A MODULAR APPROACH TO DOCUMENT INDEXING AND SEMANTIC SEARCH

Dhanya Ravishankar, Krishnaprasad Thirunarayan, and Trivikram Immaneni
Department of Computer Science and Engineering
Wright State University, Dayton, OH-45435.
dhanyars@yahoo.com, t.k.prasad@wright.edu, timmanen@cs.wright.edu
http://www.cs.wright.edu/~tkprasad

ABSTRACT
This paper develops a modular approach to improving effectiveness of searching documents for information by reusing and integrating mature software components such as Lucene APIs, WORDNET, LSA techniques, and domain-specific controlled vocabulary. To evaluate the practical benefits, the prototype was used to query MEDLINE database, and to locate domain-specific controlled vocabulary terms in Materials and Process Specifications. Its extensibility has been demonstrated by incorporating a spell-checker for the input query, and by structuring the retrieved output into hierarchical collections for quicker assimilation. It is also being used to experimentally explore the relationship between LSA and document clustering using 20-mini-newsgroups and Reuters data. In future, this prototype will be used as experimental testbed for expressive, context-aware and scalable searches.

KEY WORDS
Search and Querying, Tools, Latent Semantic Indexing, Domain-Specific Search, Modular Search Engine, Document Clustering

1. Introduction

The state-of-the-art search engines (such as Google) provide a scalable solution for flexible and efficient search of Web documents, capturing collective Web "wisdom" to rank order the retrieved documents. This approach to search cannot always be expected to work well for documents pertaining to specialized domains where implicit background knowledge and vocabulary can be exploited to improve the accuracy of the retrieved results [1]. In general, precision can be improved through disambiguation, and recall can be improved by considering meaning preserving query variations [2][3][4].

Verbatim searches can be generalized in a number of directions such as by using information implicit in the English language and in the document collection. Eliminating stop words and affixes, proximity based searches, etc can capture semantic invariance due to word inflection and permutations, improving recall. English language synonyms can be used to improve recall, but including synonyms for all possible senses can adversely affect precision. Latent Semantic Analysis approach effectively regroups the document collection on the basis of occurrences of correlated words inferred from the document collection, so that some documents that lack the query words may be retrieved, and other documents that happen to contain the query words in a different context may be skipped [5][6].

In this paper, we investigate systematic generalization of keywords-based syntactic queries to concept-based semantic queries by utilizing linguistic information (such as synonyms) available explicitly, and domain-specific information (such as term correlations or associations) available implicitly in the document collections and explicitly through controlled vocabularies. Furthermore, it is important to locate and highlight the query hits in the context of a document in order to enable access to relevant portions of the document (because the user may not be aware of the automatically included context). In order to ensure that the search tool is efficient, flexible, and usable in practice, and extensible, customizable, and evolvable in future, mature software components have been employed in developing the infrastructure.

The content indexing and intelligent search tool discussed above has been put to novel use in performing domain-specific information extraction from documents (for example, Materials and Process Specifications), by exploiting it for semi-automatic mapping of document phrases to controlled vocabulary terms. That is, one can (i) determine all controlled vocabulary terms that can (partially) match a query phrase, (ii) determine all controlled vocabulary terms that appear in a document and locate the corresponding document phrases, and (iii) determine all partially matching controlled vocabulary terms that can potentially be extended to match a document phrase, deserving further human intervention for disambiguation.

To demonstrate the extensibility of the tool for improving user experience with respect to query input and display of query response, a spell-checker module and a simple technique for organizing search results into finer groups respectively has been incorporated. It is also being used to experimentally explore the relationship between LSA and document clustering. Specifically, the empirical relationship between the number of significant eigenvalues in the SVD decomposition and the number of document clusters is being studied. In future, this prototype will be used as experimental testbed for expressive, context-aware and scalable searches.

Section 2 provides architectural and implementation details of the content-based indexing and intelligent search prototype we have built after a brief review of the software components used. Section 3 describes the prototype, and evaluates its effectiveness. Section 4 concludes with potential future work.

2. Architectural and Implementation Details

The content-based indexing and semantic search prototype, shown in Figure 1, takes a document collection and parameters specified through configuration files, and carries out the following steps:

- Creates and maintains various indexes for the document collection for performing efficient searches using Lucene. The inverted indexes are also used for generating term-document space based on LSA using JAMA library.
- For a user query and options, performs syntactic phrase matches accommodating morphological processing using Porter stemmer, wildcard pattern matches, boolean queries, query expansion using synonyms for search terms provided by WordNet via JWNL library, executes proximity queries, and uses LSA techniques to determine relevant documents.
- Highlights or tags the search results in the original document and displays excerpts of the matches to the end user.

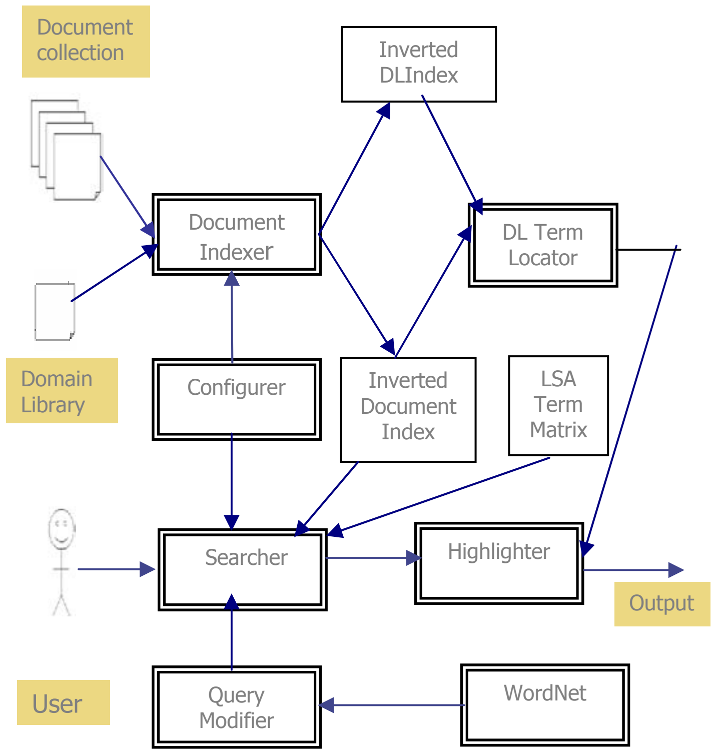
The prototype has also been employed, among other things, to index and search domain-specific controlled vocabulary of technical terms presented as XML files, to assist in semi-automatic content extraction from materials and process specifications. The implementation details of the various components of the prototype follows.

2.1 Lucene Overview

Lucene, provided by Open Source Apache project, is a Java API for indexing and searching text documents that can be tailored and integrated into applications [7]. Lucene builds inverted indexes for the documents to be searched and stores statistics about the terms contained in the text. The document text is pre-processed by Analyzers to extract indexable tokens. Lucene supports case conversions, stop words elimination, and performing other input modifications. In order to facilitate searching of the indexes, Lucene provides `IndexSearcher` and `QueryParser` interfaces that translate Google-like search expression to Lucene's API representation of the query.

Overall, Lucene is a high-performance, text search engine library with smart indexing strategies that can be used in a wide range of applications because of its flexibility.

Figure 1. Architecture of Content-based Indexing and Semantic Search Engine



2.2 Java WordNet Library

WordNet is an electronic lexical database that organizes English nouns, verbs, adjectives, and adverbs into synonym sets, each representing one underlying lexicalized concept [8]. Polysemous word forms are those that appear in more than one synset, therefore representing more than one concept. Associated with every word form is a count of the number of senses that the word form has when it is used as a noun, verb, adjective, or adverb.

`JWNL` is an API for accessing WordNet style relational dictionaries [9]. It provides functionality beyond data access, such as relationship discovery and morphological processing.

2.3 JAMA Library

`JAMA` is a basic linear algebra package for Java developed by MathWorks and NIST [10]. Specifically, it provides five fundamental matrix decompositions, namely, Cholesky Decomposition of symmetric, positive definite matrices; LU Decomposition (Gaussian elimination) of rectangular matrices; QR Decomposition of rectangular matrices; Eigenvalue Decomposition of both symmetric and nonsymmetrical square matrices, and Singular Value Decomposition (SVD) of rectangular matrices.

The SVD computation on term-document matrix for Latent Semantic Analysis is obtained through the use of JAMA library in the prototype.

2.4 Configurer and DocumentIndexer

`Configurer` object provides initial settings information such as paths to the data directory containing the documents to be indexed, controlled vocabulary file, the index directories, the location of the property file for initialization of JWNL package used with WordNet, and output path for creating copy of the retrieved documents with the search results highlighted/tagged.

`DocumentIndexer` uses values specified by the `Configurer` and maintains various indexes using different analyzers such as standard analyzers, stemmer analyzers, synonym analyzers, etc. Initially, synonym analyzers were used to generate synonyms of document words as aliases at the time of tokenization. This turned out to be very space and time consuming, and was avoided in the final prototype.

2.5 LSA Term Document Matrix Generator

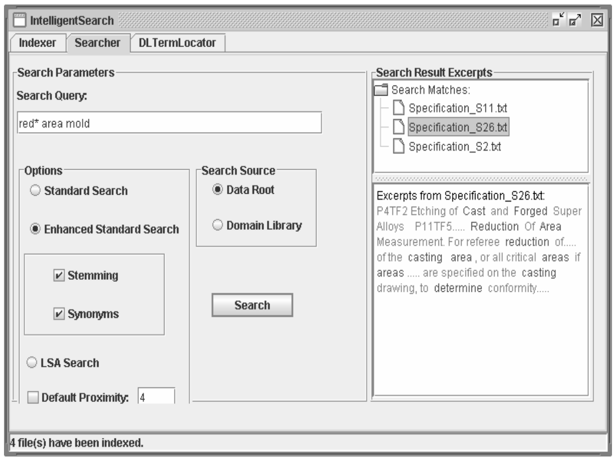
The matrix generator uses the indexing statistics obtained from the `DocumentIndexer` to generate the term weights in the term-document matrix for performing Latent Semantic Analysis of the document collection. TDM is analyzed by singular value decomposition to derive a latent semantic structure model. Because of the computational complexity of SVD, the matrices are made persistent by storing them along with the document indexes, to enable multiple search queries on the document collection.

2.6 Searcher and Query Modifier

`Searcher` object matches queries to either the document collection or the controlled vocabulary (also called the domain library). It uses Lucene APIs to translate user queries to Lucene's internal query representation. Depending on the data source, it initiates the search on the indexes stored in directories specified by the `Configurer`.

The `Searcher`, similar to the `DocumentIndexer`, makes use of the corresponding analyzer for different kinds of search. It also accepts options to change the default proximity values to be used with phrasal searches. For synonym-enabled searches, `Searcher` allows synonym expansion for explicitly selected query terms or all of the terms if no specific term has been selected. See Figure 2. A new operator '#' has been introduced and used as a prefix to specify the terms to be expanded with aliases at the time of search. Searcher is also responsible for determining relevant documents based on the LSA approach. It makes use of the term-document matrix, SVD matrices generated by LSA Matrix Generator and the term-weighted query vector corresponding to the user query to compute cosine similarity of the query with the documents in the new vector space (`k`-value = 100).

Figure 2. Enhanced search illustrating wildcard pattern and synonym expansion



Search Query: red* area mold

Options: Standard Search [✓ Enhanced Standard Search] LSA Search

[✓ Stemming] [✓ Synonyms]

Default Proximity: 4

Search Source: [✓ Data Root] Domain Library

Search Matches:
- Specification_S11.txt
- Specification_S26.txt
- Specification_S2.txt

Excerpts from Specification_S26.txt:
P4TF2 Etching of Cast and Forged Super Alloys P11TF5..... Reduction Of Area Measurement. For referee reduction of..... of the casting area , or all critical areas if areas ..... are specified on the casting drawing, to determine conformity.....

4 file(s) have been indexed.

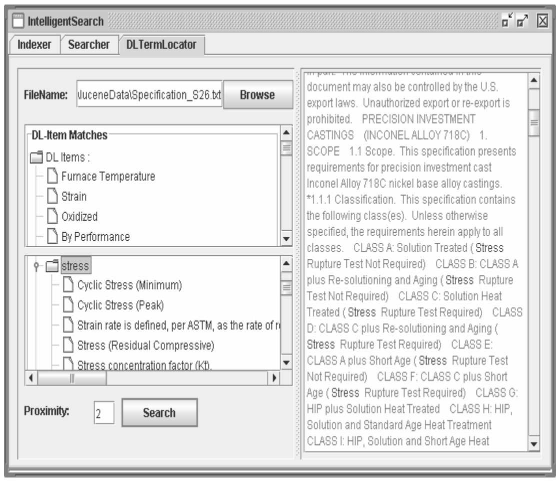
`Query Modifier` objects produce queries in Lucene's internal query format and are used in conjunction with searches to obtain altered queries that match the user specifications such as to incorporate proximity value changes, synonym expansion using WordNet, etc. In the prototype, if the number of synonyms is large, `Query Modifier` uses heuristics to select synonyms to be used for query expansion based on frequency of usage. It also incorporates user input proximity values for phrase queries.

2.7 DL Term Locator

This is a novel application of indexing and search tool infrastructure in the context of content extraction from materials and process specs, and provides an interesting alternative to the customized tools we developed laboriously [11].

A spec describes requirements on the processing of a material (alloy) in the mill, and the capabilities that the alloy should possess eventually. A domain library is a domain-specific controlled vocabulary and consists of a set of domain library items. Each item is a sequence of terms. Specification Definition Representation (SDR), together with the domain library, constitutes an ontology to articulate the semantic view of the components that comprise a spec, and capture user's interpretation of the spec. Computer assisted content extraction involves semi-automatic recognition of phrases in spec that are associated with requirements on an alloy, and subsequent synthesis of the SDR fragments, to assist an extractor. Content extraction requires formalizing spec phrases using a domain library. Given that, in general, a spec does not conform to any prescribed vocabulary, a viable semi-automatic approach is to recognize and locate as many phrases as possible that can map to unique domain library item, and for those phrases that can only partially match domain library items, generate all possible candidates for manual disambiguation. In practice, this approach can improve the quality and efficiency of the laborious manual extraction task by automating some of the routine mechanical tasks.

Figure 3. Matching DL Items; DL Term and its location in the document



FileName: \luceneData\Specification_S26.txt

DL-Item Matches
DL Items :
Furnace Temperature
Strain
Oxidized
By Performance

stress
Cyclic Stress (Minimum)
Cyclic Stress (Peak)
Strain rate is defined, per ASTM, as the rate of r
Stress (Residual Compressive)
Stress concentration factor (Kt).

Proximity: 2

document may also be controlled by the U.S. export laws. Unauthorized export or re-export is prohibited. PRECISION INVESTMENT CASTINGS (INCONEL ALLOY 718C) 1. SCOPE 1.1 Scope. This specification presents requirements for precision investment cast Inconel Alloy 718C nickel base alloy castings. *1.1.1 Classification. This specification contains the following class(es). Unless otherwise specified, the requirements herein apply to all classes. CLASS A: Solution Treated (Stress Rupture Test Not Required) CLASS B: CLASS A plus Re-solutioning and Aging (Stress Rupture Test Not Required) CLASS C: Solution Heat Treated (Stress Rupture Test Required) CLASS D: CLASS C plus Re-solutioning and Aging (Stress Rupture Test Required) CLASS E: CLASS A plus Short Age (Stress Rupture Test Not Required) CLASS F: CLASS C plus Short Age (Stress Rupture Test Required) CLASS G: HIP plus Solution Heat Treated CLASS H: HIP, Solution and Standard Age Heat Treatment CLASS I: HIP, Solution and Short Age Heat

Domain Libraries (DL), which have evolved over a decade or so and contain roughly 10,000 items, exist as XML files. `DocumentIndexer` indexes the contents of the DL after parsing these XML files using Xerces XML parser. `DL Term Locator` has the responsibility to automatically identify words/phrases in spec that also appear as DL terms/items. In other words, the `DL Term Locator` tags explicit occurrences of the DL Items in a given document. In addition, it also lists spec words that cannot be directly matched to any of the DL items, but which nevertheless occur in some of the DL items, along with the DL items wherein they appear. See Figure 3. (Recall that, in contrast, `Searcher` accepts an arbitrary phrase and tries to match it with the DL items that have been indexed.)

`DL Term Locator` component makes use of the DL Index and the document/spec index to obtain terms common to both. It also determines the DL-items wherein these terms appear and extracts those items from DL for further analysis. The probable set of items thus retrieved is searched for in the document index based upon some user specified proximity measures to determine approximate matches. The matches obtained for DL terms are returned along with the items wherein they appear, and explicitly matching DL items are returned as a list for the tool to appropriately present to the user.

2.8 Match Highlighter

If the searcher finds hits in the document collection for a specific query, then a `Highlighter` creates a file containing all the matched terms of the query properly highlighted (actually HTML bold faced) for user's convenience. It also displays excerpts from the matched passages in the "hit" documents.

Domain Library matches are treated slightly differently by the `Highlighter` because of the size of the DL files. For DL matches, the output file created contains only the matching DL items instead of reproducing the original DL file with the matches tagged. `DL Term Locator` also highlights occurrences of the DL items and the terms in the document.

3. Experimental Evaluation

The overall approach is evaluated on query expressiveness, efficiency, and effectiveness, and on modularity through extension and reuse.

3.1 Query Effectiveness

Searches were generalized both syntactically and semantically as evidenced from our experiments with materials and process specifications, and the MEDLINE collection:

(i) Syntactic variations (e.g., stemming): "Test certificate" query matched document phrases such as "certificate of test", "test certification", etc. Similarly, "dia*" matched "dia", "diameter", etc, "acc* level quality" matched "Acceptable level of quality", etc.

(ii) Semantic invariance (e.g., using synonyms): "Tensile strength" query matched the document phrase "ductile force", "part number" matched "part and lot number", "mold" matched "cast", "castings", "forge", and "forging", etc. "Insufficient immunity" matched "immune deficiency", "causes cancer" matched "induces cancer", "reasons for cancer" etc.

In order to compare the impact of LSA, the prototype was tested for all searches with the MEDLINE collection. Because of resource constraints, we used about 5000+ index terms from 500 documents in the collection. A 100-factor SVD of the above matrix was obtained and stored for later searches. It took about 12 minutes (Wall Clock time) to generate the index on a Pentium 4 machine (2.53 GHz CPU, 1GB memory) running Windows XP.

Table 1 lists the values for recall and precision obtained for some of the typical queries for enhanced search and LSA enabled search respectively. LSA-based searches consistently produced search results with better precision as compared to standard and enhanced searches. Recall was, however, generally higher for enhanced searches (with accompanying steep decline in precision). The work reported in [13] provides further analysis of the efficacy of the LSA techniques on large document collections.

Table 1. Recall and Precision on MEDLINE collection with Different Search Strategies

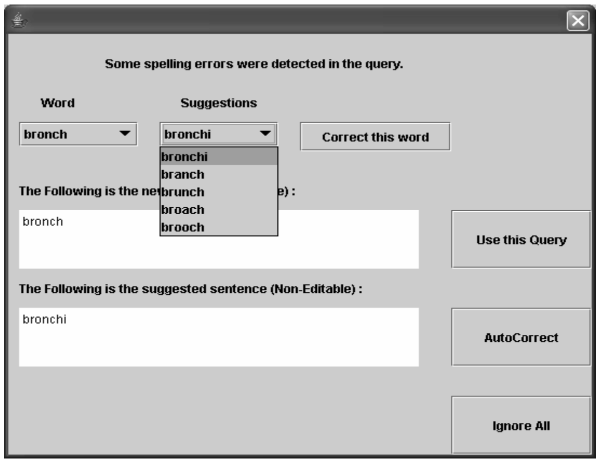
	Enhanced Search		LSA Search	
Query	Recall	Precision	Recall	Precision
"electron microscopy of lung or bronchi"	0.86	0.2	0.91	0.5
"the crossing of fatty acids through the placental barrier. normal fatty acid levels in placenta and fetus"	0.96	0.08	0.85	0.63
"the use of induced hypothermia in heart surgery, neurosurgery, head injuries and infectious diseases."	0.96	0.07	0.82	0.3
"bacillus subtilis phages and genetics, with particular reference to transduction."	1.0	0.12	**0.95**	**0.83**

3.2 Modularity through extension and reuse

Ideally, the documents retrieved as a result of a search should be grouped on the basis of word senses so as to improve precision via group labels. Given that there is no simple way of determining or expressing word senses, as a conservative approximation, we explored grouping of documents on the basis of synonyms and labeling the group with the synonym its members contain.

In order to improve query entry capability and organize the results of a query, two interesting enhancements were incorporated into the indexing and search tool. First, the input query was processed through Jazzy, a Java Open Source Spell-Checker [14], as shown in Figure 4. Second, the flat list of retrieved documents was grouped in a hierarchy based on the synonym found, so as to be able to peruse or ignore an entire sub-group of documents (to better approximate the relevant word sense) as shown in Figures 5. The search for "deficiency" generates hits for documents containing "lack", "deficiency", "want" and "insufficiency" when experimenting with the MEDLINE collection. The word "want" seems a bit remote in this query context, and hence, the documents within the folder labeled "want" can be skipped from consideration.

Figure 4. Spell-checking input dialog



Some spelling errors were detected in the query.

Word: bronch

Suggestions: bronchi

bronchi
branch
brunch
broach
brooch

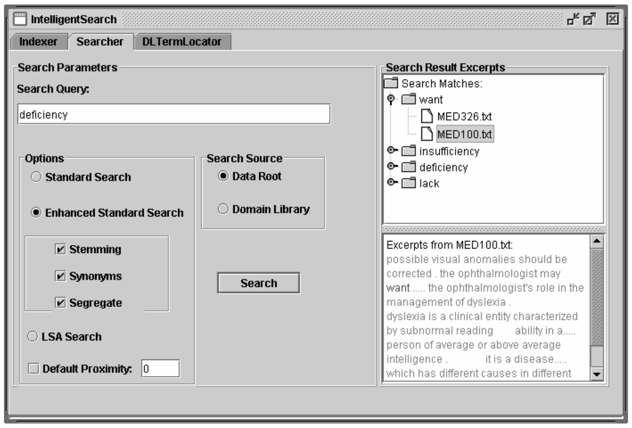
The Following is the ne…e) :

bronch

The Following is the suggested sentence (Non-Editable) :

bronchi

Figure 5. Grouping retrieved results



Search Query: deficiency

Options: Standard Search [✓ Enhanced Standard Search] LSA Search

[✓ Stemming] [✓ Synonyms] [✓ Segregate]

Default Proximity: 0

Search Source: [✓ Data Root] Domain Library

Search Matches:
want
MED326.txt
MED100.txt
insufficiency
deficiency
lack

Excerpts from MED100.txt:
possible visual anomalies should be corrected . the ophthalmologist may want ..... the ophthalmologist's role in the management of dyslexia .
dyslexia is a clinical entity characterized by subnormal reading ability in a..... person of average or above average intelligence . it is a disease..... which has different causes in different

3.3 LSI and Clustering

Under the assumption that a dataset contains clearly defined document clusters, the number of document clusters and the number of significant singular values of the term document matrix are correlated [15][16]. To explore its applicability to the text clustering benchmarks, we experimented with the 20-Newsgroups and Reuters-215781 newswire stories datasets [17]. Due to the heap space restrictions imposed on the Java runtime by the available 1G main memory, only 2K documents could be used from each dataset at a time. The 20-Mini-Newsgroups has postings from 20 Newsgroups (including cross-postings), while each Reuters' sub-collection has documents belonging to around 70 topics. With the exception of filtering the body of each Reuters' document, no other clean-up was performed. Figures 6 and 7 show the general distribution of the normalized singular values (w.r.t. the maximum singular value) as a function of the dimensions. The singular values seem to fall off rather precipitously, reducing to a seventh and a fifth of its initial value respectively around the dimension equal to the number of document clusters.

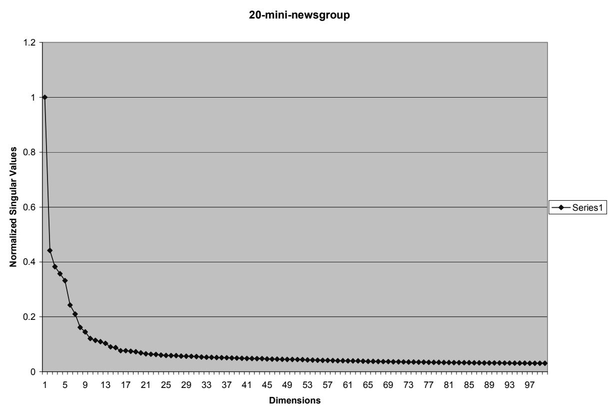


Figure 6. 20-Mini-Newsgroup dataset results

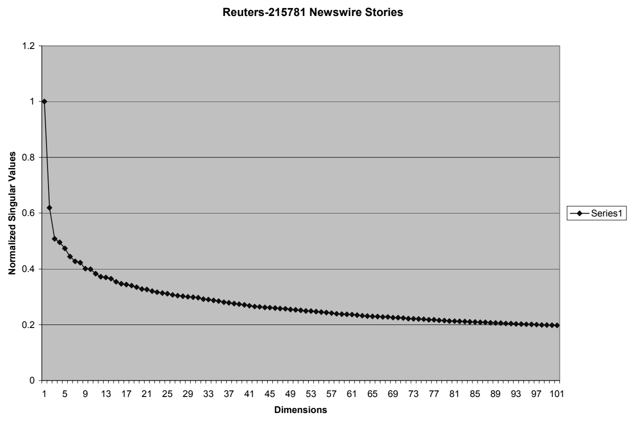


Figure 7. Reuters-21578 newswire dataset results

4. Conclusion

The prototype enhanced query effectiveness through improvements in recall and precision of search results, achieved by incorporating the language background through WordNet and exploiting document collection characteristics through LSI. It also provided partial assistance in content extraction from documents that involve use of controlled vocabularies. Overall, our approach and the implemented infrastructure can form the basis for creating expressive search and query tools underlying IE/IR systems, and for comparing existing techniques [12]. As to potential immediate enhancements, one can filter synonyms of the appropriate word sense based on the domain-specific context, and incorporate antonyms into queries and domain-specific background information into LSI.

5. Acknowledgements

We thank Nalini Pitchika for enlightening discussions.

References:

[1] T. Haveliwala, A. Gionis, Klein, and P. Indyk. Evaluating Strategies for Similarity Search on the Web, *Proceedings of The Eleventh International WWW Conference*, Hawaii, May 2002.

[2] D. Ravishankar, *Document Indexing and Semantic Search*, M.S. Thesis, Department of Computer Science and Engineering, Wright State University, 2004.

[3] Karen Sparck Jones and Peter Willett. *Readings in Information Retrieval.* Morgan Kaufmann, 1997.

[4] R. Baeza-Yates and B. Riberiro-Neto, *Modern Information Retrieval*, Addison-Wesley-Longman. 1999.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, 41(6), pp. 391-407, 1990.

[6] Latent Semantic Analysis URLs: http://javelina.cet.middlebury.edu/lsa/out/lsa_definition.htm, http://lsa.colorado.edu/, http://www.cs.utk.edu/~lsi/ (Visited: April 14, 2005)

[7] Lucene URL: http://jakarta.apache.org/lucene/docs/index.html (Visited: April 14, 2005).

[8] WordNet URL: http://www.cogsci.princeton.edu/~wn/ (Visited: April 14, 2005)

[9] Java WordNet Library URL: http://sourceforge.net/projects/jwordnet/ (Visited: April 14, 2005)

[10] JAMA Library URL: http://math.nist.gov/javanumerics/jama/ (Visited: April 14, 2005).

[11] K. Thirunarayan, A. Berkovich, and D. Sokol, An Information Extraction Approach to Reorganizing and Summarizing Specifications, In: *Information and Software Technology Journal*, Vol. 47, Issue 4, pp. 215-232, 2005.

[12] P. Husbands, H. Simon, and C. Ding: On the Use of Singular Value Decomposition for Text Retrieval, *Proc. of SIAM Comp. Information Retrieval Workshop*, 2000.

[13] G. Dupret: Latent concepts and the number orthogonal factors in latent semantic analysis, *Proceedings of SIGIR-2003*, pp. 221-226, 2003.

[14] Jazzy Spell-Checker URL: http://jazzy.sourceforge.net/ (Last visited: April 14, 2005).

[15] C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis. *Proceedings of the ACM Conference on Principles of Database Systems (PODS)*, pp. 159-168, 1998.

[16] Y. Azar, A. Fiat, A. Karlin, F. McSherry and J. Saia, Spectral Analysis of Data. *Proceedings of the ACM Symposium on Theory of Computing,* pp. 619–626, 2001.

[17] http://kdd.ics.uci.edu/summary.data.application.html (Visited: April 14, 2005)