

# On Content-Driven Search-Keyword Suggesters for Literature Digital Libraries

Sulieman Bani-Ahmad and Gultekin Ozsoyoglu  
EECS Dept. Case Western Reserve University, Cleveland OH 44120

# Online Literature Digital Libraries

## Current Status

---

- ▶ **No Search keyword-suggesters**
  - ▶ IEEE Explore
  - ▶ ACM Digital Library
  - ▶ PubMed
  - ▶ ScienceDirect
- ▶ **Search History-Based Keyword-Suggester**
  - ▶ Google Scholar – through Google Suggest
- ▶ **Content-Driven Keyword-Suggester**
  - ▶ The CompleteSearch engine, experimental



# Why Search-Keyword Suggestion?

---

- ▶ helpful for constructing search keywords for queries that are
  - ▶ more accurate,
  - ▶ less ambiguous, and
  - ▶ more focused
- ▶ users spend considerable amounts of time in search sessions
  - ▶ to properly select keywords, and
  - ▶ to modify their search keywords in order to successfully locate relevant documents.



# Google Suggest Approach

## [A Graph Query Language and Its Query Processing - Sheng, Ozsoyoglu ...](#)

Many new database applications involve querying of **graph** data. In this paper, we present an objectoriented **graph** data model, and an OQL like **graph query** ...

[citeseer.ist.psu.edu/5773.html](#) - 27k - [Cached](#) - [Similar pages](#) - [Note this](#)

## [\[PDF\] Interactive Web search by graphical query refinement](#)

File Format: PDF/Adobe Acrobat - [View as HTML](#)

represents the user's information needs using a **query graph**: a. set of **graph** objects. ... In this way, **query graphs** can visualize the information ...

[www10.org/cdrom/posters/1078.pdf](#) - [Similar pages](#) - [Note this](#)

## [GraphBlast-Graph Matching tool](#)

GraphGrep is a software package to search for a **query graph** in a database of **graphs**. Given a collection of **graphs** and a pattern **graph**, GraphGrep finds all ...

[www.cs.nyu.edu/shasha/papers/graphgrep/](#) - 17k - [Cached](#) - [Similar pages](#) - [Note this](#)

## [Query graphs, implementing trees, and freely-reorderable outerjoins](#)

We determine when a join/outerjoin **query** can be expressed unambiguously as a **query graph**, without an explicit specification of the order of evaluation. ...

[portal.acm.org/citation.cfm?id=98738](#) - [Similar pages](#) - [Note this](#)

## [\[PDF\] Deadlock Resolution in Pipelined Query Graphs](#)

File Format: PDF/Adobe Acrobat - [View as HTML](#)

While pipelining in **query graphs** can increase performance, it can also lead to runtime ... applicable to any system that uses pipelined **query graphs**. ...

[nms.csail.mit.edu/~stavros/pubs/deadlock.pdf](#) - [Similar pages](#) - [Note this](#)



graph col	
graph coloring problems	140,000 results
graphic cow	503,000 results
graphic comm central	2,220,000 results
graph complement	405,000 results
graph contraction	314,000 results
graph complexity	490,000 results
graph construction	348,000 results
graph cos 2x	34,500 results
	<a href="#">close</a>

**Left:** No suggestions, however, relevant documents exist.

**Up:** Suggesting from Search history and the expected output sizes.



# Search-History Based SKS

---

## ▶ Google's SK-Suggester

- ▶ utilizes the search history of all users as keyword suggestions
- ▶ recommends keywords from
  - ▶ (i) popular searches,
  - ▶ (ii) searches from the current user's search history,
  - ▶ (iii) current user's bookmarks.

## ▶ Drawbacks

- ▶ Noisy keywords; typos from users included as suggestions
- ▶ Incorrect search-characterization, not well-characterized search terms.
- ▶ Bias to users' interests
- ▶ Needs comprehensive search history maintenance
- ▶ High processing and maintenance costs



# The CompleteSearch Engine

The screenshot displays the CompleteSearch engine interface. On the left, the search bar contains the query 'query grap'. Below it, a 'Refine by WORD' section lists related terms: 'graph' (3824), 'graphs' (2385), 'graphical' (1330), and 'graphics' (726). The main search results area on the right shows 'Hits 1 - 7 of 5435 for query grap'. The first result is 'Graph Indexing: Tree Delta >= Graph' by Peixiang Zhao, Jeffrey Xu Yu, and Philip S. Yu, published in VLDB 2007:938-949. The second result is 'Substructure Similarity Search in Graph Databases' by Xifeng Yan, Philip S. Yu, and Jiawei Han, published in SIGMOD Conference 2005:766-777. Both results include text snippets with the word 'query' highlighted in yellow.

Searching the CompleteSearch Engine for “query grap”

- ▶ Actual execution of query.
- ▶ Text-snippets from the document collection are returned;
- ▶ needs preprocessing by the user.

# The CompleteSearch engine Approach

---

- ▶ Prepare the HYB index, pre-compute inverted lists of unions of words.
- ▶ Unions of words are identified using proximity measures between words separated by  $w$  (a pre-determined window size).
- ▶ Similar words are placed in the same block within the index
  - ▶ Maintains a good level of *locality of search*,
  - ▶ As the user enters his search words, relevant blocks determine the (*searching*) scope,



# Content-Based SK-Suggester

---

- ▶ Content-Based SK-Suggester anticipates users' search keywords.

## General Approach:

- ▶ (i) parse the document collection to be searched,
- ▶ (ii) prepare offline refinements to search keywords, and
- ▶ (iii) dynamically suggest keywords as the user types his/her keywords



# Our Approach

---

- (i) Parse the document collection: *Link Grammar parser*.
  - ▶ a syntactic parser of English,
- (ii) Group publications.
  - ▶ “most-specific” research topics (research pyramids),
- (iii) Within each research pyramid (RP), build a hierarchical structure of simple and compound tokens (phrases), islands
- (iv) Attach topic-(RP-)sensitive scores to keywords
  - ▶ use TextRank, a text summarization tool,
- (v) Use the identified *research-topics* to help user choose more focused search keywords *prior to actual search query execution*.



# Utilized Tools

The Link-Grammar English Parser

The Research-Pyramid Based Grouping Utility

TextRank Algorithm

# Linguistic Pre-Processing Step

---

## Tokenize documents

- ▶ transforms documents into a categorized block of text, called *tokens*

## Form *compound tokens*: combine two or more simple tokens at a time

- ▶ Builds syntactically and semantically correct suggestions (using the link-grammar output)

## **Example:** Tokenize the paper title:

*"The Linear Complexity of a Graph"*:

### Generate *simple* tokens

- (i) "the", "of" and "a" which are *stopwords*, and
- (ii) "linear", "complexity", and "graph" which are *non-stopwords*.
- (iii) form *compound tokens*: "linear complexity".



# Link-Grammar Parser

---

outlier.n detection.n for.p high.a dimensional.a data.n

Diagram illustrating the Link-Grammar Parser structure for the sentence "outlier.n detection.n for.p high.a dimensional.a data.n". The diagram shows the following structure:

- AN (Adjective Noun) link: outlier.n detection.n
- Op (Operator) link: for.p
- A (Adjective) link: high.a
- A (Adjective) link: dimensional.a
- N (Noun) link: data.n

a model.n for.p querying.v annotated.v documents.n

Diagram illustrating the Link-Grammar Parser structure for the sentence "a model.n for.p querying.v annotated.v documents.n". The diagram shows the following structure:

- DS (Determiner Suffix) link: a
- Mgp (Modifier Group) link: model.n
- Op (Operator) link: for.p
- A (Adjective) link: querying.v
- Op (Operator) link: annotated.v
- N (Noun) link: documents.n

Labels **L2** and **L1** are shown in red text next to the Op and A links respectively.



# Link-Grammar Parser

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

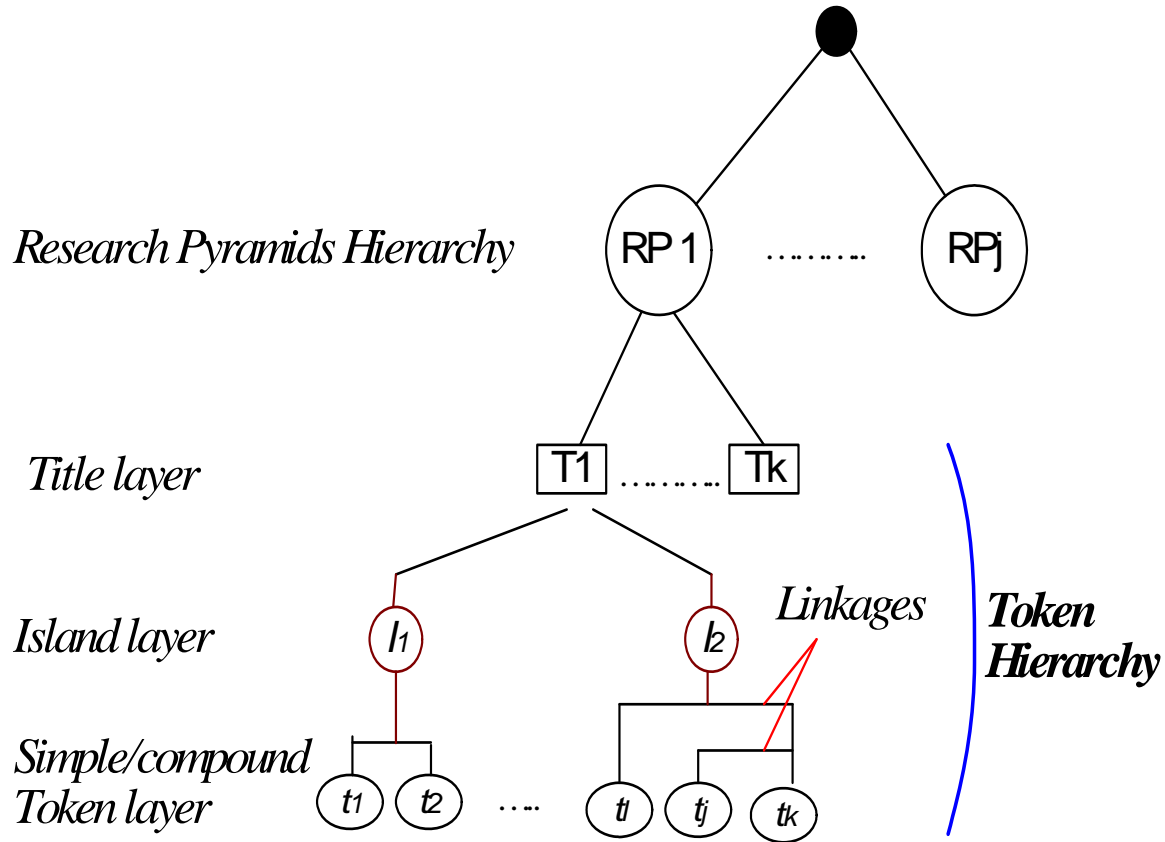
Frequency of each observed parts-of-speech token

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



# Token Hierarchy

---



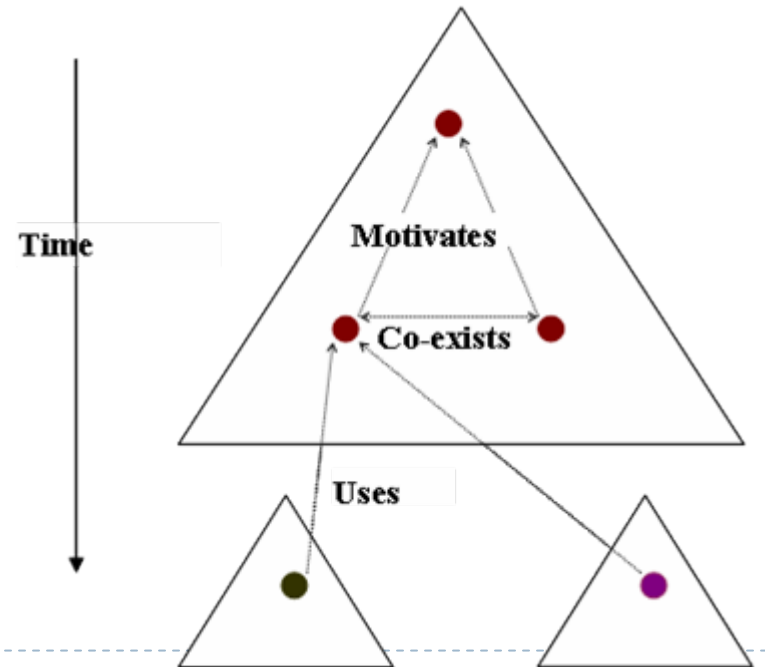
# The Research-Pyramid Model

---

- ▶ The Research Pyramid Model of Research Evolution

*Publications in any scientific discipline are 'naturally' categorized into groups called 'research pyramids'.*

- ▶ A **research pyramid (RP)** : publications that belong to a most-specific research topic.
- ▶ Each RP has a pyramid-like structure in terms of the citation graph



# The Research-Pyramid Model

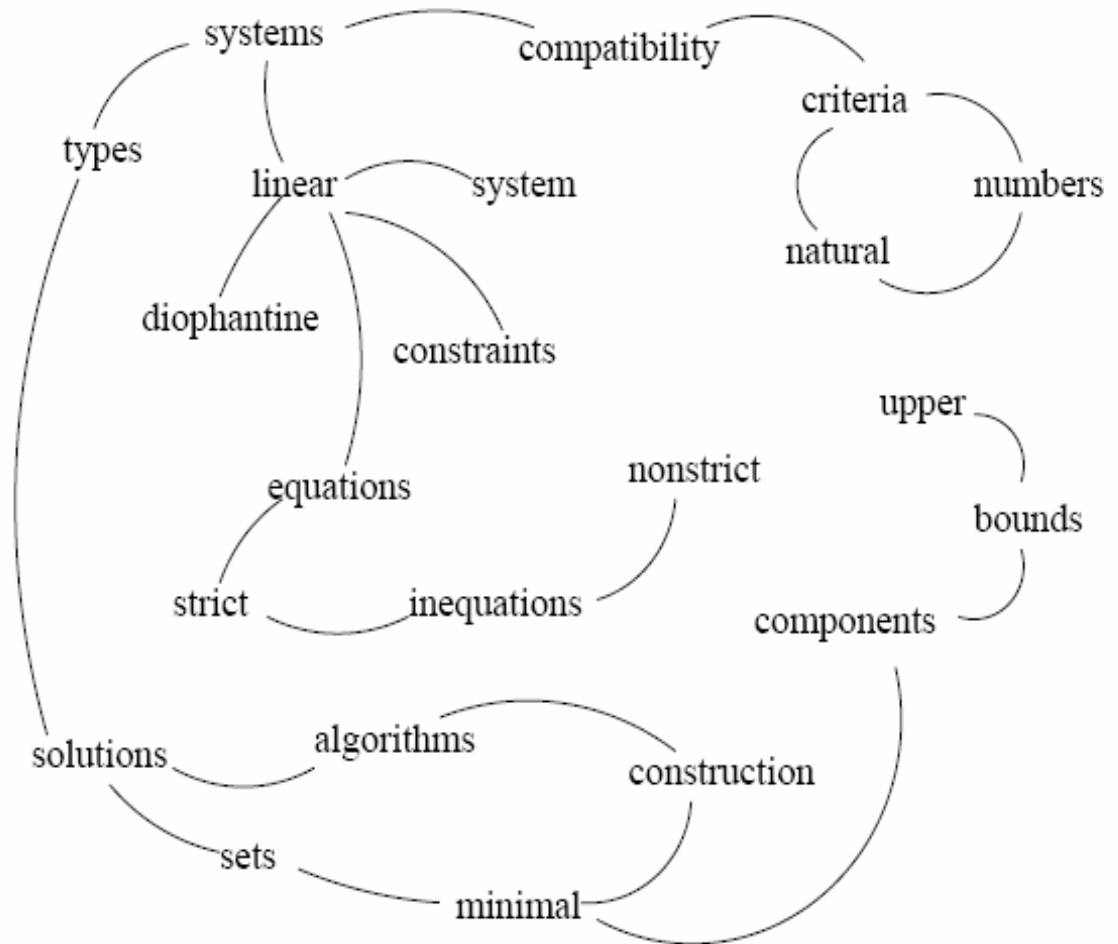
---

- ▶ The goal of each *user search session*:
  - ▶ find information about a specific topic.
- ▶ (The *locality of search* principle):
  - Within a single search session,  
the user targets documents  
within a specific topic (i.e., an RP).**
- ▶ SK suggestion scope
  - ▶ chosen as close to the topic being targeted as possible.
- ▶ As user enters more search keywords:
  - ▶ prune out topics (keywords) that are of low significance.
- ▶ Single token (word) significance => TextRank



# TextRank Algorithm

- ▶ Used to compute Topic-Sensitive Token and Phrase Weights.
- ▶ Vertices are **filtered** tokens (nouns and adjectives),
  - ▶ Syntactic filter
- ▶ Links represent linguistic/textual adjacency (in our case, they appear in the same title)
- ▶ Each graph for a research-pyramid is **topic-sensitive**.



# Advantages of Our Approach

---

- ▶ Eliminates the drawbacks of Google's search history-based SK-Suggester.
- ▶ Boosts the performance of the techniques used in the CompleteSearch.
- ▶ Has an excellent locality of access.



# Main Modules of the Content-Driven SK-Suggester

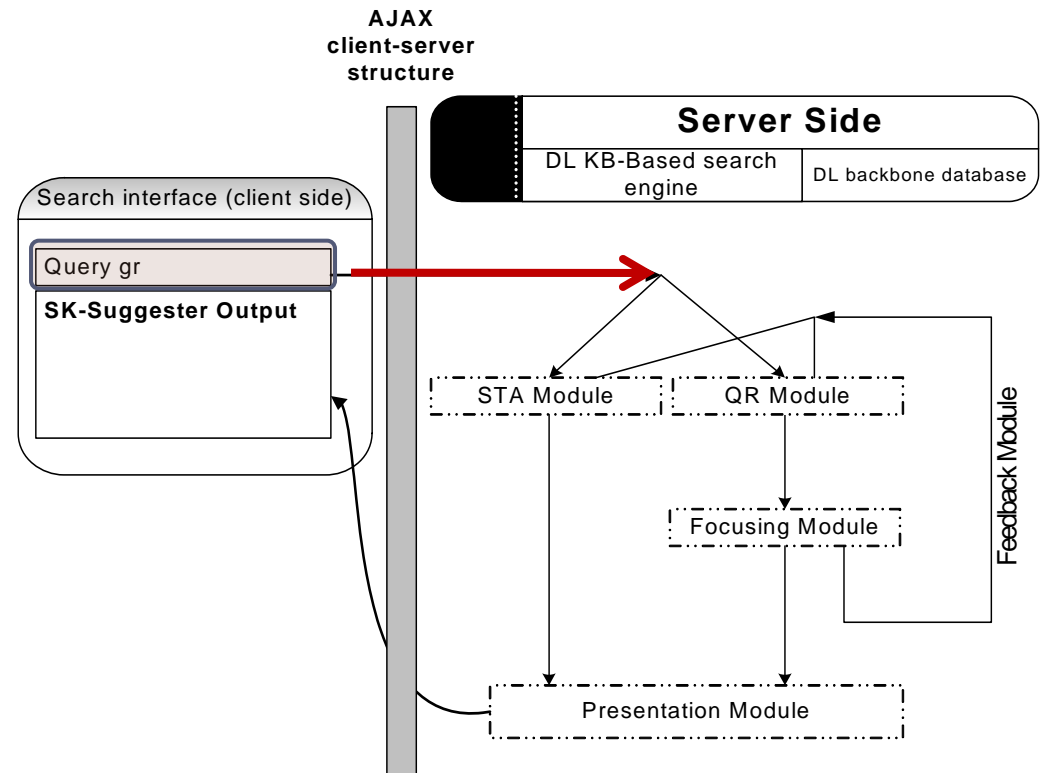
## Procedure *SK-SuggesterInterface* ()

### Input

User Input  $w$  : current search-terms

Server Input :  $R$ , and  $I$  ( stored in session status)

```
{  
(1) For  $w$   
  (1.1)  $LISK \leftarrow$  the uncompleted search keyword in  $w$   
  (1.2)  $CSK \leftarrow$  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 ( $CSK!="" \ \&\& \ LISK!=""$ )  
     $SK-List1 \leftarrow STA\_Module(LISK)$ ;  
     $SK-List2 \leftarrow QR\_Module(CSK, LISK)$ ;  
     $Join(SK-List1, SK-List2)$   
(2)  $Presentation\_Module(W')$ 
```



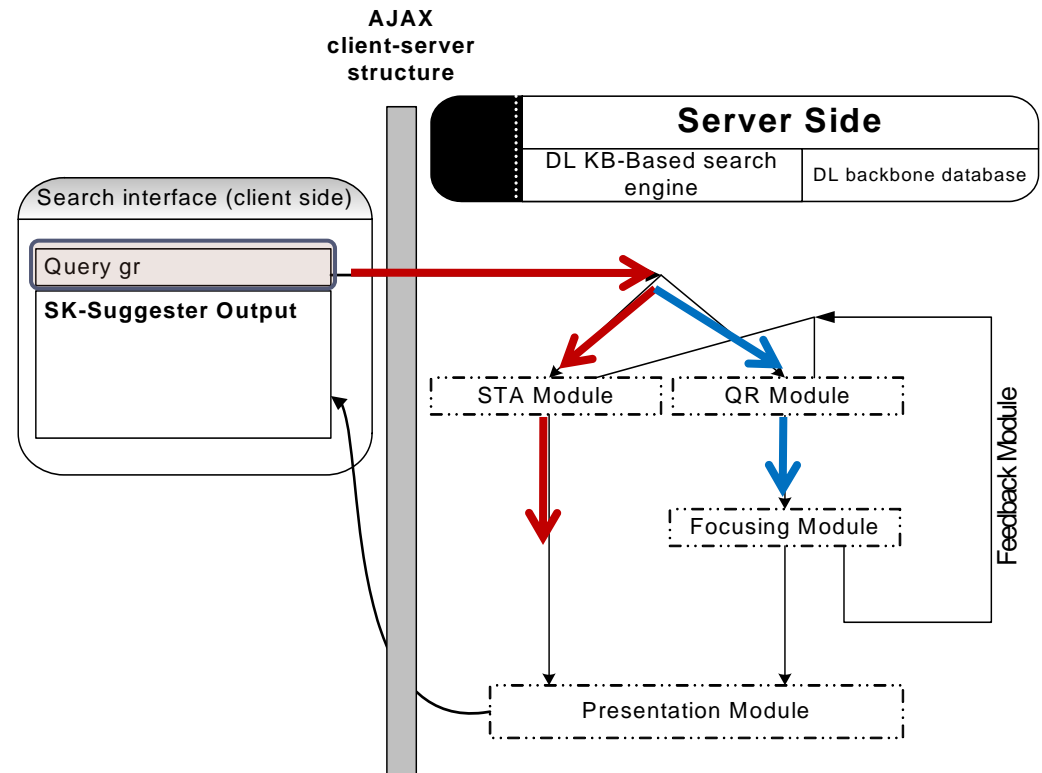
## Procedure *SK-SuggesterInterface* ()

### Input

User Input  $w$  : current search-terms

Server Input :  $R$ , and  $I$  ( stored in session status)

```
{  
(1) For  $w$   
  (1.1)  $LISK \leftarrow$  the uncompleted search keyword in  $w$   
  (1.2)  $CSK \leftarrow$  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 ( $CSK!="" \ \&\& \ LISK!=""$ )  
     $SK\text{-}List1 \leftarrow$  STA_Module(LISK);  
     $SK\text{-}List2 \leftarrow$  QR_Module(CSK, LISK);  
    Join(SK-List1, SK-List2)  
(2) Presentation_Module(W')
```



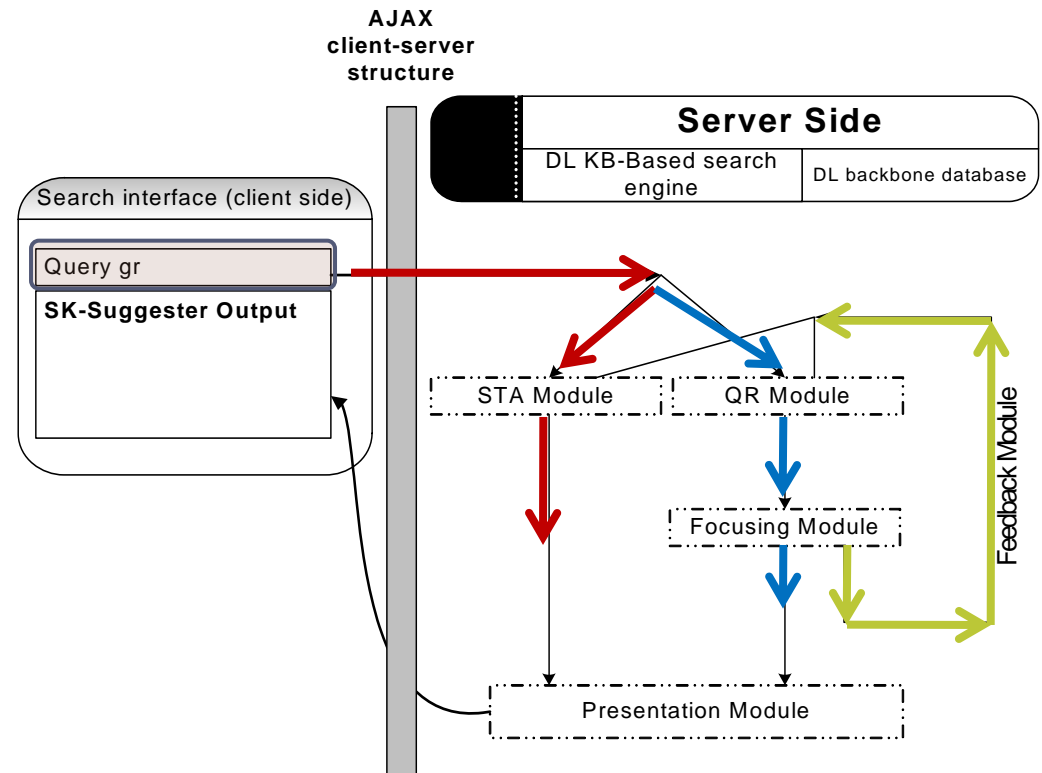
## Procedure *SK-SuggesterInterface* ()

### Input

User Input  $w$  : current search-terms

Server Input :  $R$ , and  $I$  ( stored in session status)

```
{  
(1) For  $w$   
  (1.1)  $LISK \leftarrow$  the uncompleted search keyword in  $w$   
  (1.2)  $CSK \leftarrow$  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 ( $CSK!="" \ \&\& \ LISK!=""$ )  
     $SK\text{-}List1 \leftarrow$  STA_Module(LISK);  
     $SK\text{-}List2 \leftarrow$  QR_Module(CSK, LISK);  
    Join(SK-List1, SK-List2)  
(2) Presentation_Module(W')
```



## 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***( $CSK$ ,  $LISK$ );

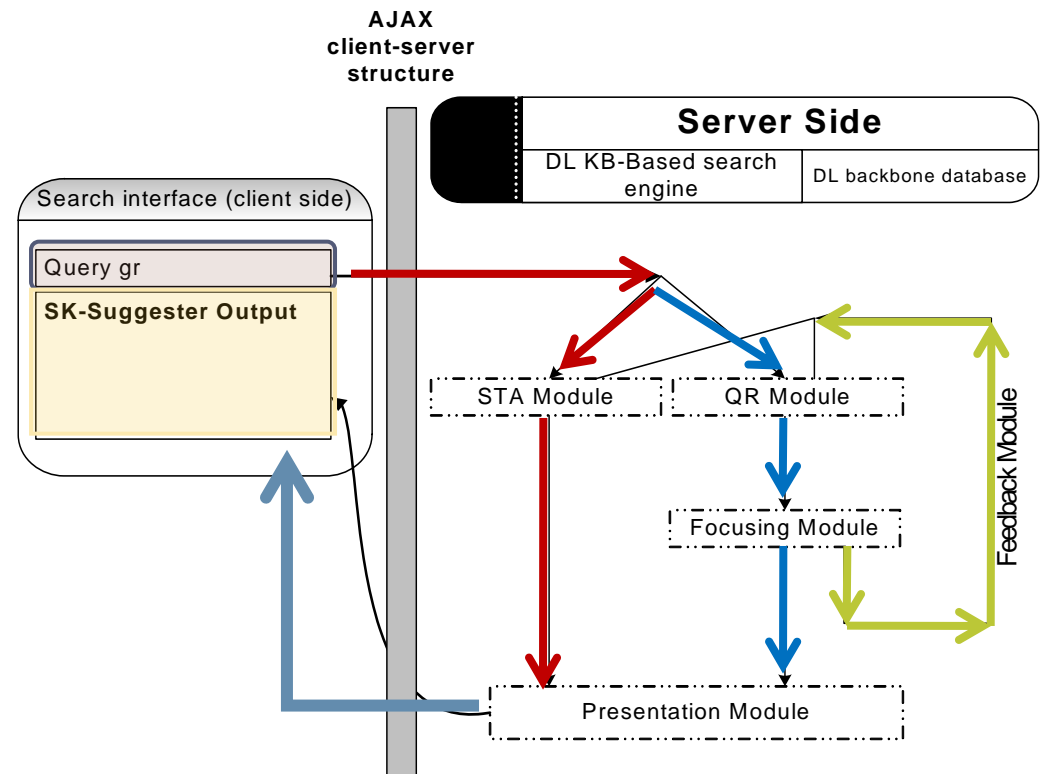
(1.5) Elseif ( $CSK!=""$  &&  $LISK!=""$ )

$SK\text{-}List1$  <- ***STA\_Module***(  $LISK$ );

$SK\text{-}List2$  <- ***QR\_Module***( $CSK$ ,  $LISK$ );

***Join***( $SK\text{-}List1$ ,  $SK\text{-}List2$ )

(2) ***Presentation\_Module***( $W'$ )



# Guiding statistics provided to the user online

---

## ▶ Suggestion Scope

- ▶ the number of research topics (or research pyramids) where the search keywords  $w$  are observed

## ▶ Topic-Sensitive Popularity of Search Keywords $TSP(W', r)$

- ▶  $W'$ : search terms
- ▶ the sum of TextRank scores of all words in  $W'$
- ▶ Query refinements ( $W'$ ): presented in the order of their **matching scores**:
- ▶  $Similarity(W', W)$ : text-based similarity between  $W'$  and the search terms  $W$

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



# Experimental Results and Observations

Accuracy of Linguistic Pre-processing and Quality of Suggestions

Scalability and Index Sizes

Convergence of Suggestion Scope

# Untagged 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

## The distribution of un-tagged tokens

- Untagged tokens are all stopwords.
- We use them
  - (i) to construct islands and
  - (ii) to connect linguistically adjacent islands.



# K-word Proximity Search

---

- ▶ The search keywords “query graph” are already identified as one island.
- ▶ Suggesting query refinements based on islands 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), which is called k-word ordered proximity search.

- (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**

Possible hits of the query “query graph”

---

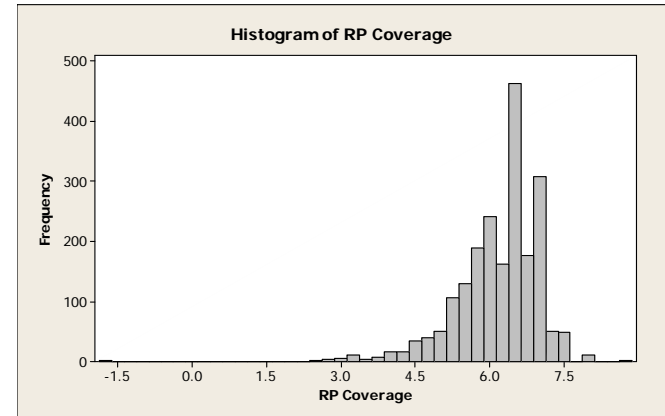


# Convergence of Suggestion Score

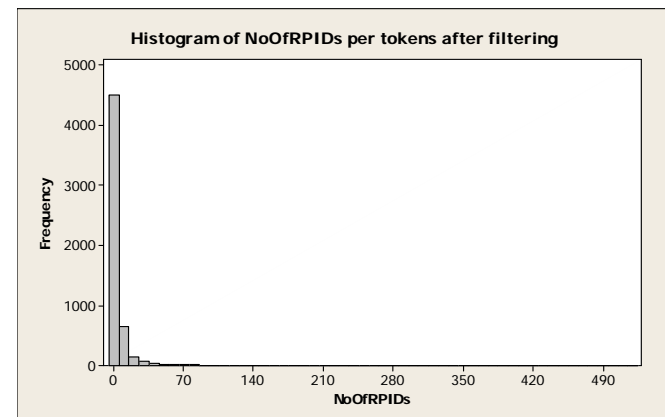
Coverage( $\mathbf{r}$ ) =  $-\log[(\# \text{ of tokens used in } \mathbf{r}) / (\text{total } \# \text{ of tokens})]$

- ▶ 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 to push refinements extracted from dominant research topic(s) up in the suggestion list.

- ▶ **Observation:** Filtered tokens have limited scope.



Distribution of RP coverage

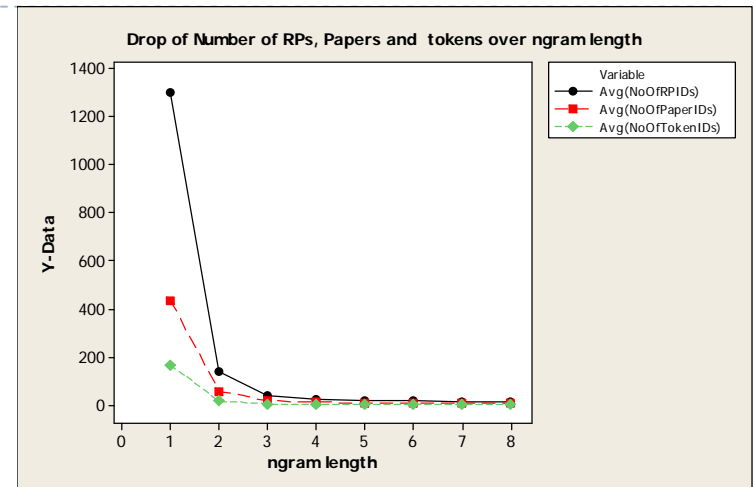


Distribution of token scope (after filtering)

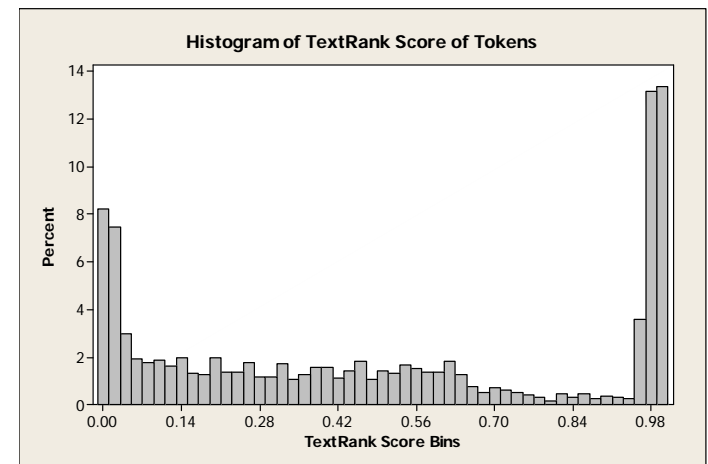


# Convergence of Suggestion Score

- ▶ Search-keyword suggestion can be optimized reducing suggestion scope
  - ▶ Suggestion scope converges fast.
- ▶ High TextRank => more significant tokens.
- ▶ Tokens that score high ( $>0.8$ ): content-bearing of the corresponding research pyramid.
- ▶ Low-scored tokens: widely used tokens.
- ▶ Order SK-suggester output
  - ▶ Significant refinements from dominant research pyramids.



Suggestion scope vs ngram length



Distribution of TextRank scores for simple tokens

# Future Work

---

- ▶ Our proposed approach can be used on any document collection given that
  - ▶ documents are clustered
  - ▶ highly topically related clusters.
- ▶ *Hybrid search-keyword suggester* (i.e. content-driven and search-history based).
  - ▶ Initially, no search-history => suggest document contents.
  - ▶ track users' search terms
  - ▶ Build search-history.
- ▶ **Why Hybrid SK-suggesters?** combine the advantages of both approaches
  - ▶ (i) accurate search-history keywords, less **incomplete** or **misspelled** words,
  - ▶ (ii) well-characterized queries. Keywords from successful searches
  - ▶ (iii) personalized search-keyword suggestions. *Which is an advantage of Google's search-history based suggester.*





Questions?

Project website:

<http://dmlab.case.edu:8080/giza/doku.php>