Integrating Distributed Information Sources with CARROT II

R. Scott Cost[1], Srikanth Kallurkar[1], Hemali Majithia[1], Charles Nicholas[1], and Yongmei Shi[1]

University of Maryland, Baltimore County
Baltimore, MD USA
{cost,skallu1,hema1,nicholas,yshi1}@csee.umbc.edu

Abstract. We describe CARROT II (**C2**), an agent-based architecture for distributed information retrieval and document collection management. **C2** can consist of an arbitrary number of agents, distributed across a variety of platforms and locations. **C2** agents provide search services over local document collections or information sources. They advertise content-derived metadata that describes their local document store. This metadata is sent to other **C2** agents which agree to act as brokers for that collection, and every agent in the system has the ability to serve as such a broker. A query can be sent to any **C2** agent, which can decide to answer the query itself from its local collection, or to send the query on to other agents whose metadata indicate that they would be able to answer the query, or send the query on further. Search results from multiple agents are merged and returned to the user. **C2** differs from similar systems in that metadata takes the form of an automatically generated, unstructured feature vector, and that any agent in the system can act as a broker, so there is no centralized control. We present experimental results of retrieval performance and effectiveness in a distributed environment.

1 Introduction

We have developed a scalable, distributed query routing and information retrieval system, CARROT II. It is the successor of an earlier project (Collaborative Agent-based Routing and Retrieval of Text) [9, 7] (originally CAFE [8]). **C2** is composed of a flexible hierarchy of query routing agents. These agents communicate with one another using KQML [10] and the Jackal platform [6], and may be distributed across the Internet. While all agents in the system are alike, they can each control widely varying information systems. Agents interact with information sources via a well-defined interface. Queries presented to any agent in the system are routed, based on the content of the query and metadata about the contents of the servers, to the appropriate destination. Agents themselves are uniform and extremely simple.

C2 contains wrappers that extend several well-known IR systems (e.g. MG [20], Telltale [16]), as well as **C2**'s own, modest IR system. These wrappers present

a basic interface to the **C2** system for operating on documents and metadata. Agents are addressable via commands that are communicated in KQML. This means that a **C2** system can be created, configured, and accessed by another information system, and so can be employed to extend the search capabilities of an existing project. We use the Jackal platform to support communication among agents in **C2** and to provide an interface to the outside world. In addition to Jackal's support for agent communication with KQML, we can use its conversation management capabilities to specify and implement higher-level behaviors for the various negotiation and management tasks required within the agent system.

Our research effort has been directed towards testing retrieval performance as well as effectiveness. Since we cannot assume static corpora, we believe in the use of an agent-based retrieval architecture based on a sophisticated communication infrastructure to handle dynamic data and its associated operations. Section 2 discusses the problems areas facing DIR and the efforts so far by the research community, Section 3 describes **C2** architecture and Section 4 describes its operations.

2 Related Work

In the past there have been attempts to introduce the concepts of agent-based information retrieval. Systems like SavvySearch [14] demonstrated a simple approach to querying web search engines and combining their results in a single ranked order.

Historically, Harvest was the first system to demonstrate the use of broker agents in distributed search. The Harvest system [2] is a distributed, brokered system originally designed for searching Web, FTP, and Gopher servers. In Harvest, "gatherer" agents collect metadata from information providers and deliver it to one or more brokers. Metadata objects are represented in Summary Object Interchange Format (SOIF), an extensible attribute-value-based description record. Harvest pioneered the ideas of brokering, metasearch, replication, and caching on the Internet.

2.1 Distributed Information Retrieval

Information Retrieval in a distributed environment normally follows three steps [3]:

1. Information Source Selection: Select best information source per query
2. Query Processing: Send query to source(s) and return ranked list of documents
3. Results Fusion: Create single ranked list from ranked lists of all sources.

For retrieval from text, one of the methods for information source selection is use of automatically generated metadata from the content. Comparing the query to metadata about the sources can reveal the possible relevance of each source

to the query. CORI [4] and gGloss [13] are examples of such metadata in information source selection. The CORI model is based on inference networks. CORI creates a virtual document containing Document Frequency (DF) and Inverse Collection Frequency (ICF). The ICF indicates importance of the term across the collections and is analogous to the Inverse Document Frequency (IDF), which is a measure of term importance in a single collection. gGloss creates a virtual document containing DF(s) and Term Frequency (TF), i.e. number of occurrences per document of unique terms of the collection. French et al. [11] showed that CORI performed better than gGloss in terms of retrieval effectiveness, however they could not provide a reason for CORI's better performance.

Gibbins and Hall [12] modeled query routing topologies for Resource Discovery in mediator based distributed information systems. Queries are routed by a referral (of a server) by the mediator or by a delegation of the query to the mediator. Liu [15] demonstrated query routing using processes of query refinement and source selection, which interleaved query and database source profiles to obtain a subset of good databases.

The final step in answering a query is fusing the ranked list from the queried sources to obtain a single ranked list. Voorhees et al. [19] showed the use of query training and query clustering to first query appropriate data sources and then merge the results. The query training approach used a dice biased by the number of documents still to be merged, whereas query clustering applied a factor to the results based on the importance of the source it was from and then rank based on the new scores. Aslam and Montague [1] showed that results fusion based on ranks alone can be as good as regular fusion techniques and that relevance scores are not required.

In general, there is a performance gain by distributing information, but distributed retrieval lags behind centralized retrieval in terms of retrieval effectiveness, i.e. percentage of relevant documents returned for a query. However Powell et al. [17] showed that a distributed search can outperform a centralized search under certain conditions.

3 C2 Architecture

A **C2** system is a collection of coordinated, distributed agents managing a set of possibly heterogeneous IR resources. These agents each perform the basic tasks of collection selection, query routing, query processing, and data fusion. In order to effect this coordination, some amount of underlying structure is required.

There are three components to the **C2** architecture. The central work of **C2** is performed by a distributed collection of **C2** agents. There is also a network of infrastructure agents which facilitate communication and control of the system. Finally, a small set of support agents facilitates access to the system, and coordinates its activities. Each of these components is described in detail below.

3.1 C2 Agents

The **C2** agent is the cornerstone of the **C2** system. It's role is to manage a certain corpus, accept queries, and either answer them itself, or forward them to other **C2** agents in the system. In order to do this, each **C2** agent creates and distributes metadata describing its own corpus to other **C2** agents. All **C2** agents are identical, although the information systems they manage may vary.

A standard interface provides methods for manipulating documents, metadata, and handling queries. The agent maintains a catalog of metadata which contains information about the documents stored by peer agents

3.2 Information Integration

A **C2** agent interfaces with an information source which may be an ordinary IR package, or a database manager.

The **C2** system currently has a wrapper that interfaces with the WONDIR[1] (in-house) IR engine. It can however be extended to support other types of Information sources. As mentioned earlier, metadata is derived from the document collection. The metadata takes the form of a vector of the unique "N-grams" of the collection and a sum of their number of occurrences across all documents in the collection. Hence unlike Harvest, **C2** metadata describes the agent's collection of documents, not a single document. **C2** uses such metadata for source selection per query. The motivation for such a form of metadata comes from relative ease of use, low cost of generation, and the ability to aggregate metadata, such that a single vector may contain metadata about multiple agents.

The **C2** metadata is different from the *CORI virtual document* in that CORI uses document frequency, i.e. the number of documents the term has occurred in the collection. **C2** uses term frequency, i.e. the sum of the number of occurrences of the unique terms over all documents. In many if not most collections, a large percentage of terms have occurrences of 0 or 1. By storing the sum of the terms **C2** metadata adds more weight to terms that appear more often in the collection either in more documents or in fewer documents with larger occurrences per document. *CORI* would favor terms that occurred only in more documents.

The same query operation can now be performed on both documents and metadata. A query operation returns a similarity score using $TF * IDF$ based cosine similarity [18]. Querying a collection returns a ranked list of the documents sorted by their similarity scores. For querying metadata collection IDF is replaced by the ICF (see Section 2). On average from empirical observations the metadata is 8-10% of the size of the document collection. The agent that creates the metadata attaches its signature information to the vector.

3.3 C2 Infrastructure Agents

In order to support the successful inter-operation of potentially very many **C2** agents, we have constructed a hierarchical infrastructure. The infrastructure is

[1] Word or N-gram based Dynamic Information Retrieval

largely dormant while **C2** is in operation, serving primarily to facilitate the orderly startup and shutdown of the system, and provide communications support.

The infrastructure is controlled by a single Master Agent, which may be located anywhere on the network. At startup, the Master Agent is instructed as to the number of agents required, and some factors regarding their desired distribution. These include the number of physical nodes to be used, as well as the degree of resource sharing at various levels.

The Master Agent starts a Node Agent on each participating machine, and delegates to it the task of creating a subset of the required agents. The node will be divided into Platforms, or independent Java Virtual Machines, each governed by a Platform Agent. The Node Agent creates an appropriate number of Platforms, and again delegates the creation of a set of agents.

Within each Platform, the Platform Agent creates a set of Cluster agents. The purpose of the Cluster Agent is to consolidate some of the 'heavier' resources that will be used by the **C2** agents. Primarily, this means communication resources. A Cluster Agent maintains a single instance of Jackal. Each Cluster Agent creates a series of **C2** agents; these run as subthreads of the Cluster Agent. Because most agents will be dormant at any one time, we allow a **C2** agent to be assigned more than one collection, creating a set of 'virtual' agents. Thus the virtual agents are the agents visible to all entities external to the system.

3.4 C2 Support Agents

While the agents in the **C2** system work largely independently, a small set of support agents serve to coordinate the system's activities. These are the Agent Name Server, the Logger Agent, and the Collection Manager.

An Agent Name Server provides basic communication facilitation among the agents. Through the use of Jackal, **C2** employs a hierarchical naming scheme, and its operation is distributed through a hierarchy of name servers.

A Logger Agent monitors log traffic, and allows the system to assemble information about the details of operation. This information can then be used to feed monitors or visualization tools.

Finally, a Collection Manager facilitates the distribution of data and metadata. It determines which collection of documents or information source will be assigned to each agent, how each agent will distribute its metadata, and what set of agents will be visible outside the system.

4 C2 Operation

Metadata distribution and query processing are the two main functions of the **C2** agents. Recall from Section 3.2 that the **C2** metadata is an automatically generated feature vector derived from the content itself.

4.1 Metadata Distribution

C2 uses a vector-based representation of metadata which describes the contents of the local corpus (see [9]). Metadata, as well as corpus documents, are managed by an underlying IR engine. This metadata is first order only, and is to be distinguished from information characterizing the set of collections known to a given agent, or higher order metadata. This form of metadata used supports interoperability. The routing of queries should not be hampered by the underlying information source, may it be an IR engine, a search engine on the Internet or an RDF or DAML+OIL [5] based system.

The distribution of Metadata has a profound impact on the system's ability to route queries effectively, and determines the "shape" of the **C2** system. Agents receive instructions on metadata distribution from the Collection Manager. There are three possible metadata distribution modes that can be used by **C2**:

1. Flood: Each agent broadcasts it's metadata to every other agent in the system. Under this scheme, any agent receiving a query would have complete (and identical) knowledge of the system, and be able in theory to find the optimal target for that query.
2. Global: As the original CARROT architecture a designated broker agent has knowledge of the entire system. All agents share their metadata with only this agent.
3. Group: Once the system reaches a certain size, however, both of the above schemes are impractical; the system would become susceptible to bottlenecking. The group scheme based on mathematical, quorum-based distributions, where a group of agents is represented by a chosen (or elected) agent. Metadata sharing would occur inside such groups and amongst the group leaders.

Metadata distribution can be effected by transferring the entire vector, a "difference" vector in case of changes in agent's corpus, or just a URL pointing to the location of the metadata, enclosed in a KQML message.

4.2 Query Processing

Once the system is running, the Collection Manager becomes the primary or initial interface for outside clients. A client first contacts the Collection Manager to get the name or names of **C2** agents that it may query. The names in this set will be determined by the metadata distribution policy. For example, in the case of group-based distribution, the set will contain the group leader agent names. The client then sends queries to randomly selected agents from the given set. It is also possible to model more restricted or brokered architectures by limiting the list to only one or a few agents, which would then feed queries to the remainder of the system.

In response to an incoming query, an agent decides whether the query should be answered locally, forwarded to other agents, or both. In flood mode, for example, the agent compares the query to its local metadata collection , and determines the best destinations. Based on the results, it may send the query to the

single best source of information, or it may choose to send it to several. One of those sources may be its own local IR engine. Once answers are computed and received, the results are forwarded back to the originator of the query. If more than one information source is targeted, the agent faces the problem of fusing the information it receives into one coherent response.

Queries may be routed through a number of different agents before finally being resolved; this depends on the scheme used for metadata distribution and the routing algorithm. For example, the simplest scheme is to have each agent broadcast its metadata to every other agent in the system. The corresponding routing algorithm would be to route to the best information source. Since all agents have the same metadata collection and employ the same algorithm, a query will be forwarded at most once before being resolved. For schemes which employ a more efficient distribution of metadata, or possibly higher order metadata, queries may pass through many **C2** agents before finally being resolved.

4.3 Implementation

The current implementation of **C2** uses the flood mode of metadata distribution. This implies that the query given to any agent in the system will return the same answer. The query can be routed either based on a metadata similarity cutoff or to the best N agents, based on the metadata similarity scores. Therefore, as stated in section 4.2, the query needs to be *forwarded* only once. The results fusion is based on Voorhees's query clustering approach [19]. But unlike their method, the importance of the collection is measured by the metadata similarity score generated. The metadata score is simply applied as a factor to the each of the individual document similarity scores of each collection's ranked list. The results are then sorted based on this new similarity score and the top N documents returned as results.

5 Conclusions and Future Work

We presented an initial prototype of a Distributed Information retrieval system that uses an agent-based architecture. We have been able to show with the initial construction of the system a proof of concept, i.e. distributed retrieval in fielded system does actually performs satisfactorily with a slightly deteriorated performance than centralized retrieval.

References

1. J. A. Aslam and M. Montague. Models for metasearch. In *ACM SIGIR*, pages 276–284, 2001.
2. C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.

3. J. Callan. *Advances in Information Retrieval*, chapter 6: Distributed Information Retrieval, pages 127–150. Kluwer Academic Publishers, 2000.
4. J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.
5. R. S. Cost, T. Finin, A. Joshi, Y. Peng, C. Nicholas, I. Soboroff, H. Chen, L. Kagal, F. Perich, Y. Zou, and S. Tolia. ITtalks: A case study in the semantic web and DAML+OIL. *IEEE Intelligent Systems*, 17(1):40–47, January/February 2002.
6. R. S. Cost, T. Finin, Y. Labrou, X. Luan, Y. Peng, I. Soboroff, J. Mayfield, and A. Boughannam. Jackal: A Java-based tool for agent development. In J. Baxter and C. Brian Logan, editors, *Working Notes of the Workshop on Tools for Developing Agents, AAAI '98*, number WS-98-10 in AAAI Technical Reports, pages 73–82, Minneapolis, Minnesota, July 1998. AAAI, AAAI Press.
7. R. S. Cost, I. Soboroff, J. Lakhani, T. Finin, E. Miller, and C. Nicholas. TKQML: A scripting tool for building agents. In M. Wooldridge, M. Singh, and A. Rao, editors, *Intelligent Agents Volume IV – Proceedings of the 1997 Workshop on Agent Theories, Architectures and Languages*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 336–340. Springer-Verlag, Berlin, 1997.
8. G. Crowder and C. Nicholas. Resource selection in CAFE: An architecture for network information retrieval. In *Proceedings of the Network Information Retrieval Workshop, SIGIR 96*, August 1996.
9. G. Crowder and C. Nicholas. Metadata for distributed text retrieval. In *Proceedings of the Network Information Retrieval Workshop, SIGIR 97*, 1997.
10. T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1997.
11. J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *SIGIR*, pages 238–245, 1999.
12. N. Gibbins and W. Hall Scalability issues for query routing service discovery. In *Second Workshop on Infrastructure for Agents, MAS and Scalable MAS at the Fourth International Conference on Autonomous Agents*, pages 209–217, 2001.
13. L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In *In Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1995.
14. A. E. Howe and D. Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.
15. L. Liu. Query Routing in Large Scale Digital Library Systems. *ICDE, IEEE Press*, 1997.
16. C. Pearce and C. Nicholas. TELLTALE: Experiments in a dynamic hypertext environment for degraded and multilingual data. *Journal of the American Society for Information Science*, June 1994.
17. A. L. Powell, J. C. French, J. Callan, M. Connell, and C. L. Viles. The impact of database selection on distributed searching. In *SIGIR*, pages 232–239, 2000.
18. G. Salton, C. Yang, and A. Wong. A vector space model for automatic indexing. *Communication of the ACM*, pages 613–620, 1975.
19. E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *SIGIR*, Fusion Strategies, pages 172–179, 1995.
20. I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.