A unified approach to high-performance, vector-based information retrieval

Kenneth Baclawski* and J. Elliott Smith
Northeastern University
College of Computer Science
Boston, Massachusetts 02115
(617) 373-4631
FAX: (617) 373-5121
{kenb,esmith}@ccs.neu.edu

Abstract

An information retrieval model based on the vector space model is proposed that unifies and extends many commonly used retrieval mechanisms. A distributed architecture and indexing algorithm for high-performance retrieval using this model has been developed. A prototype system has been built that achieves a throughput of 500 queries per second with a response time of less than one second on an 8-node network of workstations. The model and algorithm are designed for retrieval from a corpus of information objects in a single subject area. The objects need not be textual, and must be annotated with content labels. With current technology, our system can be scaled up to support a corpus of several million information objects. Finally, the model allows for content labels that are semantically more complex than just attributes, keywords and subject classifications.

1 Introduction

With the expansion of the Internet and the development of new "Information Superhighways," computer-based communication is becoming the defining technology of this decade. The amount of information that will be available over these new networks is immense: on the order of billions of objects and hundreds of terabytes of data. Information retrieval in such an environment is a monumental task but essential to its success. We propose an information retrieval model, called KEYNET, that unifies and extends many commonly used

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9117030.

IR mechanisms. We have also developed a distributed architecture and indexing algorithm for high-performance IR using the KEYNET model. Our prototype system has achieved a throughput of 500 queries per second with a response time of less than a second for more than 95% of the queries. This measurement was done locally and therefore does not include any wide area network delay times.

The KEYNET system is designed for IR from a corpus of information objects in a single subject area. It is especially well suited for non-textual information objects, for example, scientific data files, satellite images and videotapes, although some kinds of textual document, such as research papers in a single discipline, can also be supported. With current technology, KEYNET can support very high-performance IR from a corpus having up to several million information objects at approximately the same level of performance as smaller corpora.

We begin by presenting the architecture of the KEYNET system. This will explain where the various kinds of information are located, the pathways for communication, and how a user interacts with a KEYNET search engine. In section 3 we introduce the KEYNET model and explain how it can be used to implement many commonly used mechanisms of information retrieval. We then turn to the details of the indexing algorithm. The algorithm is based on the vector space model for information retrieval. It differs from the usual vector space IR systems in using distributed hash tables rather than trees for indexing. The algorithm is presented in section 4. The prototype and its performance, and in particular how it scales up, are discussed in section 5. Although the KEYNET system can be used solely to implement one or more traditional IR mechanisms, it can also support semantically richer content labeling of information objects. Some examples are presented in section 6 to illustrate this feature of KEYNET. We discuss related work in section 7, and we conclude with a summary and future work planned for KEYNET in the last section.

2 The KEYNET Architecture

The purpose of KEYNET is to assist in retrieving information objects from a corpus of them. These information objects need not be textual and may be physically located anywhere in the network. Retrieval is accomplished by means of a *content label* for each information object. These content labels are stored in a repository at the KEYNET site. The structure of the content labels is specified by an information model or *ontology*. The content labels are indexed by means of a distributed hash table stored in the main memories of a collection of processors at the KEYNET site. These processors form the *search engine*. Each content label contains information about locating and acquiring the information object. The KEYNET system is only concerned with finding information objects; users are responsible for actually acquiring (and presumably paying for) information objects.

To see more precisely where all of these components reside, and how they are connected to one another, refer to Figure 1. The user's computer is in the upper left. A copy of the ontology is kept locally at the user site. As this will require from several hundred megabytes to a few gigabytes of memory, it would generally be stored on a CD-ROM. The ontology is also the basis for the user interface to the search engine (see section 6). Queries must

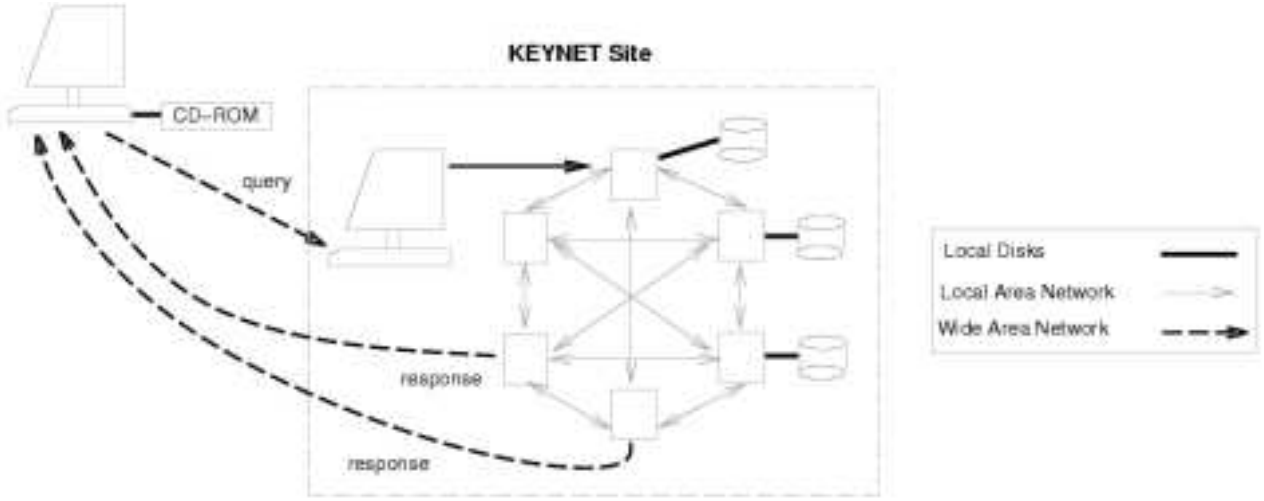


Figure 1: Architecture of KEYNET Search Engine

conform to the format specified by the ontology, and are sent over the network to a front-end processor at the KEYNET site. Responses are sent back over the network to the user's site, where they are presented to the user using the ontology. The prototype system uses a connectionless communication protocol so that no connection is required for making a query, and also so that the responses need not be sent back from the same computer that originally received the query.

At the KEYNET site, the front-end computer is responsible for relaying query requests to one of the search engine computers. The reason for having a front-end computer is mainly for distributing the workload but it also helps to simplify the protocol for making queries. The search engine itself is a collection of processors (or more precisely server processes) joined by a high-speed local area network. The search engine processors will be called *nodes*. The repository of content labels is distributed on disks attached to some of the nodes. The index to the content labels is distributed among the main memories of the nodes. The prototype differs from the KEYNET architecture only in that it randomly generates the repository as well as queries sent to it.

Since a connectionless communication protocol is unreliable, it is necessary for the user computer to resend the query if there is no response after a timeout period. The keynet protocol is stateless and idempotent, and so it works well with a connectionless communication service. There is a similar protocol for registering information objects by sending content labels to the KEYNET site, but this is not explicitly shown in Figure 1.

3 The KEYNET Model

Both queries and content labels are represented using a data structure called a *keynet*. Intuitively, a keynet is semantically intermediate between a keyword and a semantic network. We build up the formal definition of a keynet in stages. The mathematical basis for the

KEYNET model is first presented. Then the notion of a keynet ontology is defined, and it is used as the basis for defining the concept of a keynet. The keynet structure is used as the data structure for content labels, queries, index terms, etc.

The underlying mathematical structure on which keynets are based is the *directed graph*. See [CLR90, Section 5.4] for the basic definitions.[1] A directed graph consists of a set of vertices and a set of (directed) edges that link one vertex with another (possibly the same) vertex. Vertices and edges will generally have additional structure such as textual labels, informal explanations, etc. Edges, in particular, will be labeled with *type* information. When drawing a directed graph, one uses boxes for vertices and solid arrows for edges, each of which is labeled with some of the additional structure belonging to it. The *dimension* of a directed graph is the number of edges it has. A directed graph is *connected* if every pair of vertices can be linked by a sequence of edges (not necessarily all going in the same direction).

As we mentioned earlier, a KEYNET system requires a subject-specific knowledge model or ontology. The word ontology literally means "a branch of metaphysics relating to the nature and relations of being." Our use of the word is much more restrictive, dealing only with the nature of, and relationships among, concepts within a narrow subject area. Attempts to specify ontologies for scientific disciplines are very common, with most disciplines having some kind of subject classification scheme by this time.

The KEYNET system depends on having a background ontology that defines the structure and behavior of keynets. A *keynet ontology* consists of three parts:

1. A directed graph called the *schema*.

   (a) The vertices can be regarded as the set of *conceptual categories* of the ontology. They consist primarily of the subject classification categories for the subject covered by the corpus. However, a vertex can also be a property or attribute of an information object, such as an author, its year of publication, etc.

   (b) The links of the schema join conceptual categories. They are grouped into *link types*. The links in one link type share a common semantics. The best known example of a link type is the ISA link type, which is used to define the concept hierarchy for a subject classification. Other link types include "part of," "elaboration of," "cause of," etc.

2. A set of terms or concepts, called the *lexicon* or *thesaurus*. One think of the terms either as lexical terms, and hence the *keywords* of the ontology, or on higher semantic level as concepts (each of which can be expressed in many ways, by words or phrases). While a lexical term is often a word or phrase, it can also be a person, a number, or a range of names or numbers.

3. A many-to-many relation from the lexicon to the set of conceptual categories that specifies which category (or categories) a lexical term specializes or instantiates. Each lexical term is required to be an instance of at least one conceptual category.

[1] The only difference between our concept of directed graph and the standard one is that we allow more than one directed edge from one vertex to another.

The Unified Medical Language System (UMLS) [HL93] of the National Library of Medicine is an example of a subject-specific ontology. The UMLS ontology is more complex than a keynet ontology; for example, it supports lexical relationships like synonymy as well as relationships on the concept level. However, the core of the UMLS ontology can be regarded as a keynet ontology.

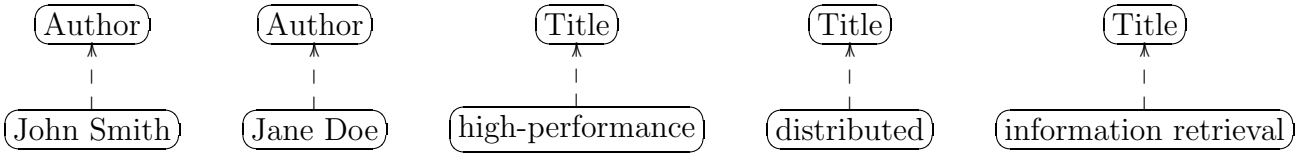
The reason for splitting concepts into the schema and lexicon levels is pragmatic. While there will generally be only about 100 conceptual categories and even fewer link types, there will typically be on the order of 100,000 lexical terms. By defining links only at the schema level, where there are fewer conceptual categories to deal with, it is easier to understand the ontology.

One commonly used IR mechanism is the use of attributes to identify documents. For example, the names of authors or words from the title. These are represented in the ontology with "Author" and "Title" concept categories. The lexicon would have the names of authors and (key)words from titles (or ranges of these if the number of authors or title words was too large). Ranges can also be used to express "wild card" matches.

The notion of a keynet ontology is the basis for the concept of a keynet. A keynet *conforming* to a keynet ontology consists of the following:

1. A directed graph.
   (a) The vertices of the directed graph are copies of vertices of the ontology. There can be several copies in a keynet of one vertex in the ontology.
   (b) The edges of the directed graph are copies of edges in the ontology. Each edge of the keynet must link copies of the source and destination vertices of the corresponding edge in the ontology: the keynet must faithfully reflect the corresponding structure in the ontology.
2. A set of lexical terms that are copies of lexical terms in the ontology.
3. A one-to-one function from lexical terms to vertices. This function specifies the keynet vertex that is being instantiated by each lexical term. Unlike the ontology where the relationship between lexical terms and categories is many-to-many, each category vertex in a keynet may be instantiated at most once by a lexical term, and every lexical term instantiates exactly one vertex.

For example, the following keynet would be used for a content label of the document "High-performance, distributed information retrieval," by John Smith and Jane Doe:



The examples given so far have not made use of any edges to link conceptual categories. In Figure 2, there is an example of a keynet conforming to the UMLS ontology. This

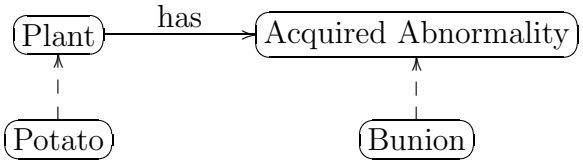
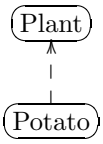


Figure 2: Can a potato get bunions?

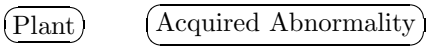
keynet can be expressed in English in several ways. Its expression as a query using English automatically generated using stock phrases is "Find all documents in which the plant potato has an acquired abnormality bunion." Its expression as a statement in a content label is "The potato can exhibit an abnormality functionally similar to bunions." Finally a more succinct colloquial expression would be: "Can potatoes get bunions?" While this may seem to be a somewhat whimsical query, it makes perfectly good sense, and we will later discuss in section 6 how a `KEYNET` system would understand and respond to such a query.

Keynets are used not only for content labels and queries but also for index terms. However, we do not use arbitrary keynets as index terms because that would result in a "combinatorial explosion" of possibilities. Rather we restrict attention to a special kind of keynet. A *fragment* of a keynet is a connected subset of the keynet. In other words, a fragment consists of a choice of vertices, edges and lexical terms from a keynet such that the set of vertices and edges so chosen forms a connected directed graph. For example, this is a fragment of the keynet in Figure 2:

and here is another:



However, the keynet



is not a fragment because it is not connected.

The main result for fragments is the fact that the number of fragments having small dimension grows linearly with the size of the keynet from which they are taken.

Theorem 1 *Let K be a keynet having v vertices. If the degree of a vertex of K is at most d, then there are at most $2 + 6(v + 1) + 4dv$ fragments of K having at most 2 edges.*

For a proof of the theorem, see [BS94]. The bound in the theorem is actually a worst-case bound, and in practice one does much better, typically just 4 times the number of vertices of the keynet. A fragment having exactly one edge is called a *clasp*, while a fragment having exactly two edges is called a *double clasp*.

The reason for limiting fragments to be at most double clasps is not just pragmatic. It arises from a well-known result from Cognitive Science known as the 7 ± 2 rule[Mil56]. This rule states that the maximum number of conceptual "chunks" that can be manipulated by a person at one time is in the range 5 to 9 chunks. A clasp consists of 3 to 5 parts (the two vertices, the edge and up to two instances of the vertices), and a double clasp consists of 5 to 8 parts. Therefore limiting to just one vertex or a single clasp would be too limiting, while a double clasp almost exactly matches the 7 ± 2 rule.

Intuitively, retrieval in the `KEYNET` system proceeds by matching fragments of a query (called *probes*) with fragments of content labels (called *index terms* or *index fragments*). The degree of relevance is then determined by a similarity measure between the vector of probes and the vector of index terms. For this to scale up to large corpora, the number of index terms must not be too large. In particular, if we allowed arbitrary subsets of a content label to be index terms, then the number of index terms would grow exponentially in the size of the content label. The theorem assures us that the size of the index will be manageable.

4 Indexing Strategy

As we have already mentioned, the basic indexing strategy is to match probes (fragments of queries) with index terms (fragments of content labels). We now discuss the details of the distributed algorithm that accomplishes this matching. This algorithm can be characterized as a "scatter-gather" technique. Queries are sent to a front-end processor in the form of datagrams. The front-end processor assigns a query id, acknowledges receipt of the query and forwards the query to a randomly chosen node of the search engine. This is the first scattering step. The node that is assigned the query is called the *home node* of the query.

At the home node, the query is broken apart into probe fragments as discussed in section 3 above. The fragmentation algorithm is more subtle than one would expect, since loops and multiple edges are allowed in keynets. Each probe is then hashed using a standard algorithm given in [Knu73, Section 6.4]. The hash value is in two parts. One part is a node number and the other part is the local hash value used at that node. The local hash value and the query identifier are then sent to the node that was selected by the hash value. This forms the second scatter step of the algorithm. The result of hashing is to scatter the probes uniformly to all of the nodes of the search engine.

Upon receiving the local hash value of a probe, the node looks it up in its local hash table. The hash table algorithm we employ is called "open addressing with double hashing," as described in [Knu73, Section 6.4]. We found that this technique is very space efficient and that collisions do not affect performance even when the hash table is 90% full. An index term in the hash table that matches a probe is called a "hit." The hits are sent back to the home node of the query. This is the "gather" step of the algorithm. Special trailer messages are used for determining when all the hits of all the probes of a query have been collected. The home node then computes the similarity measure (currently the cosine measure) of each object in the collection, and the objects are ordered by the degree of similarity. The object identifiers of the most relevant objects are then sent back to the user.

The insertion of a new content label in the index is done in a manner very similar to the query algorithm. Since content labels and queries are both keynets conforming to the same keynet ontology, they use exactly the same data structure. The same fragmentation, hashing and scattering algorithms are used for content labels as for queries. The only difference is that instead of matching entries in the hash table, index terms are inserted into the table. Note that index terms are not explicitly stored in the index, just their hash values are stored. The number of bits in the hash value is chosen to be so large that it is very unlikely that two probes would have the same hash value. As a result it suffices to store only a hash value and not the index term itself. Since an index term is nearly always much larger than a local hash value, this results in a significant savings of space with only a slight reduction in retrieval effectiveness.

The query algorithm presented so far represents the *basic* level of service. Higher levels of service can be provided by using additional scatter-gather operations. For example, the second level of service uses two scatter-gather operations. After completing the collection phase of the basic level of service, the home node sends each object identifier to the node where its content label is stored, resulting in yet another scatter step. The content label is then retrieved and sent back to the user's computer where the content labels are gathered, arranged and displayed using the keynet ontology.

5 Performance

We have developed a prototype `KEYNET` system that runs on a network of sparcstations connected by a local area network. The sparcstations are part of the network of Unix workstations maintained by the College for a large community of faculty, students, staff and guests. We ran our tests late at night on relatively unused workstations. There was less activity on the network at these times, but there was some activity even at 3:00 AM, so our results exhibit a variance that reflects this.

The largest search engine we have used consisted of an 8 node network. On each node we index the content labels of 20,000 information objects, using 16 Mbytes of memory. Although the Sparcstations have anywhere from 64 to 128 Mbytes of memory, we found that attempting to allocate more than 16 Mbytes resulted in excessive paging activity. Presumably this is a result of the other activities, some of them in the background, going on at all times throughout the network.

The prototype is fully distributed, using a pure message-passing communication mechanism. All messages are one-way: no process ever waits for a reply to a message. The memory model is local, i.e., a "shared nothing" system. It is not a parallel processing algorithm, but we are investigating whether it can be ported to a parallel machine architecture.

The individual nodes are implemented as servers. Specifically, they are implemented as connectionless, multi-threaded, interrupt-driven, stateless servers. Each server is responsible for a fixed amount of memory, 16 Mbytes, which is small enough for page faulting to be rare so that, to a first approximation, the 16 Mbytes can be regarded as physical memory.

Messages between nodes are buffered and sent in groups to improve throughput at the

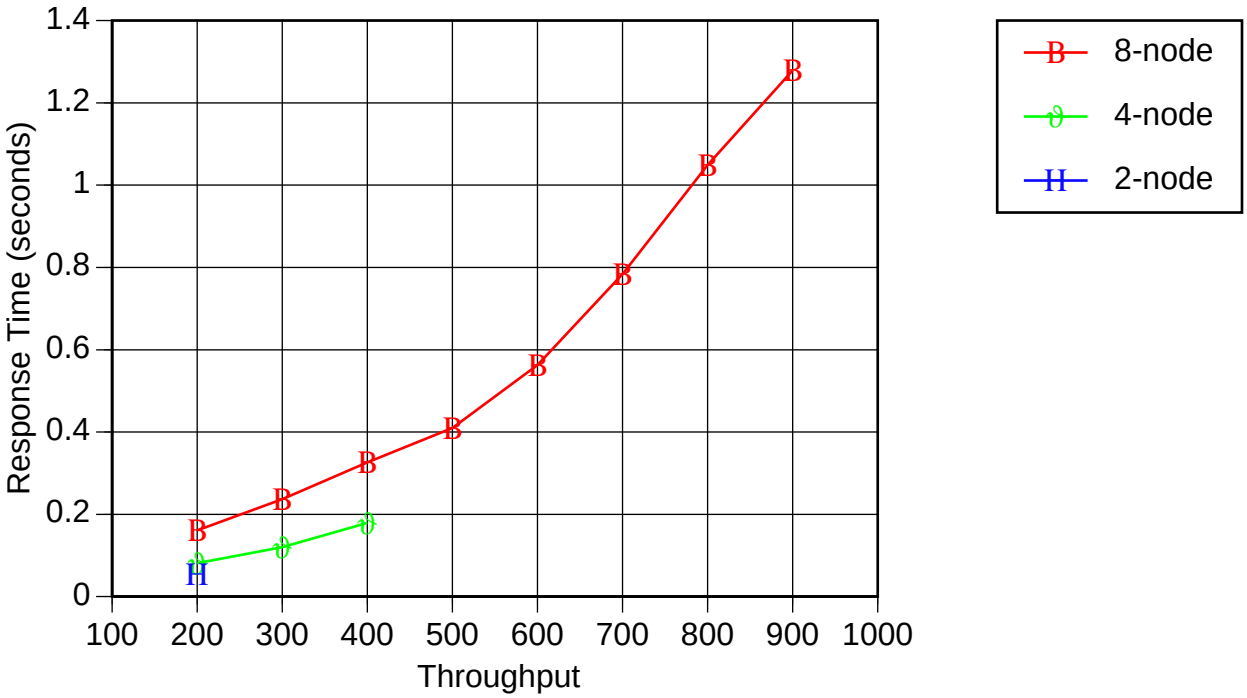


Figure 3: Median Response-Time versus Throughput

expense of some response time. The amount of buffering can be adjusted so as to maximize throughput for a given response time requirement. In the test runs, the buffer size was adjusted for each configuration so that the frequency with which packets are being sent is approximately the same. Otherwise, when one varies the number of nodes, all one is measuring is the effect of the buffer size. We have a mechanism for flushing buffers when this is deemed to be appropriate. Load balancing is done by measuring the relative performance of the nodes at the beginning of each run, and then allocating tasks to the nodes using this measurement.

In figures 3 and 4, we show the median and 95^{th} percentile response times, respectively, versus throughput for 2-, 4- and 8-node search engines. The 2-node engine has the best response time for 200 queries/second, but for a larger throughput its response time goes off the scale. Increasing the number of nodes to 4 results in a slightly slower response at 200 queries per second, but now the throughput can be increased to 400 queries per second before the response time goes off the scale. The 8-node engine has the slowest response at low throughputs, but can sustain the highest throughput before the response time goes off the scale.

The prototype is running well enough to obtain throughput and response-time data on randomly generated databases and queries, and the user interface understands UMLS. However, these two components were not yet integrated at the time this paper was written.

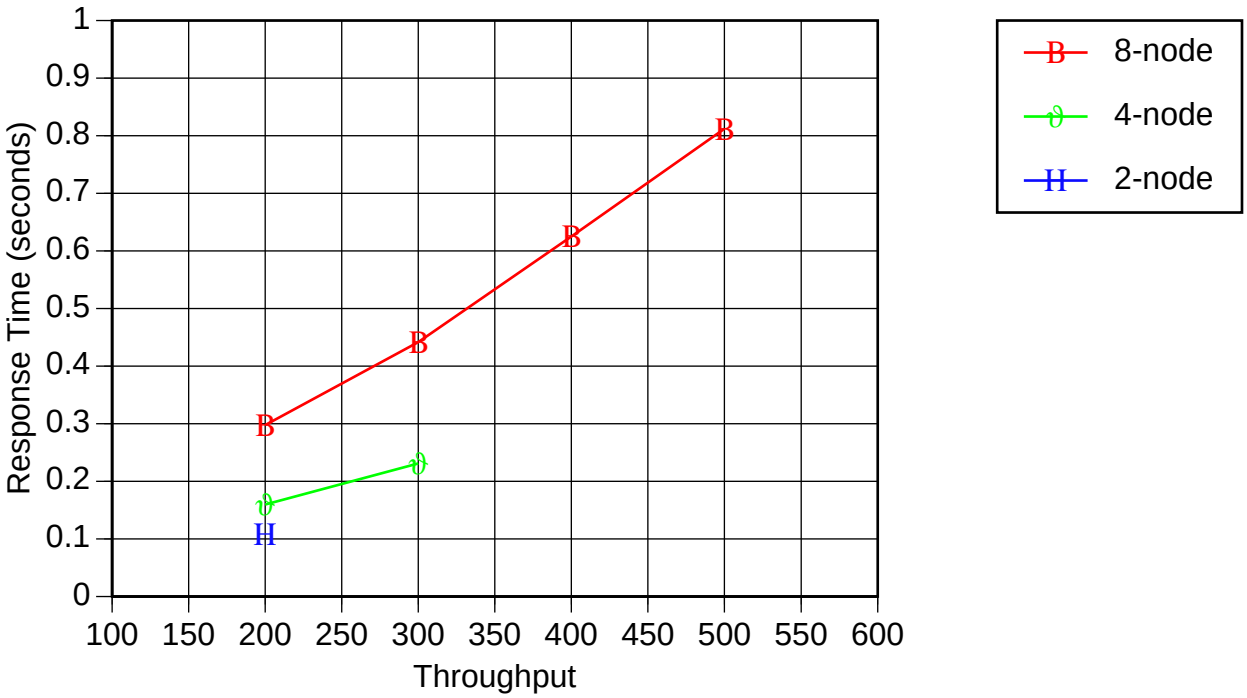


Figure 4: 95^{th} Percentile of Response-Time versus Throughput

6 Semantically Rich Information Retrieval

In addition to the commonly used IR mechanisms, KEYNET supports a semantically richer form of information retrieval. However, as Lakoff points out[Lak87], "Human categorization is based on principles that extend far beyond those envisioned in the classical theory." As a result, simple classification methods leading to taxonomies of concepts are inadequate for expressing the rich variety of human categorization techniques.

Unfortunately, since no IR systems currently use such a mechanism, one cannot evaluate whether it would improve retrieval effectiveness. Our contribution is a "proof-of-existence" that there are no technological or user interface barriers to building a high-performance IR system of this kind.

KEYNET gains potential semantic leverage relative to traditional vector space methods by responding to relations between keywords, in addition to (possibly accidental) co-occurrences. Consider the query "Can potatoes get bunions?" of Figure 2. In this case, the entire query represents one of the fragments. Other fragments may be regarded as varying degrees of generalization of the original query. For example, the specific concept 'potato' may be abstracted into the conceptual category 'plant' to yield the fragment "plants that have bunions;" or 'bunion' may be replaced by the conceptual category 'acquired abnormality' to produce the corresponding fragment "potatoes that have acquired abnormalities."

The best matches for the query "Can potatoes get bunions?" will, roughly speaking, be the content labels that include the largest number of index terms (content label fragments)

that match the query fragments. Therefore an information object that deals with plants that get bunions or potatoes with acquired abnormalities is considered a better match than an information object that discusses, say, potatoes as a treatment for bunions, or mentions potatoes and also bunions but in different contexts. This is the sense in which a KEYNET system can be said to have "understood" a query.

7 Related Work

While semantic networks from AI were influential in the development of the KEYNET model, its indexing technique is fundamentally based on the vector space model for IR [Sal89]. From the point of view of IR, KEYNET can be regarded as a mechanism for unifying a number of different IR mechanisms, and the KEYNET system can be regarded as a high-performance, distributed search engine that can be applied effectively to vector-based retrieval from a corpus of annotated documents. From this point of view, keynets serve the same role as subject classification categories, keywords and properties such as author, title or date of publication.

Despite their reputation in IR circles as cumbersome, inefficient and suitable only for small databases, at least one IR researcher has used knowledge-based indices successfully [FHL+91]. Fuhr *et al*'s AIR/X performs automatic indexing of documents using terms (descriptors) from a restricted vocabulary. Probabilistic classification determines indexing weights for each descriptor using rule-based inference. KEYNET differs from AIR/X in using a form of semantic network as part of the retrieval algorithm rather than in the extraction of suitable terms to be used for indexing. The extraction of terms (or in our case clasps) from a corpus of textual information objects is an important problem. One of the projects related to KEYNET is an effort to automate the extraction of keynets from biological research papers, in particular the Materials & Methods sections of such papers. See [BFFP93, BFH+93a, BFH+93b]. However, such extraction is independent of the architecture and algorithm employed for retrieval (and isn't even possible for non-textual information objects).

After building a system similar to AIR/X, Jacobs [Jac93] determined that "the combination of statistical analysis and natural language based categorization is considerably better than either alone." His paper describes an automated set of statistical methods for pattern acquisition that operate inside a knowledge-based approach for news categorization (an area closely related to document classification and other IR tasks). Like AIR/X, Jacobs' system does not employ semantic networks in the IR engine. Another difference between KEYNET and Jacobs' system is that KEYNET is designed for a corpus of documents in a single subject area, where it is feasible to develop a subject-specific ontology. Developing an ontology for heterogeneous textual documents is a formidable task, many orders of magnitude larger than is feasible with current technology.

The EDS TemplateFiller system [SMHC93] applies Message Understanding (MUC) text-filtering techniques to the generation of knowledge frames for one or a few specific subject areas from entire texts (computer product announcements). TemplateFiller fills in slots for frames that exist in a predefined schema of templates, ignoring subjects that are not in the

schema.

There are many other MUC-style systems; in fact, there is an annual competition among them. Such a system can automate the construction of the content labels for a collection of specialized textual documents. In a project related to the KEYNET project, we are building a MUC-style system for biological research papers [BFH+93a].

A structural model of IR was developed by a project at the University of Western Ontario[Lu90]. This model uses case relations as the structure. Case relations are a major component of case grammars which are a tool proposed by linguistic theorists and developed by computational linguists for natural language processing. The term "case" here is a refinement and generalization of well-known grammatical relationships such as "subject," "object" and "indirect object." Similarity of a query to a document is measured using a form of structural similarity. One conclusion of the study was that the proposed IR mechanism does not improve retrieval effectiveness. Although this system has some superficial similarity to KEYNET it differs in a number of important respects. The most important difference is that the Western Ontario system uses "surface" syntactic structures while KEYNET uses conceptual structures. Another important difference is that the Western Ontario system combines the two tasks of knowledge extraction from text with IR of the resulting knowledge structures. The first task is known to be very difficult, with the best such systems (the MUC-style systems discussed above) achieving only about 50% accuracy, and even this requires that the documents be restricted to a specialized topic area. Accuracy is much lower when general documents are being analyzed.

Several families of databases for semantic networks have been developed. Such databases are often called knowledge-base systems. Some of the best known of these are: Conceptual Dependency, ECO, KL-ONE, NETL, Preference Semantics, PSN and SNePs (see [Leh92]). All of these support link types, frame systems and so on, but few if any explicitly concern themselves with performance measures familiar to work in IR. Hence it is not surprising that these techniques have acquired a reputation for being cumbersome, inefficient and suitable only for small databases. The KEYNET system shows that it is possible to use a limited form of semantic network model in a high-performance IR system.

Some other examples of knowledge-based query modification systems include systems primarily for information retrieval such as those in [GS93, CD90, Har92, QF93] as well as systems designed for database systems such as the KNOWIT system of Sølvberg, Nordbø and Aamodt [SNA92] and the cooperative query answering system in [CC92]. All of these are front-end query modification systems added to an IR or database system. Such query modification techniques can also be used with a KEYNET system; in fact, a keynet ontology is very well suited to the support of such techniques; and the high-performance of a KEYNET search engine is useful for supporting the much larger queries generated by query modification techniques. However, such techniques are fundamentally a front-end for the actual search engine.

8 Summary and Future Work

An IR model has been introduced that generalizes many commonly used IR mechanisms. A distributed architecture and indexing algorithm have been developed for this model. A prototype system has been developed to measure the performance characteristics of the algorithm, and the prototype has achieved a throughput of 500 queries per second. The model can support semantically complex content labels and queries.

A number of research efforts are actively being pursued on both `KEYNET` and closely related projects.

- Since a `KEYNET` ontology is required for indexing a corpus using a `KEYNET` search engine, it is important to have good tools for building such an ontology. The ontology is also closely related to the user interface to the `KEYNET` system, making the ontology even more fundamental. The OntologyBuilder[BF93] is a research effort related to `KEYNET` that addresses the problem of building and managing ontologies. Even a small ontology can occupy hundreds of megabytes of storage, so database management is clearly an important part of this problem. The OntologyBuilder uses object-oriented database management tools for managing the very large and complex data structures of an ontology.

- The indexing strategy used by `KEYNET` is a distributed hashing method. One feature it does not yet have is expandability. Several distributed, expandable hashing methods have been proposed, and we plan to integrate `KEYNET` with one of them.

References

[BF93] K. Baclawski and N. Fridman. M&M-Query: Database support for the annotation and retrieval of biological research articles. Technical Report NU-CCS-94-07, Northeastern University, College of Computer Science, 1993.

[BFFP93] K. Baclawski, R. Futrelle, N. Fridman, and M. Pescitelli. Database techniques for biological materials & methods. In *First Intern. Conf. Intell. Sys. Molecular Biology*, pages 21–28, 1993.

[BFH+93a] K. Baclawski, R. Futrelle, C. Hafner, M. Pescitelli, N. Fridman, B. Li, and C. Zou. Data/knowledge bases for biological papers and techniques. In *Proc. Sympos. Adv. Data Management for the Scientist and Engineer*, pages 23–28, 1993.

[BFH+93b] K. Baclawski, R. Futrelle, C. Hafner, M. Pescitelli, N. Fridman, B. Li, and C. Zou. M&M-Query: Materials & Methods knowledge base and query system. Technical Report NU-CCS-93-06, Northeastern University, College of Computer Science, 1993.

[BS94] K. Baclawski and J. E. Smith. `KEYNET`: Fast indexing for semantically rich information retrieval. Technical Report NU-CCS-94-06, Northeastern University, College of Computer Science, 1994.

[CC92] Wesley W. Chu and Qiming Chen. Neighborhood and associative query answering. *Journal of Intelligent Information Systems*, 1(3/4):355–386, December 1992.

[CD90] H. Chen and V. Dhar. Online query refinement on information retrieval systems: A process model of searcher/system interactions. In *Proc. SIGIR*, pages 115–133, 1990.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[FHL+91] Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Konstadinos Tzeras, and Gerhard Knorz. AIR/X: A Rule-Based Multistage Indexing System for Large Subject Fields. In *Proc. User-Oriented Content-Based Text and Image Handling Conference (RIAO-91)*, pages 606–623, Centre de Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, 1991.

[GS93] S. Gauch and J. Smith. An expert system for automatic query reformulation. In *J. Amer. Soc. Info. Sys.*, volume 44, pages 124–136, 1993.

[Har92] D. Harman. Relevance feedback revisited. In *Proc. SIGIR*, pages 1–10, 1992.

[HL93] B. Humphreys and D. Lindberg. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170–177, 1993.

[Jac93] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, pages 13–23, April 1993.

[Knu73] Donald Knuth. *Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

[Lak87] George Lakoff. *Women, Fire and Dangerous Things*. The University of Chicago Press, Chicago, IL, 1987.

[Leh92] Fritz Lehmann. Semantic networks in artificial intelligence, parts I and II. *Computers and Mathematics with Applications*, 23(2-9), 1992.

[Lu90] X. Lu. *An Application of Case Relations to Document Retrieval*. PhD thesis, The University of Western Ontario, 1990. ISBN: 0-315-59092-0.

[Mil56] G. Miller. The magical number 7 ± 2: some limits on our capacity for information processing. *Psych. Rev.*, 63:81–97, 1956.

[QF93] Y. Qiu and H. Frei. Concept based query expansion. In *Proc. SIGIR*, pages 160–169, Pittsburgh, PA, 1993.

[Sal89] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[SMHC93] H. Kelly Shuldberg, Melissa Macpherson, Pete Humphrey, and Jamil Corley. Distilling Information from Text: The EDS TemplateFiller System. *Journal of the American Society for Information Science*, 44(9):493–507, 1993.

[SNA92] Ingeborg Sølvberg, Inge Nordbø, and Agnar Aamodt. Knowledge-based information retrieval. *Future Generation Computer Systems*, 7:379–390, 1991/1992.