines. It is an inverted index of the positions of words (e.g., [2]). Uniquely in TLM, this idea is extended to include strings of spaces and punctuation as well as words. Any document or collection of documents can be described as a list of words (or punctuation) and their position of occurrence. For example, if we were indexing the current paragraph, we would assign the position 1 to the word *The*, 2 to *first*, etc. All of these words and their positions can now be organized as in a back-of-the-book index. Each word can be connected to a list of the positions in which it appears. In TLM specifically, each word or collection of adjacent punctuation (called a *separator*) is connected to a list of the positions of that term in documents. In a collection of multiple documents, the position could include the document number or could ignore it (e.g., [5]; [3]). In our collection TLMtest, the word **`tumourogenic`** appears 10 times in 9 Medline abstracts. The index stores the word position of each of those occurrences. Similarly, the separator **`',,,, '`** (four commas and a space) appears once, and that position is stored.

Once we know the positions of each word and separator, we can ask how often two particular words occur near each other. We do this simply by comparing the lists of positions and checking certain conditions. For example, we might want to find all the (up to) four word phrases that contain *cancers* and *tumours*. Our search engine can retrieve the lists of word positions for *cancers* and for *tumours* and can iterate through those lists looking for two, three, or four word phrases that contain both. For example, suppose *cancers* appears as the 10th and 45th word of a document and *tumours* appears as the 20th and 43rd word of the same document. Scanning

[1] All example queries described in this paper were run against the TLMTest collection. For these examples, TLM was running on a 2.4 GHz AMD Opteron. The TLMTest collection is a set of 15,176,580 Medline records. A collection of important fields was included (eg., title, abstract, MeSH terms, etc.) resulting in approximately 22 Gigabytes of text. This text was indexed by TLM in approximately 18 hours. An additional 24.5 hours was used to create a list of potentially useful tags such as **`<Sentence>, <Noun>, <ContainsDigit>`**. The algorithms used for division into sentences and the part-of-speech tagging are very simple and will be replaced in future uses of TLMTest.

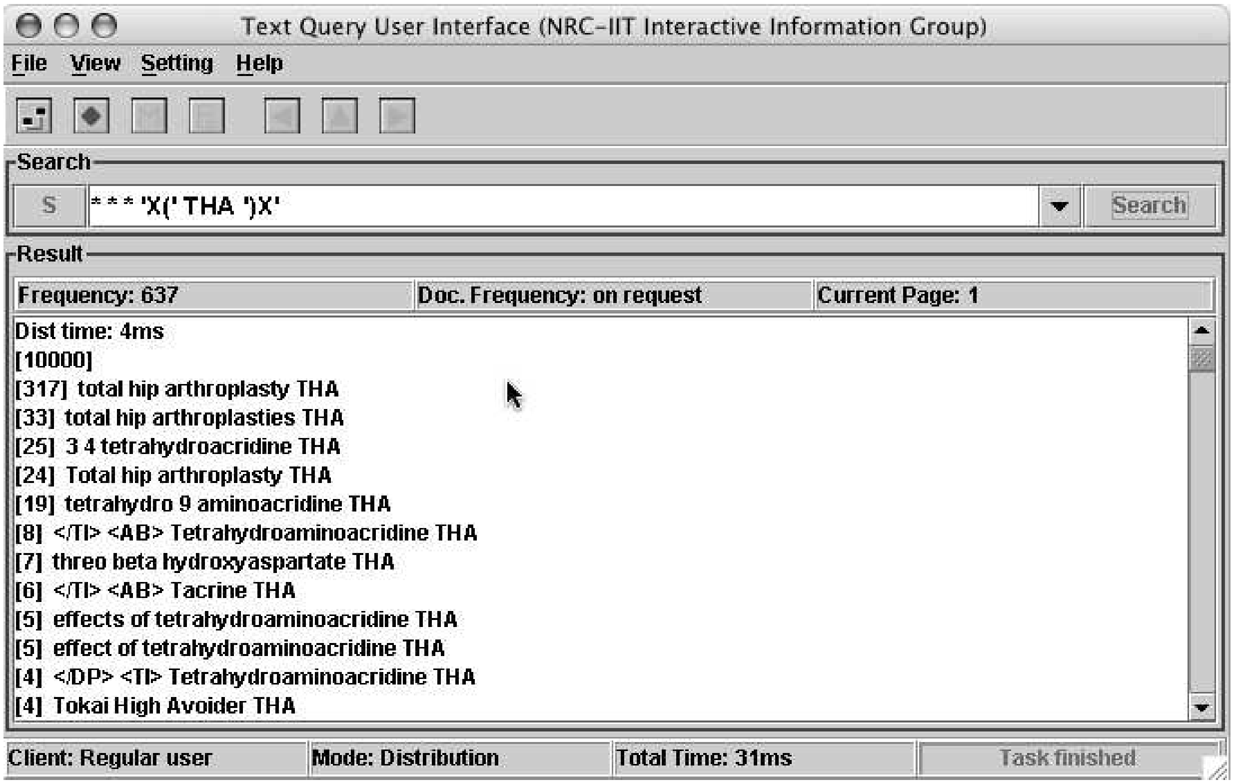


Figure 1: A screenshot of a GUI client to TLM. The figure show the overall frequency for the query, the overall number of documents that contain the query, and the time to run the query. As well, in the center frame, the results are organized in a frequency distribution.

those word position lists reveals that the range of positions from the 43rd word to the 45th word describes a phrase containing both *cancers* and *tumours*. In the TLMTest collection, we find 128 occurrences in 120 documents. One of the resulting phrases is *tumours, including cancers*, which appears twice.

The second idea is that, unlike most search engines, TLM returns parts of documents. Usually, we don't want to see the whole document and only want a snippet, like Google's several word summary that appears under every result, or a passage (e.g., [8]). TLM does this by returning a range of two positions, say the position of the word *'tumours'* and the position of the nearby word *'cancers'* (cf., [3]). If they appear as the 43rd and 45th words respectively, the range is from 43 to 45. All query operations process these ranges of word positions.

The use of ranges leads directly to the third idea, that lists of ranges can be given a tag name. This idea is similar to macros or variables in some search engines, but its simple syntax is unique to TLM. For example, all the titles in a collection of documents can be described as a list of word position ranges. If that list is given the name **`<Title>`** or **`<TI>`**, the user can easily refer to the list of ranges in later queries. These special tags could be based on XML tags that actually appear in the text or can be defined by the user during the creation of queries. TLM stores these tags in an inverted index of ranges, just like the inverted index of word and separator positions.

The fourth fundamental idea is that the result of a query can be statistical information rather than just a simple list of discovered documents. This is a common text mining activity that is unique as part of a search engine. The simplest form of such statistical information is the frequency and document frequency of a query. For example, the query **`mdm2/i | hMDM2 | mouse double minute 2`** matches 7,811 words or phrases in the TLMTest. These matches occur in 6,248 sentences, and these sentences occur in 1,769 documents. A second useful form of statistical output is a frequency distribution of the distinct matches to a query. As an example, the query, **`blood near growth near vessel*`**, is a request for phrases that have the word *blood* near (within 10 words of) the word *growth* which together are near any word beginning with the six letters *vessel*. This would tell us that there are 280 occurrences of *blood vessel growth*, 90 occurrences of *growth of new blood vessels*, 52 of *growth of blood vessels*, 6 of *growth of new capillary blood vessels*, etc.

A fifth fundamental idea is that query speed is more important than conserving disk space. The availability of low cost massive storage, allows us to store multiple indexes that each accelerate different types of queries. Many search engines have indexes based on compression technology (e.g., [13]) and advertise that they require disk space that is only a small percentage of the original text size. TLM was designed to make many aspects of text mining fast. For example, there is a case-sensitive inverted index and a case-insensitive conversion index. These two indexes allow the user to specify specific capitalization patterns. The query **`hMDM2`** is a request for matches to exactly that term. In contrast, the query **`hmdm2/i`**, is a request for matches to *hMDM2*, *HMDM2*, *hmdm2*, *hMdm2*, etc. In addition, many common queries are pre-computed with results stored in a file. As a result of all this, the indexing file system can be four times the size of the original text (or more). If any part of that is removed, some type of common query would be slower to calculate.

All that follows and all that we have tried as part of LitMiner should be possible given a text analysis engine implementing these five ideas. The engine should have an inverted index of words, separators, and tag ranges. It should return parts of documents as ranges of word positions that match the query and should permit statistical post processing before giving the user the answer. Finally, it should prioritize fast text mining over conserving disk space.

2.2 A summary of the query language.

We have already presented a few example queries, with only a simple definition of the query language. More example queries are shown in Table 1. From these examples, it is obvious that the enhanced expressive power of TLM is in exchange for increased complexity. Most internet users would prefer Google's simple syntax to these complex queries. However, in many cases, the simplicity can be restored without losing the power, by using interfaces such as LitMiner that bury the query complexity behind GUI buttons.

In general, a TLM query is composed of words, tags, or separators connected by pairwise operators. All operators describe transformations of two lists of word position ranges into a resulting list of word position ranges. There are four major operators in TLM, as well as a syntax for tag definitions.

`interact* <Adverb>`	All adverbs that appear immediately after the word stem *'interact'*.
`<NounPhrase> > (<TI> > cancer)`	All noun phrases that appear in titles that contain the word *'cancer'*.
`interact* near protein*`	All passages that have the word stem interact near (within 10 words) of the word stem protein.

Table 1: Some example queries for TLM. See section 2.2 for an interpretation of the query language's operators.

The first major operator, and the one with the highest precedence is adjacency. When two words are separated by a space in a query, **`open heart`**, it forms a request for phrases that contain the first word followed immediately by the second word.

The second major operator is the 'or' operator. It simply merges two lists of word ranges. For example, **`mdm2 | MDM2`** is a request for all the word position ranges that contain just the word *mdm2* and all the word ranges that contain just the word *MDM2*. Then it merges those lists of word ranges.

The third major operator restricts answers to have two nearby parts. There are actually two forms of this operator, **near** and **..**. A query like **word1 near word2** is a request for all the word position ranges in which the two words appear within 10 words of each other. Similarly, a query could request that the two words be nearby and in order, **word1 .. word2**. It is a request for all the ranges in which the two words appear within 10 words of each other and *word1* appears first.

Both the **near** and **..** operators can be modified with specified distances. The simplest modification is to add '/' followed by a number. The modified operation **near/2** means that two ranges must be near, within two words. Similarly, **../100** means that the two ranges must be in order and within 100 words. The distance can be further modified by specifying a minimum distance as well. For example, **near{4,10}** means that the two ranges must be at least four words apart and up to 10 words apart.

The fourth major operator tests containment and was inspired by [3]. Considering two word position ranges, it is possible for them to overlap, for one to contain the other, or for them to be non-overlapping. In TLM, queries can force all answers to contain at least one example of another range. For example, the query **<TI> > geopolitical** is a request for word ranges that are whole titles, but only the ones that contain the word **geopolitical**. This query could be reversed and be a request for **geopolitical < <TI>**, ranges of length 1 with the word 'geopolitical', but only those occurrences inside titles.

The common search engine operators **and** and **not** were purposely omitted from this description, because they are not flexible enough for text analysis. In most query languages, **and** is a request for documents containing both of two words (or boolean expressions). In TLM, a query such as **<DOC> > protein > gene** is also a request for documents that contain both words. This approach is more flexible than the operator **and**, because it also applies to smaller document segments such as abstracts, or sentences, or phrases. For example, **<TI> > protein > gene**. Similarly, **not** typically is a request for documents that do not contain a particular word. In TLM, a query such as **<DOC> /> protein** would have the same effect, while also permitting **<Sentence> /> protein**.

The TLM query language also permits the definition of variables to hold partial query results. Multiple variable assignments can appear in a single query and the variable value is available even inside the same query but to the right of the first appearance. For example, the query **($det = (the|a|an)) .. <$det>** is a request for two determiners that appear near each other. As in this example, a variable name, which always begins with a $, is assigned the results of a query using an **=** operator. That variable then becomes a tag name for future queries by simply enclosing the variable name within **< >**.

3 TLM for Text Mining

TLM is a step closer to what users need. TLM queries have greater expressive power compared to most search engines, because a wider range of textual patterns can be specified. In exchange for much more complex queries, this greater expressive power allows queries to better correspond to real world entities. In biological (or any) text mining, there is a gap between a referent, such as a gene, and how we refer to that entity, i.e., the gene. In some sense, all queries are 100% accurate because they return exactly what they are supposed to do. Practically, though, they rarely find all and only what we want them to. TLM is not perfect, but it is a step beyond many search engines.

In this section, we will consider a few examples of how TLM can be useful for biological text mining. In none of these illustrations do we prove that TLM results are more accurate than previous results, only that they are similar. The point of this exercise is that TLM can do relevant text mining and can do it rapidly. We will leave it to future work to discover the best ways to use TLM to produce the highest accuracy, precision, and recall.

3.1 MedMiner

The goals of using TLM for mining the biological literature match many of those for MedMiner ([12]). MedMiner was designed to access 'extrinsic' information about genes. It was composed of three key components: internet-based querying of multiple databases, text filtering, and a carefully designed user interface. TLM could address the querying and text filtering. Our LitMiner system is our attempt to create a carefully designed interface.

In illustrating the value of their system, the authors considered a specific biological relationship (inhibition) between two genes, MDM2 and P53. They argued for their system on the basis of the completeness of the result, the amount of irrelevant information presented, the query complexity, and the running time.

3.1.1 More complete and fewer irrelevant sentences

TLM can be used to further increase the completeness of the results. As the authors noted, MedMiner will "miss relevant concepts if they are not represented in the keywords". The interactive use of TLM with frequency distributions can partially address this problem.

The gene, MDM2, could be represented by any number of synonyms. A simple string of queries on TLM can tell us new terms to add. Each of the following queries has results that suggest new synonyms. The first query is a request for four words followed by MDM2. This query suggests that the two most frequent expansions of MDM2 are *murine double minute 2* and *mouse double minute 2*. The third column shows the accepted suggestions.

<table>
<tr><th>Query</th><th>time</th><th>Suggested synonyms</th></tr>
<tr><td>* * * * MDM2</td><td>330 ms</td><td>murine double minute 2 | mouse double minute 2</td></tr>
<tr><td>MDM2*/i | MDM*/i (2 | ii/i)</td><td>1120 ms</td><td>MDM2 | mdm2 | Mdm2 | mdm 2 | MDM 2 | Mdm 2</td></tr>
<tr><td>hMDM2*/i | hMDM*/i 2</td><td>200 ms</td><td>hMDM2 | hmdm2 | hMdm2</td></tr>
</table>

In a few seconds, we have a better query than simply **MDM2**. If we include the synonyms from Entrez Gene [10] and truncate important words, we produce a more complex query for MDM2 (Table 2). This query took about 30 seconds to create and about 9.7 s to run.

These queries can yield more complete results. In addition, like MedMiner, TLM's results for inhibition displays the phrase or the sentence that indicates the relationship rather than merely identifying the document. It is also possible to highlight the gene names and inhibition phrases, because TLM returns the positions of matches.

3.1.2 Query complexity and running time

As shown in Table 2, the TLM queries created for MDM2 and P53 are rather complex as is the query for identifying some sort of relationship between genes. However, TLM provides user defined tags which greatly simplifies later queries. After the first three complex queries in the table have been submitted, the very simple fourth query can be submitted to ask for all phrases in Medline where MDM2 and P53 are said to interact.

The MedMiner time for a similar inhibition query was approximately 60 s and the equivalent PubMed query was 30 s when that paper was first written. It is not easy to compare these times with TLM. As a preparatory step, TLM requires between 2 and 60 s to perform each of the individual gene queries like those shown in Table 2. In addition, it requires approximately 6 minutes to process the interaction verb query in row 3 of Table 2. However, after that preparation, requests between arbitrary pairs of genes require an average of 4.8 s. This

<table>
<tr><th>Run time</th><th>Frequency</th><th>Query</th></tr>
<tr><td>9.7 s</td><td>11,010</td><td>$mdm2 = MDM2/i | MDM/i 2 | HDM2/i | HDM/i 2 | MGC71221/i | P53/i bind*/i protei*/i | Mouse/i double*/i minute*/i 2 | murine/i double*/i minute*/i 2 | hMDM2/i</td></tr>
<tr><td>1.4 s</td><td>195,159</td><td>$p53 = tp53/i | tp/i 53 | Cys51Stop/i | TRP/i 53 | TRP53/i | p53/i | tp53s/i | Cys51Stops/i | TRP53s/i | p53s/i</td></tr>
<tr><td>372 s</td><td>15,664,040</td><td>$Iverb = (inhibit*/i | block*/i | reduc*/i | decreas*/i | acetylat*/i | activat*/i | target*/i | suppress*/i | stabiliz*/i | regulat*/i | phosphorylat*/i | modulat*/i | is/i ../2 conjugat*/i ../2 to/i | interact*/ i| inhibit*/i | destabiliz*/i | bind*/i | bound/i | associate*/i ../2 with/i)</td></tr>
<tr><td>4.8 s</td><td>719</td><td><$mdm2> n/5 <$p53> > <$Iverb></td></tr>
</table>

Table 2: The queries (and times) needed to find the passages describing the interaction between P53 and MDM2)

suggests a scheme where gene queries and interaction verb queries are updated nightly, allowing users to get more complete pairwise responses in only a few seconds.

Overall, TLM meets many of the same goals as MedMiner but also provides improved performance (assuming some pre-processing) and a fast interactive solution to the problem of missing relevant concepts.

3.2 Finding interactions between sets of proteins

Blaschke et al. ([1]) went beyond a single pair of genes and described a text mining system that scanned 6728 abstracts looking for the pattern **<Protein> .. <InteractionVerb> .. <Protein>**, that is two proteins separated by a verb (or nominalization) that means some form of interaction. In their first example, they scanned for six different proteins separated by several different interaction patterns. The six proteins were, *pell, dorsal, toll, tube, spatzle*, and *cactus*.

Their scan of abstracts rediscovered nine known pairwise interactions between the proteins. The authors noted that frequency of the mention of a relationship can help determine which interactions to predict.

As an illustration, we attacked this same problem with TLM. Table 3 shows the queries created to represent parts of this task and their time to run. Each query was assigned to a variable for later use. The variable called **$Protein** is a list of capitalized and lowercase protein names. That query was combined with **$InteractionVerb** to find patterns of the type sought in the original paper, **protein .. verb .. protein**. In a total time of about 75 seconds, 15 million abstracts were searched and TLM rediscovered the interactions discovered in the original paper. The time for each component query is shown in Table 3.

The query in the fourth row resulted in 57 total phrases, 55 of which were unique. Of all fifteen automatically detected interactions reported in [1], the 57 results contain at least one example interaction for each. Six of the results identified the same relationship verb. TLM did not find exactly the same results, because it was searching all of Medline, it permitted matches across sentence boundaries, and it was only looking for results of length five words or fewer.

Using TLM to follow Blaschke et al.'s example required a few minutes and returned similar results with very few irrelevant phrases. In addition, these results included suggestions of the two known interactions between Pelle and Cactus and between Dorsal and Cactus that the earlier technique missed (*"Cactus inhibits Dorsal"*, *"Pelle proteins Phosphorylation of Cactus"*). The same 57 TLM results also reveal that there is a protein called

<table>
<tr><th>Run time</th><th>Frequency</th><th>Query</th></tr>
<tr><td>1.8 s</td><td>233,899</td><td><code>$Proteins = Pelle | Dorsal | Toll | Tube | Spatzle | Cactus | pelle | dorsal | toll | tube | spatzle | cactus</code></td></tr>
<tr><td>66.7 s</td><td>11,671,613</td><td><code>$InteractionVerb = acetylat*/i | activat*/i | target*/i | suppress*/i | stabiliz*/i | regulat*/i | phosphorylat*/i | modulat*/i | is/i ../2 conjugat*/i ../2 to/i | interact*/i | inhibit*/i | destabiliz*/i | bind*/i | bound/i | associate*/i ../2 with/i</code></td></tr>
<tr><td>5.8 s</td><td>57</td><td><code><$Proteins> ../5 <$Proteins> > <$InteractionVerb></code></td></tr>
</table>

Table 3: The queries (and times) needed to find interactions among Pelle, Dorsal, Toll, Tube, Spatzle, and Cactus.

"Twist" that interacts with Dorsal and another related protein called *"Kra"* (*"Dorsal-interaction proteins (Twist and Cactus)"*, *"Kra associates with Pelle and Tube"*).

We repeated this exercise for the authors' larger protein list for cell cycle control in Drosophila. We constructed a single query (**`$CellCycleProtein`**) for the 91 proteins included in ([1]), using case insensitive searches. This created many irrelevant matches where both proteins were the same. In addition, many pairs of proteins were not matched because of intervening matches. To address these problems, we created one query for each of the protein names. This meant finding, for example, ranges containing *Myb* followed by an interaction verb, then by a cell cycle protein other than Myb.

For this second exercise, we reused the definition for interaction verbs that must occur between each pair. The TLM queries, including the redefinition of the variables CellCycleProtein and InteractionVerb, took a total of 6 minutes, 32 seconds.

The original paper ([1]) rediscovered 28 well-known interactions, 20 possible interactions, and missed one well-known interaction. In the list of 610 resulting phrases from TLM, we also found evidence for 27 of the 28 known interactions and all but five of the possible interactions. The main interaction missed by TLM was between *cdc2* and *twine*. However, TLM did detect the interaction between *cdk* and *p21* that the original paper missed.

TLM clearly supports the extraction of significant facts from large text collections. Specific entities can be identified and relationships between those entities can be correctly discovered. TLM can achieve these and similar tasks in minutes. This is fast enough to allow a tolerable interaction between the user and the text.

4 Text categorization

Another important text mining application is text categorization. Researchers have applied text categorization to label Medline abstracts as relevant or not to some task (e.g., [6]; [11]). For example, in PreBIND, text categorization was used to select papers about protein-protein interactions for later human curation.

In principle, this is similar to search engine retrieval. However, text categorization uses additional computation (slower) to improve the precision and recall (and accuracy) as compared to the results of search. A search on Google might return 100 results with only 10 of them being relevant. In that search, the precision would be 10%. If the search results completely missed 190 other relevant documents, the recall of that search would be 5%. In contrast, text categorization often results in 65% recall and precision ([11]) or even 90% recall and precision ([6]).

The other difference with Google, besides precision and recall, is the time necessary to produce the results.

Google often reports millisecond response time whereas Donaldson et al. [6] quotes a time in days to apply a text categorization model to 12 million Medline records. With other techniques ([11]) text categorization like levels of accuracy can be achieved much more quickly. Even in that second case, though, the authors suggest using a cluster of several processors to achieve fast learning and application of that learning.

TLM can be used to achieve high recall and precision without requiring days or multiple processors. To illustrate this potential for categorization, we recreated the experiment described in [6]. For this experiment, we used the following technique. From the training examples of protein-protein interaction abstracts, we extracted two general types of features: *"A appears in the document"*, *"A near/5 B"* In those features, A and B refer to one or two word phrases. Among those thousands of possible features, we selected the 5000 that individually were most diagnostic in determining whether a document was a positive example or a negative one. As in the previous study, we used information gain to select those features. Then we applied Ripper ([4]) to learn a boolean expression of the features that would select the positive documents while excluding the negative ones. These boolean expressions were translated into acceptable TLM queries allowing rapid application across all of Medline.

We divided the development set into 10 folds and performed cross-validation, each time training on 90% and testing on the remaining 10%. As a result, we found a precision of 89% and a recall of 86%. Both of these numbers are lower than, but similar to, the results reported in [6]. From past studies, we can expect this new technique to always under-perform Support Vector Machines (e.g., [7]; [14]). However, we expect the new technique to always outperform techniques such as those in [11] again based on performance in those same past studies.

In addition to the high precision and recall, TLM plus Ripper was fast. In our illustration, a single query that resulted from applying Ripper required an average of 85.25 s when submitted to TLM. This is much shorter than the hours necessary to apply an SVM. As well, it is much faster than would be possible with any non-index based technique. In fact, in contrast to the suggestions in [11], we are able to achieve reasonable performance with a single CPU and several users.

As for the case of identifying specific interactions, we have only shown that TLM can be used to achieve similar results quickly. More work has to be done to devise and evaluate a scheme to create consistently high recall and precision while still requiring only a few minutes.

5 Discussion

A text analysis engine is a necessary tool for the future of text mining in biology and other fields. In contrast to search engines, in a text analysis engine, queries are composed of not just words, the results are not just documents, and the final answer is not just a list. Queries can contain punctuation, tags, variables, etc. Results can be documents, sections, topic-based passages, paragraphs, sentences, phrases, etc. The final answers can be a list or could be multiple levels of frequency counts or a frequency distribution.

One example of such a text analysis engine is TLM. It has an inverted index of words, separators, and tag ranges. It returns parts of documents represented by ranges of word positions that match the query and permits statistical processing of the results. As well, it favours speed over conserving disk space.

In our illustrations, we have taken classic examples of text mining in biology and shown that TLM can match the reported performance and can do so very quickly. We have not shown TLM's results to be conclusively better or worse than earlier results, only that they are similar and fast.

CONTRIBUTIONS & ACKNOWLEDGEMENTS

All the code for TLM, except for a public domain sdbm implementation (by J. Chapweske), was written at NRC (engine by the author; GUI by Chengbi Dai). Chengbi Dai's client GUI is shown in Figure 1 above. All the examples in this paper were created by the author. The idea to use RIPPER for query based text categorization arose in discussion with Berry de Bruijn.

I would also like to thank all LitMiner team members for the motivation to create TLM, both on the computer side (Berry de Bruijn, Lynn Wei, Darrell Ferguson, Norm Vinson, and Jeff Demaine), and on the biology side (Hung Fang, Annie Law, Qing Liu, Maria Moreno, Brandon Smith, and Roy Walker).

References

[1] C. Blaschke, M. Andrade, C. Ouzounis, and A. Valencia. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Intelligent Systems for Molecular Biology*, pages 60–67, 1999.

[2] James P. Callan, W. Bruce Croft, and John Broglio. TREC and Tipster experiments with Inquery. *Information Processing and Management*, 31(3):327–343, 1995.

[3] Charles L. A. Clarke and Gordon V. Cormack. Shortest substring retrieval and ranking. *ACM Transactions on Information Systems*, 18(1):44–78, 2000.

[4] William W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.

[5] O. de Kretser and A. Moffat. Effective document presentation with a locality-based similarity heuristic. In *Proceedings of the Twenty Second International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 113–120. ACM Press, 1999.

[6] I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G.D. Bader, K. Michalickova, T. Pawson, and C.W. Hogue. Prebind and Textomy–mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4(11), 2003.

[7] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98, Tenth European Conference on Machine Learning*, 1998.

[8] M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *Journal of the American Society For Information Science and Technology*, 52(4):344–364, 2001.

[9] J. Martin and B. de Bruijn. Litminer. www.litminer.ca, 2003.

[10] U.S. National Library of Medicine. Entrez gene. www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene.

[11] B.P. Suomela and M.A. Andrade. Ranking the whole medline database according to a large training set using text indexing. *BMC Bioinformatics*, 6(75), 2005.

[12] L. Tanabe, U. Scherf, L. Smith, J. Lee, L. Hunter, and J. Weinstein. Medminer: an internet text-mining tool for biomedical information, with application to gene expression profiling. *BioTechniques*, 37:1210–1217, 1999.

[13] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan-Kaufmann Publishers, 2nd edition, 1999.

[14] Yiming Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, 1999.

Evaluating Publication Similarity Measures

Sulieman Bani-Ahmad, Ali Cakmak, and Gultekin Ozsoyoglu
EECS Dept,
Case Western Reserve University
(sulieman, ali.cakmak, tekin) @case.edu

Abdullah Al-Hamdani
Dept. of Computer Science
Sultan Qaboos University, Oman
abd@squ.edu.om

Abstract

Publication searching based on keywords provided by users is traditional in digital libraries. While useful in many circumstances, the success of locating related publications via keyword-based searching paradigm is influenced by how users choose their keywords. Example-based searching, where user provides an example publication to locate similar publications, is also becoming commonplace in digital libraries.

Existing publication similarity measures, needed for example-based searching, fall into two classes, namely, text-based similarity measures from Information Retrieval, and citation-based similarity measures based on bibliographic coupling and/or co-citation.

In this paper, we list a number of publication similarity measures, and extend and evaluate them in terms of their accuracy, separability, and independence. For evaluation, we use the ACM SIGMOD Anthology, a digital library of about 15,000 publications.

1 Introduction

Searching publications based on keywords is common in digital libraries. While useful in many circumstances, the success of locating related publications based on keywords depends on the choice of keywords [6]. Example-based searching, i.e., locating similar/related publications to a given publication is also becoming a common search query type in digital libraries [13]. In this work, we deal with the quality of publication similarity measures used for locating related- or similar-publications of a given publication. Existing publication similarity measures fall into two classes: (i) text-based similarity measures from the field of Information Retrieval (IR), such as the cosine similarity and the TF-IDF (term frequency-inverse domain frequency) model [14], or (ii) citation-based similarity measures based on bibliographic coupling (i.e., common citations between two publications) [8], co-citation (i.e., common citers of two publications) [15] or author-coupling (i.e., common authors between two publications). In this paper, we summarize the existing publication similarity measures, and extend and evaluate them in terms of their *accuracy*, *separability*, and *independence*. For evaluation, we use the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

ACM SIGMOD Anthology [1], referred to as *AnthP* here, a digital library of about 15,000 publications in data management.

Text-based similarity measures are based on information retrieval methodologies [14, 5]. As an example, using the vector space model of IR and the TF-IDF weighting scheme [14], the similarity between two publications may be measured by using Cosine, Jaccard, Dice or other document measures [10].

CiteSeer [2] is a literature search system for searching (presently) about 730,000 computer science and bioinformatics publications, and uses three document similarity measures, namely, word vectors, LikeIt string distance, and the Common Citation Inverse Document Frequency [7]. Google Scholar, Google scholarly literature search engine [3], does not provide publication similarity functions which are needed to answer example-based queries where the user provides an example publication and asks for similar publications.

By evaluating "multiple levels" of paper similarities based on bibliographic-coupling, co-citation and author-coupling, we make the following observations:
(a) Similarity value distribution curves are similar within the same group of similarity measures, i.e., bibliographic-coupling-based, co-citation-based, and author-coupling-based measures,
(b) Citation-based and author-coupling-based similarity measures are more separable than bibliographic-coupling-based measures,
(c) Citation-based and author-coupling-based similarity measures are all highly correlated. This phenomena is due to the citation and coauthorship behavior in the literature [11].
(d) Text-based similarity measures show low overlapping with citation-based and with author-coupling-based measures. Therefore, providing two sets of similarity scores, one text-based and another based on citation and/or author-coupling may prove to be a useful practice.

This paper is organized as follows. In section 2, we list and extend a number of publication similarity measures. In section 3, we evaluate the proposed similarity measures. Section 4 concludes.

2 Similarity Measures between Two Publications

2.1 Text-Based Similarities

The vector space model of text documents is used to evaluate title, abstract, index terms, and body similarities between two papers [14]. Consider a vocabulary T of atomic terms t that appear in each document. A document is represented as a vector of real numbers $v \in R^{|T|}$, where each element corresponds to a term. Let v_t denote an element of v that corresponds to the term t, $t \in T$. The value of v_t is related to the importance of t in the document represented by v. Using the Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme [14], v_t is defined as

$$v_t = log(TF_{v,t} + 1) * log(IDF_t)$$

where $TF_{v,t}$ is the number of times that term t occurs in the document represented by v, $IDF_t = N/n_t$, N is the total number of documents in the database, and n_t is the total number of documents in the database that contain the term t.

The cosine similarity between two documents with vectors v and w is computed as

$$cosine(v, w) = (\sum_{i=1}^{|T|} f(v_i).f(w_i))/(\sum_{i=1}^{|T|} f(v_i)^2. \sum_{i=1}^{|T|} f(w_i)^2)$$

where f() is a damping function, which is either the square-root or the logarithm function. Other similarity functions include Dice and Jaccard measures [10] where both change the normalization factor in the denominator to account for different characteristics of the data. As a preprocessing step, one needs to first remove the stopwords from the terms of a document, and then use the Porter's algorithm [12] to stem the terms.

2.2 Citation-Based Similarities

The citation-based similarity between two publications can be computed using (a) bibliographic coupling: common citations between the two publications [8], and (b) co-citation: common citers to the two publications [15]. One can then define citation-based similarity between two publications as a weighted sum of the two. In this section, we discuss various ways of computing bibliographic coupling and co-citation.

2.2.1 Bibliographic Coupling with Reachability Analysis

The bibliographic coupling-based similarity between papers P_Q and P_X, $Sim_{bib}(P_Q, P_X)$, can be defined as

$$Sim_{bib1}(P_Q, P_X) = \text{(common citations count between } P_Q \text{ and } P_X\text{)/MaxB.}$$

where MaxB is the maximum number of common citations between any two publications in $AnthP$. One problem with this definition is that it assumes that each common citation contributes to the reference similarity equally, and ignores the effects of publications that are cornerstone works leading to significant research in the field. A cornerstone publication is cited by all the publications that discuss an issue related to the field, and its citation by two publications carries a lesser significance. Hence it is quite possible for two publications about two unrelated topics to cite the same cornerstone publication.

To reduce the effect of common citations to cornerstone works, we define a new bibliographic coupling measure where each common citation contributes at a different level depending on the extent to which it is "influential". Assume that we assign importance scores to publications using the well-known PageRank algorithm [4]. PageRank scores are computed recursively using the formula $P_{i+1} = (1-d)M^T P_i + E$ where P_{i+1} and P_i are the current and next iteration PageRank vectors respectively, citation matrix C is the adjacency matrix of a graph with papers representing nodes, and citation relationships between papers representing edges, M is a matrix derived from C by normalizing all row-sums in C to 1, and, d is the "future citation probability" defined as follows. Given (a) an author A writing a new paper and citing paper u which in turn cites paper v, and (b) w, a randomly selected paper in $AnthP$, the parameter d, which we choose to be low, represents the probability that A will cite w, and $(1-d)$ is the probability that A will cite v. C is of size $N \times N$, where N is the total number of papers in the system. To guarantee that the PageRank algorithm converges, a hidden link, represented by the user-defined parameter E, is assumed to exist between each pair of graph nodes. A choice for E is simply $E_1 = d$. Another choice, used in [4], is $E_2 = d/N[1_N].P_i$ where 1_N is a vector of N ones. A highly important publication is cited by a large set of publications, and therefore, cannot provide an informative measure. On the other hand, if two publications cite a publication with a relatively low importance score, this citation information provides more clues toward the similarity of the two publications. Therefore, we assign weights to common citations, which are inversely proportional to their (importance) scores as follows.

$$Sim_{bib2-L1}(P_Q, P_X) = \sum_{P_i \in S_{QX}} (1 - P_{Score}(P_i))/MaxW$$

where S_{QX} is the set of common citations between P_Q and P_X, $P_{Score}(P_i)$ is the PageRank-based score of paper P_i. MaxW is the maximum $\sum_{P_i \in S_{QX}} (1 - P_{Score}(P_i))$ for any two publications in *AnthP*.

Another extension to bibliographic coupling similarity is to incorporate the notion of citations iteratively, which we refer to as *reachability analysis*. The formula of $Sim_{bib2-L1}$ can be considered as the $firstlevel$ (level-1) evaluation of a given citation information. We can also make use of $second-level$ and $third-level$ citation information. Due to efficiency considerations, next we consider only the most basic reachability analysis cases. Normally if a publication is cited by only one of the publications (i.e., either P_Q or P_X, but not both) then this publication is not considered in $Sim_{bib2-L1}$. Nevertheless, by following the citation information one more level, we may obtain additional information. For instance, assume that publication P_i is cited by P_Q, but not cited by P_X. It is possible that, at one level below, P_i may be cited by one of the publications, say P_j, which is in turn cited by P_X, as illustrated in Figure 1(a).

Note that second-level common citations can be used to strengthen common citation information of publications P_Q and P_X. Assume that P_i is cited by both P_Q and P_X. This common citation may lead to more similarity clues such that P_i might cite a publication P_k which is cited by P_Q, P_X or both, as illustrated in Figure 1(b). Finally, third level common citations can be considered as common citations for publications P_Q and P_X which is illustrated in Figure 1(c).

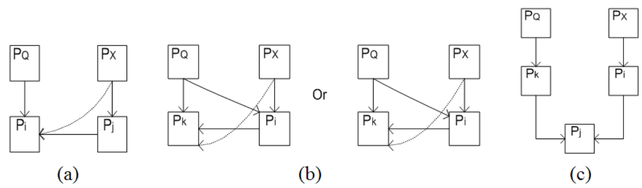


Figure 1: Illustration of citation networks (a) one level (b) two levels (c) three levels

We do not consider higher levels of co-citation information since, at each new level, publications get more diverse in terms of their contents, and their citations become less significant.

2.2.2 Co-citation Similarity with Reachability Analysis

As in multi-level bibliographic coupling, we can apply the same one, two, or three-level co-citation similarity in a similar manner. Different co-citation cases are illustrated in Figure 2, and the corresponding co-citation definitions are given next. One-level co-citation similarity between papers P_Q and P_X is defined as

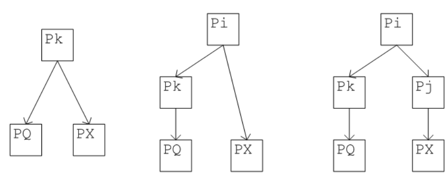


Figure 2: Illustration of three levels of co-citation similarity.

$$Sim_{co-cit1} = |C_Q \cup C_X|/MaxN$$

where C_Q, C_X are the set of publications each of which cites P_Q and P_X, respectively and $MaxN$ is the maximum number of common citers between any pair of publications in the $AnthP$. Once again, assume that we use a paper scoring algorithm, such as PageRank, to assign importance scores to publications. Then, if a publication citing P_Q or P_X is a hub (e.g., a survey paper) [9], then it will refer to many publications. To reduce the effects of hubs, we use

$$Sim_{co-cit2-L1} = \sum_{P_i \in S_{QX}} (1 - P_{Score}(P_i))/MaxC$$

where S_{QX} is the set of publications that co-cite P_Q and P_X, $P_{Score}(Pi)$ is the importance score of co-citer P_i, MaxC is the maximum $\sum_{P_i \in S_{QX}} (1 - P_{Score}(P_i))$ value of any pair of publications in $AnthP$.

If publications P_Q and P_X are cited together by more than one publication, then we can weigh the contribution of each citing publication by its ”hub score” of HITS [9]. Here we use the hub score of the citing publication because this relationship represents an outgoing link from the citing publication to P_Q and P_X. For outgoing links, in Kleinberg’s model [9], the hub score of the entity determines the strength of the outgoing link. Therefore if the citing publication is a good hub with a relatively high hub score then it contributes more than other citing publications rather than each citing publication contributing equally. Thus, we have yet another co-citation-based function:

$$Sim_{co-cit-Hub} = (\sum_{P_i \in S_{QX}} (1 - P_{HubScore}(P_i)))/MaxCh$$

where $P_{HubScore}(P_i)$ is the hub score of publication P_i, and $MaxCh$ is the maximum $\sum_{P_i \in S_{QX}} (1 - P_{HubScore}(P_i))$ value between any pair of publications in *AnthP*.

2.3 Author-Coupling-Based Similarities

We compute the *author similarity* between two publications directly via the number of common authors between the two publications (referred to as the Level-0-author overlap Sim_{AOC-L0}) or indirectly via co-authorship in other publications, e.g., two different authors, each of different publications P_Q and P_X, are co-authors in a third publication P_W (referred to here as the Level-1-author-overlap Sim_{AOC-L1}). We then use the following formula to compute the author similarity between publications P_Q and P_X:

$$Sim_{Author}(P_Q, P_X) = W_{L0} * Sim_{AOC-L0}(P_Q, P_X) + (1 - W_{L0}) * Sim_{AOC-L1}(P_Q, P_X)$$

where $0 \leq W_{L0} \leq 1$ and

$$Sim_{AOC-L0}(P_Q, P_X) = |A_Q \cup A_X|/MaxA0$$
$$Sim_{AOC-L1}(P_Q, P_X) = (1/MaxA1) \sum_{(i \in A_Q) \wedge (j \in A_X)} |(S_i - \{P_Q\}) \cup (S_j - \{P_X\})|$$

where A_Q and A_X are the sets of authors of P_Q and P_X, respectively. S_i and S_j each is the set of papers written by authors i and j, respectively, where $i \in A_Q$ and $j \in A_X$. $MaxA0$ and $MaxA1$ are the maximum numbers of level 0 (L_0) and level 1 (L_1) co-author overlaps, respectively, of any two publications in *AnthP*.

Next we assume that we have importance scores computed for authors. As an example, we may compute an author importance score as the average of importance scores assigned to the author's perhaps top-k publications. Then, as another variant, we can also consider using a different mechanism so that each shared author contributes to the similarity of publications in different proportions, depending on his/her author importance scores. This is based on the assumption that the works of important authors share a common thread. As an example, we produce a higher similarity score for two publications which share one author with a high importance score in comparison with two publications which share one author with a low ranking. On the other hand, in practice, with some exceptions, well-known authors are usually the ones who publish many high quality publications. Moreover, due to their prolificacy, it is not uncommon for these authors to publish on relatively different topics. Therefore we use a weighing mechanism which leads to author weights that are inversely proportional to their importance scores. In this way, the information that two publications share a less important author implies more towards the similarity of the publications in comparison to the case that these publications share an author with a higher importance score. Thus, we define the Level-0 and level-1 author-overlap involving author weighting Sim_{AOW-L0} and Sim_{AOW-L1} as follows

$$Sim_{AOW-L0}(P_Q, P_X) = \sum_{a_i \in A_{QX}} (1 - A_{Score}(a_i))/MaxA0$$
$$Sim_{AOW-L1}(P_Q, P_X) =$$
$$(1/MaxA1) \sum_{(i \in A_Q) \wedge (j \in A_X)} (1 - A_{Score}(a_i))(1 - A_{Score}(a_j))|(S_i - \{P_Q\}) \cup (S_j - \{P_X\})|$$

where A_Q and A_X are the sets of authors of publications P_Q and P_X, respectively, A_{QX} is the set of common authors between P_Q and P_X. $MaxA0$ and $MaxA1$ are the maximum numbers of level 0 (L_0) and level 1 (L_1) co-author overlaps, respectively, of any two publications in *AnthP*. In our experiments, we compute the score $A_{Score}(a)$ of author a as the average score of most important K papers of a, where K is 5.

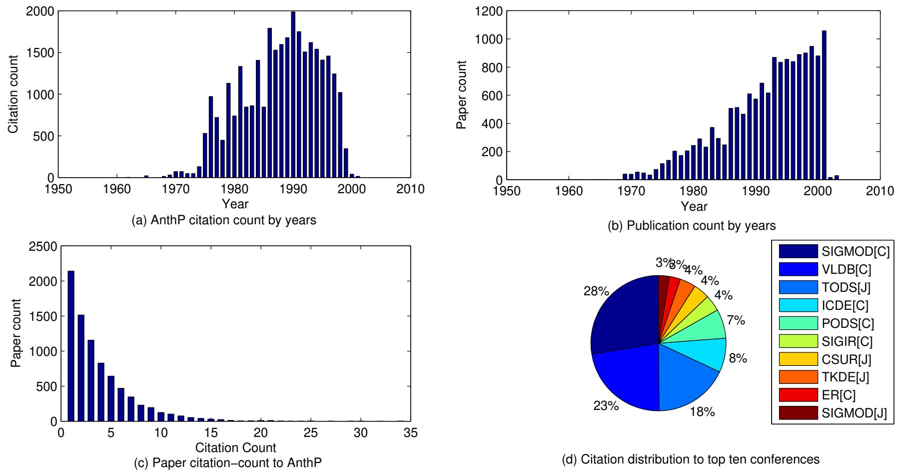


Figure 3: *AnthP* statistics

3 Empirical Evaluation of Publication Similarity Measures

3.1 Experimental Setup

For each publication in $AnthP$, we extracted titles, authors, publication venues, publication year information, and citations. The final experimental dataset included (a) 106 conferences, journals, and books, (b) 14,891 publications, and (c) 13,208 authors. *AnthP citation* refers to a citation from any publication in the *AnthP* set to a publication in the same set. *DBLP citation* refers to a citation from a publication in *AnthP* to a publication P outside of *AnthP*, but within DBLP. *External citation* of publication P is a citation from publication P to a publication outside of *AnthP* and *DBLP*.

Next we present $AnthP$ statistics. The average number of citations in an *AnthP* publication is 20. The average number of *AnthP* and DBLP citations in an *AnthP* publication is 4.289. The average *AnthP* citation count per *AnthP* publication is 2.066. Thus, the average citation reduction due to DBLP citation removal is 48.2%. Figure 3(a) displays the citation count distribution of *AnthP* publications over years, Notice that the most recent publications are not cited yet, which means that their scores will be very low even though we do not know how important they are for sure. Same comments apply to the publications published before 1974; we do not have information as to which publications cite them. The publications published before 1974 and after 2000 are very few as shown in Figure 3(b). Figure 3(c) displays the distribution of *AnthP* citation counts for the publications in *AnthP*. Figure 3(d) shows top ten venues in term of citation counts. We think that all ten venues are known to be among the best in the computer science community.

In section 3.2, we compare publication similarity measures in terms of $separability$, $independence$ and $accuracy$. $Separability$ refers to having similarity scores that distribute to a large range reasonably well. To compare similarity measures in terms of $separability$, we use similarity score distribution plots. $Independence$ refers to similarity measures that are not (highly) correlated. We evaluate $independence$ using pairwise Top-K overlapping ratios. We define the Top-K Overlapping ratio between two measures m_1 and m_2 as:

$$TKO(m_1, m_2) = Average_{(\forall p \in AnthP)}[SS_1(p) \cap SS_2(p)]/min(|SS_1(p)|, |SS_1(p)|)$$

where $SS_1(p)$ and $SS_2(p)$ are the sets of K most-similar publications to publication p based on m1 and m2, respectively. For our experiments, we used K=50. We do not consider publications with zero similarity in the set of similar publications. $Accuracy$ refers to how accurate a similarity measure is. For $accuracy$, we compute the overlapping between text-based and citation-based similarity measures, i.e., we consider text-based measure (in this case, TF-IDF and Cosine similarity) as a benchmark to which we compare citation-based similarity measures.

3.2 Experimental Results

Observation: (Figure 4): Paper similarity measure distribution within the same group of similarity measures are similar, where the groups are defined as bibliographic-coupling-based, co-citation-based, and author-coupling-based.
Observation: (Figure 4): Citation-based and author-coupling based similarity measures are more separable than bibliographic-coupling-based measures
Observation: Paper overlapping ratio within bibliographical coupling-based similarity measures outputs to the

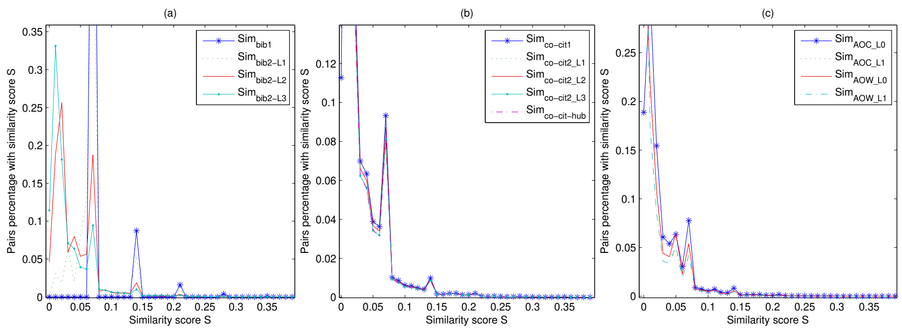


Figure 4: (a) Bibliographic-coupling-based, (b) citation-based and (c) author-coupling-based paper similarity score distributions.

same query ranges from 0.82 to .92.
The reason for the above observation is that, although a particular paper P usually deals with a limited and usually a single topic, its references cover a much wider range of research topics. This diversity increases by moving to the references of references. Thus,
Observation: In general, moving from a lower level to a higher level in bibliographical coupling-based measures creates more diversity, and in turn, smaller overlapping ratio.
Observation: Top-50 overlapping ratio between those similarity measure outputs based on bibliographical coupling and those based on co-citation ranges from 0.81 to 1.0.

The reason for the above observation is perhaps that authors usually tend to cite their own previous papers. On the other hand, most of one author's papers in general cover a small number of research interests which makes most of his/her work cite similar works. This leads to high top-50 overlapping paper ratios between the similarity measures based on bibliographical coupling and those based on co-citation.
Observation: Top-50 overlapping paper ratio between those similarity measures based on author-coupling overlapping and those based on co-citation ranges from 0.86 to 0.95.
Observation: Top-50 overlapping papers ratios between those similarity measures based on author-coupling and those based on bibliographical coupling ranges from 0.77 to 0.96.

The reason for the above observation is that, if two papers are similar based on an author-coupling measure then these papers in general are similar based on bibliographical coupling because the common authors usually have the same or at least somewhat related research interests. This makes the papers they publish commonly cite almost the same set of publications.
Observation: Text-based similarity measures show low overlapping with citation-based and author-coupling-based measures.

The above observation resulted from the way we retrieve top similar papers based on TF-IDF and Cosine similarity measure. That is, the papers that we find to be similar to a particular paper p are sorted according to their importance scores. Then we report top scored similar papers. This prevents papers that are similar, but low scored, to p also from appearing in the reported set. This in turn reduces the overlapping between text-based similarity measures in one side, and citation-based and author-coupling-based measures in the other side.

4 Conclusions

In this paper, we have presented and evaluated three groups of paper similarity measures in terms of their (i) *accuracy* (ii) *separability* and (iii) *independence*. For evaluation, we have used the ACM SIGMOD Anthology, a digital library of about 15,000 publications.

5 Acknowledgment

This research is supported by the US National Science Foundation grant ITR-0312200. S. Bani-Ahmad is supported by a fellowship from BAU-Jordan.

References

[1] ACM SIGMOD Anthology, http://www.acm.org/sigmod/dblp/db/anthology.html.

[2] CiteSeer Scientefic Digital Litrature Library, http://citeseer.ist.psu.edu/.

[3] Google Scholar (Beta), http://scholar.google.com/scholar/.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 1998.

[5] W. Cohen. The WHIRL approach to integration: An overview. In *Proceedings of the AAAI Workshop on AI and Information Integration*, Madison, Wisconsin, 1998.

[6] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1467–1479, 1999.

[7] L. Giles, K. D. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *Proc. of Intl. Conf. Digital Libraries*, 1998.

[8] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[9] J. Kleinberg. Authoritative sources in hyperlinked environments. In *the 9th ACM-SIAM Symposium on Discrete Mathematics*, 1998.

[10] G. Kowalski. *Information retrieval systems: theory and implementation*. Kluwer Academic Publishers, 1997.

[11] M. Newman. Coauthorship networks and patterns of scientefic collaboration. *PNAS*, 2004.

[12] M. Porter. An algorithm for suffix stripping. *Program*, 14(3), 1980.

[13] C. S. Lawrence, L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. In *Intl. Conf. Digital Libraries*, 1998.

[14] G. Salton. *Automatic Text Processing*. Addison Wesley, 1989.

[15] H. Small. Co-citation in the scientific literature: a new measure of the relationship between two documents. *Journal of the American Society for Information Sciences 24*, pages 265–269, 1973.

Hard Queries can be Addressed with Query Splitting Plus Stepping Stones and Pathways

Xiaoyan Yu
Computer Science Dept.
Virginia Tech
Blacksburg, VA 24061
USA

Fernando Das-Neves
Snoop Consulting
Paraguay 346 Piso 5
Buenos Aires
Argentina

Edward A. Fox
Computer Science Dept.
Virginia Tech
Blacksburg, VA 24061
USA

Abstract

A key finding of the Reliable Information Access Workshop of 2003 was that in collections like those used for TREC 6-8, there are a number of hard queries for which no current search engine can return a high quality set of results. Our Stepping Stones and Pathways (SSP) approach may yield an effective solution to such hard problems, as well as support exploration of collections of content not well known to a person (with broad interest and/or complex information needs). Our initial and promising testing of SSP had users prepare two separate short queries in order to launch processing. However, since beginning with a single information need is a more typical initial situation, we have extended the SSP research by exploring query splitting, especially as might apply to handling hard queries. This paper summarizes our recent results and identifies some of the future work needed.

1 Introduction

Searching, such as of text, is a key service of digital libraries. The quality of search results, however, is highly variable. This situation has been a key concern of the information retrieval community, and also is of interest to the database community.

Though on average results are fairly good, there is room for improvement, and in particular cases, results may be unacceptable. Accordingly, the Robust Retrieval Track of the Text REtrieval Conference (TREC), starting in 2003, has focused on individual query effectiveness rather than average effectiveness [16]. Typically, variability in retrieval effectiveness is caused by: 1) an incorrectly formulated query, 2) a collection that lacks pertinent content, or 3) an information retrieval method/system that is inadequate.

The Reliable Information Access (RIA) Workshop 2003 was initiated to investigate in-depth the reasons for retrieval variability. It approached this by studying the behavior of 7 leading search engines developed by the research community. One of the interesting results is that all of the systems failed on a subset of queries, most of which are considered to be 'hard' due to their multiple-aspect (i.e., touching on several different topics or aspects, each of which should be satisfied) property [8]. This finding suggests that the problem may be due to a combination of causes 1 and 3 listed above.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Several studies have tried to predict the variability of effectiveness or to improve the handling of queries for which performance is poor. [11] predicted a query's performance by its clarity score, basically the relative entropy between the query in the relevant document set, versus in the whole document collection set. A SIGIR workshop was held to target the problem, but in summarizing the findings of the meeting, [9] reported that research on query variability prediction is in its infancy. Information retrieval researchers lack a clear understanding of why performance is low for some queries, and cannot reliably predict which queries are 'hard'. Similarly, in the database community, Graupmann targeted this problem; his approach was to add annotations using XML to attach semantic meaning to important terms [12].

Different people have different interpretations of what a document is about. Similarly, people vary in their ability to express their information needs in the form of a query [1]. Based on the finding that the cause of poor performance for some queries might be their multi-aspect property [8], we propose our query-splitting-enabled Stepping Stones and Pathways approach to detect different aspects of a query and then to improve retrieval by identifying connections among these aspects.

We also identify two situations that may exist for some poorly-performing queries:

1. A low-relevance set is retrieved where the results are dominated by a subset of the query aspects.

2. A low-relevance result set is retrieved where it is possible to get high relevance for connected subsets of the query aspects (maybe with modification).

In order to address these two possibilities, we think it necessary to create an alternative interpretation of a user's intention. In this alternative interpretation, a query is identified as a description of two or more separable aspects. By separable aspects we mean that a significantly different set of documents representing each aspect can be retrieved for the query. The query is thus split into multiple sub-queries; without loss of generality we limit the splitting to two [17]. The next step is to retrieve a set of documents that support a valid relationship between the two sub-queries using the Stepping Stones and Pathways (SSP) approach [4]. The result returned by the SSP system is a set of topic sequences (**pathways**). Each step in each pathway is supported by documents connecting a sub-query with an intermediate topic (**stepping stone**), or connecting different intermediate topics, that provide a rationale for the connection between the two original sub-queries, i.e., the **end stones**. The two sub-queries become the endpoints which may be reached through different pathways. Each pathway connects the endpoints by a succession of stepping stones, and thus is an answer to the user information need. The SSP user interface (Figure 1) highlights the end stones, stepping stones, and pathways, and supports as well as exploratory type of search.

Query splitting have been used in the past [7]. Most commonly it converts a natural language query into two parts, so that the new query will be interpreted as of form *X or Y*. However, earlier work on query splitting did not address the problem of coherence from the user's point of view: *Are X and Y related (possibly by other intermediary concepts)?* Ours is the first study we know of that has researched both query splitting and new methods to build and display the connections of the results from query splitting.

The Stepping Stones and Pathways approach was inspired by earlier research on Literature-Based Discovery. The explosion of scientific knowledge in the last half of the 20th century resulted in many researchers being highly specialized. Different researchers may work on related problems without even being aware of each other. Discovering relations that are not explicitly stated but yet are latent in a body of knowledge is the objective of Literature-Based Discovery. Swanson [2] was the first to introduce the idea of discovering such new relations within a bibliographic database. Further work by Swanson with Arrowsmith [3] detected indirect relationships between topics in the Medline database, by finding common keywords between two document sets through an intermediate document set. Our work with SSP has extended the line of research launched by Swanson; we are extending it further through integration with new work on query splitting.

This paper concentrates on the effectiveness of query splitting as a technique for improving retrieval results. The rest of the paper is organized as follows. Our prior work with SSP is summarized in Section 2. Section

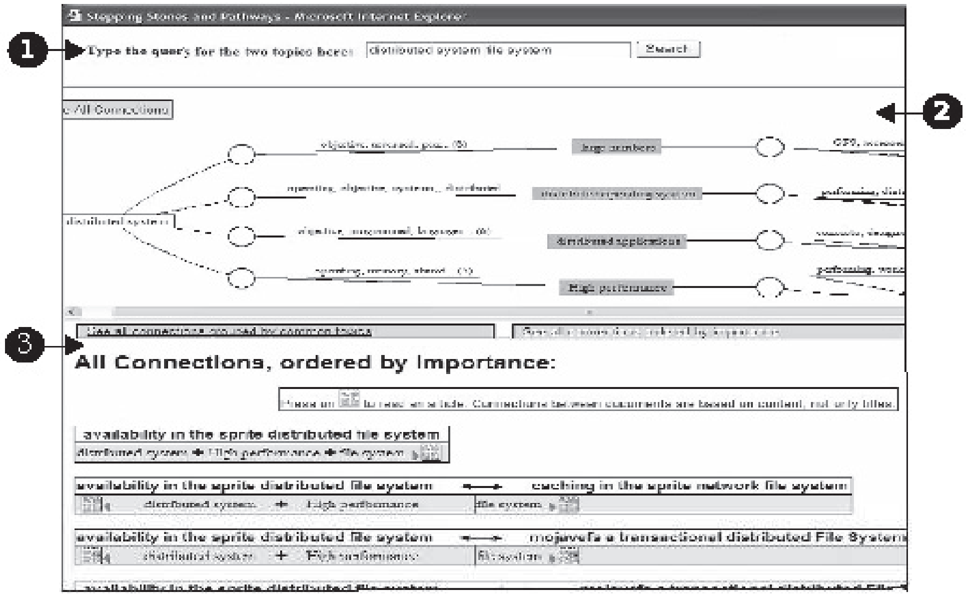


Figure 1: The areas of the Stepping Stones and Pathways user interface.

3 describes query splitting algorithms. Section 4 discusses testing of query splitting methods and the findings from our experiments. Finally, we conclude this paper and list future plans in Section 5.

2 Stepping Stones and Pathways: How it works

In order to describe the context in which we use query splitting to answer hard queries, we briefly discuss how Stepping Stones and Pathways works. In particular, we provide an overview of the user interface and the methods used to create the stepping stones and pathways. Details can be found in [4].

The Stepping Stones and Pathways user interface is divided into three areas (see Figure 1): 1) **The Query Area:** Here the user types a (two-aspect) query describing topics of interest. 2) **The Network Area:** Every time a new query is issued in the query area, this area shows the initial graph connecting the topics in which the user is interested, through a number of intermediate topics. 3) **The Document and Connections Area:** This area displays a list of documents and provides indications of how they support the corresponding connections.

From a user's point of view, a SSP retrieval session starts with the user typing a query. The query is split to create two sub-queries. SSP displays a network, in which the leftmost and rightmost nodes are labeled according to the sub-queries, and intermediate nodes are labeled according to topics connecting the sub-queries. Below the network, SSP also displays a list of documents and explains how each of them supports a connection in the network. The user can click on any node to see all the documents covering that topic, or on any edge to see any connections between the topics at each end of the edge. If the connection between any two topics is too vague, then the user can request SSP to add more intermediate topics (stepping stones) between those topics.

From an implementation point of view, a SSP session works as shown in Table 1.

1.	(Off-line) Index the document collection, using the document text and also references, if available. Document text is indexed using a tfidf word weighting, after filtering stop words.
2.	Split and process the query, trying to match all words in the sub-queries first. If that fails, relax the sub-queries by making words optional. To calculate the similarity between two documents, use the formula $sim(d_1, d_2) = 1 - (1 - P_w(d_1, d_2)(1 - P_{cocit}(d_1, d_2))(1 - P_{ref}(d_1, d_2))$
3.	Create the endpoint nodes (end stones) of the graph by: 3.1. Retrieving two document sets, one from each of the user sub-queries; 3.2. Creating a document cluster for each document set; 3.3. Calculate a cluster centroid from the top 10 documents in the cluster; 3.4. Label the cluster using Suffix-Tree Clustering [6].
4.	Create intermediate Stepping Stones and Pathways by: 4.1. Using the endpoint centroids as queries to find two document sets; 4.2. Creating an intermediate document set with the documents that appear in both retrieved sets; 4.3. Finding relevant connections between the documents in the endpoint clusters and the documents in the intermediate set; 4.4. Eliminating all documents in the intermediate set that are not part of a connection; 4.5. Clustering and labeling documents left in the intermediate set; the clusters become stepping stones.
5.	Visualize and display to users the stepping stones and pathways.

Table 1: Overview of SSP working steps

3 Query Splitting

In this section we explain how we decide if a query has multiple aspects. In general, we employ clustering algorithms, heuristic rules, and thresholds set to control the splitting. If there is not enough difference between two clusters, for a given splitting threshold, we call the query a single-aspect one; no splitting is appropriate. On the other hand, when the difference is sufficient, we have identified a multiple-aspect query, and split it into two sub-queries.

3.1 Algorithms for Query Splitting

In order to determine how best to split queries, we have devised and implemented three algorithms.

Relevance-Feedback-Based Clustering (RFC). Borodin [7] also believed a user would search for multiple concepts using a single query. He made good use of relevance feedback information from users, to retrieve more documents for different concepts. We adopt Borodin's method here. q_0 is a user's original query. New queries are generated iteratively as follows.

1. Retrieve the n highest-ranking documents, not previously retrieved, of the current query q_i, for relevance judgment. n is 5 initially.
2. Generate different groups from the documents judged as relevant documents. We put each such pair of documents, d_s and d_t, into different groups if $(correlation\, of\, d_s\, and\, d_t) <= \tau \times [(correlation\, of\, d_s\, and\, q_i) + (correlation\, of\, d_t\, and\, q_i)]/2$. Here the cosine correlation is calculated, with τ as the splitting threshold. So there can be zero (if no relevant documents are retrieved), one, or multiple groups.

3. For each group j, get top m words as a new sub-query q_{i+1}^j of the current query q_i by $q_{i+1}^j = q_i + \sum(r)/|R_i^j| - \sum(nr)/|nonR_i|$, where R_i^j are relevant documents of the group j (if none, then omit this expression). $nonR_i$ are at most two non-relevant documents with highest ranking. $r \in R_i^j$ and $nr \in nonR_i$.

4. Repeat the above steps for each newly generated sub-query, if any, but let n be 3 in the first step.

Term-Based Clustering (TC). Based on the observation that the representative terms in a user query might reflect different aspects, we characterize a query using its expanded term list, considering the top retrieved documents. The algorithm basically consists of the following steps.

1. Get the top m words from a query, plus the top $|R|$ retrieved documents for the query, using Rocchio's query expansion algorithm. That is, $q' = \alpha q + \beta \sum(r)/|R|$ where $r \in R$.

2. Represent each word as a list of documents.

3. Calculate the word distances using cosine correlation.

4. Cluster words using the agglomerative hierarchical clustering algorithm, requiring complete linkage.

5. Cut the cluster tree into groups based on the splitting threshold τ, so that each group represents a sub-query.

Document-Based Clustering (DC) Similar to Term-Based Clustering, we also propose an algorithm to cluster a user query based on the diversity of its top retrieved documents. It is like the TC algorithm, but with terms swapped for documents, and vice versa, except that: we use the top k retrieved documents for a query, and in the last step we cut the tree into groups based on the splitting threshold τ, and get the top m terms in the centroid of each group as a sub-query q'. More specifically, $q' = \alpha q + \beta \sum(r)/|R|$, where R represents the documents in a group and $r \in R$.

3.2 Term Scoring Functions and Parameter Values

Term scoring plays an important role in every query splitting algorithm. The different term scoring functions used in the algorithms are: 1) **Term Scoring Functions in First Pass Retrieval:** For all the approaches, we used the Okapi BM25 [13] term scoring function in the first pass retrieval. 2) **Term Scoring Functions in Clustering:** We also used Okapi BM25 [13] for the dividing groups step (i.e., the second step of RFC), to weight query terms and document terms. We tried Okapi BM25 [13] and pivoted tfidf (Ptfidf) [14] for weighting terms in all the clustering parts (i.e., the second step in both TC and DC). 3) **Term Scoring Functions in Query Expansion:** For all the approaches, we employed a Kullback-Leibler Distance (KLD) based method [10] for selecting and weighting expansion terms, with normalization based on dividing by the maximum term weight.

It also is essential to assign values to the parameters properly. Multiple experiments by Carpineto [10] on TREC 7-8 showed that an increasing number of pseudo-relevant documents decreased the retrieval performance nearly monotonically. Further, an increasing number of selected terms in query expansion just slightly increased the retrieval performance. Therefore, we selected $m = 20$ and $|R| = 12$ in the TC method, since that combination yielded good results in [10]. For consistency with the TC algorithm parameter settings, we set $k = 24$, since the ideal case in the DC method is to generate two equal-size clusters. In this case, the number of pseudo-relevant documents in each cluster will be 12, which is the value of $|R|$. For the same reason, we set $m = 10$ in the DC method and the RFC method. Further, $\alpha = 1.0$ and $\beta = 1.5$ are commonly used with the Rocchio algorithm, and so are used in the TC method. For the DC method, we set $\alpha = 0.2$, since we found $\alpha = 1.0$ makes two sub-queries very alike, and to a large extent hides their differences.

$Precision_n$	The minimal value of the rankings, falling within the top n, of each relevant document retrieved by each sub-query.
$Overlap_n$	The maximal value of the rankings, falling within the top n, of each relevant document retrieved by each sub-query.
$Difference_n$	$(Precision_n - Overlap_n)/Precision_n$; This measure is limited to only two sub-queries generated. A more complicated measure should be used to compute the difference among more than two sub-queries.
P_{avg}	The maximum of the P_{avg} values, as used in TREC, for the sub-queries.
$Precision_n$ average, $Overlap_n$ average, $Difference_n$ average or P_{avg} average	The total of the values of $Precision_n$, $Overlap_n$, $Difference_n$, or P_{avg}, divided by the number of detected multiple-aspect queries.

Table 2: A new evaluation strategy.

4 Experiments

In order to evaluate the effectiveness of our approach, three kinds of experiments are required: comparing the three query splitting algorithms in Section 3, evaluating SSP itself, and evaluating the combination of query splitting and SSP. The third experiment is underway. Also, we will not describe here the evaluation of SSP as an effective tool to discover connections among documents and topics, since that is detailed in [5]. Nevertheless, we must recall one of the interesting findings, i.e., that SSP can help users explore many implicit connections between a query pair. In this paper we focus on the first experiment, so as to generate a good split, based on a user's information need, and to provide sub-queries as input for SSP.

4.1 Evaluation Strategy

Since there are no well-known techniques to evaluate the prediction of query difficulty [9], it is hard to apply widely used evaluation strategies when judging the quality of our query splitting algorithms. Also, though it adds complexity to the evaluation problem, we must continue our focus regarding query splitting, wherein we detect poorly-performing queries based on their having the multi-aspect property. Thus, we adopt the requirement of SSP, whose starting point is query splitting. Accordingly, we claim that the quality of a query splitting algorithm depends on three factors: retrieval performance, difference between sub-queries, and overlap of sub-queries.

The most important factor deciding the quality of the results of a sub-query is still the number of relevant documents retrieved. Since the reason to propose a query-splitting algorithm is to improve information retrieval, it is reasonable not to expect a degradation of retrieval performance when using an algorithm. Regarding the second factor, we note that only with enough overlap of sub-queries is it feasible to find the intermediate concepts, i.e., the stepping stones. Regarding the third factor, we observe that a very small difference among sub-queries makes building a bridge among them unnecessary, since the sub-query topics can be directly connected and discussed in a single document to be retrieved by the original query.

We evaluate these three factors based on the retrieval results of the sub-queries. Thus, we make use of the relevance judgment information available in TREC, and evaluate the three query splitting approaches using a $relevance \times rank$ matrix. Each row in the matrix contains all the relevant documents for a specific query; each column corresponds to one query splitting approach; and each cell value is the corresponding relevant document's rank when documents are retrieved by a sub-query generated when the corresponding approach is employed. We define the measures in Table 2 based on the matrices.

	Total number	Total number detected by RFC at the threshold of 12.5	Total number detected by DC and TC at the threshold of 1 using Okapi and Ptfidf	Total number detected by TC at the threshold of 200 using Ptfidf
Queries selected manually	20	14(70%)	20(100%)	16(80%)
Hard queries	17	14(82%)	17(100%)	11(65%)
Union	34	25(74%)	34(100%)	24(71%)
Overlap	3	3(100%)	3(100%)	3(100%)

Table 3: Query split results using different algorithms

4.2 Experimental Setup

Collections and Queries. Our test collection is the one used in the Robust Track of TREC 2004 [15]. Two sets of queries were chosen from the TREC queries. The first set consists of 17 queries selected due to the multiple-aspect property pointed out in the analysis in [8, 15]. The other set consists of 20 queries we selected as likely to have multiple aspects, based on reading the title, description, and narrative. There are three queries that are common to both sets; thus we have 34 unique queries.

Upper Bound Experiment. Since we do not know in advance what should be the proper splitting threshold in each algorithm, we ran an upper-bound experiment first using trial-and-error to find the threshold value under which the corresponding algorithm performs the best. Then we compared the algorithms using their optimal settings. We evaluated each algorithm's performance based on the evaluation strategy in Table 2. More specifically, the metrics are $num_{detected}$ (the number of detected multiple-aspect queries), p_{avg}, sum_n ($precision_n$ + $overlap_n$ + $difference_n$), where $n = 20$, 30, 50, and 100. We consider each metric to be of the same importance, so we normalize each one by its total value. The higher the value of each metric, the better the algorithm performs.

4.3 Findings

Splitting Thresholds. By the upper bound experiment, we identify the optimal settings for each algorithm: 1) RFC: the splitting theshold $\tau = 12.5$; 2) TC when using Okapi as the term scoring function in clustering: $\tau = 1$; 3) TC when using Ptfidf as the term scoring function in clustering: $\tau = 200$; 4) DC when using Okapi as the term scoring function in clustering: $\tau = 1$; 5) TC when using Ptfidf as the term scoring function in clustering: $\tau = 1$. More details are in [17].

Query Splitting Results. We summarized the number of queries split by each algorithm in their optimal settings in Table 3. At least 70% of the queries that we selected by manually judging the multi-aspect property are split by all the algorithms. At least 65% of the hard queries identified with multi-aspect property by [8] are split as well. All the algorithms split all the common three queries. Further, we note that all the algorithms split at least 24 of the 34 queries (71%).

Comparison of Best Cases of All the Algorithms. In general, the results follow the pattern $P_{RFC} > P_{DC} > P_{TC}$, where P stands for performance, as can be seen in Figure 2.

We also measured the performance of retrieval of the original query without a splitting process, and the refined query with a query expansion (QE) process, using the p_{avg} average, since other metrics are not applicable.

original	QE	TC (w=Okapi,τ=1)	TC (w=Ptfidf,τ=200)	DC (w=Okapi,τ=1)	DC (w=Ptfidf,τ=1)	RFC (τ=12.5)
0.157597	0.171934	0.155587	0.156439	0.179454	0.175758	0.26489

Table 4: Comparison of all the best cases with the retrieval performance by original queries without query splitting and the performance by refined queries with only query expansion (QE) in terms of p_{avg}.

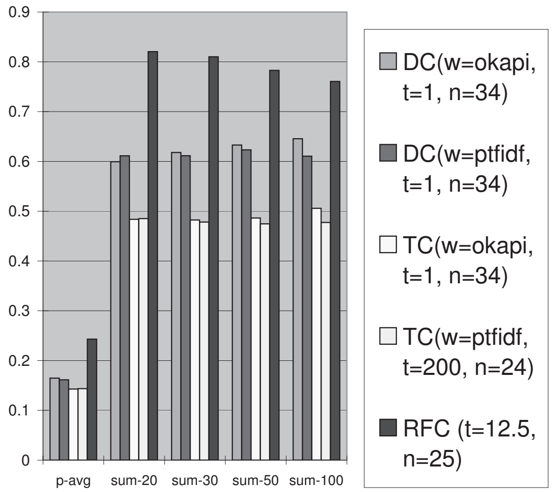


Figure 2: Comparison of all the best cases on the metrics of p_{avg}, sum_{20}, sum_{30}, sum_{50}, and sum_{100}.

The refined queries are produced using the method in [10], which is also the method to select top terms in all the query splitting algorithms. The results are shown in Table 4. Considering only the p_{avg} average, the performance of the TC algorithm is very close to the original query retrieval performance, while the DC algorithm using Okapi and Ptfidf are 13.9% and 11.5% better than the original query performance. The RFC algorithm is even better, 68.1% over the original query retrieval performance. The refined queries perform 9.1% better than the original ones, on average, and also are close to the DC algorithm.

When using different term weighting mechanisms, the performance values for the same algorithm are very close to each other, except concerning the metric sum_{100}. For the DC algorithm, the result when using Okapi is 5.7% better than when using Ptfidf; for the TC algorithm, the one using Okapi is 6% than when using Ptfidf. However, there are more interesting results when we consider the $precision_n$, the $overlap_n$, and the $difference_n$, respectively. Table 5 shows the results and reveals a pattern: the TC algorithm gets a rather low value on the overlap metric, which leads to a relative low result of sum_n.

	P_{20}	O_{20}	D_{20}	P_{30}	O_{30}	D_{30}	P_{50}	O_{50}	D_{50}	P_{100}	O_{100}	D_{100}
DC(w:Okapi,τ:1)	0.197	0.249	0.153	0.203	0.255	0.160	0.214	0.266	0.153	0.216	0.278	0.152
DC(w:Ptfidf,τ:1)	0.193	0.247	0.172	0.202	0.250	0.159	0.201	0.268	0.154	0.201	0.273	0.137
TC(w:Okapi,τ:1)	0.171	0.068	0.245	0.174	0.051	0.257	0.178	0.041	0.268	0.184	0.037	0.285
TC(w:Ptfidf,τ:200)	0.164	0.069	0.253	0.160	0.057	0.261	0.160	0.045	0.270	0.163	0.034	0.280
RFC(τ:12.5)	0.275	0.368	0.177	0.261	0.386	0.163	0.247	0.380	0.155	0.236	0.379	0.146

Table 5: Comparison of all the best cases on the $P_i(precison_i)$, $O_i(overlap_i)$, and $D_i(\ differece_i)$ metrics, where $i = 20,\ 30,\ 50,\ 100$.

4.4 Discussion

The term-based clustering and document-based clustering algorithms perform better, if not the best, when all the queries are split. Even considering the relevance-feedback-based clustering, there is a trend that the performance is better when more queries are split. More importantly, as can be seen in Table 2, the performance (measured by p_{avg}) of each algorithm in its optimal settings is not worse, and sometimes is better, than when there is no splitting. Also, all of these algorithms, except for TC, perform even better than does the refined query resulting from query expansion. Hence the majority of the query samples we selected are, indeed, multi-aspect queries, since the splitting process did not hurt performance. We expect that integrating query splitting with SSP will yield an even better result since SSP can find more relevant documents by means of discovering the connections within a multi-aspect query.

Relevance-feedback-based clustering is a special variation of the document-based clustering, since its basic idea is to cluster documents and use a cluster centroid as a sub-query. However, it clusters already-known relevant documents (from relevance judgments) instead of top retrieved documents. Hence much less noise should be included when representing the aspects of an original query. However, in practice, this kind of relevance information only can be obtained implicitly or explicitly from users. How to collect such information accurately, but not intrusively, is still an open question.

In our experiments document-based clustering performed in general better than term-based clustering. As we pointed out in Section 4.3, the utility of term-based clustering for splitting was poor due to its rather low overlap metric value. Term-based clustering divides the term candidate representatives for a query so that there are no overlap terms in the sub-queries of the query, hence decreasing the probability of the overlap of top retrieved results. On the other hand, the document-based approach clusters the document candidate representatives for the query and generates sub-queries containing common terms, from a document cluster. Since the query splitting results are to be fed into SSP, which finds connections between the query parts, we expect that different enough sub-queries will yield a bigger search space for intermediate concepts as bridges connecting the sub-queries. Our future experiments on the combination of SSP and document-based clustering and term-based clustering, respectively, should yield further insight.

The term scoring function used had no significant effect on the algorithm performance, though Okapi results generally were slightly better than Ptfidf. Consequently, in future work, we will use Okapi as the term scoring function for clustering (when testing performance on the combination of SSP and query splitting).

5 Conclusions and Future Work

We have studied an approach to handle poorly-performing queries where a possible reason for the poor results is their multi-aspect property. We have shown the feasibility of splitting this type of queries without decreasing the retrieval performance.

We plan further experiments to test how much retrieval improvement will result from using SSP, taking the split results as input. The experiments will consist of automatic runs and user studies. We will run SSP on the TREC collection, already used in the query-splitting experiment, and will evaluate the results using p_{avg} and other reasonable measures. It is also important to get feedback from real users with respect to their subjective impression of the query splitting results. Accordingly, we plan a user study on the split results, and also one on SSP, with those results as input.

6 Acknowlegments

Our work was funded in part by NSF grant IIS-0307867.

References

[1] Cleverden, C.W. (1991). The Significance of the Cranfield Tests on Index Languages. In Proceedings of ACM SIGIR91, pp. 3-12, ACM Press, 1991.

[2] Swanson, R. (1986). Fish oil, Raynauds syndrome, and undiscovered public knowledge. Perspectives in Biology and Medicine, 30(1): 7-18.

[3] Swanson, R., Smalheiser, N, Bookstein, A. (2001). Information Discovery from Complementary Literatures: Categorizing Viruses as Potential Weapons. Journal of the American Society for Information Science and Technology, 52(10): 797-812.

[4] Das Neves, F. (2004). Stepping Stones and Pathways: Improving Retrieval by Chains of Relationships between Documents. Ph.D. dissertation, http://scholar.lib.vt.edu/theses/available/etd-11012004-003013/restricted/dissertation.PDF.

[5] Das Neves, F., Fox, E.A., and Yu, X. (2005). Connecting topics in document collections with Stepping Stones and Pathways. In Proceedings of the 2005 ACM CIKM, Bremen, Germany, 31 October - 5 November.

[6] Zamir, O., Etzioni, O. (1998). Web document clustering: a Feasibility Demonstration. Proceedings of SIGIR98, pp. 45-54. ACM Press.

[7] Borodin, A., Kerr. L., Lews, F. (1968). Query Splitting in Relevance Feedback Systems. Scientific Report No. ISR-14, Dept. of Computer Science, Cornell University, Ithaca, NY.

[8] Buckley, C. (2004). Why current IR Engines Fail. In Proceedings of SIGIR2004. ACM Press.

[9] Carmel, D., Yom-Tov, E., and Soboroff, I. (2005). Predicting Query Difficulty: Methods and Applications. In SIGIR 2005 workshop.

[10] Carpineto, C., et al. (2001). An information-theoretic approach to automatic query expansion. ACM Transactions on Information Systems (TOIS), 19(1): 1-27.

[11] Cronen-Townsend, S., Zhou, Y., and Croft, W.B. (2002). Predicting Query Performance. In SIGIR 2002. Tampere, Finland: ACM.

[12] Graupmann, J., Schenkel, R., and Weikum, G. (2005). The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents. In VLDB.

[13] Robertson, S.E., Walker, S., and Beaulieu, M. (1999). Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive. In Proceedings of the 7th Conference on Text Retrieval. Gaithersburg, MD.

[14] Singhal, A., Buckley, C., and Mitra, M. (1996). Pivoted document length normalization. In Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval. pp. 21-29.

[15] Voorhees, E.M. (2005). The TREC Robust Retrieval Track. ACM SIGIR, June, p. 39.

[16] Voorhees, E.M. (2003). Overview of the TREC 2003 robust retrieval track. In Proceedings of TREC 2003. Gaithersburg, MD.

[17] Yu, X., Das Neves, F., and Fox, E.A. (2005). Query Splitting. Virginia Tech Department of Computer Science Technical Report, November.

Ten-Year Cross-Disciplinary Comparison of the Growth of Open Access and How it Increases Research Citation Impact

Chawki Hajjem < $hajjem.chawki@uqam.ca$ >
Stevan Harnad < $harnad@uqam.ca$ >
Yves Gingras < $gingras.yves@uqam.ca$ >
Institute of Cognitive Sciences
Université du Québec à Montréal
Montréal, Québec, Canada H3C 3P8
http://www.er.uqam.ca/nobel/cogsci2/isc/

Abstract

In 2001, Lawrence found that articles in computer science that were openly accessible (OA) on the Web were cited substantially more than those that were not. We have since replicated this effect in physics. To further test its cross-disciplinary generality, we used 1,307,038 articles published across 12 years (1992-2003) in 10 disciplines (Biology, Psychology, Sociology, Health, Political Science, Economics, Education, Law, Business, Management). We designed a robot that trawls the Web for full-texts using reference metadata (author, title, journal, etc.) and citation data from the Institute for Scientific Information (ISI) database. A preliminary signal-detection analysis of the robot's accuracy yielded a signal detectability d'=2.45 and bias $\beta = 0.52$*. The overall percentage of OA (relative to total OA + NOA) articles varies from 5%-16% (depending on discipline, year and country) and is slowly climbing annually (correlation r=.76, sample size N=12, probability* $p < 0.005$*). Comparing OA and NOA articles in the same journal/year, OA articles have consistently more citations, the advantage varying from 25%-250% by discipline and year. Comparing articles within six citation ranges (0, 1, 2-3, 4-7, 8-15, 16+ citations), the annual percentage of OA articles is growing significantly faster than NOA within every citation range (*$r > .90$*, N=12,* $p < .0005$*) and the effect is greater with the more highly cited articles (r = .98, N=6,* $p < .005$*). Causality cannot be determined from these data, but our prior finding of a similar pattern in physics, where percent OA is much higher (and even approaches 100% in some subfields), makes it unlikely that the OA citation advantage is merely or mostly a self-selection bias (for making only one's better articles OA). Further research will analyze the effect's timing, causal components and relation to other variables, such as, download counts, journal citation averages, article quality, co-citation measures, hub/authority ranks, growth rate, longevity, and other new impact measures generated by the growing OA database.*

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

1 Introduction

With the advent of the Internet and the Web, more and more researchers are making their research openly accessible (OA) by self-archiving it online [8, 18] to increase its visibility, usage and citation impact [5, 6, 16]. In 2001, Lawrence reported that OA articles in computer science are cited more. We have since replicated this OA citation advantage based on a single large central OA archive in physics [10, 11] and have begun testing it more widely [7]. We here report the generality of this effect across biological and social sciences, using a robot that trawls the Web for full-texts based on reference and citation data from the Institute for Scientific Information (ISI) database.

2 Method

Using the reference metadata for 1,307,038 articles published in peer-reviewed journals covered by the CD-ROM version of ISI's Science and Social Science Citation Indices (SCI and SSCI), our robot trawled the Web to estimate how many of the articles did (OA) or did not (NOA) have a full-text version freely accessible on the web. The 10 disciplines covered were: administration, economics, education, business, psychology, health, political science, sociology, biology, and law, for 12 years: 1992-2003.

The robot's search algorithm was the following: (1) Send request to ISI database for metadata of article (first-author name and article title). (2) Send request (name, title) to: Yahoo, Metacrawler, Vivissimo, Eo, AlltheWeb and Altavista. (3) Extract external (irrelevant) links. (4) Remove duplicate URLs. (5) Sort URLs to process PDF and PS files first (probable full-texts). (5) Convert files (PDF, PS, Latex, HTML, XML, RTF, and Word) to text. (6) Parse files to test for full-text of reference article (name/title in first 20% of text, references in last 20%). (7) If, in parsing HTML file, title found but not full text, extract and follow links in file further as references possibly leading to the full text (to depth of 3 levels). (8) Sort articles by discipline/journal/issue/year; calculate percent OA articles within each; then by discipline/journal; and finally for each discipline. (9) Sort articles by discipline/journal/issue/year, calculate citation ratio as (OA-NOA/NOA) within each, then by discipline/journal and finally for each discipline. (10) Exclude data for all journals that are 100% OA (OA journals) from both the article counts and the citation counts (as we are only doing within-journal comparisons for NOA journals); exclude data from all single issues that are 100% OA (to eliminate denominators).

3 Signal detection analysis of the robot's accuracy

To test the robot's accuracy, we performed a preliminary signal detection analysis [4]. From the 633,410 articles in Biology we took a sample of 100 articles the robot had called OA and 100 it had called NOA and hand-checked them for correctness. This yielded four possibilities :*Hits* (correct positives: OA is called OA), *Correct rejections* (NOA is called NOA), *False alarms* (NOA is called OA) and *Misses* (OA is called NOA). In a sample of 100 articles tagged by the robot as OA and 100 tagged as NOA, the Robot had 6 Misses and 19 False Alarms according to a manual check of its accuracy.

Signal detectability (d') was found to be 2.45, indicating that the robot was fairly sensitive. The robot's bias $\beta = 0.52$ indicates some tendency toward false alarms (overestimating OA). If $\beta = 1$ the robot is neutral, favoring neither false alarms nor misses; $\beta > 1$ favors misses and $\beta < 1$ favors false alarms. As there are in fact about ten times as many NOA articles as OA articles, this means there is some overestimation of the percentage of OA articles and hence some underestimation of the size of any OA citation advantage we might find.

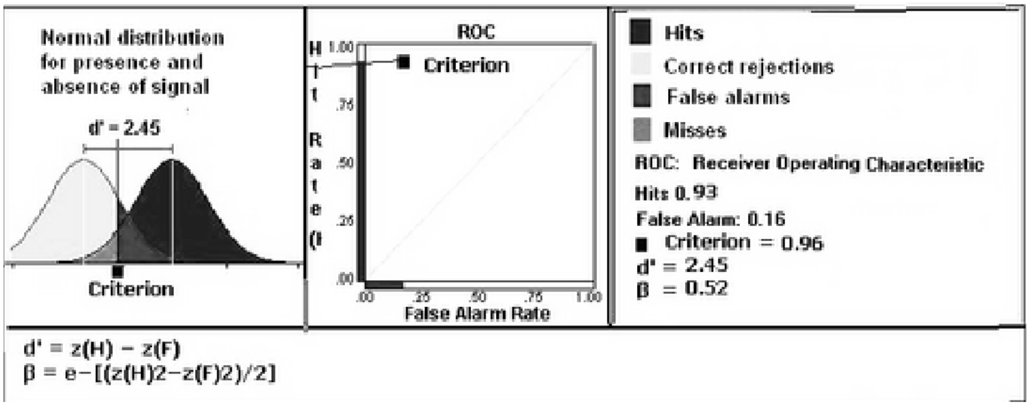


Figure 1: **Signal detection Analysis of robots Accuracy**. (Graph generated using the applet provided by Wise Project $http://wise.cgu.edu/sdt/sdt.html$)

N=12	r
OA Citation Advantage x Year	0.25 NS
OA Citation Advantage x Total articles	0.21 NS
OA Citation Advantage x %OA articles	-0.02 NS
Total articles x Year	0.65 $p < 0.01$
Total articles x %OA articles	0.31 NS
%OA articles x Year	0.76 $p < 0.005$

Table 1: **Correlation between Year and OA Growth.** Significant correlation between year and percent OA articles: %OA is growing annually. (Total articles is also growing yearly; no other correlations are significant.)

4 Results

Figure 2.a shows the 12-year average for the percentage of OA articles (dark bars) in each of our 10 reference disciplines, ordered by total number of articles (OA + NOA, with Biology on the high end and Law on the low end). Percent OA varies from 5%-16%. There is a clear and consistent OA citation advantage (OA-NOA/NOA calculated within each individual journal issue, then averaged across journals, but not counting issues that had 100% or 0% OA articles) across all the disciplines, varying from 36%-172% (white bars): OA articles have more citations. Figure 2.b shows that this OA citation advantage is present across all countries (based on 1st-author affiliation and ordered by total article output).

We now look more closely at the fine-structure of the OA citation advantage and OA growth across time. Figure 2.c shows pooled results across all the disciplines for total annual articles (OA + NOA, gray curve), percent OA (black bars, log scale) and percent OA citation advantage (white bars, log scale). Both total articles and annual percent OA are growing (slowly) from year to year (r=.65 and .76, respectively, Table 1; no other correlations are significant).

We next look at the time course of total percentage growth in OA (for all 10 disciplines) within specific citation ranges OA_c (c= 0, 1, 2-3, 4-7, 8-15, 16+). Figure 3.a should be read backwards, 2003-1992, because citations grow with time, older articles accumulating more citations across the years. So it is perhaps not surprising that the percentage of OA articles among those articles with zero citations, OA_0 decreases with time (at first rapidly, from 2003 till about 1998, and then slowly leveling off). For articles with one or more citations, the corresponding effect is the opposite, OA_c grows (backwards) with time (first rapidly from 2003 till about 1998, then likewise leveling off). But this is not a specific OA effect at all, for the inset shows the very same pattern is for NOA articles too. The specific OA effect only becomes apparent when we examine the corresponding ratio OA_c/NOA_c within each citation range (Figure 3.b).

The OA effect only becomes apparent when we look at OA_c/NOA_c. This ratio is growing year by year

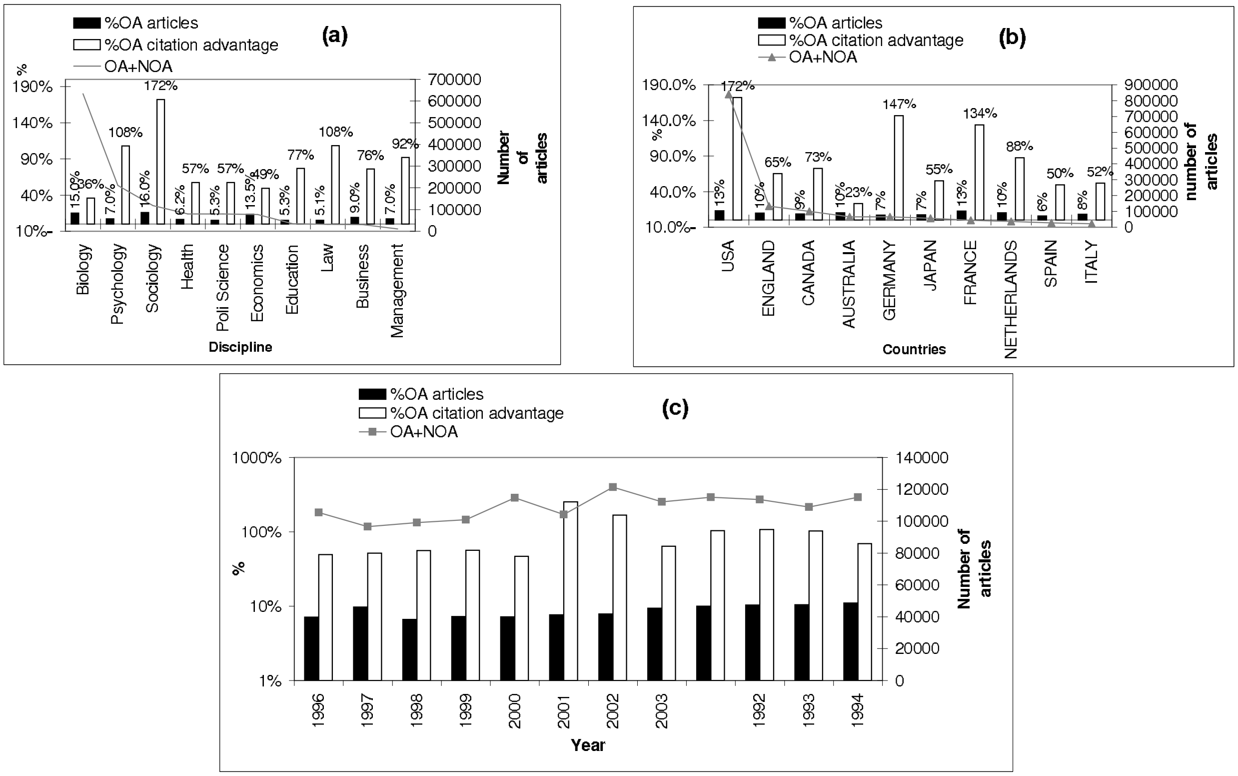


Figure 2: **(a): Open Access Citation Impact Advantage by Discipline.** Total articles $(OA{+}NOA)$, gray curve; percentage OA: $(OA/(OA + NOA))$ articles, black bars; percentage OA citation advantage: $((OA - NOA)/NOA)$ citations, white bars, averaged across 1992-2003 and ranked by total articles. All disciplines show an OA citation advantage. **(b): Open Access Citation Impact Advantage by Country.** Total articles (gray curve), percent OA articles (black bars), and percent OA citation advantage (white bars); averaged across all disciplines and years 1992-2003; ranked by total articles. **(c): Open Access Citation Impact Advantage by Year.** Total articles (gray curve), percent OA articles (black bars), and percent OA citation advantage (white bars): 1992-2003, averaged across all disciplines. No yearly trend is apparent in the size of the OA citation advantage, but %OA is growing from year to year (see Table 1). Note that percent scale is logarithmic (to make the OA growth visible).

(Figure 3.b) which means that within each citation range, the percentage of articles that are OA is growing faster than the percentage of articles that are NOA (correlations are all positive and very high, Table 3). This growth differential also increases with the citation range, being lowest for uncited articles and highest for articles with over sixteen citations. This confirms the pattern reported for computer science articles by [15].

If we look at our total sample of 1,307,038 articles across all disciplines and years, we see that 793494 (61%) of them are uncited; of the remaining 513544 (39%), 155265 (12%) have 1 citation, declining to 53838 (4%) with 16+ citations (Figure 4, gray curve). 156845 (12%) of the total articles are OA. Of those, 85794 (55%) are uncited, and their numbers in each higher citation range fall off much the way the totals do (Figure 4, dark curve). However, if we again look at the ratios between the percentages among OA and NOA articles for each range, c, expressed as (OA_c-NOA_c)/NOA_c (bars in Figure 4), we see that this ratio is positive for all nonzero citation ranges, beginning at 1 citation (16% OA advantage), peaking at about 4-7 (c. 22% OA advantage), and falling off again toward 16+ citations (10% OA advantage). This means that the proportion of articles within each citation range is greater among OA articles than among NOA articles except zero, the most populace category (61%), where it is NOA articles that have the -12% NOA *dis*advantage.

In and of themselves, these correlations and temporal patterns cannot determine causality. It is a logical possibility that the cause of the OA advantage is merely a self-selection bias: that authors tend to self-archive their better papers (or better authors tend to self-archive their papers) and better papers are simply cited more.

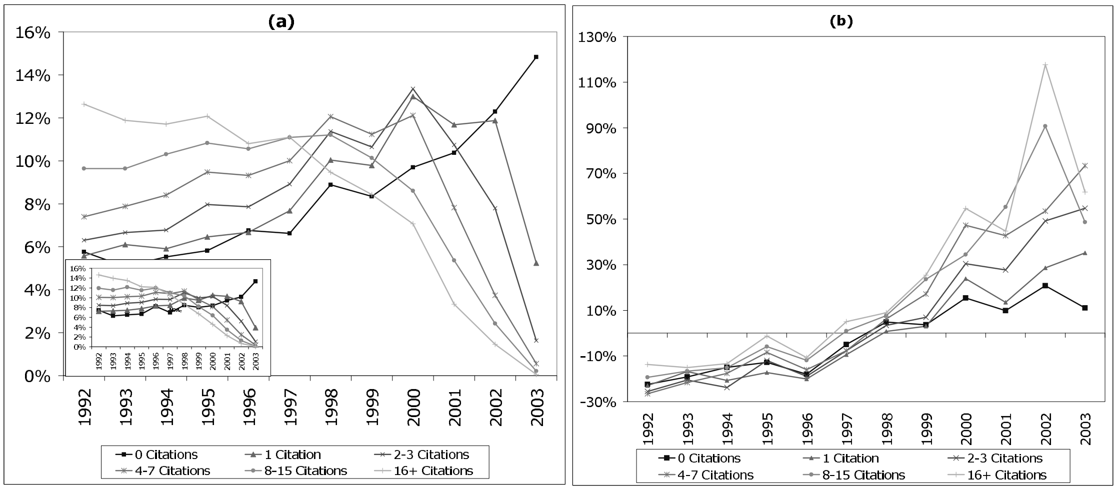
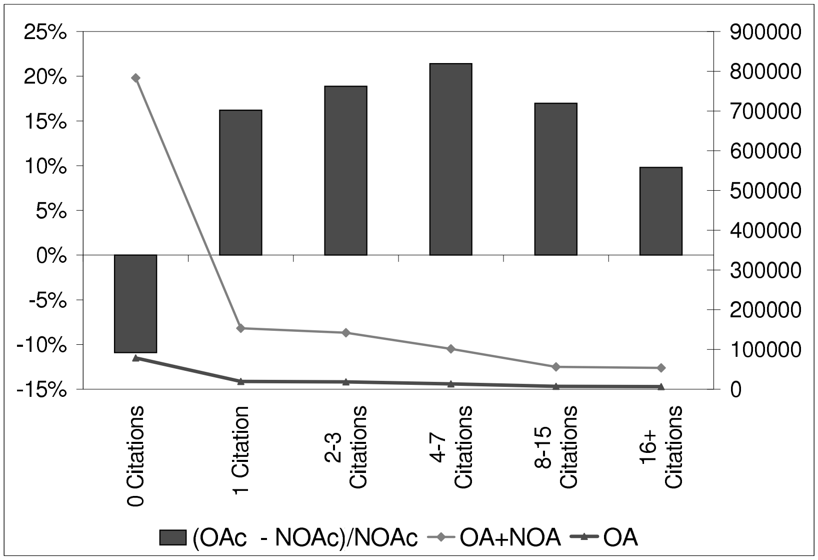


Figure 3: **(a): Yearly OA and NOA in each Citation Range.** The yearly percentage (OA_c) of the articles with c citations (c = 0, 1 2-3, 4-7, 8-15, 16+) that are OA (1992-2003). This graph (figure 3.a) should really be read backwards, as citations increase cumulatively as an article gets older (younger articles have fewer citations). Reading backwards, for articles with no citations (c=0), the percentage OA_c decreases each year from 2003-1992, at first rapidly, then more slowly. For articles with one and more citations ($c > 0$), OA_c first increases rapidly from 2003 till about 1998, then decreases slowly 1998-1992. Notice that the rank order becomes inverted around midway (c. 1998), the percentages increasing from c=0 to c=16+ for the oldest articles (1992) and the reverse for the youngest articles (2003). The pattern is almost identical for NOA articles too (see NOA_c inset), so this is the relationship between citation ranges and time for all articles, not a specific OA effect. **(b): Yearly Growth of OA Relative to NOA in Each Citation Range.** The yearly ratio OA_c/NOA_c between the percentage of articles with c citations (c = 0, 1 2-3, 4-7, 8-15, 16+) that are OA and NOA (all disciplines). This ratio is increasing with time (as well as with higher citation counts, c), showing that the effect first reported for computer science conference papers by Lawrence (2001) occurs for all disciplines.

This is unlikely to be the sole or even the primary cause of the OA advantage for three reasons, two empirical and one commonsensical: (1) The first empirical reason is that if the OA advantage were solely a self-selection bias, it would have to shrink or disappear as the percentage of OA articles approaches 100%. Our sample's average percent OA content was low (around 9%), but prior studies in disciplines where the self-archiving rate is much higher – well over 50% in some areas of physics [10, 11] and near or at 100% in astronomy and astrophysics [12] – have found OA citation advantages that were of the same size as the ones found here. (2) The second empirical reason is that OA has also been shown to increase article downloads [1, **?**], and that increased downloads are in turn correlated with increased citations [2, 17, 19]. Causality is more directly evident there. (3) The commonsensical reason to assume that OA is causal is that access is a necessary (if not a sufficient) condition for usage and citation, and no researcher's institution can afford access to anywhere near all journals [http://www.arl.org/stats/arlstat/]; OA self-archiving supplements that access, increasing potential

N=12	r
O Citations OA_c x Year	0.94 $p < 0.005$
1 Citations OA_c x Year	0.60 $p < 0.025$
$2-3$ Citations OA_c x Year	0.10 $p < 0.05$
$4-7$ Citations OA_c x Year	-0.36 $p < 0.05$
$8-15$ Citations OA_c x Year	-0.74 $p < 0.005$
$16+$ Citations OA_c x Year	-0.93 $p < 0.001$

Table 2: **Correlation between Year and Percent OA in Each Citation Range.**Significant correlations between year and the percentage of OA articles in each citation range, OA_c: Percent OA is growing annually (negative correlation) in the higher citation ranges and shrinking in the lower ones; but the correlation pattern is the same for NOA articles, hence this is not an OA effect. It just shows that citations increase with time.



Page 1

Figure 4: **OAc/NOAc Ratio in Each Citation Range (All years, All Disciplines).** Ratio of the percentage of articles with c citations (c = 0, 1 2-3, 4-7, 8-15, 16+) that are OA to the percentage that are NOA (across all disciplines and years), expressed as a difference from equality (OA_c-NOA_c)/NOA_c. This ratio increases as citation count (c) increases (r = .98, N=6, $p < .005$). The percentage of articles with 0 citations is relatively higher among NOA articles, but it becomes higher among OA articles with 1 citation and higher. This shows that the more cited an article, the more likely that it is OA. (The gray curve is the total number of articles (OA + NOA) in each citation range, and the dark curve is the number of OA articles scale for both curves is on right.)

online accessibility to 100%.

5 Conclusion

Research is conducted (and funded and published) in order to be used, applied and built upon. It is for this reason that citation impact is rewarded by researchers' institutions and funders [3, 20]. It follows that whatever increases research access and impact increases benefits to research, researchers, their institutions and their funders. Our estimate of the current percentage of OA articles in the 10 disciplines tested is between 5% and 15% (mean 9%; median 7% ; SD 4.26) and that OA is associated with citation impact that is 25% to 250% higher (mean 83 %; median 77% ; SD 39.49). To extend this benefit to the remaining 85-95% of research, "publish or perish" needs to be extended, in the online age, to "publish and self-archive" so as to maximize research access and impact

N=12	r
O Citations OA_c/NOA_c x Year	0.94 $p < 0.001$
1 Citations OA_c/NOA_c x Year	0.94 $p < 0.001$
$2-3$ Citations OA_c/NOA_c x Year	0.96 $p < 0.001$
$4-7$ Citations OA_c/NOA_c x Year	0.96 $p < 0.001$
$8-15$ Citations OA_c/NOA_c x Year	0.91 $p < 0.001$
$16+$ Citations OA_c/NOA_c x Year	0.87 $p < 0.001$

Table 3: **Correlation between Year and OAc/NOAc Growth Ratio in Each Citation Range.** Significant correlations between year (1992-2003) and the ratio OA_c/NOA_c between the percentage of articles with c citations (c = 0, 1 2-3, 4-7, 8-15, 16+) that are OA and the percentage with c citations that are NOA (all disciplines). This ratio is growing annually in every citation range.

[21]. In addition to the direct impact benefits, as the OA database approaches 100%, many rich new measures of research usage and impact will become possible, including both citation and download counts, growth curves, and latencies; co-citation counts; hub/authority ranks, semantic indices [14] and many other online performance indicators. These will be usable not only for navigation and evaluation, but also for analyzing and predicting research directions and influences.

References

[1] Bollen, J., Van de Sompel, H., Smith, J. and Luce, R. (2005) Toward alternative metrics of journal impact: A comparison of download and citation data. Information Processing and Management, 41(6): 1419-1440 http://arxiv.org/abs/cs.DL/0503007.

[2] Brody, T., Harnad, S. and Carr, L. (2005, in press) Earlier Web Usage Statistics as Predictors of Later Citation Impact. *Journal of the American Association for Information Science and Technology (JASIST)*. http://eprints.ecs.soton.ac.uk/10713/

[3] Diamond, Jr. , A. M. (1986) What is a Citation Worth? *Journal of Human Resources* 21:200-15, 1986, http://www.garfield.library.upenn.edu/essays/v11p354y1988.pdf

[4] Egan, J.P. (1975) *Signal detection theory and ROC analysis*. Academic Press. Garfield, E. (1955) Citation Indexes for Science: A New Dimension in Documentation through Association of Ideas. Science, Vol:122, No:3159, p. 108-111

[5] Garfield, E. (1955), http://www.garfield.library.upenn.edu/papers/science_v122(3159)p108y1955.html

[6] Garfield, E. (1973) Citation Frequency as a Measure of Research Activity and Performance, in *Essays of an Information Scientist*, 1: 406-408, 1962-73, Current Contents, 5, http://www.garfield.library.upenn.edu/essays/V1p406y1962-73.pdf

[7] Hajjem, C., Gingras, Y, Brody, T., Carr, L. & Harnad, S. *(submitted)* Open Access to Research Increases Citation Impact. http://www.ecs.soton.ac.uk/ harnad/Temp/hajjem-draft.doc

[8] Harnad, S. (1994) A Subversive Proposal. In: Ann Okerson & James O'Donnell (Eds.) *Scholarly Journals at the Crossroads: A Subversive Proposal for Electronic Publishing*. Washington, DC., Association of Research Libraries, June 1995.http://www.arl.org/scomm/subversive/toc.html

[9] Harnad, S., Carr, L., Brody, T. and Oppenheim, C. (2003) Mandated online RAE CVs linked to university eprint archives: Enhancing UK research impact and assessment. *Ariadne*, issue 35, April 2003 http://www.ariadne.ac.uk/issue35/harnad/

[10] Harnad, S. & Brody, T. (2004) Comparing the Impact of Open Access (OA) vs. Non-OA Articles in the Same Journals, *D-Lib Magazine* 10 (6) June. http://www.dlib.org/dlib/june04/harnad/06harnad.html

[11] Harnad, S., Brody, T., Vallieres, F., Carr, L., Hitchcock, S., Yves, G., Charles, O., Stamerjohanns, H. and Hilf, E. (2004) The Access/Impact Problem and the Green and Gold Roads to Open Access. *Serials review* 30(4). http://eprints.ecs.soton.ac.uk/10209/

[12] Kurtz, Michael J.; Eichhorn, Guenther; Accomazzi, Alberto; Grant, Carolyn S.; Demleitner, Markus; Murray, Stephen S.; Martimbeau, Nathalie; Elwell, Barbara. (2003) The NASA Astrophysics Data System: Sociology, Bibliometrics, and Impact. *Journal of the American Society for Information Science and Technology*. http://cfa-www.harvard.edu/ kurtz/jasis-abstract.html

[13] Kurtz, M. J. , Eichhorn, G. , Accomazzi, A. , Grant, C. S. , Demleitner, M. , Murray, S. S. (2004) The Effect of Use and Access on Citations, *Information Processing and Management*, 41 (6): 1395-1402 http://cfa-www.harvard.edu/ kurtz/IPM-abstract.html

[14] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. *Discourse Processes*, 25, 259-284.

[15] Lawrence, S, (2001) Online or Invisible?, *Nature* 411 (2001) (6837): 521. http://www.neci.nec.com/lawrence/papers/online-nature01/

[16] Moed, H. F. (2005a) *Citation Analysis in Research Evaluation*. NY Springer.

[17] Moed, H. F. (2005b) Statistical Relationships Between Downloads and Citations at the Level of Individual Documents Within a Single Journal, *Journal of the American Society for Information Science and Technology*, 56(10): 1088-1097. http://www3.interscience.wiley.com/cgi-bin/abstract/110506743/ABSTRACT

[18] Odlyzko, A. M. (2002) The rapid evolution of scholarly communication. *Learned Publishing*, 15(1), 7-19. http://www.dtc.umn.edu/ odlyzko/doc/rapid.evolution.pdf

[19] Perneger, T. V. (2004) Relation between online ”hit counts” and subsequent citations: prospective study of research papers in the British Medical Journal. *British Medical Journal* 329:546-547. http://bmj.bmjjournals.com/cgi/content/full/329/7465/546

[20] Smith, A. and Eysenck, M. (2002) The correlation between RAE ratings and citation counts in psychology *Technical Report*, Psychology, Royal Holloway College, University of London, June 2002. http://psyserver.pc.rhbnc.ac.uk/citations.pdf

[21] Swan, A. and Brown, S. (2005) Open access self-archiving: An author study. *JISC Technical Report*. http://eprints.ecs.soton.ac.uk/10999/

A System of User-Guided Biological Literature Search Engine

Meng Hu
EECS, Case Western Reserve University
Meng.Hu@case.edu

Jiong Yang
EECS, Case Western Reserve University
Jiong.Yang@case.edu

Abstract

Efficiently finding most relevant publications in large corpora is an important research topic in information retrieval. The number of biological literatures grows exponentially in various publication databases. The objective of the study in this paper is to fast locate useful publications from large biomedical document collections based on users' preferences.

In this paper, a new iterative search paradigm is introduced which integrates biological background knowledge in organizing the results returned by search engines, and utilizes user feedbacks to filter irrelevant documents. A term weighting scheme based on Gene Ontology is introduced to improve similarity measurement of documents in biomedical domain. A prototype text retrieval system has been built based on this iterative search approach. Experimental results show that the system can filter a large number of irrelevant documents while keep most of the relevant documents with limited user interactions.

1 Introduction

Text retrieval is an important problem in information retrieval. Searching for relevant publications from large literature corpora is a frequent job to biologists and biomedical researchers. With the abundance of biomedical publications available in digital libraries in recent years, efficient text retrieval becomes a more challenging task. For example, PubMed [1] now contains over 14 million publications. It is crucial to efficiently and accurately identify those documents most relevant to users' interests from such large document collections.

It has been recognized that one limiting factor of the traditional search engine technology is the low precision of the results returned. When users search by a few keywords, a large number of matched results could be returned. Users spend a significant amount of time to browse these results to find out those documents they are truly interested in. Keyword-based search is currently the most commonly employed search strategy in biomedical digital libraries. The publications returned by keyword searches may not be organized properly, forcing the users to browse thousands of publications. In most cases, it is impossible for users to manually read every returned entry, thus leads to loss of many truly relevant publications.

Many efforts have been done to improve the efficiency and effectiveness of literature retrieval in public domain and biomedical discipline. For example, document ranking is introduced for indexing entries in large literature collections. PageRank [2] and HITS [3] are both citation-scoring functions for evaluating the importance of documents. [4] presented a method to rank documents in MEDLINE using the differences in word content between MEDLINE entries related to a topic and the whole of MEDLINE. On the other hand, text

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

categorization has been studied to organize the search results. In [5], a machine learning model based on text categorization is built to identify high-quality articles in a specific area of internal medicine. SOPHIA [6] is an unsupervised distributional clustering technique for text retrieval in MEDLINE.

In this paper, a new iterative searching paradigm which aims to solve the above problem by incorporating biological background knowledge and user feedbacks is proposed. The iterative approach works as follows. First a set of documents returned by the keywords-based search is organized in a clustering manner, then users interact with system to provide objective evaluations on a small set of representative documents from these document clusters. Biological background knowledge descried in a controlled vocabulary is integrated to help the document clustering process. Next the system takes advantage of user feedbacks to refine the document set by filtering those user-rated irrelevant documents. Users can stop the iterative search at any time if the number of remaining documents is small enough for them to review, or the search process terminates automatically if a pre-defined number of remaining documents is reached. In this system, the number of documents that users examined is significantly reduced and the size of retrieved document set could also shrink with the help of the pruning process. This approach is particularly useful when labeling text is a labor-intensive job and when there is a large amount of results returned for a keywords-based search.

Since our text retrieval system focuses on the biological domain, we believe the background knowledge in this area could benefit the document clustering process, and add explanatory power to the organization of documents. The background knowledge we exploit in this paper is Gene Ontology [8]. Gene Ontology is a structured, controlled vocabulary that describes gene products in terms of their associated biological processes, cellular components, and molecular functions. We consider Gene Ontology as a hierarchical organization of biological concepts, and incorporate this hierarchical structure in measuring the similarity between biological publications. Users' evaluation on representative documents is utilized to prune the document set. Documents in clusters whose representatives are evaluated as relevant by users are kept for the next iteration.

Document clustering is one of the research areas most relevant to this paper. In [7] a core ontology WordNet is integrated in text clustering process as background knowledge. Concepts in the core ontology are compiled into the representations of text documents. However, their methods may not work for specific biomedical domain, and also the formal concept analysis used for conceptual clustering is known to be slow and impractical in real applications. Therefore, in this paper, a new term weighting scheme based on biomedical ontology is proposed to improve the similarity metric of biological publications.

The remainder of this paper is organized as the following. In Section 2, the terminology and metrics are formally defined and the methodology of our system is described in details. Experimental results are presented in Section 3. Last, we concluded our work in Section 4.

2 System and Methods

We have developed a prototype system to help users to retrieve useful biological literatures from a large amount of publications. The users will provide the keywords as input and interact with the system during the retrieval process. In this prototype system, Gene Ontology is utilized as the background knowledge to organize documents, and the user feedbacks are used to refine the retrieved documents. Finally, the system returns a small set of documents that are considered as most relevant to users' preference. In this section, we formally defined some terminologies. The methodology of our system is described and the three main steps in the system are explained in details.

2.1 Pre-Processing

In order to improve the response time of the system, pre-processing is done before users interacting with the system. Every time a document is imported to the database, the pre-processing described below is conducted.

During pre-processing phase, Gene Ontology, which is originally described in a DAG (directed acyclic graph), is transformed to a tree hierarchy. If a term has multiple parents, it will have multiple instances in the transformed GO tree because it has different paths to the root term, which is important for the feature weighting discussed in a later section. For example, term ”RNA transport”(GO:0050658) has two parent terms: ”nucleic acid transport”(GO:0050657) and ”establishment of RNA localization”(GO:0051236). Therefore, ”RNA transport” has two instances in the transformed GO tree: one is at level 8 as a child of ”nucleic acid transport”, and the other one is also at level 6 as a child of ”establishment of RNA localization”.

After the transformation of the Gene Ontology structure, the occurrences of GO terms are collected from the documents. The synonyms of GO terms defined in Gene Ontology are also considered equally as GO terms themselves. That is to say, if a synonym of a GO term appears in a document, the GO term is also considered occurred in the document. For instance, when searching for ”peroxisome targeting sequence binding”, ”PTC binding” is also searched. By searching all documents, the number of occurrence of each GO term in each document is collected. Other statistical information are also collected at the same time, such as the length of every document, occurrence of every other word in each document, etc. Non-informative words, such as ”the”, ”we”, are removed from the documents based on a given English stop-word list.

2.2 Feature Selection and Weighting

Traditionally documents are considered as a bag of words, and are represented by a set of feature words. Feature selection is the process to select the set of words to represent documents. It benefits the clustering and classification by reducing the feature space and eliminating noisy features. In our system, the mutual information as defined in [9] is used as the criteria for feature selection. 2000 words with the most mutual information throughout all the documents in each iteration are selected as the feature terms. For example, if in the first iteration, the system returns 5000 documents matching users' keywords out of 100,000 documents, 2000 words with the most mutual information in these 5000 documents will be selected as feature words. Besides this, a set of GO terms is also chosen as feature terms. A feature level is selected in the transformed GO tree, and all distinct GO terms at this level serve as the feature terms.

The 2000 words with most mutual information and all the GO terms at the feature level in GO tree form the feature set. In our prototype system, level 8 in Gene Ontology, which contains around 3500 GO terms, is selected as the feature level.

After obtaining the feature words to represent documents, we construct a vector of real numbers for every document by assigning each feature term a numerical weight. The weight of a term is dependent on two factors: the importance of the term throughout all the documents and the strength of the term in a particular document. Therefore, the weight of term t consists of two parts: the global weight and the local weight. The global weight(gw) of a term t is defined as $\frac{|D|}{df(t)}$, where $|D|$ is the total number of documents in database, and $df(t)$ is the number of documents that contain term t.

A definition of the local weight of a term t in a document d based on Poisson distribution ([10]) is given as below:

$$lw(t) = 1/(1 + exp(\alpha \times dlen) \times \gamma^{f(t,d)-1}) \tag{1}$$

where $\alpha = 0.0044$, $\gamma = 0.7$, $dlen$ is the length of document d, and $f(t, d)$ is the frequency of term t in d.

For those feature terms obtained by the most mutual information, their weights in a document are just the multiplication of the global weight and the local weight: $tw(t) = gw(t) \times lw(t)$. A more complex weighting scheme is used for those feature terms from Gene Ontology. The original term weight computed from the above will be distributed and aggregated based on Gene Ontology structure. The weight of a term not at the feature level is distributed or aggregated to its ancestor or descendant terms at the feature level. If the term is at a lower level than the feature level, its weight is aggregated to all ancestors of this term in the feature level. If the term is in a higher level than the feature level, its weight is uniformly distributed to its children level by level until the feature level is reached. After obtaining the term weight vector for each document, the similarity between two documents is defined as the cosine similarity of their term weight vectors.

Figure1 illustrates an example of the distribution and aggregation process. A part of the Gene Ontology hierarchy is shown in Figure1. The two numbers beside each term at the feature level are the original weights computed for a document and the final weights after distribution and aggregation, respectively. If the second level in this figure is selected as the feature level, then only ”Transport”, ”Secretion” and ”Establishment of RNA localization” will serve as the feature terms when computing the document similarity. In this case, although the term ”Establishment of RNA localization” never appears in the document, the weights of its children terms will be aggregated to the second level. Therefore, term ”Establishment of RNA localization” will gain weight of 0.25 from its children terms ”RNA Transport” and ”establishment of pole plasm mRNA localization”. However, the weights of ”Amide Transport”, ”Ion Transport” and ”Boron Transport” are not aggregated to the second level, because their ”Transport” is a substring of its children terms, and the occurrences of ”Transport” has already been counted. Meanwhile, the weight of term ”establishment of localization”, which locates in the first level, is distributed uniformly to its children terms. Therefore, the final weight of feature terms ”Transport”, ”Secretion” and ”Establishment of RNA localization” in this document will be 0.76, 0.16 and 0.33 respectively.

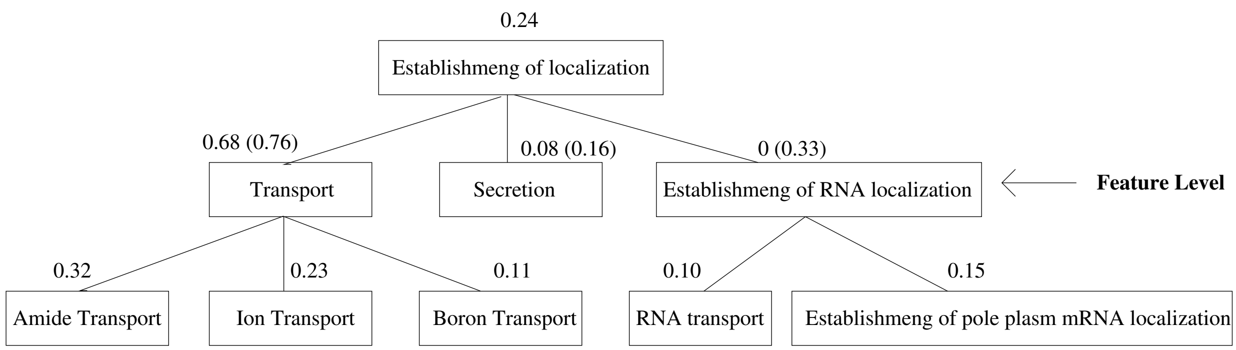


Figure 1: Distribution and Aggregation of term weights

2.3 Clustering and Representative Selection

Document clustering has been considered as an important tool for browsing and navigating large document collections. In our prototype system, after users input the keywords to search, a set of documents is returned from the document corpus by exact keyword matching. To organize these documents in a meaningful way, these documents are clustered into groups according to their mutual similarities. Traditional document clustering methods only consider the distribution of words in documents, but ignore the fact that prior knowledge could be important in organizing the documents. In stead of measuring the document similarity directly by the distribution of words, our idea is to compile the background knowledge provided by biological lexicon into the similarity measurement, which is described in the earlier section.

In our system, Bi-Section-KMeans clustering method ([7]) is used for clustering purpose, which has been shown to perform as good as other clustering algorithm, but much faster than others in document clustering. Bi-Section-KMeans is essentially a variant of KMeans clustering algorithm, which keeps partitioning the largest cluster until the desired number of clusters is reached.

After obtaining the document clusters, one representative document is selected from each cluster. In our prototype system, the centroid document of each cluster, which has the maximum average similarity to all other documents in the cluster, is chosen as the representative document. The user will review all the representative documents and rate each one as relevant or non-relevant. In each iteration, documents are clustered and representative documents are selected. The number of clusters is a parameter of the system and can be set by users. Users will read the representatives and provide their evaluations. The system will use their evaluations to refine the document set, then reduce the number of documents. Documents in those clusters whose representative documents are rated as ”relevant” by users are then kept for next iteration. By looking at a small number of documents in each iteration, users save a significant amount of time from manually reading all search results.

3 Experimental Results

A prototype search system is implemented in Perl based on the methodology proposed in this paper. 100,000 abstracts from PubMed, which are stored as plain text files in a 7200 rpm hard drive, are used to test our prototype system. These abstracts serve as the document universe in our experiments. In this section, experimental results are presented to demonstrate the effectiveness and efficiency of our proposed method.

The following experimental method is conducted for evaluating the prototype system. First we search a set of keywords, referred to as reference keywords, by exact keywords matching, then a set of documents are returned for this search query. This set of documents are considered as the benchmark and serve as the reference result set. Then some keywords are removed from the reference keywords to generate a reduced keyword set. Naturally, the reduced keyword set will result in a larger document set, which is referred to as initial document set. The system organizes these documents by document clusters, and users will review the representatives selected from these document clusters in each iteration. Finally the system will return a set of documents after several iterations. In our experiments, recall is used to evaluate the search performance of our prototype system. We denote the set of documents obtained by searching reference keywords as D_r and the set of documents our prototype system returns by taking the reduced keywords as input is denoted as D_o. The recall is defined as $\frac{|D_o \cap D_r|}{|D_r|}$.

One reference keywords set used is ”metabolism”, ”expression”, ”regulation”, ”phenotype”, ”protein”, ”mRNA” and ”yeast”. By doing an exact keyword matching on this set of keywords, 300 documents are returned from our testing document universe. Then we use the following three reduced keyword sets: ”regulation, Phenotype and yeast”, ”metabolism, expression, regulation, Phenotype, protein and mRNA” and ”regulation, mRNA and yeast” as the input keyword sets of our system. Each of the three reduced keyword sets will result in thousands of documents by exact keywords matching. In this experiment, the system was set to terminate when the number of remaining documents reaches half of the initial result document set. The number of document clusters was set to 10 in each iteration.

The results show that our prototype system can identify over 70% of the benchmark documents while removing thousands of irrelevant documents in several iterations. Since we reduced the size of the result document set to half, but achieved a recall over 50%, the precision of the results was also improved compared to the initial results returned by exact matching on the reduced keywords. Similar results were obtained by other keyword sets such as ”protein, kinase, enzyme, synthetase, DNA and ligase” and ”nucleotide binding, promoter, enzyme, expression and regulator”. To evaluate the robustness of our system against different input size, we also chose keyword sets to vary the size of initial document set, which is returned by exact keyword-matching on the reduced keyword set. The experimental results show that the recall varied insignificantly around 68%, although the response time rose with the increase of initial document size.

One parameter of our system is the number of document clusters in each iteration. We tested the performance of our system under different settings of this parameter. The results show that if the cluster number is not set too small, the system performed almost steadily, and was able to identify 70% of the reference document set. The reason for this observation is that a partitioning clustering algorithm is used in our system, and in each iteration of the clustering process, it only splits the larger cluster. When the size of the larger cluster is not too large, users tend to have the same evaluations on two clusters split from one larger cluster. Therefore the system performs robustly when the number of clusters is not set too small. However, the number of clusters can not be

Table 1: Performance on keyword set

	Iterations	Response Time	Recall
Test Set 1	4	600 s	74%
Test Set 2	5	645 s	69%
Test Set 3	4	570 s	70%

set too large in practice, because this parameter is actually the number of representatives users will review in each iteration. A reasonable setting of the number of clusters is from 5 to 15.

4 Conclusions

In this paper, a new iterative search paradigm is proposed. In our approach, document clustering is adopted to organize documents, and the user feedbacks are used to refine the retrieved documents. A new term weighting scheme is defined based on Gene Ontology, which benefits the document clustering by considering the hierarchy of biological concepts in the document similarity measurement. By this approach, users review a much smaller number of representative documents and the system filters a large number of irrelevant documents according to user feedbacks. A prototype biomedical literature search system has been built upon this iterative search paradigm. Experimental results demonstrate the effectiveness, efficiency and robustness of our prototype system.

References

[1] PubMed, available at http://www.ncbi.nlm.nih.gov/entrez/

[2] Brin S. and Page L., The Anatomy of A Large-scale Hypertextual Web Search Engine, *WWW7 Conf., 1998*

[3] Kleinberg J., Authoritative Sources in A Hyperlinked Environment, *9th ACM-SIAM Symposium on Discrete Algorithms, 1998*

[4] Suomela BP and Andrade MA., Ranking the Whole MEDLINE Database According to A Large Training Set Using Text Indexing, *BMC Bioinformatics. Mar 2005*

[5] Aphinyanaphongs Y., Tsamardinos I., Statnikov A., Hardin D., and Aliferis CF, Text Categorization Models for High Quality Article Retrieval in Internal Medicine, *J Am Med Inform Assoc. 2005;12*

[6] Vladimir D., David P., Mykola G., and Niall R., SOPHIA: An Interactive Cluster-Based Retrieval System for the OHSUMED Collection, *IEEE Transctions on Information Technology in Biomedicine, June 2005*

[7] Andreas H., Steffen S., and Gerd S., Text Clustering Based on Background Knowledge, *Technical Report*

[8] The Gene Ontology Consortium, available at http://www.geneontology.org.

[9] Slonim N. and Tishby N., Document Clustering Using Word Clusters via The Information Bottleneck Method, *ACM SIGIR 2000*

[10] Kim W., Aronson AR and Wilbur WJ., Automatic MeSH Term Assignment and Quality Assessment. *Proc AMIA Symp 2001*

Call for Papers and Proposals

http://aitrc.kaist.ac.kr/~vldb06

VLDB2006

32nd International Conference on Very Large Data Bases

Very Large Data Bases

The Convention and Exhibition Center(COEX), Seoul, Korea /September 12-15, 2006

Aims of the Conference

VLDB 2006 is a premier international forum for database researchers, vendors, practitioners, application developers, and users. We invite submissions reporting original results on all aspects of data management as well as proposals for panels, tutorials, demonstrations, and exhibits that will present the most critical issues and views on practical leading-edge database technology, applications, and techniques. We also invite proposals for events and workshops that may take place at the Conference site between September 10th and 11th before the VLDB 2006 conference.

Topics of Interest

VLDB 2006 strongly encourages the submission of creative work that goes beyond improvements of already known results. Submissions may cover novel approaches in data management, visions that present new viewpoints and challenges, or a description of the implementation or deployment of advanced database technology in an industrial or application setting. Furthermore, since new challenging applications appear on the horizon, papers that describe those with respect to their technical substance, their impact, and their importance and relate them to today's database technology are also solicited.

Paper Submission Guidelines

Research Papers

Papers must adhere to the conference's duplicate submission policy, must be formatted according to the conference's camera-ready format, and are limited to 12 pages. Paper submission must be done electronically using the conference management tools for the Core Database Technology or the Infrastructure for Information Systems track. For each paper, its authors must submit an abstract by March 9th, 2006 (5:00 p.m. Pacific Standard Time). The full paper must subsequently be submitted electronically, in pdf format, by March 16th, 2006 (5:00 p.m. PST). Authors will be notified of the results by May 30th, 2006. Further questions may be addressed to:

Core Database Track: David Lomet (lomet@microsoft.com)
Infrastructure Track: Gustavo Alonso (alonso@inf.ethz.ch)

Industrial, Applications, and Experience Papers

Full papers or extended abstracts must be submitted electronically, in pdf format, by March 16th, 2006 (5:00 p.m. Pacific Standard Time) using the conference management tool. The conference's duplicate submission policy and the formatting requirements also apply. In particular, each paper must be formatted according to the conference's camera-ready format and the page length is restricted to at most 12 pages. Authors will be notified of the results by May 30th, 2006. Further questions may be addressed to:
Guy Lohman (lohman@almaden.ibm.com)

Proposal Guidelines

Demonstration Proposals

Demonstration proposals must be submitted electronically, in pdf format, by March 16th, 2006 (5:00 p.m. Pacific Standard Time) using the conference management tool. Proposals should be focused on new database technology, advances in applying databases, or innovative use of database techniques. Proposals must be submitted in camera-ready format and are limited to 4 pages. They should describe the demonstrated system, indicate what is going to be demonstrated, and state the significance of the contribution to database technology or applications. Proposals must not be published or under consideration for publication elsewhere. Authors will be notified of the results by May 30th, 2006. Demonstration papers will appear in the proceedings. Further questions may be addressed to:
Tore Risch (tore.risch@it.uu.se)

Tutorial Proposals

Tutorial proposals must clearly identify the intended audience and its assumed background. Tutorials whose audience is broader than the database research community are encouraged. Proposals must be no more than 5 pages and must provide a sense of both the scope of the tutorial and depth within the scope. The intended length of the tutorial (1.5 or 3 hours) should also be indicated, together with justification that a high-quality presentation will be achieved within the chosen time period and the indication of the main learning outcomes. Proposals should also include contact information (name, email address, telephone number, and FAX number) and a brief bio of the presenters. If the proposed tutorial has been given previously, the proposal should include where the tutorial has been given and how it will be modified for VLDB 2006. Proposals must be submitted electronically by March 16th, 2006 (5:00 p.m. Pacific Standard Time) to:
Christos Faloutsos (christos@cs.cmu.edu)
Tutorial presentations will be published and made available to VLDB participants, and must be ready for publication by July 12th, 2006.

Panel Proposals

Panels should address timely and, preferably, controversial issues and must be debate-oriented rather than series of short presentations. A proposal should include the topic title; a short statement about the importance and relevance of the panel and the potential issues of controversy; a tentative list of questions that will be posed to the panelists; a list of confirmed participants along with their affiliations; and a short bio of each participant. Proposals must be submitted electronically by March 16th, 2006 (5:00 p.m. Pacific Standard Time) to:
Michael Carey (mcarey@bea.com)
Short panel summaries will appear in the proceedings.

Workshop Proposals

VLDB 2006 will feature a number of co-located workshops on broad topics related to data management, and will be held before the main conference. Proposals for workshops are hereby invited. A proposal should be no more than 5 pages and include the workshop title, technical description of the topic and issues, justification, potential officials, duration, and history (if any) of the workshop. Proposals should be submitted electronically by February 7th, 2006 (5:00 p.m. Pacific Standard Time) to:
Ming-Chien Shan (ming-chien.shan@hp.com).
The decision on the proposal will be notified by March 1st, 2006.

Important Dates

Feb. 7th, 2006:	Workshop Proposal Deadline.
Mar. 9th, 2006:	Abstract Submission Deadline.
Mar. 16th, 2006:	Paper, Panel, Demonstration and Tutorial Submission Deadline.
May 30th, 2006:	Notification of Acceptance.
June 23rd, 2006:	Camera Ready Papers Due.
Sep. 12-15th, 2006:	Conference in Seoul, Korea.

ICDE'06

Sponsored by the
IEEE Computer Society

CALL FOR PARTICIPATION

22nd International Conference on Data Engineering

April 3 - April 7, 2006

JW Marriott Buckhead Hotel, Atlanta, Georgia, USA

http://icde06.cc.gatech.edu Mirror: http://icde06.ewi.utwente.nl

WELCOME TO ICDE 06 IN ATLANTA

Data Engineering deals with the use of engineering techniques and methodologies in the design, development and assessment of information systems for different computing platforms and application environments. The 22nd International Conference on Data Engineering provides a premier forum for sharing and exchanging research and engineering results to problems encountered in today's information society.

ICDE 06 HIGHLIGHTS

We have an exciting program designed to appeal both to researchers and to data engineering professionals. It includes:

- Three keynote talks;
- Panels and advanced technology seminars (no additional fee);
- Presentations of 90 research papers and 52 research posters out of over 465 submissions;
- 13 industrial papers and 16 research demos;
- 12 special topic workshops in conjunction with the main conference;

KEYNOTE TALKS

We will have three keynote addresses by distinctished speakers from IBM, Microsoft and Google.

PANELS

The panels are organized by Marek Rusinkiewicz, Telecordia and will be announced shortly.

ADVANCED TECHNOLOGY SEMINARS

- Schema and data translation (Paolo Atzeni)
- Query co-Processing on Commodity Processors (Anastassia Ailamaki, Naga K. Govindaraju and Dinesh Manocha)
- Foundations of Automated Database Tuning (Surajit Chaudhuri, Gerhard Weikum)
- Mining, Indexing, and Similarity Search in Graphs and Complex Structures (Jiawei Han, Xifeng Yan, Philip S. Yu)
- Models and Methods for Privacy-Preserving Data
- Publishing and Analysis (Johannes Gehrke)

REGISTRATION FEES (until March 1st, later)

Conference

Member:	500 USD / 600 USD
Non-Member:	625 USD / 750 USD
Student:	275 USD / 330 USD

Workshops

Member:	250 USD / 300 USD
Non-Member:	310 USD / 375 USD
Student:	200 USD / 250 USD

WORKSHOPS

- Second International Workshop on Database Interoperability (InterDB'06), April 3, 2006, http://www.fundp.ac.be/InterDB2006/
- IEEE International Workshop on Multimedia Databases and Data Management (MDDM'06), April 8, 2006, http://www.cs.fiu.edu/mddm06
- IEEE Workshop on Electronic Chronicles (eChronicle'06), April 8, 2006, http://www.eChronicle.org
- International Workshop on Security and Trust in Decentralized/Distributed Data Structures (STD3S'06), April 8, 2006, http://lsirwww.epfl.ch/std3s/
- Second IEEE International Workshop on Networking Meets Databases (NetDB'06), April 3, 2006, http://www.cs.brown.edu/research/db/netdb06
- Second International Workshop on Challenges in Web Information Retrieval and Integration (WIRI'06), April 3, 2006, http://itkaken.ex.nii.ac.jp/WIRI/wiri2006/
- Semantic Web and Databases (SWDB'06), April 8, 2006, http://lsdis.cs.uga.edu/swdb06
- Workshop on Workflow and Data Flow for Scientific Applications, April 8, 2006, http://www.cc.gatech.edu/~cooperb/sciflow06
- International Workshop on Semantics enabled Networks and Services (SeNS'06), April 3, 2006, http://lsdis.cs.uga.edu/SeNS/
- Third International Workshop on XML Schema and Data Management (XSDM 2006), April 7, 2006, http://www.ntu.edu.sg/home/assourav/XSDM/main.html
- Workshop on Privacy Data Management (PDM'06) April 8, 2006, http://www.ceebi.curtin.edu.au/PDM2006
- Second International Workshop on Databases for Next-Generation Researchers (SWOD06), April 7, 2006, http://www.ics.ritsumei.ac.jp/SWOD2006
- PhD Student Symposium/Workshop

ABOUT ATLANTA

Atlanta is the home of Georgia Aquarium opened on November 23rd 2005 as the World's Largest Aquarium. With over 8 million gallons of fresh and marine water, and 100,000 animals representing 500 species from around the globe, you're sure to see things you've never seen before!.

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398