IADIS International Journal on Computer Science and Information Systems
Vol. 4, No. 3, pp. 1-19
ISSN: 1646-3692

ON CITATION-BEHAVIOR-GUIDED SEARCH-PHRASE SUGGESTERS FOR ONLINE DIGITAL LIBRARIES

Sulieman Bani-Ahmad. *Department of Information Technology, Al-Balqa Applied University. Salt, Jordan.*
Sulieman@case.edu

ABSTRACT

A content-driven *search-keyword (SK-) Suggester* for keyword-based search in digital libraries is proposed. Suggesting search terms while the user is entering search terms is helpful for constructing correctly-typed and focused search terms for digital library queries. The proposed SK-Suggester is based on pre-analyzing step of the publication collection to be searched. The pre-analysis step consists of the following. (i) We parse the document collection using a Link-Grammar parser, a *syntactic parser of English*, next, (ii) we group publications based on their research topics, (iii) after that, the parser output is used to build a hierarchical structure of simple and compound tokens to be used to suggest search terms. In order to sort the suggested terms, we use the TextRank algorithm, a text summarization tool, to assign topic-sensitive scores to the simple and compound tokens. The identified research topics are used to help user entering focused search terms prior to the actual search query execution. The topic-sensitive TextRank scores are further refined to incorporate the user's citation behavior model proposed in [Bani-Ahmad, S., Ozsoyoglu, T. 2009].
We experimentally show that the proposed framework promises a more scalable, high quality, and user-friendly SK-Suggester when compared to its competitors. We validate our proposal experimentally using a subset of the ACM SIGMOD Anthology digital library as a testbed, and by employing the research-pyramid model to identify the research topics.

KEYWORDS

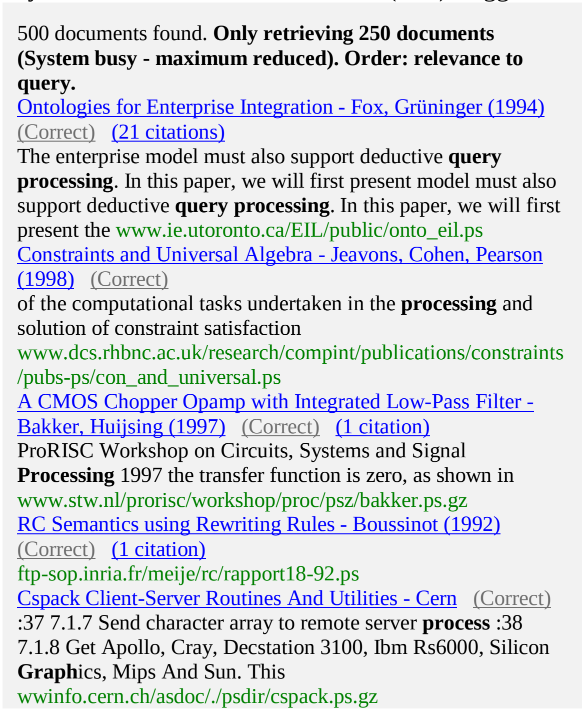
Online digital libraries, Search keyword suggesters, The research-pyramid model.

1. INTRODUCTION

Equipping keyword-based search user interfaces with efficient and user-friendly SK-Suggesters has proven to be useful [The CompleteSearch engine][H. Bast, I. Weber. 2006][Google search engine]. Studies show that users spend considerable amounts of time in search sessions to properly select keywords [Derek Sisson][D. O. Case and D. M. Higgins.

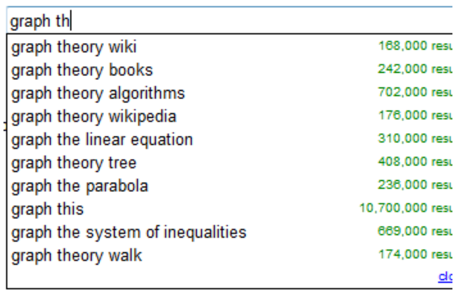
2000], and to modify their search terms in order to successfully locate documents that they are searching for. Specific benefits of suggesting search terms while typing are (i) saving the time required for typing (ii) catching typing mistakes early before executing queries, and (iii) identifying documents with exact-matching to search terms early to place them on top of search results [Google Suggest]. Consequently, with a SK-Suggester utilized, users are less likely to face unsuccessful search attempts. In the case of literature digital libraries searching for "query processing using query graphs" using CiteSeer [CiteSeer] (a digital library from the computer science domain), a list of 500 documents are identified (see figure 1). Furthermore, the top-5 relevant documents to the query are of low relevancy to search terms. Thus, guiding the user selection of search terms prior to actual query execution is an important problem.

In the case of web queries, frequently, users are not sure as to how to characterize the search using keywords [Derek Sisson], and gradually build more focused search terms [iProspect Inc. 2006]. One scenario where users find difficulty formulating their queries is when a search term has synonyms that the user does not remember. As an example, the "Big O notation", which is a mathematical notation used to describe the asymptotic behavior of functions, is also referred to as "Landau notation" or "asymptotic notation"[Wiki]. Another scenario is when the same keyword has different meanings in different contexts, i.e., polysemy [Krovetz, R. 1997]. This may force the user to add more keywords to prune out irrelevant contexts. A possible approach to solve these problems is to provide users with immediate feedback on the digital library contents as well as on how focused their search terms are, at an early stage, i.e., as they enter search terms. In this paper, we propose and evaluate such a system which we call Search term (SK-) Suggester.



500 documents found. **Only retrieving 250 documents (System busy - maximum reduced). Order: relevance to query.**
Ontologies for Enterprise Integration - Fox, Grüninger (1994)
(Correct) (21 citations)
The enterprise model must also support deductive **query processing**. In this paper, we will first present model must also support deductive **query processing**. In this paper, we will first present the www.ie.utoronto.ca/EIL/public/onto_eil.ps
Constraints and Universal Algebra - Jeavons, Cohen, Pearson (1998) (Correct)
of the computational tasks undertaken in the **processing** and solution of constraint satisfaction
www.dcs.rhbnc.ac.uk/research/compint/publications/constraints/pubs-ps/con_and_universal.ps
A CMOS Chopper Opamp with Integrated Low-Pass Filter - Bakker, Huijsing (1997) (Correct) (1 citation)
ProRISC Workshop on Circuits, Systems and Signal **Processing** 1997 the transfer function is zero, as shown in
www.stw.nl/prorisc/workshop/proc/psz/bakker.ps.gz
RC Semantics using Rewriting Rules - Boussinot (1992)
(Correct) (1 citation)
ftp-sop.inria.fr/meije/rc/rapport18-92.ps
Cspack Client-Server Routines And Utilities - Cern (Correct)
:37 7.1.7 Send character array to remote server **process** :38 7.1.8 Get Apollo, Cray, Decstation 3100, Ibm Rs6000, Silicon **Graph**ics, Mips And Sun. This
wwinfo.cern.ch/asdoc/./psdir/cspack.ps.gz

Figure 1. Searching CiteSeer for keywords "*query processing using query graphs*".



graph th

graph theory wiki	168,000 resu
graph theory books	242,000 resu
graph theory algorithms	702,000 resu
graph theory wikipedia	176,000 resu
graph the linear equation	310,000 resu
graph theory tree	408,000 resu
graph the parabola	236,000 resu
graph this	10,700,000 resu
graph the system of inequalities	669,000 resu
graph theory walk	174,000 resu

cl

Figure 2. Google suggest refinements search-keywords for the search terms "*graph th*"

In contrast with our approach, Google Scholar provides an SK-Suggester through Google Suggest (Figure 2) which employs users' *search-history repository* [Google Scholar]. Google's SK-Suggester utilizes the search history of all users as keyword suggestions, and recommends search terms from (i) popular searches, (ii) searches from the current user's search history, and (iii) current user's bookmarks. Studies show that this approach has multiple limitation aspects that make it inadequate for the literature digital library domain [S. Bani-Ahmad, G. Ozsoyoglu. 2006].

"Content-Driven" SK-Suggesters, as opposed to Google's Search-History-Driven SK-Suggester, recently received more attention [The CompleteSearch engine][Bast, H., et al. 2007.][H. Bast, I. Weber. 2006]. In general, a SK-Suggester foresees users' search terms by (i) parsing the document collection to be searched, (ii) preparing offline refinements to search terms, and (iii) dynamically suggesting keywords as the user types his/her keywords.

In this paper, we present the framework of a SK-Suggester that boosts the performance of the auto-completion tool proposed in [H. Bast, I. Weber. 2006] and implemented in the CompleteSearch engine [The CompleteSearch engine]. We experimentally verify that the proposed enhancements result in a more scalable, higher quality, and a more user friendly SK-Suggester than the CompleteSearch engine autocompletion tool, and also overcomes the shortcomings of Google Suggest.

Our proposed SK-suggester is based on an a priori analysis of the publication collection of the digital library at hand. We (i) parse the document collection using the Link Grammar parser, a syntactic parser of English, (ii) group publications based on their "most-specific" research topics (using the notion of research pyramid [Bani-Ahmad, S.. Ozsoyoglu, T. 2007]), (iii) use the parser output to build a hierarchical structure of simple and compound tokens to be used to suggest search terms, (iv) use TextRank, a text summarization tool, to assign topic-sensitive scores to keywords, and (v) use the identified research topics to help user aggregate focused search terms prior to actual search query execution.

To properly establish the basis to compare our proposed SK-Suggester to the CompleteSearch engine autocompletion tool [H. Bast, I. Weber. 2006], we start with an overview of our proposed framework through an example from the literature digital library domain. In the linguistic pre-processing step, we start by tokenizing documents of the publication repository, which transforms documents into a categorized block of text called tokens. At this stage, we ignore the stopwords; later, when forming complex tokens, i.e., combining more than one token into one complex token, we consider the stopwords to guarantee syntactically and semantically correct suggestions. For performance issues, one may choose to parse properly selected parts of each document. Experimentally, we have found that it is advantageous to parse two parts: (a) publication titles since (i) the number of tokens in a title are an order of magnitude less in count than the tokens of the full document, and (ii) publication titles are significantly less likely to have ambiguous tokens (like impersonal pronouns) than the full document even though, in rare occasions, authors choose for their articles humorous, but irrelevant, names, for instance, "On saying enough already in SQL" by Michael J. Carey and Donald Kossmann. Having said this, such titles are humorous and thus easy to be remembered by users, and they have great value in navigational queries in which the user has a particular target that s/he is searching for [U Lee, Z Liu and J Cho. 2005]. On the other hand, these titles negatively affect the performance of informational queries, in which the user is looking for sources that provide background knowledge about the search topic [U Lee, Z Liu and J Cho. 2005]. To remedy this approach, we also suggest preprocessing (b) abstracts of publications in addition to titles. We give an example.

Example: Tokenizing the title "The Linear Complexity of a Graph" generates the following simple tokens (i) "the", "of" and "a" which are stopwords, and (ii) "linear", "complexity", and "graph" which are non-stopwords that are expected to appear in user search queries. Stopwords are useful in forming compound tokens through combining two or more simple tokens at a time. For instance "linear complexity" and "graph" can be linked using "of" to form the full title. Simple and compound tokens then serve as building blocks for expected user search terms.

Next, we organize the collection of simple and compound tokens into a token hierarchy. During a search query session, the proposed SK-Suggester recommends search terms by traversing the token hierarchy as follows: at the beginning of the search session, the Single Token Anticipator, STA, is called to make suggestions based on the first few letters entered by the user. The STA is called each time the user enters a new search term during the session; however, the suggestion scope is continually reduced based on the previously fed terms within the same session. The suggestion scope is defined as the set of most-specific research "topics" where suggestions are extracted. Starting from simple tokens (i.e., the "most general" suggestions) towards higher levels of compound tokens (i.e., "more focused" or "more specific" suggestions), the SK-suggester guides the user towards building successful search terms. During this process, the user has the choice to stop further focusing his/her search terms when the following items are acceptable: (a) the expected query result size (i.e., the number of publications), (b)topic-sensitive significance, computed using TextRank [R. Mihalcea and P Tarau. 2004], and (c) scope, i.e., the number of relevant research topics [Bani-Ahmad, S.. Ozsoyoglu, T. 2007].

Research topics are represented by research pyramids, where a research pyramid is a set of publications that are related to the same research topic.. For more details on research pyramids, see [Bani-Ahmad, S.. Ozsoyoglu, T. 2007].

Next, to compare our approach, we briefly present how the CompleteSearch engine works [The CompleteSearch engine][H. Bast, I. Weber. 2006]. First, an index, named HYB, is prepared by preprocessing the document collection to pre-compute inverted lists of compound tokens. Compound tokens are identified using proximity measures between words separated by w, that is, the pre-determined window size. To maintain a good level of locality of search, similar words are placed in the same block within the index in the form of document-word pairs. As the user enters his search words, relevant blocks, i.e., blocks where search terms are observed, are identified and, thus, (searching) scope, or context, narrows down to only relevant documents.

In summary, the main contribution of this paper is to design and evaluate a content-driven SK-Suggester that (i) eliminates the drawbacks of Google's search history-based SK-Suggester, and (ii) boosts the performance of the techniques used in the CompleteSearch.

Since our proposal extends CompleteSearch suggester, our approach maintains all the advantages of CompleteSearch. For instance, our approach has an excellent locality of access. Moreover, the completion of subwords and phrases is automatically supported since phrases are linguistically pre-computed.

The remainder of the paper is organized as follows. In section 2 we present the search-keyword suggestion problem and draw the reader's attention to the major design principles of the proposed SK-Suggester. In section 3 the language-based pre-processing steps that generate the *token hierarchy* is described. In Section 4, we present our approach of computing topic-sensitive significance scores of filtered tokens, which are then used to order the computed refinements as well. In section 5, we describe how query refinements are made. We also briefly describe the techniques used in building the search interface. Finally, in section 6 we present the experimental results and the observations related the performance of the proposed SK-Suggester framework.

2. PROBLEM STATEMENT

The SK-Suggester problem involves the anticipation of the search terms that the user is attempting to specify. We define our SK-suggestion problem as follows:

Definition: An SK-Suggestion query is a 5-tuple Q(W, I, R, βs, βp), where W is all possible completions of the last word that the user started typing, and R and I are the sets of relevant topics (research pyramids) and compound tokens from the preceding query. βs and βp are thresholds for the maximum scope and the minimum popularity required to control the number of suggestions made available to the user. Processing query Q involves the following steps: (i) compute the subsets W' of W, and a word in W' that occurs in at least one compound token in I, (ii) compute R' and I' that form the set dominant research topics and compound tokens respectively, where $I' \subseteq I$ and $R' \subseteq R$. Alternatively, the user may choose to be shown a fixed number of suggestions as in Google Suggest and the CompleteSearch engine, in which case the query becomes a 4-tuple of the form Q(W, I, R, βk).

The main design goals of the proposed SK-Suggester are:

- The SK-Suggester should provide instant feedback to users prior to query execution. Studies show that search sessions usually have multiple queries [Y Zhang and A Moffat. 2006], and, that 82% of users who face unsuccessful searches modify their search terms to better target what they are searching for [iProspect Inc. 2006]. Further studies show that unsuccessful searches are followed by probably multiple clickthroughs before keyword refinement [Y Zhang and A Moffat. 2006]. This is probably because users become more knowledgeable of what is available, and thus refine their search terms accordingly. The primary goal of the proposed SK-Suggester is to help focus user's search term to what is already available prior to performing the search, and thus reduce the time spent on search failure. To meet this goal, the content-based SK-Suggester provides instant feedback as to how focused the search terms are to the user, prior to query execution.
- The SK-Suggester should suggest linguistically valid search terms. Having two words that frequently co-occur (or are similar to each other via a syntactic proximity measure) does not necessarily imply that we can put them together and provide the combination of the two as a meaningful suggestion. To meet this goal, we utilize an English language parser to tokenize and parse the digital library collection contents, and to build linguistically valid search terms. An alternative approach, used in the CompleteSearch engine [The CompleteSearch engine][H. Bast, I. Weber. 2006], is to show the user snippets of text; however, this approach needs preprocessing and more effort by the user to interpret them.
- The SK-Suggester should provide guidance to the user, as (s)he builds up the search terms. We achieve this by providing statistics on the search output prior to search execution. The proposed SK-Suggester provides the user with (i) the scope of each of the search terms (i.e. the set of papers to be returned), which also warns the user against

keywords that are very common and may lead to large search outputs. Notice that the number of documents where search terms are observed is not a good indicator of how focused search terms are. Given that the user is interested in a particular research topic, some research topics (represented as research pyramids) are large because many researchers are working on that topic [Bani-Ahmad, S. Ozsoyoglu, T. 2007], and thus large numbers of documents may be found relevant to a user query. Consequently, the fact that a search term is observed in large numbers of documents does not necessarily indicate that the keywords are not focused enough. A better indicator of how focused search terms are, is the number of relevant research topics, which is the number of research pyramids.

- As a part of the guidense provided to the user, suggestions retrieved from most recently published papers should probably given proveded with a "good" chance to be presented to the users first. The motivation behind this is that users (authors in the domain of literature digital libraries) are probably more interested in recently published works than relatively old works. In [Bani-Ahmad, S., Ozsoyoglu, T. 2009] it has also been observed that the probability that a publication receives new citations drops as it (the publication) gets older. This is refered to as the "user citation behavior" and a statistical model for it is proposed in [Bani-Ahmad, S., Ozsoyoglu, T. 2009]. This statistical model can be utilized to modify the topic-sensitive significance score of tokens to incorporate the age of the publication from where those tokens (simple or compound) were observed.
- The SK-Suggester should work online efficiently, and suggest refinements to keywords on the fly. For efficiency, our approach involves parsing properly selected parts of each document in the collection, and recognizes nouns, adjectives and verbs a priori. Further, all of the time consuming tasks are performed offline. We use (i) the link grammar-based parser, developed at Carnegie Mellon University [D. Temperley, et. al. 2005], and (ii) TextRank text summarizing algorithm [R. Mihalcea and P Tarau. 2004] to identify the most significant compound tokens that will be used to suggest refinements to user search terms.

3. CONSTRUCTING TOKEN HIERARCHY

In this section we summarize how the *Token Hierarchy* is built. Our discussions and examples are retrieved from a prototype digital library with a repository of around 15,000 publications from ACM SIGMOD Anthology, a digital library from the field of data management.

The token hierarchy involves the following levels

(1) The single token level.
(2) The keyphrases (compound token level)
(3) The publication title level
(4) The research pyramid level, each research pyramid represents a specific research topic.

The research-pyramid model was first proposed by Aya et. al. [S. Aya, et. al. 2005, and validated by Bani-Ahmad and Ozsoyoglu [Bani-Ahmad, S.. Ozsoyoglu, T. 2007]. This model suggests that citation relationships between research publications produce multiple, small, and pyramid-like structures [S. Aya, et. al. 2005][Bani-Ahmad, S.. Ozsoyoglu, T. 2007].

A research pyramid represents publications related to a *highly specific research topic*, and usually has a pyramid-like structure in terms of its *citation graph*. A citation graph G(V,E) of a given publication set, is a directed graph where publications represent the vertices. An edge is established between publications x and y (from x to y) if x cites y.

According to the research-pyramid model, a publication citation-graph evolves through the stimulation of most-specific research topics from one another as follows: [S. Aya, et. al. 2005][Bani-Ahmad, S.. Ozsoyoglu, T. 2007]. (i) A publication identifies a new *specific research problem* and proposes the first solution for it. (ii) More publications appear, addressing the same problem and proposing enhanced or refined solutions to that problem. In time, the research problem (a) is either solved, (b) settles down with “good-enough” solutions, or (c) subdivided into more specific research problems (i.e., new research pyramids).

Our SK-Suggester uses the identified research-pyramid structures to assign topic-sensitive significance score to tokens and refinements. This helps to propose refinements from research topics where the user’s entered keywords are of most significance.

We first present a number of natural language (English Language (EL) properties and definitions that will be used throughout this paper.

EL property 1: Simple Sentence types include (1) declarative, (2) interrogative, (3) imperative, and (4) conditional types. Compound sentences have the format “<simple sentence> <conjunction> <simple sentence>”. Declarative sentences consist of a subject and a predicate. Subject may be simple (i.e., consists of a noun phrase or nominative personal pronoun) or compound (i.e., consists of multiple subjects combined with conjunctions).

Figure 3 shows the parser output for the title "Outlier detection for high dimensional data" with multiple linkages identified within the title. Each linkage represent a linguistic relationship between two tokens (see EL Property 3 next). A full list of linkages that the parser can identify is available in [D. Temperley, et. al. 2005]; however, only a few of them are common and observable in publication titles.

To suggest linguistically valid keywords, we utilize the linkages identified by the parser to form compound tokens out of simple tokens. The following definitions and observations form the basis of our discussion on how the token hierarchy is built.

Definition: A simple token is a categorized block of text consisting of indivisible characters. A compound token is a linguistically valid combination of one or more simple tokens. □

As an example, "sort", "merge" and "join" are simple tokens. "sort-merge" and "sort-merge-join" are linguistically valid compound tokens; but, "join sort-merge" is linguistically invalid as the adjective should precede the noun in English.

Note that, not all linguistically valid compound tokens are “observed” in a digital library. For instance, "merge-sort join" is linguistically valid, but there is no such join algorithm in the data management field. We will refer to linguistically valid, but not necessarily observed, compound tokens as unrealistic compound tokens.

EL property 2: Part-of-speech token types include (1) articles, (2) nouns (subjects or objects), (3) adjectives, (4) adverbs, (5) pronouns, (6) conjunctions, (7) verbs, and (8) prepositions. □

To form realistic compound tokens, we identify part-of-speech tokens that are linguistically adjacent. The goal is to make keyword suggestions that make sense to the user. We use the link-grammar-based parser proposed in [D. Temperley, et. al. 2005] to identify linguistically adjacent tokens and build the token hierarchy of the publication set.

EL property 3: Possible linguistically adjacent or related token type cases include (1) (subject, verb), (2) (verb, object), (3) (adjective, noun), (4) Compound subjects, (5) Compound objects (6) (noun-possessive, noun), (7) (article, noun), (8) (adverb, verb). □

Example: the title "Adaptive Rank-Aware Query Optimization in Relational Databases" has the following compound tokens. (i) (Adjective, noun): "relational databases", (ii) (compound adjective): "rank-aware", (iii) (adjective, noun) "adaptive query", (4) "query optimization". Note that more complicated combinations of tokens are also possible, e.g., (i) (compound subject, verb), (ii) (verb, compound object), or (iii) (simple subject, verb, object).

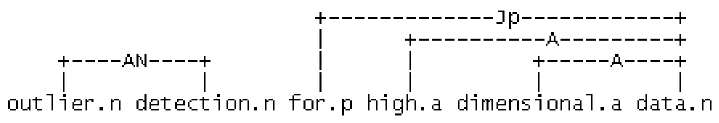
The parser is used as a tool to parse the titles. Table 1 presents a list of the linkage types observed in titles. A full list of all linkage-types can be found in [D. Temperley, et. al. 2005].

Table 1. The observed linkage types and their percentages

Linkage type	% across ACM SIGMOD Anthology	Information
(A)	24.35	Connects adjective to noun
(AN)	23.43	Connects noun-modifier to noun
(J)	18.28	Connects preposition to its objects
(D)	8.42	Connects determinator to noun
(M)	8.69	Connects noun to post-noun modifiers
(MV)	4.46	Connects verbs to adjectives
(O)	6.15	Connects transitive verbs to objects

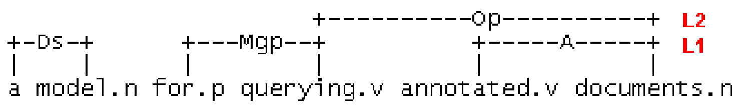
Table 2. The frequency of each observed parts-of-speech

Part of Speech	Frequency	Part of Speech	Frequency
Nouns	47.32	Adverbs	0.076
Adjectives	15.23	Clauses	0.069
Verbal nouns	10.65	Relative clauses	0.025
Prepositions	4.48	Un-tagged	21.14



```
                        +-------------Jp------------+
                        |      +----------A---------+
  +----AN----+          |      |         +-----A----+
  |          |          |      |         |          |
outlier.n detection.n for.p high.a dimensional.a data.n
```

Figure 3. A sample output from the Link-grammar parser.



```
                            +-----------Op-----------+  L2
+--Ds-+        +---Mgp--+               +------A-----+  L1
|     |        |        |               |            |
a model.n for.p querying.v annotated.v documents.n
```

Figure 4. levels of linkages and super-linkages.

Building blocks of the token hierarchy are nouns, adjectives and verbal nouns (which are sometimes identified as verbs or gerunds by the parser), altogether forming around 75% of the identified tokens in titles. The rest are stopwords, which are not totally ignored while constructing the hierarchy; we keep them in order to build meaningful compound tokens.

We construct the token hierarchy by collapsing the observed linguistically adjacent tokens into compound tokens. Any two tokens that are linked via a linkage are considered to be linguistically adjacent even if they are separated by other tokens or stopwords. A super-linkage, that is, a linkage that encompasses one or more linkages, is used to construct further compound tokens. We give an example.

Example: Figure 4 shows the parser results for the title "a model for querying annotated documents". Two levels of linkages are identified: (i) the 'A' linkage is at level 1 and used to form the compound token "annotated documents", (ii) the super-linkage 'OP' at level 2 (which encompasses the 'A' linkage) is used to form the compound token "querying annotated documents".

Figure 5 shows the different layers of the token hierarchy. To illustrate we give an example.

Example: (Compound token) In the parser output shown in figure 3, the simple tokens (outlier, detection, high, dimensional and data) are located in the lowest level of the token hierarchy. Using the identified linkages of this title, we construct the compound tokens “outlier detection” and “high dimensional data”. Each one of the two compound tokens forms a compound token because there is no linkage identified between the compound tokens (only linkages between filtered tokens are considered).

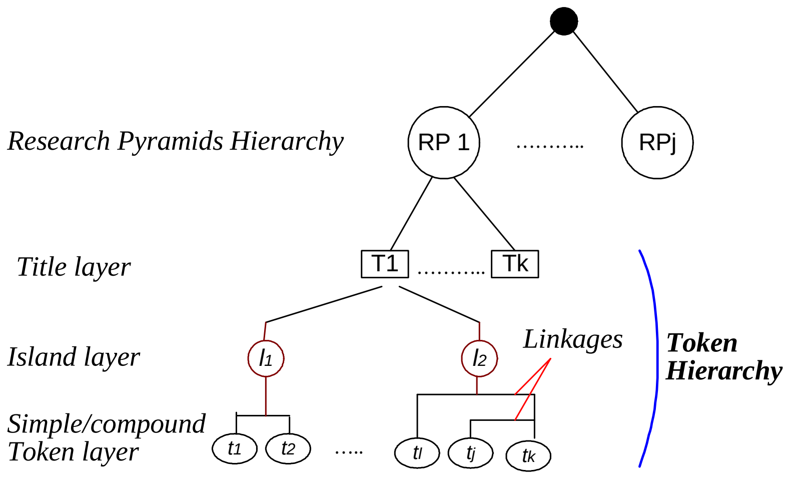


Figure 5. Layers of the *Token Hierarchy*

Publication titles, in turn, belong to papers that are clustered into research pyramids. Each research pyramid includes publications that deal with highly specific research topics [Bani-Ahmad, S. Ozsoyoglu, T. 2007]. Consequently, the full hierarchy that is utilized by the proposed SK-Suggester consists of four layers as illustrated in figure 5: (i) the research pyramids layer, (ii) the title layer, (iii) the compound token layer, (iv) the simple/compound token layer.

4. TOPIC-SENSITIVE TOKEN WEIGHT

Each user search session can be viewed as aiming at finding information about a specific topic. This implies that the user's suggestions of search terms should be chosen as close to the topic being targeted as possible. However, the topic being targeted is unknown to us. Thus, we use the already entered search terms to prune out topics where these keywords are not or rarely observed. We refer to this phenomenon as the locality of search principle.

Observation: (The locality of search principle): Within a single search session, the user targets documents within a specific topic.

This principle allows us to narrow down the suggestion scope as the user enters more search terms. The token-hierarchy relation is then accessed once in between keystrokes each time the user modifies the search terms by typing one more character. Given our hypothesis that the document(s) that the user is looking for belongs to a specific research topic or few related topics, we can reduce dramatically the diversity of the collection set by suggesting keywords from the most relevant research topic(s), which we refer to as the suggestion scope.

One issue is that a term may be used in more than one research topic. To solve this problem, we weigh tokens within each research topic. The goal is to identify the significance of tokens in each research topic, and thus prevent search-keyword refinement from topics where keywords are of lesser significance.

To weigh tokens in each research-pyramid, we use the TextRank algorithm [R. Mihalcea and P Tarau. 2004]. Briefly, TextRank algorithm constructs a graph between a properly selected set of tokens of a document (nouns and adjectives), where an edge between two tokens exists only when they co-appear together in a window of some size. Then we apply the PageRank algorithm on the formed graph to identify the most important tokens. PageRank is a an algorithm applied on graphs to measure relative importances of vertices [Brin, S., Page, L. 1998]. Finally, phrases are manually constructed out of the top-scored tokens; these phrases represent keyphrases of the document.

We use TextRank at research pyramid level to compute topic-sensitive significance score of terms. We apply TextRank on each research pyramid r as follows. (i) The titles of all publications that belong to r are tokenized and annotated with part-of-speech tags using the link-grammar parser [D. Temperley, et. al. 2005]. (ii) The tokens are filtered through a syntactic filter which selects only lexical units of certain parts of speech, namely; nouns (as well as verbal nouns and gerands) and adjectives, that give the best results [R. Mihalcea and P Tarau. 2004]. (iii) A graph Gr(V,E) is formed using the tokens returned by the filter. V, or the set of vertices, is the set of tokens. E, i.e. the edge list, is constructed such that an edge is created between any two tokens that appear in the same title. (iv) PageRank [Brin, S., Page, L. 1998] is used to measure relative importance of all tokens. Tokens that have high PageRank scores are expected to be more significant and better representatives of the research pyramid r.

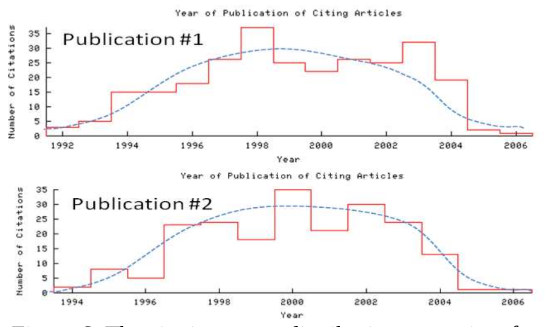


Figure 6. The citation-count distributions over time for three publications found in CiteSeer.

The topic-sensitive TextRank scores can be further refined to incorporate the user's citation behavior model proposed in [Bani-Ahmad, S., Ozsoyoglu, T. 2009]. In [Bani-Ahmad, S., Ozsoyoglu, T. 2009], the researchers observed that in technology-driven fields (such as computer science and life sciences), authors tend not to cite old publications.

The two plots in Figure 6 show citation counts of two relatively highly cited publications from CiteSeer [CiteSeer]. Notice that the citation-counts of the two publications have dropped significantly after year 2004. In [Bani-Ahmad, S., Ozsoyoglu, T. 2009] it has also been observed that the probability that a publication receives new citations drops as it gets older. This observation can be utilized to modify the topic-sensitive significance score of tokens to incorporate the age of the publication from where those tokens (simple or compound) were observed.

5. SUGGESTING SEARCH KEYWORDS

In this section we present how to suggest refinements to users' search terms. This task is performed online; thus, real-time performance is critical.

The SK-Suggester is triggered online "in-between keystrokes". After each keystroke, the search terms already entered are sent through an AJAX-enabled interface form to SK-Suggester STA (Single-Token-Anticipation) and QR (Query-Refinement) Modules at the server side (see figure 7). An AJAX-enabled search interface is needed in this application in order to provide an immediate, flexible, and responsive interaction [J Wusteman and P O'hIceadha, 2006][Bast, H., et al. 2007.] [J Wusteman and P O'hIceadha, 2006].

Online steps of our approach are:

```
Procedure SK-SuggesterInterface ()
Input  User Input w : current search-terms
       Server Input : R, and I ( stored in session status)
{
(1) For w
   (1.1) LISK <- the uncompleted search keyword in w
   (1.2) CSK <- the completed search keywords in w
   (1.3) If (CSK="" && LISK!="")
         STA_Module(LISK);
   (1.4) Elseif (CSK!="" && LISK="")
         QR_Module(CSK, LISK);
   (1.5) Elseif (CSK!="" && LISK!="")
         SK-List1 <- STA_Module( LISK);
         SK-List2 <- QR_Module(CSK, LISK);
         Join(SK-List1, SK-List2)
 (2) Presentation_Module(W')
```

Figure 7. The SK-Suggester interface procedure.

(i) Single token anticipation. The STA (Single-Token-

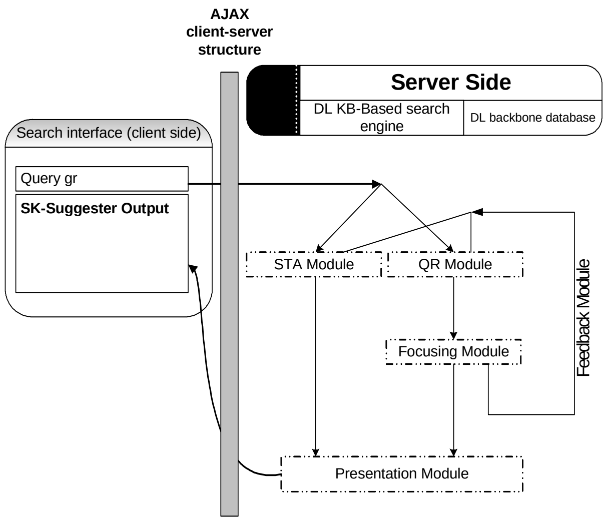


Figure 8. SK-Suggester Query Execution Modules

Anticipation) Module (figure 8) is triggered each time the user starts entering a new search term. This module suggests completions to the incomplete term entered by the user from the current suggestion scope (by using R and I in definition 1). At the beginning, the suggestion scope is all the research pyramids and all the compound tokens, which is the most time-consuming step [H. Bast, I. Weber. 2006].

(ii) Search term refinement suggestion. The QR (Query-Refinement) Module in figure 7 suggests the top-scored compound tokens I to the user as possible refinements to the user's search terms.

(iii) Focusing suggestion scope (the feedback module in figure 7): In this step, the subsets R' and I' are computed and saved in the search session status structure to be used in query refinements after the next keystroke.

(iv) Post-processing suggestions (the Presentation module in figure 7).

Next, we list and discuss the advantages of our proposed framework as compared to [H. Bast, I. Weber. 2006]:

(i) Tokens, simple and compound, observed in the same research pyramid, or multiple strongly related research pyramids, are stored within the same block as in [H. Bast, I. Weber. 2006]. This gives better locality of search and reduces I/O operations especially after suggestion scope reduces to few relevant research pyramids (see subsection 7.3 on conversion of suggestion scope).

(ii) A term may be used in more than one research topic. To solve this problem, we weigh tokens within each research topic. The goal is to identify the significance of tokens in each research topic, and thus prevent search-keyword refinement from topics where keywords are of lesser significance.

(iii) In Bast and Weber [H. Bast, I. Weber. 2006], suggestions are presented to the user as snippets of text from the documents in the literature digital library. This puts an extra burden on the user to isolate useful information from the presented text. Users usually type fast and may not have enough time for post-processing the presented suggestions. In our case, the repository is linguistically preprocessed to identify compound tokens, or compound tokens, that will be presented to the user isolated from the surrounding text.

(iv) Scalability: in order to suggest phrases instead of single words, Bast and Weber [H. Bast, I. Weber. 2006] use text-based adjacency (within a predetermined window size) as an indicator of token-to-token proximity. We observed that this proximity measure generates long lists of possible phrases which (a) significantly increases the index size [H. Bast, I. Weber. 2006]; this problem is solved by viewing the auto-completion problem as a multi-dimensional range searching problem [H. Bast, I. Weber. 2006] and (b) may result in meaningless phrases. Our proposal uses linguistic adjacency (see EL property 3) which produces meaningful and much smaller lists. Consequently, our approach is more scalable.

In order to match completions with the being entered query word, Bast and Weber [H. Bast, I. Weber. 2006] store the positions of terms within each document in an array separate from the index. We refer to this technique by the text-based adjacency (see section 6.2 in the experimental results). Online processing of this extra array takes time. In our case, we use linguistic linkage-based proximity of tokens to build compound tokens (see section 6.2 in the experimental results). This gives more realistic and better results as tokens (nouns and adjectives, for instance) may be separated by intermediate words but still linguistically related. Thus, depending on the assigned proximity window, some close terms may be missed in the case of small widow sizes, or false positives may appear in the case of big window sizes. Our approach, in some sense, uses proximity windows with variable sizes.

5.1 SK-Suggester Query Execution

As stated at the beginning of section 2, an SK-Suggestion query is a 5-tuple Q(W, I, R, βs, βp) where W is all possible completions of the last word that the user started typing, i.e. the STA output, and I (compound tokens) is a list of the most promising refinements of the already entered search terms. The parameter R is the set of dominantly relevant topics (or research pyramids). βs, βp are thresholds of the maximum scope and minimum popularity required to control the number of suggestions.

Processing the query Q involves the following steps: (i) compute the subsets W' of W, and a word in W' that occurs in at least one compound token in I, (ii) compute R', $R' \subseteq R$ and I', $I' \subseteq I$, that form the set dominant research topics and Compound tokens respectively. Alternatively, the user may choose to be shown a fixed number of suggestions as in Google Suggest and the CompleteSearch engine, in which case the query becomes a 4-tuple of the form Q(I, R, W, βk).

Compound tokens are used as refinements to user queries, and vary in their sizes. To avoid proposing a long suggestion, compared to user search terms, we propose a gradual expansion of the user query as follows. Given user's search terms W, refinements of length up to e_f * |W| are presented to the user, where e_f is the expansion factor, and |W| is the number of tokens in W.

We empirically observed that initially choosing the expansion factor to be 1.5 gives good results allow for a gradual expansion during user's search term construction. However, when the user chooses terms that are separated by a relatively large distance, i.e., separated by long series of words which is the case in large compound tokens, a particular choice of an expansion factor may fail to retrieve refinements. To remedy this problem, we propose dynamically choosing e_f through probing as follows. First, we choose e_f =1.5. If no suggestions can be retrieved, the value of e_f is increased up to e_{fmax} which is chosen as the length of largest compound token observed. The amount by which e_f is increased is left to the digital library server to estimate, based on how many online users are available and whether real-time performance is achieved or not.

Figure 6 sketches the SK-Suggester search interface procedure. The procedure receives user's search-terms from the client, calls either the STA or the QR modules depending on w as follows: (LISK is the last uncompleted search term, and CSK is the set of completed search terms in w). If LISK is not empty, the STA Module is triggered. If CSK is not empty, the QR Module is called. If both, QR and STA modules are called, and suggestions from both modules are joined such that suggestions from similar research topics are coupled, and suggestions from dominant research topics are propagated to the presentation module.

5.1 Guiding Statistics

Next we present a list of statistics that are used to guide the user selection of search terms.

Suggestion Scope: Research Pyramid based suggestion scope considers the number of research topics (or research pyramids) where the search terms w are observed, that is, the scope is

Scope(w)=(# of RPs where w appears)

Topic-Sensitive Popularity of Search terms: For a set of words (W'), the topic-sensitive popularity of W' with respect to the topic represented by some research pyramid r, i.e., TSP(W', r), is computed as the sum of TextRank scores of all words in W'. TextRank scores are topic-sensitive and computed within each research pyramid r. The suggestions are retrieved from dominant research pyramids computed by the feedback module in figure 7.

Query refinements (W') are presented to the user in the order of their matching scores which we define as follows:

$$M_{Score(W',W)} = Similarity(W',W) * Max_{r \in RP(W')}[TSP(W',r)]$$

Where $Similarity(W',W)$ is the text-based similarity between the suggested refinement W' and the search terms entered already entered by the user. And $Max_{r \in RP(W')}[TSP(W',r)]$ is the maximum TSP value observed for the refinement W' in all research pyramids where W' is observed.

One more statistic used is the Specifity of Individual terms. Specifity of token t is measured as

Specifity(t)=-log[(# of Docs where t appears)/(total # of Docs)]

We use this number to color user's already entered terms to indicate how general his/her individual search terms are. This helps when user's search terms consist of stopwords or terms used in a wide range of research topics.

6. EXPERIMENTAL RESULTS

The document collection used in our experiments includes 14,891 publications from the ACM SIGMOD Anthology, a digital library from the field of data management.

Experimental results section is organized as follows. In section 6.1, we list our observations on the accuracy of the linguistic pre-processing step and the quality of suggestions. Our observations and on the scalability of our approach and the convergence of suggestion scope are presented in sections 6.2 and 6.3, respectively.

6.1 Accuracy of Linguistic Pre-processing and Quality of Suggestions

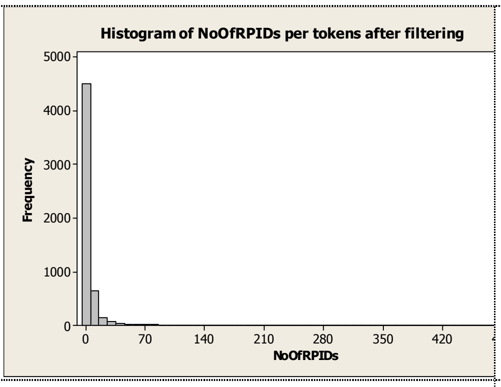


Figure 9. Distribution of token scope (after filtering)

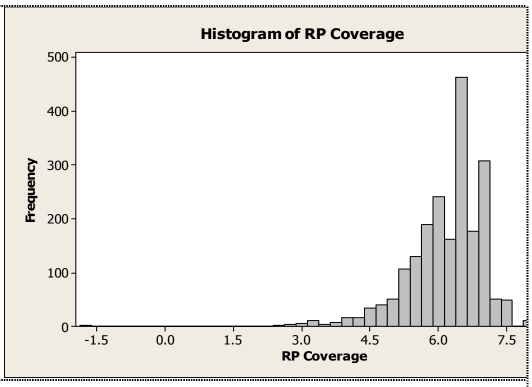


Figure 10. Distribution of RP coverage

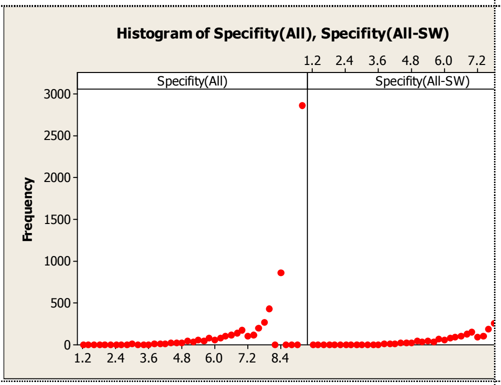


Figure 11. Distribution of specifity values of (a) all tokens (left) and (b) all tokens except stopwords

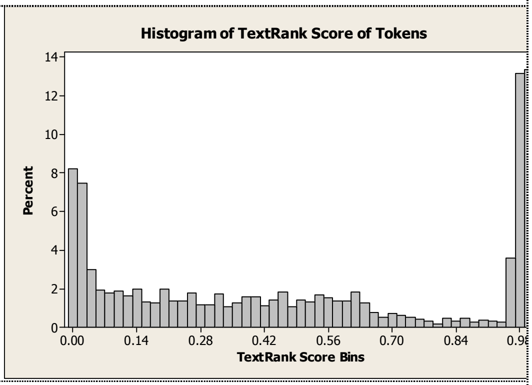


Figure 12. Distribution of TextRank scores for simple tokens.

The untagged tokens are distributed as follows.

Table 3. The distribution of un-tagged tokens.

Token	% of untagged tokens	Token	% of untagged tokens
of	17.39	to	4.72
a	14.76	an	4.51
in	14.30	on	3.43
and	14.12	with	3.34
the	11.92	from	1.62

In the document collection, untagged tokens are all stopwords. We do not totally ignore stopwords, but rather we use them to construct compound tokens and connect linguistically adjacent compound tokens.

(1)	Multiple Query Processing In Deductive Databases Using Query Graphs
(2)	Query Graphs Implementing Trees And Freely Reorderable Outerjoins
(3)	Effective Graph Clustering For Path Queries In Digital Map Databases
(4)	Query By Diagram, A Graphic Query System

Figure 13. Possible hits of the query "query graph"

In the following example, we show how the proposed SK-Suggester also serves in early construction of successful search terms for k-word proximity search, which is a very useful technique in narrowing down the results to more relevant ones, and at the same time allowing users to better express what they are looking for [Gupta, Chirag. 2008].

Example (k-word Proximity Search): In figure 13, notice that the search terms "query graph" are already identified as one compound token. Suggesting query refinements based on compound tokens may help towards a successful proximity search. Notice that item (3) is probably irrelevant to the query at search time since this publication most probably belongs to different research pyramid from the first and second hits; this false positive is pruned or pushed down in ranking query results. Informing users of the linguistic proximity of search terms prior to query execution can thus be useful. Furthermore, informing the user of the order in which terms appear may help eliminate false hits like hit (4) in figure 11, which is called k-word ordered proximity search [Gupta, Chirag. 2008].

6.2 Scalability and Index Sizes

Our approach uses linkages to construct compound tokens. This technique generates significantly smaller numbers of constructs than the text-based adjacency used in Bast and Weber [H. Bast, I. Weber. 2006]. For instance, by parsing the titles of around 9 thousand publications from the repository, 5,652 tokens were retrieved (6,896 tokens including stopwords). And, around 5 thousand compound tokens are constructed. Considering text-based adjacency using the same window sizes generated 220,000 of links between tokens. These links are to be processed further to identify the most significant compound tokens.

The value of using linguistic pre-processing to identify compound tokens comes from the quality of pre-computed compound tokens that can be constructed. Along with the post-processing required by the text-based adjacency approach, both factors balance the time needed to perform the linguistic pre-processing step (which is done offline).

6.3 Convergence of Suggestion Scope

One more factor that is critical in producing search-keyword suggestions in real-time is the locality principle of search and the convergence speed of the suggestion scope.

Figure 8 shows the distribution of scope of the observed filtered tokens, i.e., excluding the stopwords.

Observation (figure 8): Filtered tokens have limited scope.

The above observation is important as it significantly affects the QR module performance. Considering this observation, we may prune the scope of the suggestions, which is the set of dominant research pyramids from where suggestions are retrieved.

Notice that some research topics may have wide range of origins. This makes the diversity of terms used in such research topics wide as well. For example, the publication "TextRank: Bringing order into text" have origins in linguistics (tokenizing and parsing), graph theory and graph-based ranking.

To measure how wide and diverse the origin of the research topic (or a research pyramid) r is, we use the notion of RP coverage computed as follows:

Coverage(r)= -log[(# of tokens used in r)/(total # of tokens)]

This means that the higher the coverage factor of r is, the less diverse its tokens become. Zero coverage of r indicates that all tokens ever observed in the collection are used in r.

Figure 9 shows the distribution of coverage values of all research pyramids. Coverage values range between 3 and 8, which means that (i) the tokens within each research pyramid are of low diversity, and (ii) this signifies the importance of ranking tokens within research topics. This serves in pushing refinements extracted from dominant research topic(s) up in the suggestion list. We achieve this goal by using the topic-sensitive popularity (TSP) of search terms to order the list of computed refinements.

One critical factor that affects the STA module performance is the speed of convergence of the suggestion scope at the beginning of each search session.

We have experimentally observed that, usually within 3 characters entered by the user, the suggestion scope significantly decreases. We have also observed that the suggestion scope of STA reaches a saturation region within 4 characters entered by the user.

Figure 10 shows the distribution of specifity values of all tokens extracted from the document collection. High specifity value of a token t indicates that t is observed in many documents. Figure 10 shows that high percentage of observed tokens have high specifity values, and thus, may lead to large search output lists. This further signifies the importance of warning users against such popular tokens and encourage him/her properly choose tokens of lesser specifity values.

Figure 10 shows TextRank score distribution of all simple tokens that pass through the syntactic filter. High TextRank scores indicate popular and more significant tokens. Thus, tokens that score high (>0.8) are content-bearing and better represent the research topic of the corresponding research pyramid. At the other extreme, low-scored tokens are usually widely used tokens.

We use topic sensitive popularity scores of tokens to order computed refinements such that the most relevant refinement from dominant research pyramids appear close to the top of the suggestion-list.

Since the post-processing module processes the final selected list of single token completions and the computed refinements, it is scalable and takes constant time to finalize the SK-Suggester output in the proper HTML format. Further, this module can be run at the client side using client-side scripting language.

4. CONCLUSIONS

We have proposed and experimentally validated a content-driven SK-Suggester. We have experimentally shown that the proposed framework, which is optimized to work on literature digital libraries, promises a more scalable, high quality, and user-friendly SK-Suggester when compared to its competitors. We have also shown that, as it (i) pre-computes topic-sensitive scores and (ii) directs user's choice of search terms toward most-specific research topics; our approach has an excellent locality of access.

REFERENCES

Journal

J Wusteman and P O'hIceadha, 2006. *Using Ajax to Empower Dynamic Searching, Information Technolgy and Libraries,* Vol. 25 No 2, June 2006, PP 57-64.

D. O. Case and D. M. Higgins. 2000. *How can we investigate citation behavior? A study of reasons for citing literature in communication,* Journal Of American Society of Information Science, 51(7):635-645, 2000.

Conference paper or contributed volume

S. Aya, et. al. 2005. *Citation Classification and its Applications*, ICKM.

G. Jeh and J. Widom. 2002. *SimRank, A Measure of Structural-Context Similarity,* ACM KDD Conference.

Bani-Ahmad, S., Ozsoyoglu, T. 2007. *Improved Publication Scores for Online Digital Libraries via Research Pyramids.* ECDL.

Bani-Ahmad, S., Ozsoyoglu, T. 2009. *On Publication Popularity: Popularity Grawth and Decay Phases of Publications.* Innovations in Information Technology.

iProspect Inc. 2006. *iProspect Search Engine User Behavior study*, iProspect.

U Lee, Z Liu and J Cho. 2005. *Automatic Identification of User Goals in Web Search*, WWW.

Y Zhang and A Moffat. 2006. *Some Observations on User Search Behavior*, the 11th Australian Document Computing Symposium, Australia.

R. Mihalcea and P Tarau. 2004. *TextRank: Bringing Order into Texts*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Barcelona, Spain.

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

Bast, H., et al. 2007. *Efficient interactive query expansion with complete search*. CIKM.

H. Bast, I. Weber. 2006. *Type less, find more: fast autocompletion search with a succinct index.* SIGIR 2006: 364-371

S. Bani-Ahmad, G. Ozsoyoglu. 2006. *elGiza, A Research-Pyramid Based Search Tool for Vertical Literature Digital Libraries*, DBRank 2008.

Gupta, Chirag. 2008. *Efficient K-Word Proximity Search*, MS Thesis, CWRU, EECS Department.

Krovetz, R. 1997. *Homonymy and polysemy in information retrieval*. In Proceedings of the Eighth Conference on European Chapter of the Association For Computational Linguistics (Madrid, Spain, July 07 - 12, 1997). European Chapter Meeting of the ACL.

Kunihiko Sadakane, Hiroshi Imai. 1999. *Text Retrieval by Using k-word Proximity Search.* International Symposium on Database Applications in Non-Traditional Environments (DANTE'99).

Web sites

Wiki, http://en.wikipedia.org

Google Suggest Labs, http://www.google.com/webhp?complete=1&hl=en

ACM Digital Library, http://portal.acm.org/dl.cfm

IEEE XPlore, http://ieeexplore.ieee.org

CiteSeer, www.citeseer.com

Google Scholar, http://scholar.google.com/scholar

D. Temperley, et. al. 2005. *Link Gramar Parser 4.1 – http://www.link.cs.cmu.edu/link*.

Derek Sisson. A Thoughtful Approach to Web Site Quality, http://www.philosophe.com/. Viewed in December 2007.

The Onix full text indexing engine, The most widely used stopword list, http://www.lextek.com/manuals/onix/stopwords1.html

The CompleteSearch engine, http://search.mpi-inf.mpg.de/