Addressing the Requirements of a Dynamic Corporate Textual Information Base

Peter G. Anick, Rex A. Flynn, and David R. Hanssen

Intelligent Information Applications Development Group
Digital Equipment Corporation
290 Donald Lynch Blvd. DLB5-2/B4
Marlboro, MA 01752-0749

ABSTRACT

AI-STARS is a lexicon-assisted full-text Information Retrieval system, designed for use in a dynamic corporate environment. In this paper, we explore how the requirements of such an environment have influenced many key aspects of the design and implementation of the AI-STARS system. We promote the use of "views" to create logical partitions in large, heterogeneous databases, and argue that storing not only article instances, but also class definitions, stored queries, display templates and linguistic data in a single object repository has consequences that can be exploited for schema and lexicon evolution, security and subject filtering, information navigation, and data distribution.

1 INTRODUCTION

The AI-STARS project is an on-going research program at Digital Equipment Corporation, investigating methods for improving full-text information retrieval. Our target audience is Digital's Customer Support specialists, for whom ready access to on-line technical information is indispensable for quick and accurate handling of a wide range and heavy volume of customer inquiries. For the past six years, Digital's Support Centers have been using the internally developed full-text information retrieval system, STARS. This has made it possible for the AI-STARS team to observe first-hand the strengths and weaknesses of full-text information retrieval in a specific application environment. While a major focus of our research continues to be directed at improving query processing and interactive query reformulation via the incorporation of computational linguistic techniques and direct manipulation graphical interfaces, the recent growth in the quantity and variety of on-line data at the Customer Support Centers has led us to consider a number of other architectural issues as well.

A key aspect of our target domain is that the on-line textual information is intended to serve the needs of a worldwide, heterogeneous user community made up of thousands of service specialists, engineers, and external customers. As with many large corporate database applications, data security, schema evolution, and data distribution are as important as effective querying facilities and short response times. As a result, a good portion of our research effort has been aimed at developing an infrastructure that will allow us to garner the benefits of adding Computational Linguistics and Artificial Intelligence technology to full-text Information Retrieval within the practical constraints imposed by the actual operational environment.

This paper provides an overview of our experience to date. We begin by introducing the approach taken in AI-STARS for full-text indexing and searching. We then turn to the operational requirements of the Customer Service application and indicate why we chose an object-oriented approach to representing semi-structured information objects. Next we consider querying on semi-structured objects and show how the notion of storing queries can be exploited to support multiple logical "views" of large databases. We then describe how our lexical and linguistic knowledge can take advantage of the object-oriented infrastructure as well, arguing that a uniform representation of heterogeneous objects simplifies the tasks of data security, partitioning, and distribution. We conclude with a discussion of implementation status and issues, related research, and user interface implications.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1991 ACM 0-89791-448-1/91/0009/0163...$1.50

2 FULL-TEXT INDEXING AND SEARCHING IN AI-STARS

We refer to AI-STARS as a "lexicon-assisted" retrieval system because lexical information, both grammatical and relational, is maintained on-line for use in full-text indexing, natural language query processing, and interactive query reformulation. Utilizing a morphological analyzer in conjunction with an on-line lexicon, AI-STARS indexes articles according to the (uninflected) citation forms of all known words and phrases in the text. In this way, the end-user does not have to use truncation operators to match texts containing morphological variants of a search term nor use proximity constraints to indicate when a sequence of search terms is intended to make a canned phrase. The user queries the system with a "natural language" query, which is converted into a Boolean expression composed of the underlying citation forms. In addition, a direct manipulation "Query Reformulation Workspace" window [ANICK90] is optionally available to the user to inspect the results of the natural language analysis of the query and reformulate the query interactively, with the help of an integrated on-line thesaurus. Figure 1 shows a sample natural language query and the accompanying Query Reformulation Workspace. As the intent of the workspace is to provide a convenient way to visualize and manipulate Boolean search expressions, it can be employed both to quickly reformulate one-shot search expressions as well as to hand tailor carefully designed queries for later (repeated) reuse.

3 REQUIREMENTS OF A DYNAMIC INFORMATION ENVIRONMENT

The past several years have seen a spectacular increase in the amount of on-line information available to Customer Support specialists. Their 20,000 article database of symptom-solution descriptions, generated by specialists who wished to share their problem-solving experience with other specialists and customers, has mushroomed into over 150,000 articles from a vast variety of sources, including corporate publications, internal bulletin boards, and external support documentation. While this growth has greatly increased the information potentially available for problem solving, it has also brought with it a host of new requirements for on-line information management. In this section, we will enumerate these concerns in order to motivate the discussion of system design in the sections that follow.

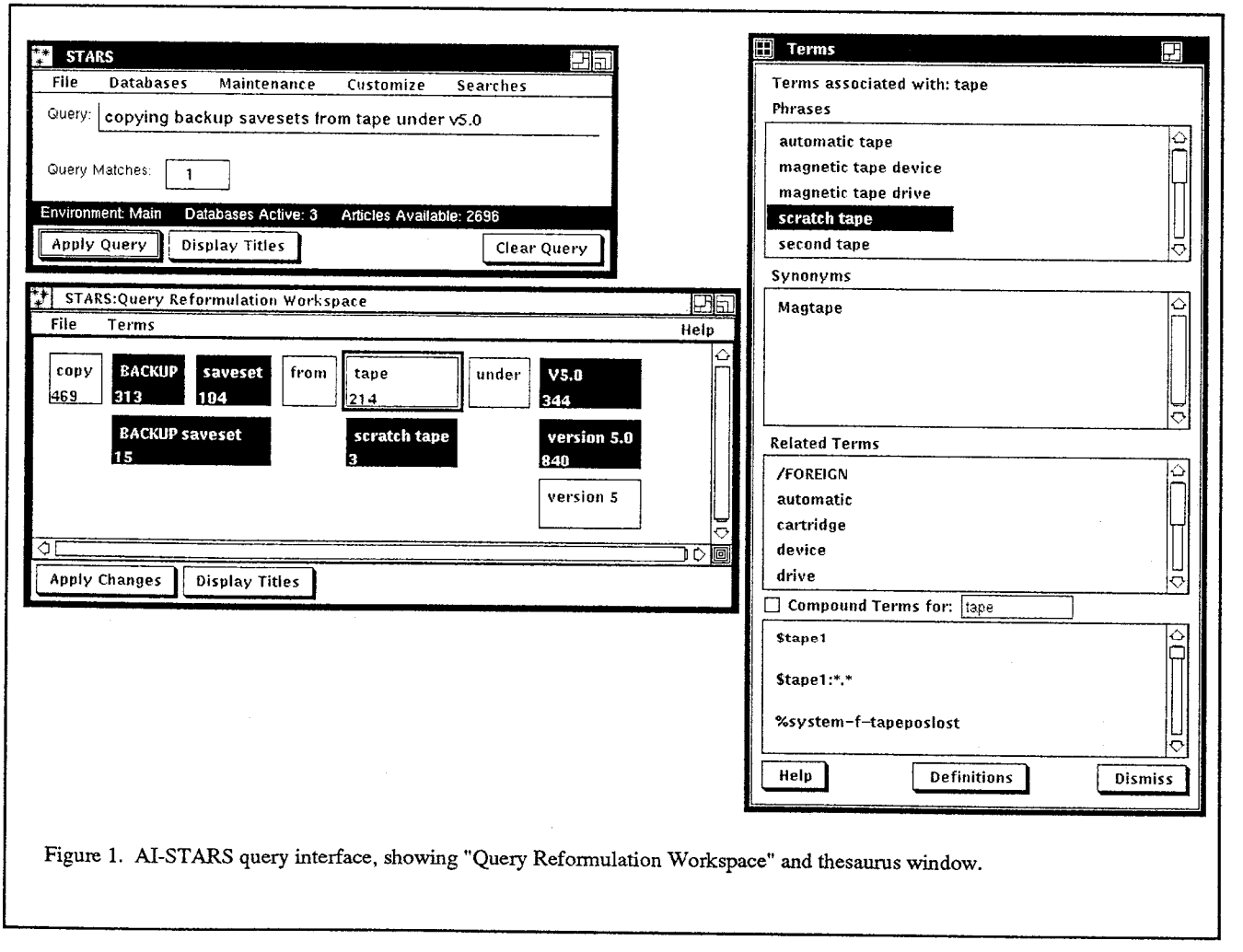

Figure 1. AI-STARS query interface, showing "Query Reformulation Workspace" and thesaurus window.

3.1 INFORMATION PARTITIONING

The rapid growth of on-line information has exacerbated the perennial problem of organizing the data for effective retrieval. In the current STARS system, administrators assign each article to a database on the basis of the product it is mainly about as well as the kind of article (e.g. problem report or symptom-solution article). However, many articles deal with multiple products, making it difficult to decide how to categorize articles into the set of databases available, and, conversely, making it difficult for searchers to know which databases to open for a given query.

3.2 HETEROGENEOUS DATA

The typical information object in STARS is a 1-2 page "chunk" of information, composed of both structured fields and fields containing unstructured text. Although most objects share many of the same fields, there are some classes of objects with specialized sets of fields. For example, articles about crash dumps contain fields for CPU, operating system, and program counter location in addition to the textual description of the problem and solution. As the current version of STARS supports only a single data format, such specialized fields are not queryable. The specialized information must be included as part of the unstructured textual data. Clearly, support specialists could be more effective retrieving relevant articles if this specialized structural knowledge could be queried explicitly.

3.3 DYNAMIC SCHEMA

If the system is to support heterogeneous article classes, it must, in addition, support the evolution of the database over time. Not only will new classes of information need to be added to the system as new sources of data become available, but the existing classes may require modification as business needs change and new fields are found to be useful.

3.4 HYPERINFORMATION LINKS

Certain kinds of information being imported into the on-line database, such as bulletin board postings and replies, are most naturally represented as linked collections of information. On retrieving a posting, one may want to navigate through the replies. Likewise, large documents broken into display-size "chunks" will need **previous** and **next** pointers to allow for sequential access in addition to the "random" access provided by content-based retrieval. Beyond such basic requirements, the value of hyperinformation links for connecting objects in a large information space is well known [NIELSEN89] and should be made available to specialists as an alternative search strategy.

3.5 DATA SECURITY

With thousands of users, including many levels of internal specialists and numerous external customers, on-line information must be appropriately protected. Junior specialists may need to be restricted from retrieving articles that have not yet been reviewed for accuracy. Customers must not be allowed access to articles containing sensitive or proprietary information. A flexible security mechanism which does not depend on separate physical databases for different audiences is required.

3.6 DATA DISTRIBUTION

The on-line Customer Service databases are used by support personnel world-wide. Although remote access to a central database is one approach to sharing data, many sites prefer to store copies of data locally to guarantee fast access to important subsets of the information base. In a database in which the schema may change over time, this means distributing not only article data but also data about new types of articles and changes to existing types.

4 AN OBJECT-ORIENTED DATABASE APPROACH

The need to support heterogeneous data in a dynamically evolving information environment has led us to pursue an object-oriented approach to data representation. In an object-oriented database (OODB) [ZDONIK90], the behavior of an object is typically defined by the *class* (or *type*) it is an *instance* of. Classes in turn are usually organized into an inheritance hierarchy, in which the behaviors of superclasses are inherited by their subclasses, unless explicitly overridden. The state of an object, as calculated by the *methods* defined for it by its class, may include references to other objects. OODBs also typically provide mechanisms to dynamically add or modify class definitions and support versioning of objects (e.g. [AHLSEN84]). In this section we discuss how we are applying these concepts towards satisfying the requirements of a corporate information base as described above.

4.1 APPLYING OBJECT-ORIENTED DATABASE TECHNIQUES TO AI-STARS

The object-oriented approach addresses the needs of our application environment in the following ways:

- It allows the addition and modification of information classes without altering system software.
- It allows classes to be organized into inheritance hierarchies where subtypes can augment the fields defined by their parent classes.
- It allows the definition and support of multiple heterogeneous classes, and enables uniform mechanisms to query against all instances in the database, regardless of their class.
- It provides a framework for dealing with semi-structured objects, composed of combinations of structured fields such as dates, and unstructured fields such as text bodies. Queries express Boolean combinations of restrictions based on fields and their associated values.
- Hyperinformation links are supported by defining fields which refer to other objects, allowing us to create networks of information objects relating, for example, product descriptions to problem reports to bug fixes.

- By adopting a self-describing metamodel, we obtain a uniform representation for classes (schema), information object instances, stored queries, and all other information which is needed at runtime (and which requires distribution among sites.)
- One can develop a single uniform mechanism for distributing any object regardless of its class.
- It provides a framework for managing multiple versions of objects, and synchronizing the view of an object in the context of an evolving schema.
- Through versioning, we are enabled to work in a distributed environment where updates propagate at finite velocities, and different sites may have different definitions of the same classes.

While we borrowed the above ideas freely from object-oriented databases, we chose not to fully encapsulate instance behavior within methods. This was a useful short term expedient, simplifying the design and development of the system.

4.2 AI-STARS DATA MODEL

AI-STARS has a built-in self-describing data model much like that of SMALLTALK [GOLDBERG83]. Figure 2 shows the basic classes and instances which form the top level of our object hierarchy. In order to create an object-oriented model suitable for use in an actual Information Retrieval application, these classes must be augmented with application-specific classes.

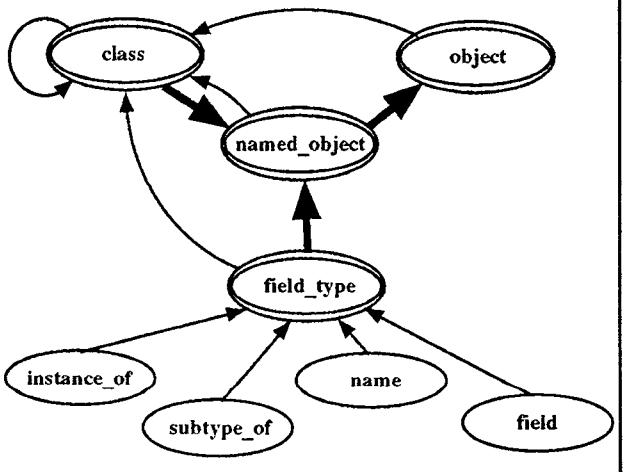

Figure 2. Top level of the AI-STARS data model. Thin arrows indicate **instance-of** links while bold arrows indicate **subclass-of** links. Double ovals indicate class objects; single ovals indicate instances.

For example, classes may be defined to correspond to the structure of mail messages, bulletin board entries, interdepartmental memos, product releases, manuals, etc. Such classes will typically be created either as direct subclasses of the "Object" class, or as subclasses of other application-specific classes. Inheritance facilitates the construction of specialized classes from more abstract class defintions.

As needed, users can also create application-specific fields, such as **reviewer_name**, **modification_date**, etc. New fields are defined by creating instances of the "Field-Type" class, which specify the new field's value type. AI-STARS supplies a number of predefined value types such as "Text", which is parsed prior to indexing; "String", which is not parsed; "Object", which allows reference to arbitrary objects; and "Date". There are also predefined enumerated value types which are used to control storage class, versioning, index generation, etc.

Field names are globally defined to facilitate their consistent use across all object classes. Because queries are usually constructed with respect to multiple classes at the same time, it is important that all classes use field names in a consistent manner, and do not use different field names for the same semantic property.

4.2.1 QUERYING ON SEMI-STRUCTURED INFORMATION OBJECTS

The need to support queries that include both structured and unstructured field values and return article sets containing objects of multiple classes has required extending our query interface, which was originally designed for full-text querying only. In our original user interface (see figure 1), the system's interpretation of a natural language query was made accessible for viewing and modification by the user through a graphical "Query Reformulation Workspace" window. In our new interface (figure 3), we allow the user to explicitly specify fields and relational operators in addition to their values. We associate a separate Query Reformulation Workspace with each field restriction, displayable on demand. By default, field-value restrictions are ANDed together. A special top-level workspace is available for modifying the Boolean relationships among the various field-value restrictions via direct manipulation of graphical tiles. This division into multiple workspaces permits users to interactively adjust any portion of the query independently, without affecting the other query components.

We allow an object to contain multiple text fields with different field names, so that, for example, a single class could have a **symptom_text** field and a **solution_text** field. User queries may be expressed which refer to these specific field names, or may refer to a special system-supported **all-text** field Queries with respect to the **all-text** field refer to any field whose value is of type "Text". In this way, we allow querying on textual information without requiring the user to know all the different ways people have chosen to subdivide their texts (similar to the approach in [McALPINE89] and [BERTINO88]).

Control over the formatting of objects for display is provided through *templates*. Templates define which fields of object in-

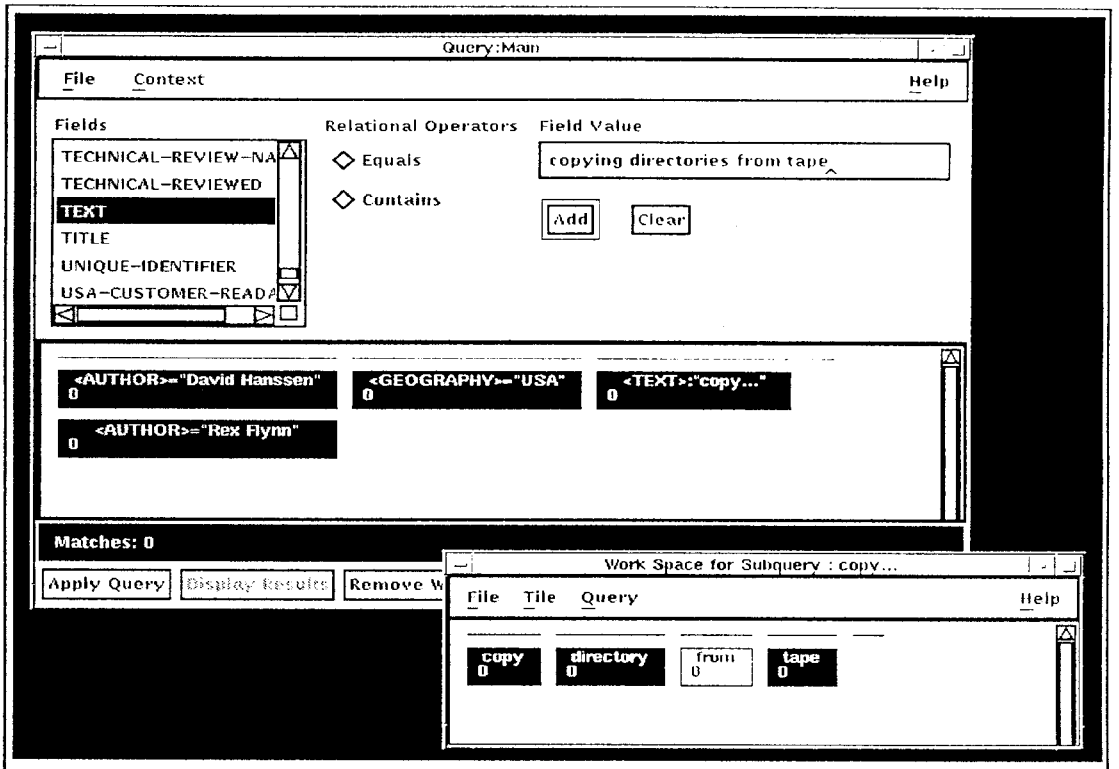

Figure 3. Query interface for queries containing multiple restrictions. Subquery window shows the workspace for one of the restrictions.

stances of a given class are to be presented on the screen, as well as their relative positioning. An object class may have one or more different templates, which are provided by database administrators as part of object class definition. Since a single query may return a heterogeneous set, we associate article classes with icons and display the icons along with article titles in the title list returned as the result of a query. If users wish to further differentiate the results of a search according to article class, they can request that the title list be sorted by class.

5 STORED QUERIES AND VIEWS

5.1 VIEWS

In Relational Database systems, a user can define a "view" of a database using a relational expression [DATE90]. A view is a *virtual* table, providing a dynamic window into the actual database tables. Views have a number of advantages, such as allowing the user to focus solely on the subset of the data that is of concern, and providing automatic security for data hidden from the view. A device which provides some of the same capabilities in an Information Retrieval system is the stored query, which defines a dynamic subset of the information objects in the database. We have chosen to utilize stored queries in AI-STARS to address the issues of data security and partitioning.

5.2 STORED QUERIES

Stored queries may be constructed by database administrators or by the users themselves. As objects defined within the AI-STARS class hierarchy, they have names and textual description fields. Users can construct queries which return these stored queries based on the stored textual descriptions, and can add queries so obtained as conjuncts or disjuncts within the context of other queries.

When a stored query is employed as part of a larger query, it is represented visually in the user's "Query Reformulation Workspace" as a single tile, just like any other query component, such as a word or phrase. If refinement of the stored query is required for query reformulation, the user may open the stored query tile for manipulation of its components as shown in figure 4. Stored queries may contain references to other stored queries, and may be nested to arbitrary depth.

5.3 VIEWS FOR SECURITY

As noted earlier, articles in the Customer Service Center databases contain not only text but also a collection of structured fields, indicating, among other things, whether the article has been technically reviewed for accuracy and whether it is intended to be read by external customers. Based on such flags, the articles can be extracted into separate databases to be made available to specific audiences, such as junior specialists or out-

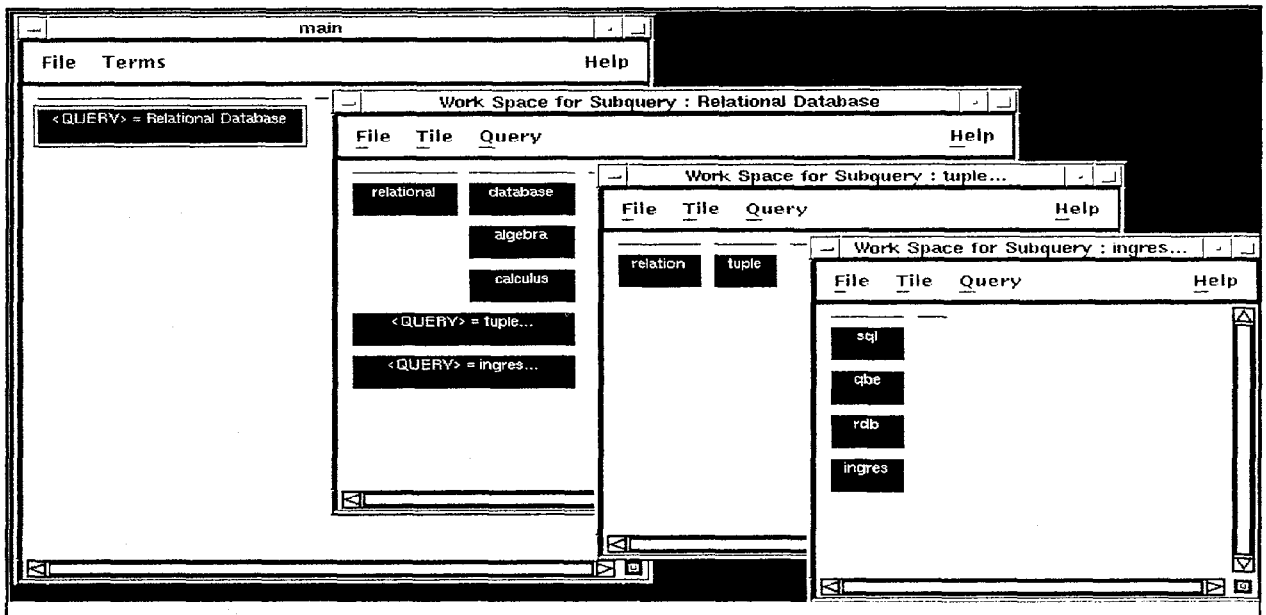

Figure 4. Visual representation of a stored query with nested subqueries. The corresponding Boolean expression is (OR (AND "relational" (OR "database" "algebra" "calculus")) (AND "relation" "tuple") (OR "sql" "qbe" "rdb" "ingres")).

side customers. Alternatively, one can use stored query views to create alternate windows on the same physical data. In AI-STARS, each user has a profile which includes a security view. This query is run against the full database when the user logs in in order to create a logical subset of data viewable by the user for the remainder of the session. Consequently, the level of granularity for security is the information object, not the database. If an information object is modified such that it comes to satisfy a view that it previously did not (as in changing the value of the customer-readable field to "true"), then this object becomes visible to all users who share that security view.

5.4 VIEWS FOR SUBJECT AREAS

One of the problems that has intensified as the size of the information base has expanded is that of classifying articles into named databases, where the database names typically refer to specific products. This is due to the fact that many articles either fit into many possible product categories or would be better classified according to some other dimension. In order to circumvent this problem, we have opted to create the illusion of a single universal database. Stored queries may be applied to this universal database in order to partition it along any number of (potentially ad hoc) dimensions. The work of creating such queries is left to a database administrator knowledgeable about the contents and the needs of the end-user community. The administrator creates a number of stored query views; these are presented to the end-user through the user interface as selectable subject areas. The user has the option of either querying the entire on-line information base, or first restricting his/her view along the twin dimensions of subject area and article class. Thus, for example, a user may select to limit his/her view to the subjects "relational databases" and "query interfaces" and to the classes "problem report" and "product description". Thereafter, all queries issued by that user are additionally filtered through the view defined by the intersection of the security, class, and subject views. Unlike the security view, the class and subject views may be altered at any time during the session.

This portion of our interface has much in common with the facet-based search interface of OAKASSIST [MEADOW89], [BORGMAN89]. Just as the OAKASSIST user may edit search terms within a "facet" window, the AI-STARS user may optionally refine selected "subjects" by direct manipulation of the contents of the stored queries. By giving users the ability to construct queries via both direct selection of malleable topics and natural language query, we hope to combine the advantages of these two basic input modes to facilitate progressive search formulation, in the spirit of [BELKIN90] and [CROFT87].

6 REPRESENTING LEXICAL KNOWLEDGE

AI-STARS indexes articles with respect to an on-line lexicon of words and phrases. If the lexicon could be assumed to be static, then a special purpose static in-memory data structure containing all lexicon items could be compiled and linked into the Information Retrieval software to support morphological analysis. On the other hand, if the lexicon is dynamic, allowing the addition of new entries over time, then lexical entries must be stored in a malleable form. Because of the dynamic nature of technical vocabulary, we chose to implement a dynamic lexicon, making use

of the storage and retrieval machinery already developed to handle our textual information objects.

6.1 LEXICON ENTRIES

Each lexical entry is actually an aggregate of three separate object types, the *stem*, the *lemma*, and the *word sense*. In morphological analysis, a surface form is reduced to one or more potential stem forms and corresponding sets of constraints, such as the expected part of speech and inflectional paradigm. These constraints are used to construct a query to retrieve any stems in the lexicon which meet the stipulated conditions. As shown in figure 5, each stem object in the database is associated with a single lemma object, which serves as a handle for the entire dictionary entry. From a lemma, one can reach all the stems for a given word as well as all the word senses. Word sense objects can optionally contain textual definitions and semantic links to other word senses, thereby supporting the construction of an on-line sense-disambiguated thesaurus. Although it would be highly desirable to index articles with respect to word senses [KROVETZ89], due to the difficulty of word sense disambiguation, we currently index instead with respect to citation forms, which are stored as strings in the lemma objects reachable through morphological analysis of surface forms.

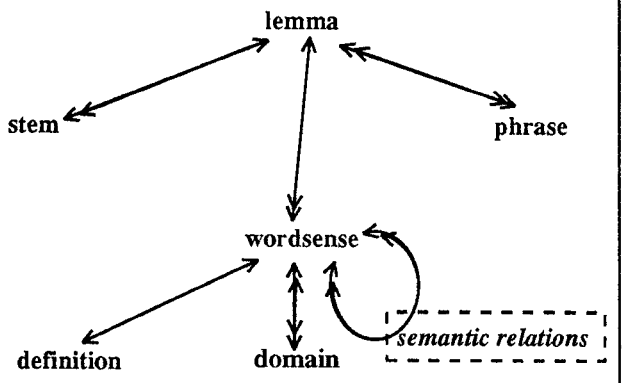

Figure 5. Relationships among lexicon objects. Single and double arrow heads indicate whether relationships are one-to-one, one-to-many, or many-to-many.

6.2 LEXICON VIEWS

In a corporate environment, certain words (such as names of internal projects or unreleased product code names) may be classified. These terms, their definitions, and any thesaurus links based on the classified senses should only be visible to users with the proper security rights. By treating lexicon objects as full-fledged information objects, one can attach fields to them for the purpose of security classification and use the stored query view mechanism to assure that only the authorized audience has access to these terms.

6.3 LEXICON DOMAINS

The words used in articles may belong to many overlapping technical domains. For example, the noun "generation" is used in computing both to describe a particular version (of a file in a source code control system) and to describe the process of configuration (setting the parameters for an operating system). The existence of such semantic ambiguities may diminish the effectiveness of the on-line thesaurus. In the case of the word "generation," if the thesaurus provided the related concepts "version" and "configuration," then a query reformulation strategy may bring in both terms when only one is appropriate.

The general solution to this problem involves performing word sense disambiguation. Since we have chosen not to index with respect to word senses, we are exploring a few weaker (and simpler) alternatives. The first arises from the observation that a querier on AI-STARS uses the lexicon within the context of a particular view of the database. We have therefore chosen to restrict the lexicon (and hence the thesaurus) dynamically to only those words and phrases that appear in articles in the querier's current view. There is little information content in displaying words and phrases which are in the lexicon, but match no articles. As a consequence, in the case where a querier's view contains only articles about version control, the word "configuration" is not likely to show up, and the ambiguity may be avoided. Another alternative we are exploring is tagging word senses with a domain field. This will allow a user to pre-select a subset of the word senses with the same mechanisms already available for defining views, e.g. by executing a query on the value of the domain field.

6.4 STORING LINGUISTIC RULES

In our current protoytpe, the linguistic rules for both morphological and syntactic analysis are compiled from their textual representation into C source code files, which are in turn compiled and linked into the AI-STARS image. This has the disadvantage that changing a rule or adding a new ruleset (as, for example, a morphological ruleset for a language other than English) requires a software upgrade. In a corporate environment with perhaps hundreds of sites using the software, it pays to minimize the frequency of such upgrades. One approach we are exploring is to compile rulesets into ruleset objects which can then be loaded into the system as data and distributed in a manner similar to other database objects.

7 IMPLEMENTATION

The first prototype of the AI-STARS system was completed in August 1990. Informal demonstrations of this system met with an enthusiastic repsonse from current STARS users. However, transforming the prototype into a fully functional substitute for STARS, to enable true on-the-job evaluation, has required a complete reimplementation, as we needed to address the additional problems outlined in section 3.

We will shortly begin a test of a new AI-STARS system with all the features described above except for the mechanisms for dis-

tribution. The corporate environment in which we intend to deploy this system necessitates a high level of performance, and work to achieve these performance goals is on-going. While a thorough treatment of implementation details and issues is beyond the scope of this paper, we will present here a short discussion of our performance requirements and some of the implementation techniques we have pursued as a consequence.

7.1 PERFORMANCE REQUIREMENTS

As with other IR systems, our system must be optimized towards performing queries. The desirability of enabling specialists to answer a question in the context of a brief telephone conversation dictates that the response time for a query must be on the order of seconds. What we have added to this problem is the need to access a dynamic lexicon on the fly to interpret the query, and the complexities of dealing with heterogeneous objects.

What may be a less obvious problem is that of loading existing databases into AI-STARS. Loading an article involves lexical access at the inner loop, as well as a massive amount of index updating. On the whole, adding an article to the database need not be a fast operation (it can be an order or two magnitude slower than querying). However, in converting from the existing STARS to our new database, we need to load 150,000 articles. A quick calculation shows that at the rate of 30 sec. per article, the amount of time required to re-load the database is 52 days.

7.2 CACHING

Performing a couple of I/O's to retrieve the citation form for every word in an article with 500 words (a small article) alone would take 15 sec. (for 2 I/O's per word, 15 ms. per I/O). We have therefore implemented a cache for all objects and indexes that flushes the least recently used entries. Our intention for queries is to cache the terms in the lexicon that the querier uses repeatedly. Our intention for loading information objects is to cache as much of the lexicon as is normally used in processing the texts.

In theory, such a cache ought to have performance comparable to a static in-memory lexicon. In practice, we have much more tuning to do in this area. This is for a number of reasons:

- There is a considerable amount of computation still involved in performing lexicon operations on the in-memory structures.
- The dynamic nature of the cache necessitates the dynamic acquisition and freeing of memory.
- Our memory structures are not yet specialized for lexicon access alone, but are generic objects. This means essentially that our structures are not optimized for space, thereby taking up more of the available cache.

We are currently pursuing implementing specialized data structures and caching the results of morphological analysis, i.e. associating the surface string with its resulting computation.

One of the benefits of having a single object repository for our system is that providing caching techniques improves access to all the objects. Thus we expect our caching of indexes to benefit users who repeat terms in subsequent queries. The other side is that our cache must be more sophisticated, in that it must deal with entities which vary widely in size and use.

7.3 INCREMENTAL UPDATE

Burkowski [BURKOWSKI90] discusses a mechanism for speeding up indexing of articles by updating the indexes in memory and deferring writing out index blocks until they have been filled with multiple article updates. Since we believe atomic transactions are necessary in any corporate database to preserve data integrity, we have reformulated this idea as one of providing incremental update during a long transaction. We have implemented, but not tested out extensively, an "update" cache, where objects and indexes are updated in memory during the transaction, and those that are updated most infrequently fall to the bottom of the cache, and are written out. Our hope is that this kind of updating will allow for a balance of CPU and I/O activity, while deferring the writing of the most frequently updated indexes.

7.4 METADATA INTERPRETATION

Since our system is self-describing, we use the class for each object to interpret an object instance when it is being accessed or updated. When performing queries, the number of classes involved is generally small, so we expect our caching to remove the I/O cost of bringing in the classes for each object. However, there is a computational burden. We are not clear currently on how great this computational burden is; so far we have chosen to avoid it by means of specialized structures for accessing lexicon objects, while incurring it for the other objects. As discussed in section 4.1, a more rigorous object-oriented database implementation than ours would encapsulate access to its objects via methods. These methods could be compiled, requiring little or no data structure navigation to process a method invocation.

7.5 FURTHER WORK

Although we believe there are many benefits arising from using a unified object-oriented infrastructure, these benefits bring with them a more complex implementation, and the requirement to pay attention to many performance details. We expect to have to do much more work improving the performance of our cache. Working in a complex programming environment with multiple object types and inter-object references guarantees that there will be issues involving dynamic memory management. We have addressed some of these but expect the work in memory management to be on-going, coordinated with our work on cache management. Finally, if the need arises, we may have to do additional work in tuning the interpretation of metadata.

The major functional change we intend to work on next is to extend our system to handle distributed access. We have the requirements defined for distribution, and have done some of the design. We expect the implementation of a distributed AI-STARS system to bring up many more performance issues.

8 DISCUSSION

8.1 RELATED WORK

Our model of Information Retrieval has much in common with that developed for the EUROMATH project [McALPINE89]. Designed to support mathematicians in their research work, EUROMATH adopts an object-oriented approach to integrating, within a uniform interface, a number of retrieval techniques for shared, heterogeneous data. Like EUROMATH, AI-STARS can be considered a "Knowledge Worker Support System," and we see our own experience as affirming and extending the model proposed in [McALPINE89]. The fact that additional considerations, such as ease of data distribution, security, schema evolution and natural language processing support can also be accommodated within the same basic model is, we feel, indicative of the value of employing object-oriented techniques within Information Retrieval applications.

Our work has also been influenced by the research on Information Lens [MALONE87] and its successor, Object Lens [LAI88], which explore the virtues of semi-structured information objects as a medium of inter-personal communication for cooperative work. An important focus of this effort is information filtering, the role instantiated by stored query views in our system. Indeed, by associating views with individuals' mail addresses, the AI-STARS system could be employed as an "Anyone" Server [MALONE87], actively broadcasting incoming articles to those parties registering specific interest.

The benefits of linking heterogeneous information objects (including "concepts" connected by thesaurus relationships) have long been a major theme in the work of Croft and his colleagues [CROFT87], [THOMPSON89], [CROFT90]. Although we have done some work on the integration of a thesaurus with our query interface [ANICK90], we have not yet incorporated the wide range of browsing mechanisms developed in systems like I^3R.

The relationship between full-text indexing and hypertext has been investigated by Coombs [COOMBS90]. The rich hypertext environment of the IRIS Intermedia system, with its information webs and within-article anchors provides a level of information navigation not achievable solely with the field-object links currently available in AI-STARS.

[CELENTANO90] employs object-oriented representational techniques to encode domain knowledge in an office document retrieval system, showing how knowledge of office proceedures and document relationships can add yet another dimension to effective document retrieval.

The use of stored queries has been further developed in the RUBRIC system [McCUNE83]. RUBRIC adds a fuzzy logic weighting scheme to stored query trees composed of words, phrases and other stored queries, called "topics" in the system. While the use of fuzzy logic is an intriguing enhancement, it is unclear how much value it can ultimately add in practice over simpler Boolean concept trees, especially given the difficulty of fine-tuning the weights. Since the precise intent of a "topic" is likely to change from user to user, we currently opt to let users do their own fine-tuning of Boolean stored queries at run-time via interactive query reformulation.

8.2 USER INTERFACE

One of the most difficult aspects of complex system design is constructing a user interface that hides that complexity from the end-user. In [ANICK90], we describe how the Boolean interpretation of natural language queries can be made accessible to the user for query reformulation through a direct manipulation interface. Our hypothesis is that by facilitating user controlled query reformulation, the traditional shortcomings of Boolean query can be overcome without recourse to more complicated and costly retrieval models, such as the vector-space or probabilistic models [SALTON89]. Furthermore, in our application domain, users often want the output sorted by temporal recency of the article or its frequency of use in problem solving, making it difficult to take direct advantage of the ranked output produced by these other methods.

A second interface issue regards how much of the underlying object hierarchy the end-user should be aware of. The Object Lens system [LAI88] exposes the entire object hierarchy and allows end-users full control over constructing and modifying classes. Because of the security and consistency needs of the AI-STARS database, we take a more conservative approach, dividing the world into two sets of users: adminstrators and searchers. Administrators are responsible for modifying the class hierarchy, defining fields, managing the knowledge bases, and creating stored query views representing areas of potential interest. For searchers, we "flatten" out the list of visible classes, excluding system metadata and classes for which there are no user accessible instances. User-visible classes have a set of visible fields, which may be used to construct queries. Thus, the searcher's view of the database is as a set of subject areas, article types and fields; one chooses the subject areas and classes of interest, then constructs queries.

9 CONCLUSIONS

In this paper, we have touched upon a number of concerns that arise when lexicon-assisted full-text Information Retrieval is applied to a large corporate text database that is dynamic, heterogeneous, and distributed. Notions from database theory and Artificial Intelligence, such as views, object caching and self-describing representations, have played an important role in our current solution, as have hyperinformation concepts. With an ever-increasing quantity of semi-structured information becoming available to organizations on-line, we believe that the proper mix of these disciplines for a variety of real-world applications is likely to continue to be a major theme for Information Retrieval research for some time to come.

ACKNOWLEDGEMENTS

The evolution of the ideas embodied in AI-STARS and its implementation have been a group effort. Jeffrey Robbins was the principal designer and implementor of the original STARS system. Jeff, along with Bryan Alvey, Norman Lastovica, and James Wagner, of Digital's Colorado Springs Customer Support Center, are collaborating with Intelligent Information Applications Development Group members Suzanne Artemieff, Jong Kim, Clark Wright and the authors on the development of AI-STARS. The entire team's contributions have been essential for the creation and realization of the ideas presented in this paper.

REFERENCES

[AHLSEN84] Ahlsen, M. , A. Bjornerstedt, A. Britts, S. Hulten, and L. Soderlund. An Architecture for Object Management in OIS. ACM Transactions on Office Information Systems, 2(3), 1984.

[ANICK90] Anick, P. G., J. D. Brennan, R. A. Flynn, D. R. Hanssen, B. Alvey and J. M. Robbins. A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query, in Proceedings of ACM/SIGIR '90, Brussels, 1990.

[BELKIN90] Belkin, N. J. and P. G. Marchetti. Determining the Functionality and Features of an Intelligent Interface to an Information Retrieval System, in Proceedings of ACM/SIGIR '90, Brussels, 1990.

[BERTINO88] Bertino, E. et al. Query Processing in a Multimedia Document System. ACM Transactions on Office Information Systems, 6(1), 1988.

[BORGMAN89] Borgman, C. L. and C. T. Meadow. The Design and Evaluation of a Front-End User Interface for Energy Researchers. Journal of the American Society for Information Science, 40, 1989, pp. 99-109.

[BURKOWSKI90] Burkowski, F. J. Surrogate Subsets: A Free Space Management Strategy for the Index of a Text Retrieval System, in Proceedings of ACM/SIGIR '90, Brussels, 1990.

[CELENTANO90] Celentano, A., M. G. Fungini and S. Pozzi. Knowledge-Based Retrieval of Office Documents, in Proceedings of ACM/SIGIR '90, Brussels, 1990.

[COOMBS90] Coombs, J. H. Hypertext, Full Text, and Automatic Linking, in Proceedings of ACM/SIGIR '90, Brussels, 1990.

[CROFT87] Croft, W. B. and R. T. Thompson. I^3R: A New Approach to the Design of Document Retrieval Systems. Journal of the American Society for Information Science, 38, 1987, pp. 389-404.

[CROFT90] Croft, W. B. and R. Das. Experiments with Query Acquisition and Use in Document Retrieval Systems, in Proceedings of ACM/SIGIR '90, Brussels, 1990.

[DATE90] Date, C. J. An Introduction to Database Systems. Addison-Wesley, 1990.

[GOLDBERG83] Goldberg, A. and Robson, D. Smalltalk-80: The Language and Its Implementation. Addison-Wesley, 1983.

[JARDINE71] Jardine, N. and C. J. van Rijsbergen. The Use of Hierarchic Clustering in Information Retrieval. Information Storage and Retrieval, 7(5), 1971, pp. 217-240.

[KAY80] Kay, M., Algorithm Schemata and Data Structures in Syntactic Processing, Xerox Palo Alto Research Center, Tech Report no. CSL-80-12, 1980.

[KROVETZ89] Krovetz, R. and W. B. Croft. Word Sense Disambiguation Using Machine Readable Dictionaries, in Proceedings of ACM/SIGIR '89, Cambridge, 1989.

[LAI88] Lai, K., W. Malone and K. Yu. Object Lens: A "Spreadsheet" for Cooperative Work. ACM Transactions on Office Information Systems, October, 1988.

[MALONE87] Malone, T. W., K. R. Grant, F. A. Turbak, S. A. Brobst and M. D. Cohen. Intelligent Information Sharing Systems. Communications of the ACM, 30(5), 1987.

[McALPINE89] McAlpine, G. and P. Ingwersen. Integrated Information Retrieval in a Knowledge Worker Support System, in Proceedings of ACM/SIGIR '89, Cambridge, 1989.

[McCUNE83] McCune, B. P., R. M. Tong, J. S. Dean, D. G. Shapiro. RUBRIC: A System for Rule-based Information Retrieval. COMPSAC '83.

[MEADOW89] Meadow, C. T., B. A. Cerny, C. L. Borgman and D. O. Case. Online Access to Knowledge: System Design. Journal of the American Society for Information Science, 40, 1989, pp. 86-98.

[NIELSEN89] Neilsen, J. Hypertext and Hypermedia. Academic Press, 1989.

[SALTON78] Salton, G. and A. Wong. Generation and Search of Clustered Files. ACM Transactions on Database Systems, 3(4), 1978, pp. 321-346.

[SALTON89] Salton, G. Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, 1989.

[SALTON90] Salton, G. and C. Buckley. Approaches to Global Text Analysis, in Proceedings of ASIS '90, Toronto, 1990.

[THOMPSON89] Thompson, R. H. and W. B. Croft. Support for Browsing in an Intelligent Text Retrieval System. International Journal of Man-Machine Studies, 30, 1989, pp. 639-668.

[ZDONIK90] Zdonik, S. B. and D. Maier. Fundamentals of Object-Oriented Databases, in S. B. Zdonik and D. Maier, Readings in Object-Oriented Database Systems. Morgan Kaufmann: San Mateo, 1990.