KEYNET: An architecture and protocol for high-performance semantically rich 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

January 25, 1995

Abstract

We propose an architecture and protocol for semantically rich information retrieval from a subject-specific corpus of information objects. The technique assumes that information objects have been labeled using small directed graphs, called content labels. A content label subsumes the role of an abstract. Both content labels and queries are required to conform with a general framework (ontology) for the subject area. Our method avoids explicit isomorphism testing by decomposing queries into fragments of small maximum size and using tables of these fragments to accomplish comparisons. The index structure is compatible with distributed processing architectures and scales up well, allowing very large collections to be searched very quickly using semantically complex queries.

1 Introduction.

With the expansion of the Internet and the development of new "Information Highways," computer-based communication is becoming the defining technology of this decade. A number of proposals have been made to build a coherent structure over these new high-bandwidth networks and thereby convert them into a National Information Infrastructure (NII). The

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

amount of information that will be available in an NII is immense: on the order of billions of objects and hundreds of terabytes of data. Information Retrieval (IR) in such an environment is a monumental task but essential to the success of the infrastructure. Any solution to this problem must satisfy two important requirements: it must be scalable to handle the large number of information objects and it must be semantically rich enough to support effective information retrieval.

Our architecture and indexing strategy provides a search engine that can satisfy these requirements. The KEYNET system supports information retrieval for a corpus of information objects in a single subject area, such as a collection of biological research articles, a set of court cases, files containing remote geophysical sensor data, or even collections of software programs and modules. KEYNET unifies and extends many commonly used IR mechanisms, and can be used effectively not only for corpora consisting of annotated information objects, but also for object-oriented databases in general. A distributed architecture and indexing algorithm has been developed for high-performance IR using the KEYNET model[BS94c, BS94b]. The 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[BS94d].

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, such as scientific data files, satellite images and videotapes. For example, the literal content of a satellite image does not include the geographic coordinates of the boundaries of the image or other cartographic abstractions. 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.

A KEYNET system requires the development of a subject-specific concept ontology that is understandable to a literate practitioner of the field. A keynet ontology represents knowledge using a directed graph of conceptual categories and relationships between them. The Unified Medical Language System (UMLS) developed by the National Library of Medicine is an example of such an ontology[HL93, LHM93]. KEYNET further assumes that each information object in its collection has been annotated with a content label that indicates what portion of the subject-specific ontology relates to the content of the object. [1]

Both content labels and queries have the same data structure called the *keynet* structure. A keynet may be regarded as a kind of semantic network[Lev92], although in practice it is semantically intermediate between keywords and semantic networks. The keynet framework generalizes many commonly used mechanisms for information retrieval, such as: subject classification schemes, keywords, document abstracts, reviews, content labels for non-textual information objects, properties such as author or date of publication, ranges of text strings such as “wild card” match strings, and ranges of quantities. The KEYNET system allows a uniform treatment of these disparate techniques in a system that permits a great deal of

[1] We are misusing the word “annotate” slightly here. Although a content label may refer to portions of its information object, its role is to classify or abstract rather than to explain the portions of the information object to which it refers.

flexibility compared to traditional database and information retrieval systems. For example, one can combine all of the above mechanisms in a single system, and easily add new features to the ontology, such as new attributes and keywords. In addition, the KEYNET framework allows for sequences of concepts linked by relationships and expressed in natural language using phrases, clauses, sentences and paragraphs.

A content label is similar to an abstract or review of a document both in size and in being separately accessible from its corresponding information object. Using a tool such as our M&M-Query System[BF93], a content label can be generated by the author of the information object with no more effort than is now taken to write the abstract or to select the keywords.

2 Example

For a simple example of a content label consider an imaginary paper entitled, "POOQ: A parallel, object-oriented query system." Suppose that the paper uses some known dynamic programming algorithms to optimize queries for use on parallel machine architectures. A keyword-based approach could classify this paper by using phrases such as "parallel algorithms," "object-oriented databases," "dynamic programming," and "query optimization."

Keywords can include topically relevant words and phrases that do not appear in the information objects themselves. Furthermore, some information objects (like images, scientific data and software) have no text that can reasonably be used by traditional IR technology. Keynets enrich the semantics possible with keywords (simple subject classifications) by adding relationships between keywords. In addition to being more expressive than sets of keywords, keynets exhibit more structure and are generally larger, although still much smaller than the entire information object.

To describe a content label for the imaginary POOQ paper above, we must first describe a hypothetical ontology for Computer Science. Assume that one of the classes in this ontology is concerned with the concept of translation or transformation. The content label for the POOQ paper begins with an instance of the transformation class which is linked via an attribute edge labeled "input" to an object having the subtype "declarative language," as shown in Figure 1. The label of each node consists of its type followed by its value (if any) separated by a colon. Since the output language is well-known, no further elaboration is needed. However, the input language, POOQ, is not well-known, so it must be specified further with attributes like its name and other attributes not shown.

As an example of a query, consider "What systems use dynamic programming to generate C^* code?" This is translated into the keynet in Figure 2.

After converting a query into a keynet, it is decomposed into components of bounded size. These fragments are called *probes*. The fragmentation algorithm is given in [BS94a]. The probes are indexed using a distributed hash index. Content labels are also fragmented and the fragments inserted into the index. The search step consists of hashing the probes and looking for matches with fragments of content labels. The POOQ document in the example has fragments that match every probe of the query.

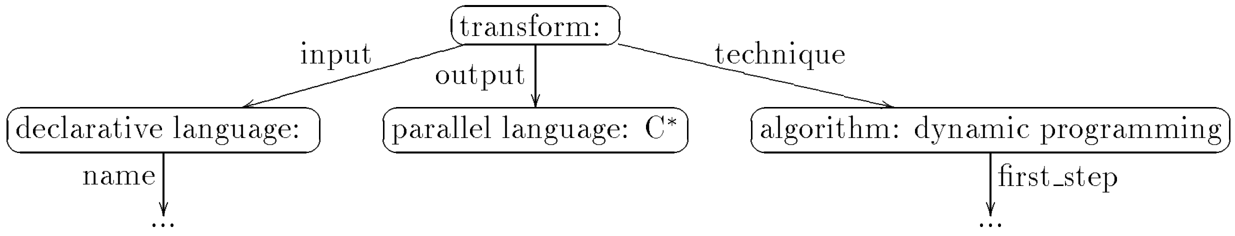


Figure 1: Example of part of a content label

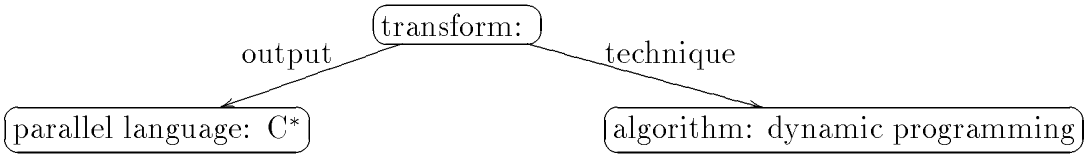


Figure 2: Semantic network of a query

3 Information Retrieval.

Information retrieval systems are primarily concerned with the problem of providing mechanisms for a user to select a small set of relevant documents (or parts of documents like chapters, figures, tables, and so on) from a large document collection. This objective differs from database management in a number of ways. Generally speaking, IR systems are not concerned with answering detailed queries about the content of the documents in the collection. On the other hand, database systems can answer detailed queries very precisely, but most are not capable of answering the vaguely worded queries about relevance that an IR system can handle.

A database system takes for granted that a query is precisely stated, and the issue becomes how efficiently the query can be evaluated. By contrast, since IR queries are not required to be precise, one measures performance in a different manner. The two most common general measures of retrieval quality in IR research are called *recall* and *precision*. The former is the ratio of documents retrieved versus the number of available documents relevant to the query, i.e., the fraction returned out of all desirable documents (a measure of alpha risk or type I error). The latter is the ratio of the number of relevant documents retrieved versus the total number of documents retrieved, or the useful fraction of what was actually retrieved (beta risk or type II error).

Recall and precision presume that categories assigned by human experts are correct, complete, and well-specified; yet emergent concepts are seldom clearly formulated. An ideal information retrieval mechanism must not only capture poorly expressed concepts, but also somehow adapt as ideas change; its conceptual ontology must evolve over time.

Most commercial IR systems are based on a boolean model of relevance. The query terms are matched to keywords or to words in the document content, and relevance is determined by the satisfaction of a boolean expression specified by the user. A variety of techniques such as word stemming, truncation, thesauri and lexicons have been used to extend this model[Sal89].

In contrast to boolean methods, the so-called "vector" methods use a notion of relevance that is less sharply defined: a document has a degree of relevance (salience) rather than simply being relevant or irrelevant. Documents are represented as points in a multidimensional vector space. One can then compare documents to each other as well as to queries. One commonly used measure is the cosine of the angle between the points regarded as vectors [Sal89]. Other possible measures of salience include path distance between nodes in a graph, or the number of levels that must be traversed to connect two categories in a hierarchy of abstractions.

While vector methods use probability and statistical methods to improve retrieval effectiveness, they are the same as boolean methods in their reliance on simple linguistic units, such as combinations of words or phrases, as the basis for retrieval. Since fragments of natural language do not always communicate a concept unambiguously for every combination of speaker, listener, writer or reader, retrieval errors inevitably arise from irrelevant discourse. Consequently, to improve retrieval effectiveness IR systems often label documents using keywords or phrases that may never appear in the document itself.

Moreover, relevance requires relative judgement; material irrelevant for categorization may still be relevant for other user purposes. Retrieval that merely matches against text in a document body presumes concepts can be completely characterized by statistical correlations. Yet as Jacobs [Jac93] observes, "statistical methods must be an aid rather than a replacement for knowledge acquisition". Text statistics are best used to identify patterns that depend on specialized words and phrases not obvious to casual readers. Mechanisms that recognize complex ideas are better constructed by human experts, whose understanding of a concept may transcend language.

Unfortunately, knowledge-based approaches that utilize semantic networks are currently considered so inefficient that they are explicitly omitted from some IR textbooks. For example, [FBY92] dismisses them on the basis of "the amount of manual effort that would be needed to represent a large document collection."

4 Related Work.

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.

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 information retrieval tasks).

Both of these systems attempt to automate the process of generating index terms. KEYNET, by contrast, is concerned not with how the index terms are obtained but instead with their structural relationships.

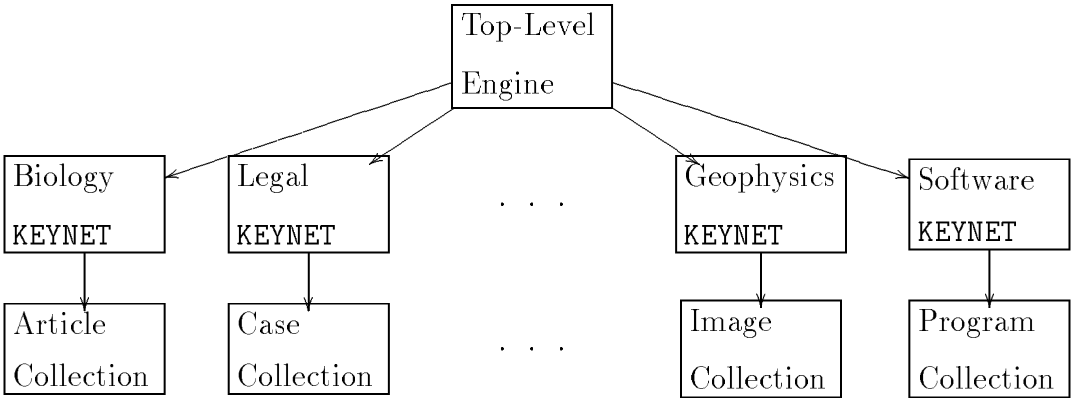
Several distinct families of databases for semantic networks have been developed. Such databases are often called knowledge-base systems. Some of the best known of these are: Conceptual Dependency, ECO, KL-ONE, NETL, Preference Semantics, PSN and SNePs (see [Leh92]). All of these support link types, frame systems and so on, but few if any explicitly concern themselves with performance measures familiar in traditional database work, such as minimizing the number of *disk accesses* required to retrieve complex structures (i.e., graphs assembled from multiple frames and their typed relations). Contemporary knowledge-base systems traverse semantic networks one frame/relation at a time. Unless the knowledge-base fits entirely in the main memory of a single processor, these traversals result in large amounts of virtual memory paging.

In [Lev91], Levinson describes a technique for pattern-oriented (graph) retrieval. Levinson's approach discovers common structures among graphs in a database, and then uses pattern associativity to reduce the required number of tests for subgraph isomorphism. A separate paper, [Lev92], compares techniques for subgraph isomorphism testing suitable for pattern-oriented retrieval. The baseline retrieval method described in [Lev92] considers a flat set of N networks with no pattern associativity information, in which each query requires N isomorphism tests. The first improvement indexes commonly occurring substructures in the graphs, and tests only a subset of the entire database. A second improvement creates a multilevel index of subgraph relationships between successive levels of substructures and the domain of graphs in the database (establishes a partial order). A final elaboration applies multilevel indexing to connectivity and label information required during subgraph isomorphism testing (via the refinement method), rather than to the graph substructures themselves. This produces a tree of "node descriptors" in order of increasing specificity, with actual index pointers at its leaves.

In KEYNET we *approximate* subgraph isomorphism testing to determine the relevance of information objects that were previously annotated with content labels. Our approach compiles fragments of content labels into a hash table, reducing lookup computation (while increasing database construction overhead). Although some ambiguity remains possible when matching fragments instead of whole graphs, vertex overlap between components that contain vertex-incident edges ensures similarity of graph structure between a query and a content label that is retrieved using its fragments. This approximation further allows "simultaneous" (multi-process, distributed or parallel) matching of different regions of the query and content labels (rather than serialized path-oriented inspection).

The KNOWIT system of Sølvberg, Nordbø and Aamodt [SNA92] embeds a semantic network in a front-end query refinement system. Queries to the ESA-QUEST bibliographic database are expanded according to a semantic model that describes the meaning of a concept entirely in terms of its relations to other concepts in the model. The KNOWIT system is a front-end to a traditional IR system, whereas KEYNET uses semantic networks as part of its search strategy.

The Government Information Locator Service (GILS)[GIL] is an example of a second-level retrieval mechanism in which the result of a search is another search engine rather than an information object. The KEYNET model can also be applied to to instances of itself, producing a quasi-encyclopedic classification of information objects. The following diagram suggests how one could organize search engines in the NII:



Using currently available technology, each search engine could support collections having a few million information objects. The top-level search engine could, in turn, support one million search engines. The entire structure would therefore index 10^{12} information objects having an aggregate storage size of 10^{16} bytes, i.e., 10,000 Terabytes.

5 System Architecture.

The KEYNET information retrieval technique is designed to be used in a highly distributed environment, and assumes that the information objects themselves are widely distributed. Information objects need not be textual and may be physically located anywhere in the network.

Retrieval is accomplished by means of *content labels* for each information object. These content labels are stored in a repository at the KEYNET site. Structure that exists among the content labels is defined by a schema called the *ontology* that is a substantial database in its own right. For more details about its structure, see [BS94a]. 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 actual information object. The KEYNET system is only concerned with finding information objects. Acquiring (and paying for) objects is a separate issue.

To see more precisely where all of these components reside, and how they are connected to one another, refer to Figure 3. The user's computer, responsible for presentation (user-interface) services, is at the upper left. A copy of the ontology is kept locally at the user site. As this will typically require several hundred megabytes of memory, it will generally be stored on a CD-ROM. The ontology is also the basis for the user interface to the search

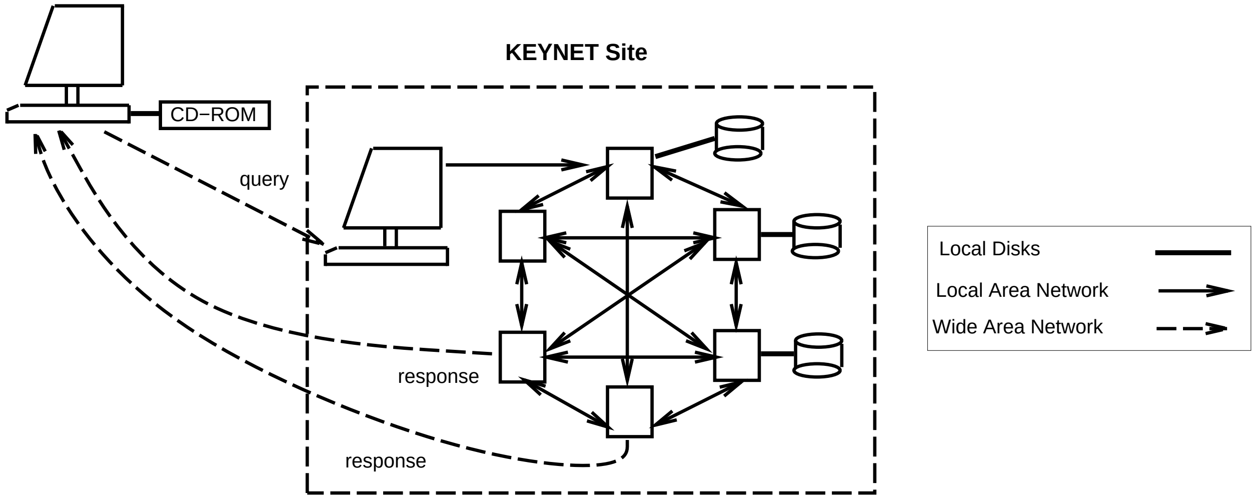


Figure 3: Architecture of KEYNET Search Engine

engine[BF93]. Queries must conform to structure 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 (making use of the ontology to display them).

At the KEYNET site, the front-end computer is responsible for relaying query requests to one of the search engine computers. The purpose of the front-end computer is mainly to distribute 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 are 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.

Queries are answered by fragmenting them into small graph components, called *probes*, each having a bounded size. Components similarly obtained from content labels are called *index terms*.

The result of the fragmentation step is a large number of probes that can be indexed in parallel. The indexing step is analogous to the technique used by biologists to study the genome. A chromosome (a very long strand of DNA) is probed using small pieces of DNA which can attach to the chromosome only where they match in a precise fashion. In fact, a KEYNET system can be used for the problem of mapping long strands of DNA called *clones* to their locations in the chromosome by regarding the clones as "information objects" which are "retrieved" using probes.

The basic indexing strategy of KEYNET is to match probes (fragments of queries) with index terms (fragments of content labels). We now give an outline of the distributed algorithm that accomplishes this matching. For more details see [BS94b]. This algorithm can be characterized as a "scatter-gather" technique. Queries are sent to a front-end processor that 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 earlier. Each probe is then hashed using a standard algorithm. 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. 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. 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. The home node then computes the similarity measure (currently the cosine measure) of each object in the collection, and the objects are ordered by their degree of similarity. The object identifiers of the most relevant objects are then sent back to the user. More sophisticated techniques (including explicit graph isomorphism testing) can be applied as post-processing filters.

The insertion of a new content label in the index is done in a manner very similar to the query algorithm. 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.

6 Formats and Protocol.

KEYNET uses the UDP protocol of the TCP/IP communications protocol suite. In this section we describe the format and protocol for KEYNET communication. Since KEYNET assumes that the ontology is available at both the sender and the receiver, it is unnecessary for the format to include any textual information, and all fields consist of integers. For convenience in communicating between different machine architectures, all integers are represented as *network integers*. For simplicity in the following, we have omitted the specification of acknowledgement and error structures.

6.1 Keynet Structure.

Both queries and content labels use the same data structure, called the keynet structure and defined as follows:

1. **Ontology Identifier.** Each ontology is assigned a unique integer identifier.

2. **Major version number.** Since ontologies can evolved over time, a version number is used to distinguish both major and minor versions of an ontology. Minor versions differ from one another only by the addition of new concepts, conceptual categories and relationship types. Major versions may differ in more substantial ways, including the splitting of categories, merging of categories, as well as more complex alterations in the ontology.

3. **Minor version number.**

4. **Count.** Most keynets are expected to be small enough to fit into a single UDP packet, but there is a mechanism for larger keynets. The count field specifies that the keynet has been split into a number of pieces as specified in this field. Normally the value of this field is 1.

5. **Sequence number.** When a keynet is in several pieces, this field will have the sequence number of this piece. The values range from 0 to one less than the count field above.

6. **Vertex count.** This is the count of the number of vertices that will be specified in the list immediately following this field.

7. **Vertex list.** This is a set of zero or more vertex specifications. Each vertex specification consists of three integers as follows:

   (a) **Vertex id.** Each vertex of a keynet has a unique identifier within the keynet.

   (b) **Type id.** Each vertex has a type or conceptual category. The types and their identifiers are listed in the ontology.

   (c) **Instance id.** A vertex may be instantiated using one of the instances given in the ontology. Instance identifiers are nonzero integers. If this field has value 0, then the vertex has not been instantiated.

8. **Edge count.** This is the count of the number of edges that will be specified in the list immediately following this field.

9. **Edge list.** This is a set of zero or more edge specifications. Each edge specification consists of three integers as follows:

   (a) **Source vertex.** This is the vertex identifier of the source of this edge in the keynet.

   (b) **Destination vertex.** This is the vertex identifier of the destination of this edge in the keynet.

   (c) **Edge type id.** Each edge has a type. The edge or relationship types and their identifiers are listed in the ontology.

6.2 Query Structure.

The query structure is used to send queries to a `KEYNET` search engine. This same structure is also used within the search engine for sending queries from one node to another. The query structure has the following structure:

1. **Message type specifier.** The first field is used to specify to a `KEYNET` search engine that this packet contains a query or a part of one.

2. **Source internet number.** This is the internet number of the internet node that originated the query. If this node sets this field to zero, then the engine will fill in its value.

3. **Source port number.** This is the port number used to send the query. Like the internet number, if this is set to zero by the originator, then the search engine will fill in its value.

4. **Source identifier.** This is a source-generated query identifier which may be used by the system that originates the query to distinguish its queries from each other. However, it does not distinguish queries originating from different locations.

5. **Engine identifier.** This is an identifier supplied by the search engine that uniquely distinguishes queries from each other within the search engine.

6. **Keynet structure.** See above for this structure.

6.3 Content Labels.

The structure of a content label is almost identical to that of a query. The only difference is that the first field contains a message type specifier appropriate to a content label rather than a query. This structure is used for registering a content label in the search engine repository.

6.4 Query Responses.

A query response structure is somewhat more complex than the structure for a content label because there will generally be several responses to a given query, and it is necessary to specify the query that is being answered using a given query response packet.

1. **Message type specifier.** This specifies that this structure contains a query response.

2. **Response count.** This is the number of responses that were generated by the query.

3. **Response rank.** This the sequence number of the response within the list of all responses to one query in order by their salience with the first response being the one judged to be most salient.

4. **Weight.** This is a measure of the salience of the response. Normally this will be expressed as a real number between 0 and 1. To store this number in an integer field, it is multiplied by a denominator (such as 1,000,000) that is specified in the ontology.

5. **Source identifier.** This is the source-generate query identifier from the original query.

6. **Engine identifier.** This is the engine-generated identifier from the original query.

7. **Keynet structure.** See above for this structure.

6.5 Protocol.

The protocol for queries is as follows. The query originator sends a query packet to the front-end of the search engine. The front-end immediately responds with an acknowledgement packet that includes the identifier assigned to the query by the search engine. The front-end fills in missing field of the query packet and forwards the query to a randomly chosen home node for the query. The search engine uses the keynet algorithm to find and rank the desired responses. Each of these is sent back to the query originator from whichever search engine node finds it first. The query originator is responsible for collecting the responses, arranging them in the correct order, and then displaying them.

7 Summary.

KEYNET is a graph-oriented method for information object indexing and retrieval. Information Objects must be annotated with small semantic networks that represent their key concepts. A larger semantic network (part of a subject-specific ontology) determines which node and link types (basic concepts and relations) are considered pertinent to a subject during information object retrieval.

The query graph actually used for retrieval may substitute more general or specific concepts for those specified by the user. Retrieval does *not* match large components of the query graph against whole content labels. Instead, the query graph is *fragmented* into small probes of bounded size. These fragments are matched against content labels, and resulting retrieval sets of potentially relevant information objects are combined using fragment-oriented weights.

The graph representations, their fragmentation, and post-retrieval merging of information object sets associated with distinct fragments naturally introduce a “fuzziness” appropriate to information retrieval notions of relevance, and facilitate use of distributed or parallel processing resources (at appropriate stages).

The KEYNET system employs a number of optimizations to ensure that it scales up to large corpora and so that it has high performance. Fragmentation combined with pattern associativity of graph structures and linear hashing techniques produces tractable complexity of communications and computation, despite necessary isomorphism testing and index manipulation. The system is therefore compatible with the requirements for search engines needed in proposals for an NII.

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.

[BS94a] K. Baclawski and D. Simovici. An abstract model for semantically rich information retrieval. *Information Systems*, 1994. to be submitted.

[BS94b] K. Baclawski and J. E. Smith. A distributed approach to high-performance information retrieval. In *Proc. IEEE Sympos. Par. Distr. Processing*, 1994. Submitted.

[BS94c] K. Baclawski and J. E. Smith. High-performance, distributed information retrieval. Technical Report NU-CCS-94-05, Northeastern University, College of Computer Science, 1994.

[BS94d] K. Baclawski and J. E. Smith. A unified approach to high-performance, vector-based information retrieval. In *RIAO'94*, 1994. submitted.

[FBY92] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval / data structures and algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1992.

[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.

[GIL] Government information locator service (GILS): Draft report to the Information Infrastructure Task Force. Available by anonymous ftp from 130.11.48.107 as /pub/gils.txt.

[HL93] Betsy L. Humphreys and Donald A.B. Lindberg. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170, Apr 1 1993.

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

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

[Lev91] Robert Levinson. A self-organizing pattern retrieval system and its applications. *International Journal of Intelligent Systems*, 6:717–738, 1991.

[Lev92] Robert Levinson. Pattern associativity and the retrieval of semantic networks. *Computers and Mathematics with Applications*, 23(6-9):573–600, 1992.

[LHM93] D.A.B. Lindberg, B.L. Humphreys, and A.T. McCray. The Unified Medical Language System. *Methods of information in medicine*, 32(4):281, Aug 1 1993.

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

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