Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1992

Implementation of an Information Retrieval System (ANIRS) with Ranking and Browsing Capabilities

Fazli Can* Kevin McCarthy†

*Miami University, commons-admin@lib.muohio.edu

†Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-001

Implementation of an Information Retrieval System (ANIRS) with Ranking and Browsing Capabilities

Fazli Can and Kevin J. McCarthy

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

IMPLEMENTATION OF AN INFORMATION RETRIEVAL SYSTEM (ANIRS) WITH RANKING AND BROWSING CAPABILITIES

by

Fazli Can Kevin J. McCarthy
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-001 April, 1992

Implementation of An Information Retrieval System (ANIRS) With Ranking and Browsing Capabilities

Fazli Can Kevin J. McCarthy

Department of Systems Analysis

Miami University

Oxford, OH 45056

April 1992

Abstract

This report describes an implementation of a cluster based information retrieval system with statistical ranking facilities, ANIRS. ANIRS uses the vector space model to represent the document database. In this model, the database is defined by a document by term, D, matrix. In this matrix, each row represents the terms in a single document and each column represents the documents that contain a single term.

In ANIRS, two matching methodologies are allowed: a full database search and a cluster based search. The system uses a natural language query interface. It incorporates suffix stripping for term conglomeration. Two methods of query refinement are used: relevance feedback and document seed searching. Cluster browsing, the ability to look at all the documents in a single cluster, is also implemented.

ANIRS was written in Pascal on an IBM VM/CMS environment.

Key words: Information retrieval, clustering, query matching, statistical ranking, browsing.

1. INTRODUCTION

The power of all information is limited by the ability of people to find it. Every day, more and more facts, articles, journals, and other publications come into existence -- all for the sake of distributing information. Without the ability to organize and access this information, it is useless. With the advent of computers, more powerful and advanced methods for storing and retrieving information became possible, such as CD-ROM storage. Information retrieval is a rapidly growing field of research in the area of computing.

This paper is a description of one such prototype information retrieval system, ANIRS, or AN Information Retrieval System. This is a medium to large scale system incorporating some of the more recent ideas in information retrieval.

The overall organization of this report is as follows. Section 2, following this introduction, is a discussion of the query-database matching methodology choices presented by ANIRS. Section 3 discusses the specific features incorporated into ANIRS. Section 4 discusses its actual implementation, getting into the specifics of the programming language and operating system details.

2. MATCHING METHODOLOGIES

2.1 Statistical Ranking/Clustering

ANIRS is an implementation of a cluster based information retrieval system with statistical ranking facilities. When a query is matched against the database, the output is a set of documents, sorted by a generated similarity value. For more information on similarity values, see (5,6).

There are several term weighting approaches ANIRS allows the user to choose from. It allows the choice of a Term Frequency Component (TFC), Collection Frequency Component (CFC), and a Normalization Component (NC). For a detailed description of term weighting approaches see (5).

ANIRS allows the user to employ document-cluster matching instead of a full database search. Using the cover-coefficient based clustering methodology (1,2), the database is partitioned into clusters. Each cluster is a group of documents that are statistically similar. Each cluster is represented by a centroid -- the average document for the cluster. Instead of searching the entire database, relevant clusters are first selected and then searched.

2.2 D Matrix/Inverted Term Lists

The entire database is represented by a D Matrix, or document matrix. The rows of the matrix are documents -- each row is a separate document. The columns of the matrix are terms in the database. The intersection of each column and row is an integer value representing the number of occurrences of the term in the document.

The D-matrix can be sparse for most databases, resulting in a lot of wasted storage for zeroes. Thus, two other equivalent representations of the data are used in ANIRS: document vectors and inverted term lists.

Each document vector, instead of containing an entry for each term, stores only term numbers with frequency greater than zero, along with the frequency value. For a sparse matrix, this results in significant storage reduction.

Another representation used in ANIRS are inverted term lists. For each term in the database, a sorted list of documents and the frequency of the term in each document are stored for documents with term frequency greater than zero.

2.3 Implementation of Document Searching

There are two matching methodologies used in ANIRS, full search and cluster based search. The actual implementation of each is described below.

2.3.1 Full Search

A full search loops through each term in the query. For each term, it loads its associated inverted term list. For each document in the inverted term list, it adjusts the frequency of the term in the query and document based of the chosen term weighting function (5). It then multiplies these two weights together and stores the value associated with that document. Subsequent values generated with other terms are added to the previous value and stored there. After each term has been run through, the documents are sorted by the stored similarity value and returned as relevant documents. The document with the highest similarity value is considered to be the most relevant document.

2.3.2 Cluster Search

In a cluster search, a full search is first performed on a centroid D-matrix -- each centroid represents the average document in the cluster. The centroids are then sorted in descending order by similarity value and the first x clusters

(the number x being controlled by the user) are chosen as relevant clusters.

The surprising part is that a full search is then performed on the entire database -- returning a list of documents sorted by similarity. Each document in the list is then filtered as to whether it belongs to a relevant cluster or not. A current study shows this to be the fastest cluster retrieval method, despite searching the entire database (3).

3. SYSTEM FEATURES

3.1 Databases

ANIRS is currently configured to use two databases: TODS and INSPEC. The TODS database contains 322 documents and 2602 terms. It is partitioned into 46 clusters.

The INSPEC database contains 12,684 documents and 14,573 terms. It is partitioned into 475 clusters.

3.2 Suffix Stripping

ANIRS uses a natural language query interface. The user types in key terms and phrases relevant to desired documents.

The database then searches for terms in the database. Each term is assigned a weight of 1 for each occurrence in the query.

Each database first uses a suffix stripping algorithm to conglomerate variations of terms. For example, "computer" "computers" and "computation" might all be stripped into a single form: "comput". This leads to a large reduction in the number of terms stored in the D-matrix at the expense of some precision.

The TODS database employs the Porter's suffix stripping algorithm. For a detailed description of Porter's algorithm, see (4). After the term was suffix stripped, a binary search was employed on the key term list to see if the term was in the database.

The suffixing algorithm for INSPEC was not known during the construction of ANIRS. Therefore, a closest match algorithm was employed. No suffixing takes place on the query word. Instead, a binary search is performed on the key word list using the whole term. If an exact match is found, that term is used. Otherwise, we backtrack through the list until a match of at least two characters is found or until no match is possible. A match is defined as the largest prefix substring of the query word. For example, if the query word were "mathematics" and the term list contained "thematics", "math", and "mat", a the match would be "math" because it is the largest prefix of mathematics.

"thematics" is a longer match, but is not a prefix of "mathematics".

Once the query terms have been located. ANIRS allows the user to add, delete, and modify the weight of the found query terms. The user may also look at terms nearby a term. This compensates for suffixing and the closest match algorithm, which don't always find the best term.

3.3 Query Refinement

Once selected documents have been returned, the user may wish to refine and reperform a query. Two methods are provided by ANIRS for this: relevance feedback and document seed search.

3.3.1 Relevance Feedback

The Ide dec hi method of relevance feedback, as document by Salton (6), was used. In this method, one document is chosen by the user as a nonrelevant document. Note that if no documents are nonrelevant, then why refine the query? Then any number of documents may be selected as being relevant. The relevant documents terms and term weights are added to the original query vector. Up to 50 terms are allowed to be added. These are sorted by term occurrence, and the 50 most common terms are added. All of the nonrelevant documents term weight are

subtracted from the query vector. Negative term weights are not allowed. Then a query is performed using the modified query vector.

The selected relevant and nonrelevant documents are automatically filtered from the matched documents.

3.3.2 Document Seed Search

An alternate way to redefine a query is using document seed searching. The user selects a single document that is highly relevant. This document's document vector is then used to query the database.

3.4 Cluster Browsing

The database, as previously explained, is partitioned into clusters of documents that are statistically similar.

The user may choose to browse the cluster of any retrieved document. This provides some expansion of recall ability, as not all documents in a cluster are relevant, but they are related in ways not always accessible through a query.

4. IMPLEMENTATION NOTES

ANIRS is implemented using VS Pascal on an IBM VM/CMS environment. This provides the opportunity for many students to view the code in addition to experimenting the system, since Pascal is a commonly known programming language

To implement database features, direct access files were used. Commands such as seek , get, and @ (file buffer reference) could then be used for speedy results.

Variable length records could not be implemented. Therefore, to allow for different length D-matrix records, each document was broken into records to 15 terms each. Indexes were used to point to the beginning of each document's set of records.

For suffix stripping algorithm implementation, IBM VS Pascal's string manipulation abilities were used, such as SUBSTR, DELETE, LTRIM, and TRIM.

VM/CMS's filedefs were used to link file variables to physically files. This allowed the same file variables to link to different databases. Therefore, to link a new database to ANIRS, new variables need not be coded in. The program must be exited and run again to use a different database.

5. CONCLUSION

ANIRS is a medium to large scale prototype information retrieval system. The primary features incorporated into ANIRS are as follows:

1) Statistical ranking facilities
2) Clustered databases
3) Full or cluster database searching
4) Natural language query interface
5) Suffix stripping term conglomeration
5) Query refinement through relevance feedback and document seed searching
6) Cluster browsing

The system currently uses the TODS and INSPEC databases, but can accommodate databases of any size.

References

1. Can, F., Ozkarahan, E.A. "Concepts and Effectiveness of the Cover-Coefficient-Based Blustering Methodology for Text Databases." ACM Transactions on Database Systems 15, 4 (Dec., 1990), 483-517.

2. Can, F. "Incremental Clustering for Dynamic Information Processing." ACM Transactions on Information Systems

3. Can, F. "On the Efficiency of Best-Match Cluster Searches" Information Processing and Management (to appear).

4. Porter, M.F. "An Algorithm for Suffix Stripping" Program, 14, 3 (July, 1980) 130-137.

5. Salton, G., Buckley, C. "Term-Weighting Approaches in Automatic Text Retrieval" Information Processing & Management 24, 5. 513-523.

6. Salton, G., Buckley, C. "Improving Retrieval Performance by Relevance Feedback" Journal of the American Society for Information Science 41(4), 288-297.

7. Salton, G. Automatic Text Processing. Addison Wesley, Reading, Massachusetts, 1989.