technical report NUIG-IT-250901

An Architecture for Information Retrieval from Distributed Heterogeneous Information Sources

J. Ryan (Compaq, Galway)
C. O'Riordan (NUI, Galway)

An Architecture for Information Retrieval from Distributed Heterogeneous Information Sources

John Ryan,
Service Infrastructure Engineering,
Compaq,
Galway.
Sean.ORiain@compaq.com

Colm O' Riordan,
Dept. of Information Technology,
NUI, Galway.
colmor@geminga.nuigalway.ie

Abstract

This paper provides an overview of the initial design of a system to provide accurate responses to users' queries against a set of information repositories. We present an overview of techniques from traditional information retrieval and filtering, techniques from the more specialised field of distributed information retrieval (paying particular attention to the problems of source selection and result fusion) and, finally, we discuss the initial design for our system.

1 Introduction

This paper provides an overview for the initial design of a system to provide accurate responses to user's queries against a set of information repositories. We present an overview of techniques from traditional information retrieval and filtering, techniques from the more specialised field of distributed information retrieval (paying particular attention to the problems of source selection and result fusion) and, finally, we present the initial design for our proposed system.

The field of distributed Information retrieval has come to the fore recently in the past few years with the increased presence of distributed information sources. The need for efficient accurate techniques to satisfy users' information needs against a distributed set of information repositories is well recognised.

2 Information Retrieval and Filtering

2.1 Introduction

A typical IR system for discussion purposes may be considered as comprising the following components: document pre-processing, query processing, document and query representation, comparisons of query representation to document representation, presentation of retrieved documents, user feedback and query modification. The objective of any IR system is to accurately satisfy a user's information need. In this section we overview approaches and techniques used and developed within the field of information retrieval.

2.2 Document Preprocessing

The document preprocessing phase involves applying a set of well-known techniques to the document collection to convert it to a format more suitable to the task at hand. Common approaches include:

- Stemming: Stemming algorithms remove common suffices from terms occurring in the documents. The goal is to reduce similar words to a common root form by identifying morphological derivations of words. Commonly used algorithms include Lovin's stemming algorithm[14] and Porter's stemming algorithm[15].

- Thesauri construction: This is often used to identify synonyms within the texts. Thesauri can be constructed via manual or automated approaches. The former is created with knowledge of the language at hand; the latter is based on calculating statistics relating to co-occurrences of terms.

- Stop word removal: This involves the removal of highly frequent terms from documents. These terms (typically including conjunctions, prepositions etc.) add little to the semantic meaning of the document.

2.3 Query Processing

Query processing involves query tokenisation, syntax interpretation and query expansion. The terms within a query are often also subjected to stemming and stop-word removal algorithms. The query format can vary from system to system. The most commonly used are:

- Query terms augmented with Boolean operators, proximity operators and wild-cards.

- Natural language text.

However, graphical based query languages and form-based approaches have also been used.

2.4 Comparison

The type of comparison effected between the user's information need and the document set is determined, to a degree, by the representation chosen. Approaches include:

- String matching possibly augmented with proximity and Boolean operators.

- Vector-Space model[19] in which documents and queries are represented as vectors of dimension m, where m is the total number of terms used to identify content. Each of these terms has an associated weight representing its relative importance (based on frequency within a document and across the document collection).

- Latent Semantic Indexing (LSI)[7] attempts to overcome the problems associated with word-based methods, especially the vector space approach, by organising textual information into a semantic/conceptual structure more suitable to information retrieval. The phrase "latent semantic" refers to the inherent underlying associations between words used to express a particular concept.

- Connectionist approaches to IR in which each node is used to represent an individual keyword. The search mechanism usually used in these systems is the *spreading activation search* (SAS). In this search strategy, activity is propagated through the document representations and nodes with a high level of activity are returned as the result of the search. Connectionist approaches include Kwok[11] and Belew[2],[3].

2.5 Presentation of results

The most common approach to presenting results to the user is the ranked list where documents are ordered in decreasing order of relevancy. Other approaches have also been used but are not in widespread use. These include graphical representations of the closeness of query components to returned documents (e.g. the Tilebar system[10] and the VQuery system[17] or the placing of the returned documents with respect to their relation to other documents[13].

2.6 Feedback and Query Modification

Relevance feedback has proved to be highly effective for improving information filtering and retrieval. Upon receiving returned articles, the user may provide relevance judgments for these articles. These relevance judgments may subsequently be used to guide the matching function for the retrieval/filtering system.
Typically, on presentation of the results from the filtering system, the user is asked to identify which documents are relevant and which are not. This information, along with the current user query Q$_k$, is then used to form a new query Q$_{(k+1)}$.

One well-known relevance feedback technique used in the vector space model is the Rocchio feedback model[16] in which a more effective query representation is iteratively generated:

$$P_{k+1} = P_k + \beta \sum_{k=1}^{n_1} \frac{R_k}{n_1} - \gamma \sum_{k=1}^{n_2} \frac{S_k}{n_2}$$

where P_{K+1} is the new profile, P_k is the old profile, R_k is a vector representation of a relevant article k, S_k is a vector representation for non–relevant article k, n_1 is the number of relevant documents and n_2 is the number of non–relevant documents. The values β and γ determine the relative contributions of positive and negative feedback, respectively.

Relevance feedback using this technique has been shown to result in a significant improvement in retrieval performance[18].

3 Distributed Information Retrieval

The problem of distributed information retrieval has become a field of much interest in recent years given the large move towards the distributed paradigm. Factors that have influenced this move include the performance improvement possible due to parallelism and also, possibly more importantly, the increased presence of distributed information sources. The domain provides many very open research questions. These include:

1. Site descriptions: what techniques are suitable (and possible) to describe various different sites containing information repositories.
2. Collection partitioning: If a centralised repository is present, what means are suitable to distribute the collection across a set of sites?
3. Collection Selection: Upon issuing a query, what techniques are useful for collection selection?
4. Interoperability in Searching: Many problems exist in distributing the same query to a set of sites with different query interfaces. How can this desired interoperability be best achieved?
5. Result merging: Which techniques are suitable for merging results from a set of sites in order to return results with high accuracy to the user?
6. Metrics: What metrics are suitable to test the quality of solutions to the above problems, particularly those of site selection and results merging?

The actual focus of this research is in the area of result-merging and associated algorithms. We intend to look at the feasibility and effectiveness of merging retrieval run results from separate autonomous collections into an effective combined result. This is termed collection fusion[21] and is closely related to the field of database merging. We also present information on the other aspects of distributed information retrieval, but the emphasis is placed on those areas of most interest to our research.

3.1 Collection Partitioning

Collections can be either distributed via some priori semantic categorisation or randomly distributed to provide some form of load balancing at query time. The documents may be semantically partitioned where the documents are organised into semantically meaningfully collections by topic or by a categorisation procedure such as clustering.

3.1.1 Interoperability

A broker, if utilised, would have to pass the query to all systems in parallel for evaluation against its collection, resulting in the production per collection of a locally ranked list. These intermediate lists would then be merged into to a final ranked list. Binary heap queues and round robin are examples of approaches previously used in merging but they rely upon globally consistent document scores.

Queries can be issued from, and documents results returned to, a central server. A broker such as STARTS (Standard Proposal for Internet Meta Searching)[9] could be employed to accept user queries, distribute them to the target systems, collect results from the server evaluating the query, combine the results and pass them back to the end user. The selection of a search protocol for result transmission and retrieval is therefore important. STARTS is a proposed protocol designed for heterogeneous distributed searching. It also has the added advantage that it addresses algorithmic issues in distributed retrieval such as result merging.

3.1.2 Available Statistics

Global term statistics may be collected in two ways—compute global terms at indexing time and maintain these statistics at each relevant collection, or alternatively, have two round trips in each query (on the first round, collect statistics from each search process, the combine them into a single set of global statistics and then on the second round, issue the query together with the global statistics). The first approach represents a more efficient approach for executing queries.

In order to facilitate this first approach, each collection would require its own inverted file. The global statistics will also have to be generated and would have to be distributed to all sites.

Approaches can be categorised into two categories. These possible approaches depend on whether collection wide statistics are available or not; in other words, can centralised indexing be constructed or does the set of distributed sites include a number of incompatible and independent collection statistics (termed *uncooperative providers* by Callan[4]).

Voorhees et al.[21],[8],[20] adopted the basic assumption that no machine has access to complete statistics on all documents when evaluating their approaches (query clustering and modelling relevant document distributions).

Callan, et al.[4] worked on the basis that the central server has access to all statistics for each collection. The statistics were then compiled into a single structure that ranked the relevance of each collection to a given query. Moffet et al.[1] qualified this position by assuming that the central server had complete statistics on each document collection but used it only as a filter in discarding or including specific collections.

Danzig et al.[6] provided a mechanism for maintaining similar groupings automatically by using broker agents to maintain centralised indexing on a periodic basis by remote collection querying.

3.2 Ranking

Ranking typically occurs post query processing at the collection server and again prior to the return of the query search results to the user. Depending on the nature of the algorithm employed and statistics available the ranking strategies may vary considerably. Ranking algorithms assign weights to terms in the document and query, compare the weighted document term to the query term and finally rank the results.

When the suitable collections (those which will be queried) have been selected, the retrieval system merges all rankings from the individual collections into a single ranked list. If the listings are not ordered then this task is easily completed as a straight merge. If on the other hand each individual ranking list is ordered per collection then the problem posed becomes more complex.

Voorhees et al.[21] investigated ranking collections by using similarity of training queries to new queries was investigated. The number of documents to retrieve from each collection was evaluated by the relevance judgments for the most similar training query (this approach for dynamic collections would not be as effective if used against static collections). Voorhees investigated interleaving the rankings where only the document rankings were available. Results showed that interleaving via the round robin approach performed poorly in comparison to uneven interleaving weighted by the relevance of the collection to the query.

Callan et al.[4] ranked using inference (the collection similarity was calculated as part of source selection). One system was used for ranking both documents and collections based upon document frequency and inverse collection frequency scores. The mean square was then used to compare the effectiveness of variations to the basic collection-ranking algorithm. They found that ranking based upon weights calculated for document scores and collection weights were approximately as effective as weights based upon normalised scores.

By concentrating collections into blocks of documents Moffet et al.[1] investigated the generation of centralised indexes on these blocks. For each block of documents in the centralised index the query returns the block identifier on its first pass and then only searches the most highly ranked blocks. Results demonstrated that this approach caused dramatic decreases in precision as the number of retrieved documents increased.

More recently Craswell et al.[5] looked at feature distance algorithms to assist with ranking. Feature distance is based upon statistics in a downloaded document header and term positioning in the document relative to other terms and document attributes. When used in combination with reference statistics rather than collection statistics they were found to be more successful than existing techniques in an isolated server environment and just as effective in an integrated-server environment.

3.3 Source selection

Many approaches exist to identify a subset of relevant sources against which to search when attempting to satisfy a given query. One could assume that each is equally likely; this is viable if documents are randomly partitioned. On the other hand, if semantically organised, one could:

1. Search each relevant site; potential to miss relevant documents at a site is deemed irrelevant

2. Map each collection as a large document and create an index of these. At query time, compare the query to this index to identify best sites. This suffers from the same problem mentioned above.

3. Build a model based on test queries. This involves the maintenance of a database of queries and distribution of relevant documents. For each new query, previous similar queries are found and an estimate regarding the distribution of relevant documents is produced.

4. Rule-based approaches

Due to the fact that in large distributed environments there will be collections containing documents that will be more relevant than others, we should have a process or algorithm to assist in determining which of the collections will contain relevant documents and therefore should be sent the query for processing. If we assume that each collection will have an equal probability of containing a relevant document, the query can be forwarded to each collection. For documents occurring on a random distribution basis across the collections this approach works well but as we have some degree of document portioning it would be better to rank each collection as to the probability of containing relevant documents.

3.4 Merging

Strategies for merging can be divided into two categories, namely isolated methods and integrated methods[20]. Integrated methods require the server to provide specific information for use in merging while isolated methods do not. The strategies will define the statistics available when ranking and merging and the approaches taken to generate the merged ranking lists will therefore vary.

3.4.1 Integrated

Collection statistics are required for ranking algorithms in singular or unified collections to provide for effective searching. The merging strategy adapted typically utilises server software and communication protocols such as STARTS to assist in the collation of server collections. In this way statistics such as document frequencies, used to determine relevance can be generated across all collections to produce comparable document scores.

The collection server uses these statistics together with a ranking algorithm (used by all servers) to produce a set of comparable document scores. The client or central server then applies a ranking algorithm to generate a merged ranking of all documents based upon relevance score.

An obvious advantage to this approach is that allowable document scores are generated across all collections. The disadvantages to this approach (and they depend on the environment that the solution is being applied to) are that:

- A database selection algorithm is used to decide which collection should receive the query.
- When the queries are issued, they will, by necessity, include server statistics and the software will resultantly be complex.
- There must be some form of standard client-server communication protocol adapted

Another approach requires that searched servers provide collection statistics with the query search results[9]. The client then combines rankings of the various sets of collection information from each server into a single merged list. The document score may be supplied with the collection statistics or information extracted for down-loading the document itself from its server. The user query will, therefore, if using a standard statistics gathering protocol, not require prior targeting as in the first approach.

Rather than collating the entire set of collection statistics, one need only collate a set of reference statistics to produce a reference statistics database[5]. The reference database would contain statistics for a smaller set of documents. Craswell et al.[5] suggest 10% of the overall collection. The reference statistics would then be substituted in place of collection statistics and fed to a ranking algorithm

Due to the additional server functionality and communication overhead the efficiency of relevant retrieval is affected but the availability of collections statistics can improve merging effectiveness.

3.4.2 Isolated

Isolated methods utilise information available from document and search servers obtained without special protocol or functionality. Potential informational sources for probabilistic, vector space and inference include:

- Score based
  Documents are assigned scores to determine relevance. These raw scores are server specific and are generated from collection statistics that are not shared. Consequently, they are incomparable.
- Rank based
  A document is assigned a rank number based upon a collection from a particular server. These are then used to produce the ranked server list that can be interleaved. Document distance from a list may be weighted in proportion to server promise as an N-Sided dice and used to determine document ordering as in Voorhees Interleaving[20].
- Content based
  The document is down-loaded from the collection server to the client where it is analysed and receives a merged ranking without the use of collection statistics (e.g. INQUIRIS[12]). As documents are downloaded the merged list will always be current and not impacted by individual collection additions or removals.

Voorhees, et al.[21][20],[8] presented the two isolated merging strategies of:

- Modelling Relevant Document Distribution (MRDD)
  The relevant document distribution can be modelled

by calculating the average of the most relevant document distributions from the most similar training queries. Training queries are used to model the content and search behaviour of each collection. Amalgamating each document set will provide a single set of documents for retrieval but will not impose a ranking system. Instead MRDD will order documents probabilistically.

- Query Clustering (QC)
  The system learns the measure of the quality of a search for a particular topic in the collection. Topic areas are represented as centroids of query clusters. Similarity measures per collection from the set of training queries are clustered using the common documents retrieved. Queries are clustered using Wards clustering method.

Both MRDD and QC are dependent on the training set and the effectiveness of its final ranked list. They are not really effective in environments where user queries differ greatly.

3.5 Merging Algorithm Selection

Available approaches are based upon the rankings and available statistics. At the most basic level if we consider that all collections have equal numbers of documents, the document rankings can then be interleaved. As already noted this form of simple interleaving will not provide satisfactory results.

Some systems return, in addition to the document ranking, a score indicating relevance of the query to the document. If the collections have used the same processing algorithm or they are compatible the document score rankings can be merged. This is termed a raw score merge. Voorhees et al[21] using SMART, evaluated a merge-sort approach where the same assumption is made. They found that results were less relevant than that received from a single collection.

If incompatible, then we are faced with the problem that weightings assigned to words can vary considerably. This has both positive and negative repercussions. It can indicate the term's importance in a collection or if the term is used randomly or commonly across a number of collections then the behaviour will be erratic (Dumais [13]). The terms importance can be measured for a collection by its idf or inverted document frequency score. Incompatibly can be addressed by application of normalisation to the scores.

Incorporating collection statistics along with the documents score, weights and weighting scores can be generated. Assuming that similar collections have similar weights, the collection's score can be used rather than its weight to provide the following example of how w can be calculated. The product of the documents weight and score provides the ranking ensuring that documents from collections with high scores receive high relevance ratings and relevant documents in collections with low scores are also allowed good ratings.

$$w = 1 + |N| \times \frac{c - c_m}{c_m}$$

Where $|N|$ is the number of collections searched, c is the collection score and c_m the mean collection score.

When the merging strategy has access to document frequency statistics for each collection other approaches can be considered.

Incompatible scores can be resolved in certain cases by normalising the document frequencies using a weighting such as idf generated using the collections. This will provide the same result as if the collections were merged into a single collection.

4 Proposed Architecture

4.1 Introduction

It is assumed that a number of search servers are available each with the capability of generating their own and others indexes. These servers should be considered as isolated uncooperative providers as there may be no visibility or control over the collections document score generation.

In order to provide for increased extensibility we will make few assumptions regarding the content of any site or the querying capabilities provided.

4.2 Overview

The challenge is to develop an integrated distributed IR system whose skeleton incorporates the following:

- A set of collections that may or may not have some a priori categorisation. Some collections may be topically organised, either automatically or by an individual. Other collections (e.g. email correspondences) may be organised chronologically. We will make no assumptions regarding the nature of any individual collection regarding any semantic or chronological categorisation. This will enhance robustness and extendibility of any algorithms we employ for source selection or result fusion. Any preference for one site over another in either the source selection of result fusion components will be derived from evidence obtained from user-feedback.

- Local indexes available at each collection. Each site will provide an interface that will accept a query from the user and return a ranked list. Again, the actual implementation or algorithms employed cannot be assumed to be known in advance. In reality, initially, the sites will be largely homogeneous in nature employing some well-known accepted comparison algorithm (e.g. vector space).

- Communication query via a web user-interface to the search servers
- A means to accept individual ranked lists from distributed servers
- Localised central merging to determine relevance and presentation to the user of final merged list rankings. The algorithm employed here should ideally take into account available information such as:
  - Individual/group profiles: learning can be incorporated via feedback from individual users. This can be interpreted for individual users or at a group level where users will have overlapping interests.
  - Learning based on query types: Certain queries may be common throughout the whole set of users. From any evidence garnered from feedback (implicit or explicit) we can modify the results. Note that feedback for both these cases can be used to modify the query using traditional feedback mechanisms but can also be employed to modify our heuristics for source selection.
- Source selection and ranking based on updated values learned from user feedback.

4.3 Experiments

Given the above architecture, it will necessary to test the performance of the system. It is envisioned to use the TREC data collection (or subset of) and distribute it across a set of sites.

The standard metrics of precision and recall will be employed to illustrate and gauge the performance of our system.

We will wish to test the behaviour of our learning algorithms based on user feedback by comparing our results to those obtained by a straightforward round-robin approach.

4.4 Summary

In this paper we have presented an overview of work in information retrieval and distributed information retrieval. The main open research questions in the domain of distribute information retrieval are discussed with particular emphasis on those of relevance to the architecture proposed. We have also briefly outlined the basic architecture and experimental set-up to test a number of features of our design.

References

[1] Moffet A. and Zobel J. Informational retrieval systems for large documents. *Proceedings of the Third Text REtrieval Conference (TREC-3)*, 1994.

[2] R. K. Belew. *Adaptive Information Retrieval: Machine Learning in Associative Networks.* Phd thesis, Univ. Michigan, CS Department, 1986.

[3] R. K. Belew. Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents. *Proceedings of the Twelfth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, 1989.

[4] James P. Callan, Zhihong Lu W., and Bruce Croft Computer. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[5] Nick Craswell, David Hawking, and Paul Thistlewaite. Merging results from isolated search engines. *Australasian Database Conference*, 1999.

[6] Peter Danzig, Jongsuk Ahn, John Noll, and Katia Obraczka. Distributed indexing: A scalable mechanism for distributed information retrieval. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1991.

[7] S. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods Instruments and Computers*, 2(23):229–236, 1991.

[8] Towell G., Voorhees E., Gupta N, and Johnson-Laird B. Learning collection fusion strategies for information retrieval. *Proceedings of the Twelfth Annual Machine Learning Conference*, July 1995.

[9] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. STARTS: Stanford protocol proposal for Internet retrieval and search. Technical Report SIDL-WP-1996-0043, CS Dept., Stanford University, 1996.

[10] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of CHI*, 1995.

[11] K. L. Kwok. A neural network for probabilistic information retrieval. *Proceedings of the Twelfth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, 1989.

[12] Steve Lawrence and C. Lee Giles. Inquirus, the neci meta search engine. *Proceedings of the 7th World Wide Web Conference*, August 1998.

[13] Xia Lin, Dagobert Soergel, and Gary Marchionini. A self-organizing semantic map for information retreival. *14th ACM SIGIR Conference*, pages 262–269, 1991.

[14] J.B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 1:22–31, March 1968.

[15] M.F Porter. An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137, July 1980.

[16] J.J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART REtreival System : Experiments in Auotmatic Document Processing*, pages 313–323, 1971.

[17] Jones S. Dynamic query result previews for a digital library. In *Proceedings of ACM Digital Libraries*, pages 291–292. ACM, 1998.

[18] G. Salton. *Automatic Text Processing: The transformation, Analysis, and Retreival of Information by Computer*. Addison-Wesley, 1989.

[19] G.A. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill International, 1983.

[20] E. Voorhees. Siemens trec-4 report: Further experiments with database merging. In D. K. Harman, editor, *Proc. Fourth Text Retrieval Conference (TREC-4)*, pages 121–130, November 1995.

[21] Ellen M. Voorhees, Narendra Kumar Gupta, and Ben Johnson-Laird. The collection fusion problem. In *Text REtrieval Conference*, 1994.