Using Induction to Refine Information Retrieval Strategies

Catherine Baudin * **Barney Pell** **
Artificial Intelligence Research Branch
NASA Ames Research Center
MS 269-2, Moffett Field CA 94035
baudin@ptolemy.arc.nasa.gov pell@ptolemy.arc.nasa.gov

Smadar Kedar
Institute for the Learning Sciences
Northwestern University
1890 Maple ave. Evanston, IL 60201
kedar@ils.nwu.edu

Abstract

Conceptual information retrieval systems use structured document indices, domain knowledge and a set of heuristic retrieval strategies to match user queries with a set of indices describing the document's *content*. Such retrieval strategies increase the set of relevant documents retrieved (increase recall), but at the expense of returning additional irrelevant documents (decrease precision). Usually in conceptual information retrieval systems this tradeoff is managed by hand and with difficulty. This paper discusses ways of managing this tradeoff by the application of standard induction algorithms to refine the retrieval strategies in an engineering design domain. We gathered examples of query/retrieval pairs during the system's operation using feedback from a user on the retrieved information. We then fed these examples to the induction algorithm and generated decision trees that refine the existing set of retrieval strategies. We found that (1) induction improved the *precision* on a set of queries generated by another user, without a significant loss in *recall*, and (2) in an interactive mode, the decision trees pointed out flaws in the retrieval and indexing knowledge and suggested ways to refine the retrieval strategies.

1. Introduction

Conceptual information retrieval systems [Tong 89], [Mauldin 91], [Baudin et al. 93] use structured patterns, rather than keywords or words from a text, to represent the *content* of the information in a document. These systems use domain knowledge and can *reason* about how to select pieces of information related to a user query using the conceptual indices associated with the documents. In particular, when there is no direct match to a particular query, the relationships in the domain model can be used to predict where the answers to a query *might* be documented (increase a system's *recall*) or to narrow down the search for a document that would best answer the query (increase *precision*). This is based on the observation that pieces of information in a document are usually not isolated islands of ideas but are related by implicit rules.

This type of knowledge-based retrieval requires: (1) structured indices that can represent the objects and relationships in a piece of information, (2) domain knowledge about the concepts and their relationships and (3) heuristic retrieval strategies about how to match a query with an index as in RUBRIC [Tong 89] or DEDAL [Baudin et al 93]. The following example shows how heuristic knowledge is used in the conceptual information retrieval system DEDAL [Baudin 93] in the mechanical engineering design domain.

In this sample interaction, a mechanical engineering designer queries DEDAL to retrieve information about the design of a new variety of shock absorber. The query is about the "function of the solenoid". The system cannot find an index that exactly matches the query, so it uses a retrieval heuristic: "to find information about the *function* of a subcomponent, look for information about the *operation* of a mechanism that includes this component". This heuristic reflects the engineer's intuition that a paragraph describing how a mechanism works is likely to provide information about the function of its subparts and about how they interact. In this case, if *solenoid* is part of the *force generation mechanism*, DEDAL will assume that information about the function of the solenoid will be located in the same paragraph or nearby a piece of information which describes how the *force generation mechanism* works. Here the system used knowledge about the shock absorber domain (the solenoid is part of the force generation mechanism) to extend the user query and find related information.

Such heuristic retrieval strategies increase the set of relevant documents retrieved in response to a user query (increase recall), but at the expense of returning additional irrelevant documents (decrease precision). It is difficult for the system builder to manage this tradeoff by hand. For instance in the previous example the system could have retrieved a portion of a videotape showing the *force generation mechanism* but no close view of the *solenoid*. This paper discusses the application of standard induction algorithms to refine the heuristic retrieval strategies of a conceptual information retrieval system. Our goal is to refine the retrieval knowledge of such system over time as it gains experience with different users in different domains. To this end, We gathered examples of query/retrieval pairs during the system's operation using feedback from a user on the retrieved information and we fed these examples to an induction algorithm to refine the existing set of retrieval strategies.

* Catherine Baudin is an employee of RECOM Software Inc.
** Barney Pell is an employee of the Research Institute for Advanced Computer Sciences.

Section 2 discusses the specifics of the conceptual retrieval system that we used and gives examples of retrieval heuristics in the mechanical engineering domain. Section 3 describes how we use relevance feedback techniques and induction to refine the retrieval heuristics. Section 4 presents experiments and results on the impact of the refinements on the retrieval performance. Section 5 discusses the system's limitations and our plans for future work.

2. Background: Document Indexing and Retrieval in DEDAL

We have tested our approach on the conceptual retrieval system DEDAL described in [Baudin et al. 93] and will use examples from the mechanical engineering design domain to illustrate our method. DEDAL stores multimedia mechanical engineering design documents such as meeting summaries, pages of a designer's notebook, technical reports, CAD drawings and videotaped conversations between designers. It uses a conceptual indexing and query language to describe the content and the form of design information. [Baya *et. al.*, 1992], For instance: "The inner hub holds the steel friction disks and causes them to rotate" is part of a paragraph in page 20 of the record: report-333. It can be described by two indexing patterns:

<topic **function** subject **inner-hub** level-of-detail **configuration** medium **text** in-record **report-333** segment **20**>.

<topic **relation** subject **inner-hub** and **steel-friction-disks** level-of-detail **configuration** medium **text** in-record **report-333** segment **20**>

The queries have the same structure as an index and use the same vocabulary. A question such as: "How does the inner hub interact with the friction disks?" can be formulated in DEDAL's language as the query:

<get-information-about topic **relation** subject **inner-hub** and **steel-friction-disks** with preferred medium **equation**>.

The domain model includes aspects of the artifact structure, the requirements, the main design decisions and alternatives considered as well as relations such as: *isa*, *part-of* and *depends-on*. For instance, *solenoid* is a kind of *actuator* and is part of the *force generation mechanism*. The attribute *force* generated by the solenoid depends on the value of other attributes such as *current of the solenoid* or *power of the car battery*.

2.1 Heuristic Retrieval Strategies

The analysis of different types of design records such as engineering notebooks and structured progress reports led to the identification of a set of heuristics to help match a user's query with a set of indexing patterns describing the documents. These heuristics are activated when no index exactly matches a query or when the user is not satisfied with the references retrieved. They look for regions in the documentation that contain related information and assume that the required information will be found near these regions (on the same page, or in the same subsection depending on the type of record). For instance:

equation-to-schemata: In an engineering notebook, an equation describing a mechanism will usually be found next to a drawing representing this mechanism.

Some heuristics use the relations among the concepts in the task vocabulary:

performance-to-analysis: Information about the *performance* of a particular assembly and the *analysis* of this assembly are likely to be located in nearby regions of the documentation.

Others exploit the hierarchical relations in the domain model:

operation-to-function: In a structured document such as a progress report, the *function* of a component X in a mechanical assembly Y might be found near where the *operation* of assembly Y is described.

While these heuristics have been shown to significantly increase the recall of the system when compared to the system without the heuristics and to a base-line boolean retrieval system [Baudin et al. 93], they also decrease the precision of the retrieval by a small but non negligible amount. There are several reasons why a heuristic can fail to retrieve relevant information (see Section 6). One reason is that it is difficult for a knowledge engineer who creates these heuristics to take into account the influence of every possible document specific characteristic (such as the size of the document, the medium or the level of detail of the information) and determine what is relevant to the retrieval heuristics.

3. Using Induction to Refine Heuristic Retrieval Strategies

In order to refine the retrieval heuristics in DEDAL we first gather examples of query/retrieval pairs generated by successful and failed retrievals. We process these examples and reformulate them in terms of the language of the retrieval heuristics and of "interesting" concepts that could discriminate positive and negative examples. We then feed these examples to an induction algorithm to generate a decision tree that refines the set of heuristics. We define criteria to evaluate the impact of the induction mechanism on the retrieval performance of the system and on our understanding of the flaws in the initial set of heuristics.

3.1 Gathering examples through relevance feedback

First, we gather examples of relevant and irrelevant retrievals through a relevance feedback method [Salton 89]. Given a query, DEDAL first tries to find an index that exactly matches the query. If the retrieval fails, it applies a retrieval heuristic and presents the user with a prioritized list of references. The user then has the option to provide feedback to the system on the relevance of the reference retrieved. If the user is not satisfied by the answers returned he or she can ask the system to resume its search and apply another set of heuristics.

There are mainly two ways of using feedback from the

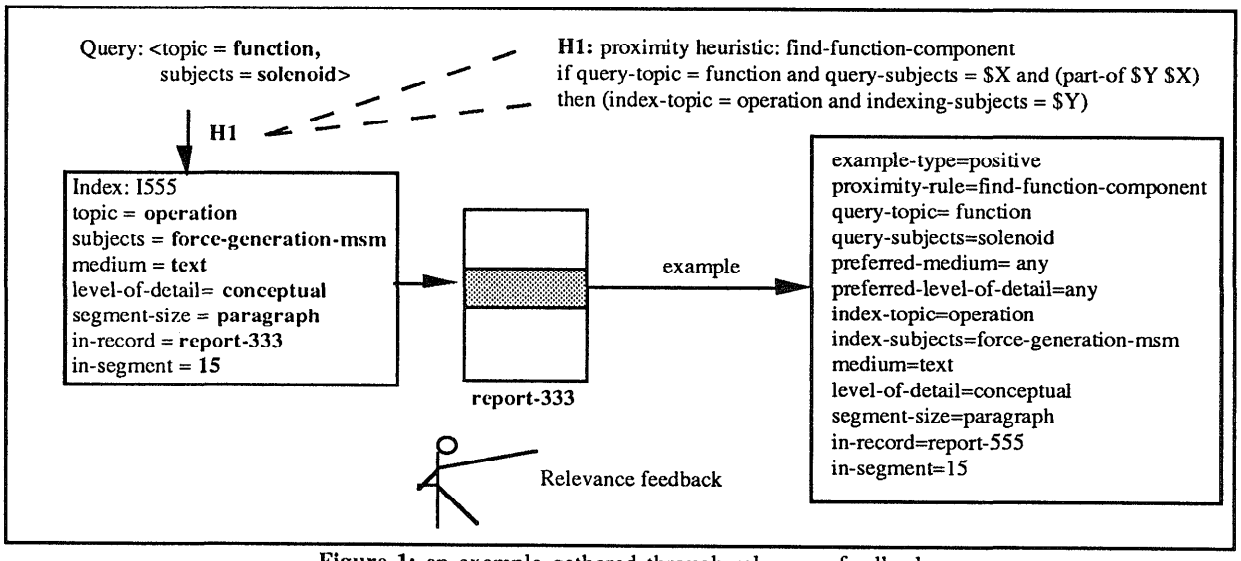

Figure 1: an example gathered through relevance feedback

user to improve the behavior of the system: The first way is to change the priority of the heuristics based on their rate of success. When the user provides feedback on the relevance of the references retrieved, the system updates a success coefficient associated with the heuristic. This coefficient is used by the retrieval mechanism in DEDAL to choose which heuristic to try first and to order the references proposed to the user. This re-prioritizing mechanism works well for the cases where a heuristic is very good or very bad, but does not improve the behavior of the system in the majority of cases where the heuristic is "somewhat" good, but needs slight adjusting.

Another way is to *refine* the heuristics so as to still cover the relevant references retrieved in answer to a user question (positive examples), while minimizing the number of irrelevant references presented (negative examples). This involves being able to modify applicability conditions of the heuristics - that is, to add and remove the tests that lead to the selection of a given document in response to a query. The idea is to take the examples of successful and failed query/retrieval examples generated by the user feedback, feed them to an induction mechanism, and generate a decision tree that refines the initial set of heuristics by refining tests leading to additional irrelevant references.

Figure 1 illustrates how a query/retrieval pair is acquired from the user using the example discussed in the introduction. In this phase we record the queries asked by a user, the answers returned by the retrieval heuristics and the relevance assessment of the user. The shaded information segment is described by one index and is about the operation of the force-generation-mechanism, the medium is text, the level of detail is conceptual, the record is the report-333 and the segment is a paragraph in page 15. That information segment is retrieved by the index I555 using the heuristic H1. The user provides a relevance assessment (relevant or irrelevant) and a corresponding positive or negative example of query/retrieval is stored in a database.

3.2 Using Induction to Refine the Retrieval Heuristics in DEDAL

Our initial goal is to decrease the number of irrelevant documents presented to a user (increasing precision) while maintaining the same number of relevant documents (preserving recall).

We use an off-the-shelf ID3 induction algorithm [Quinlan 86] on the examples of successful and failed retrievals to generate a decision tree. Figure 2 shows a query being processed through the set of initial heuristics and through the decision tree. When a query/retrieval example is selected by the existing set of heuristics, this example is filtered by the decision tree in an attempt to detect irrelevant references and assign them low priority.

To use ID3 we had to process the examples generated by

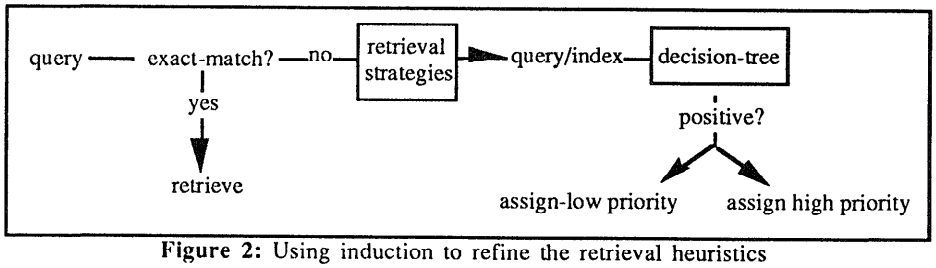

Figure 2: Using induction to refine the retrieval heuristics

the heuristics so as to (a) select features that could be used to discriminate positive and negative examples, and (b) remove noisy examples.

Selecting features to describe the examples: The first step is to select a set of features to describe the examples. This involves selecting and adding features to describe the characteristics of a query/retrieval example. In DEDAL an example has three types of features corresponding to three levels of generality:

(1) *documentation* dependent features such as a particular reference to a segment of information in a record.

(2) *domain* dependent features (subjects). These are references to elements dependent on a particular design. For instance in our example *solenoid* and *force generation mechanism* are both components of a particular mechanical device.

(3) *task* dependent features. These are features that do not depend on a particular design document. They only depend on the fact that the documentation is about the task of *mechanical design*. Features such as topic, media and level of detail are task dependent characteristics that can be reused across domains.

If a branch of the decision tree refers to document specific features, the corresponding heuristic refinement will only be usable for the documents used in the training set. In the same way, if a branch of the decision tree refers to domain specific features, the refinements should be valid for other documents related to the same design problem. Finally if a branch of a tree only refers to task specific features, the refinements should transfer across domains for documents associated with any mechanical design project.

In addition, to make possible the generation of domain independent refinements we added some features to each example. These features represent different types of relations between the subjects in the query and the subjects in the matching index. For instance the feature "subjects-query-equal-subjects-index =no" means that the subjects in the query are not the same as the subjects in the matching index. In the same way, "subjects-query-is-part-of-subjects-index = yes" means that the subjects in the query are part-of some subjects in the matching index. Figure 3 shows an example: the bold italic features are document dependent features, and the bold features are domain dependent features. The features in plain text are task dependent. The additional task features that are automatically added to each example are shown at the bottom of Figure 3.

```
(example 165 positive
generated-by-rule: find-function-component
query-topic = function
query- subjects= solenoid
index-topic = operation
index-subjects =
force-generation-mechanism
index-medium = text
index-level-of-detail = conceptual
in-record: damper-spring-1989
in-segment: 15

subjects-query-equal-subjects-index = no
subject-query-is-attributes = no
subject-query-is-requirements = no
subjects-query-is-part-of-subjects-index = yes
subjects-index-is-part-of-subjects-query = no
subjects-query-is-kind-of-subjects-index = no
subjects-index-is-kind-of-subjects-query = no
subjects-query-depends-on-subjects-index = no
subjects-query-influence-subjects-index = no
... )
```

Figure 3: A positive example of query/index match

Removing noise in the training examples: The examples are processed so as to detect the cases where the same answer to the same query led to different relevance assessments. If an answer was found relevant one time and irrelevant the other time it is counted as a positive example. This is because several factors other than faulty heuristics can lead to the retrieval of an irrelevant reference in answer to a query. For instance, an answer might be irrelevant because the user misformulated a query, because he or she did not scan the information thoroughly and missed the answer or because of a flaw in the indexing knowledge.

3.3 Evaluation

We evaluate our approach in terms of: (1) the impact on the *retrieval performance* of DEDAL: as a start our goal is to decrease the number of irrelevant references retrieved (increasing the system's precision) by the heuristics while maintaining the same number of relevant references (maintaining the system's recall), and (2) the *informativeness* of the fixes proposed by the induction algorithm in terms of the ability of a knowledge engineer to understand the information in the decision tree and detect flaws in the retrieval heuristics, in the indices or in the domain model.

4. Results

To evaluate our method we selected a set of examples generated by two mechanical engineers using DEDAL to retrieve answers about the "rotary-friction-damper" design, an innovative electromechanical shock absorber designed for Ford Motor Corporation [Baudin et al. 93]. The users queried the system while solving a redesign problem involving the modification of the shock absorber design. Each of the references retrieved was rated by the users as relevant or irrelevant. In this experiment when a proximity heuristic was used, the user was asked to rate all the references retrieved by the heuristic.

4.1 Experiments

We extracted 300 examples of relevant and irrelevant heuristic retrievals generated from the interaction with the first user (user1) as our training set, and another set of 81 examples generated by the second user (user2) as our test set. We then selected sets of features for the examples and generated three different decision trees corresponding to different levels of generality in the features selected. Table

1 summarizes the results in terms of the impact of the decision tree generated from the training examples on the percent of negative and positive examples filtered for the queries generated by user2.

In our first experiment the examples include features at all three levels of generality: the *document*, the *domain* and the *task* features. This means that information about specific documents is included in the training examples. In this case, the induction algorithm picked the document specific features as the most discriminant features to separate the positive and negative examples. As a result the induction tree used information about specific documents and pages to refine the heuristics. In the second experiment, we removed the document specific information but kept the domain specific information. As a result the refinement tree generated from the induction algorithm does not refer to any specific document, it does however refer to domain concepts. Finally in the third experiment both the document and the domain specific features were removed from the examples.

The effect of this manual feature selection is to force the induction algorithm to generate filters at different levels of generality. However, because ID3 naturally picks the most discriminant features, the effect of this forced generalization is to trade predictive power for an increase in generality. The following subsection shows results from these experiments.

4.2 Impact on the Performance of the Retrieval

We measured the impact of the decision tree in terms of its ability to filter the negative examples in the test set while preserving the positive examples.

We present in Table 1 the number of positive and negative examples filtered by the system. In the table, the column '%filtered negatives' lists the percentage of negatives filtered by the system out of the total number of negatives in the test set, and similarly for '%filtered positives'. The total number of examples generated by user2 is 81, with 54 negative examples and 27 positive examples. The cases that are undecided are counted as positive. As seen in Table 1:

a. Document dependent repair: the decision tree generated for experiment1 would have reduced by 44% the number of irrelevant references presented as high priority references to user2, while assigning low priority to only 7% of the relevant references. However, this filter is document dependent as most branches of the decision tree refer to specific regions of information in particular documents.

b. Domain dependent repair: In Experiment 2 the document dependent features (record and segment in figure 3) were removed from the examples, forcing ID3 to generate a more general tree. As a result, in experiment2 the branches of the decision tree do not refer to any specific document but still refer to subjects of the domain. For instance one branch states that: if an information segment is about *rotary-friction-damper* and *wedge* and the query is about *rotary-friction-damper* then the match is irrelevant. Because there is no reference to any specific document, this knowledge should be applicable to any design document in the "rotary-friction-damper" domain. Here, as expected the percentage of filtered negatives (33%) is smaller than for the document dependent tree, while the misclassification of relevant retrievals (11%) is higher.

c. Task dependent repair: Finally both the document dependent and the domain dependent features were removed from the examples in an attempt to generate a domain independent filter. The resulting tree only involves tests on task dependent features. Table 1 shows that the resulting decision tree would filter 26% of the irrelevant references presented to the user but at the expense of also filtering 41% of the relevant ones. As a result of this high rate of misclassified relevant references, we consider this tree ineffective as an automatic filter. However, as discussed in the next section, some branches of the tree were still informative for the knowledge engineer.

4.3 Informativeness

We found two main advantages in having a knowledge engineer examine the branches of the decision tree: (1) even in the cases where the tree does not improve the performance of the retrieval, some branches of the tree suggested interesting local fixes that appealed to the knowledge engineer's intuition although they cannot yet be evaluated automatically by the system, and (2) the branches of the tree pointed out flaws in other parts of the system such as the indices themselves or the domain model.

Even though we do not have any formal measure of the informativeness of the fixes proposed by the system, we present three examples of particularly informative repairs suggested by the decision trees generated from our experiments. Currently the knowledge engineer examines these suggestions and manually modifies heuristics.

Example 1: In Experiment3 most branches of the tree refine the heuristics by adding conditions on the *medium* and *level of detail* of the information retrieved. For instance in the case of the heuristic "operation-to-construction", the initial applicability conditions state that any query about the *operation* of a mechanism X can be matched with an index about *construction* of X (information about the construction of X shows how X is structured and how it is assembled).

User 2 on user1 1	*# filtered negatives*	*# filtered positives*	*% filtered negatives*	*% filtered positives*
with document features	24	2	44%	7%
with domain features	18	3	33%	11%
only task features	14	11	26%	41%

Table 1: Impact of induction on retrieval performance

H1: Operation-to-construction
if (query operation-of $X)
then (index construction-of $X)

The applicability tree (see Figure 4) refines the applicability condition of this heuristic by specifying that any information about construction of X may be relevant to a query about operation of X *only if the medium of the document is text* , not if it is say, a schemata. This proposed repair to the "operation-to-construction" heuristic corresponds to the intuition that a picture convey structure but not actions and therefore a text is more likely to give an idea of how a mechanism works. The system displays the branches of the tree and the knowledge engineer can then decide to update the retrieval heuristic "operation-to-construction".

H1': operation-to-construction
if (query operation-of $X)
then (index construction-of $X
with **index-medium text)**

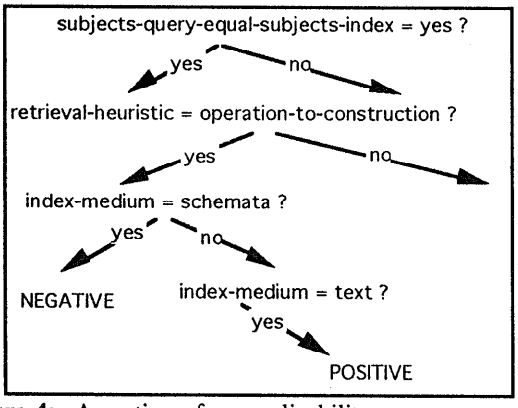

Figure 4: A portion of an applicability tree generated by the ID3 induction algorithm.

Example 2: Another portion of the tree generated in Experiment 3 rediscovered a heuristic that was suggested later by the expert of the domain in a newer version of the proximity heuristics: the heuristic generated by the decision tree is: "if query-topic = location and query-topic-is-part-of-index-topic = yes and medium-index = picture then the example is positive.". This branch corresponds to the following rule: "If the query is about the location of a component X, and X is a subpart of Y, a picture describing how Y is assembled might provide information about where the component X is located in the assembly." The system here discovered that a picture of an assembly Y might be a good way of locating a subcomponent of Y. The tree also suggested that this heuristic will fail if the medium of the retrieved information is text. Even though the test set generated by user2 could not be used to validate this rule (there were no questions about the location of a component), the following new rule can be manually added by the knowledge engineer.

if (query location $X)
(part $Y $X)
then (index construction $Y
with medium = picture)

Example 3: The branches of the decision tree generated by Experiment 2 pointed out flaws in the domain model and in the way some of the information was indexed. For instance, one branch of the tree suggests that: "if an information segment is about *rotary-friction-damper* and *wedge* and the query is about rotary-friction-damper then the example is negative" This rule pointed out the fact that a piece of information was about *rotary-friction-damper* and *wedge*, whereas *wedge* is not a component used in the current design. This piece of information in fact described an alternative design solution and should not be indexed using the concept *rotary-friction-damper* which refers to the current design solution. This failure pointed out to a problem in the index itself. Such a flaw in the indexing knowledge would have been difficult to discover without the help of the induction mechanism.

5. Discussion

The difficulty in using examples collected during the system's interaction with end-users to refine information retrieval heuristics is that there might be several causes involved in the failure of the system to satisfy a user: flaws in the domain model, in the way a piece of information is indexed, in the way the query is formulated or in the heuristic itself. Consequently, we expect two types of results out of the decision trees generated from the examples: (1) to improve the performance of the system in terms of the precision and recall of the retrieval during the system's operation with the *end-user*, (2) to help a *knowledge-engineer* find flaws in the knowledge of the system. In addition we want (3) to be able to *transfer* the retrieval heuristics learned to other documents and other domains.

With respect to these three goals, our experiments have shown that by using induction in the heuristic retrieval component: (1) we improved the precision of the system at the most specific document level by reducing the number of irrelevant references retrieved by 44% with minimum impact to the system's recall, (2) some of the domain and document specific rules pointed out flaws in the way the information was indexed and in the domain model, these flaws would have been difficult to detect manually, and (3) the decision trees suggested fixes to the proximity heuristics that showed the influence on the relevance of the retrieval heuristics of document specific characteristics such as the medium and the level of detail of the information. In addition, some of these repairs can transfer to other domains for mechanical engineering design documents. However, currently the domain independent decision tree is not an effective filter to improve the precision of the retrieval as it misclassifies too many relevant references (41%). This is partly due to the fact that we need more training examples in different domains. We also might need to add new domain independent features to describe the

examples.

5.1 Limitations

The examples used to generate the decision trees were pre-filtered by the initial set of proximity heuristics. As a result the decision tree must still be used in conjunction with these heuristics. To be able to generate new heuristics from the examples, the induction algorithm needs to operate on the negative examples that are filtered by the initial set of retrieval heuristics.

To be fully operational, the system needs to be able to maintain several decision trees in parallel and must be able to monitor their performance in the background as in [Maes et al. 93] in order to understand when and at what level of generality the result of the induction algorithm can improve the performance of the retrieval.

Our goal in these experiments was to increase the precision of the retrieval while maintaining the same level of recall. In the future, we need to address the cases where the rules are too specific, so as to increase the coverage of the rules and increase the system's recall. This in turn might lead us to extend the interactive capability of the system to help modify the domain knowledge itself--that is, the taxonomy of domain concepts and the relations between these concepts--as in [Bareiss et al.89].

Finally we need to run more experiments not only with different classes of users but with different domains. In particular, we are now running the system to retrieve text, graphics and video records for the design of an innovative bioreactor, a device that will enable NASA life scientists to study microbial growth.

6. Conclusion

This paper discussed the use of induction to refine conceptual information retrieval strategies. Our approach improves retrieval performance in a noisy environment, using relevance feedback from the end-user during the system's operation. The result of the induction algorithm is useful in two modes: (1) in an *automated* mode, as a filter to increase the precision of the retrieval, and (2) in an *interactive* mode: to help the knowledge engineer detect flaws in the different components of the system.

Acknowledgments

Thanks to Vinod Baya, Ade Mabogunje and Jody Gevins Underwood who helped us develop, and evaluate DEDAL. Thanks to Larry Leifer and to the other members of the GCDK group for their feedback and support on this project, to Wray Buntine and members of NASA Ames. Thanks to Michel Baudin for his help on early drafts.

References

Bareiss, R., Porter, B.W., Murray, K.S., Supporting Start-to-Finish Development of Knowledge Bases. *Machine Learning*, 4(3-4):259-283, 1989.

Baudin, C., Gevins, J., Baya, V., "Using Device Models to Facilitate the Retrieval of Multimedia Design Information", in proceedings of IJCAI 93 Chambéry, 1992.

Baudin, C., Kedar, S., Gevins, J., Baya, V., Question-Based Acquisition of Conceptual Indices for Multimedia Design Documentation. *Proceedings of the Eleventh National Conference on Artificial Intelligence*; Washington, D.C., 1993.

Baya, V, Gevins, J, Baudin, C, Mabogunje, A, Leifer, L. "An Experimental Study of Design Information Reuse", in proceedings of the 4th International Conference on Design 1992.

Maes, P., Kozierok, R., "Learning Interface Agents" *Proceedings of the Eleventh National Conference on Artificial Intelligence*; Washington, D.C., 1993.

Mauldin, M. "Retrieval Performance in FERRET", Proceedings of the ACM SIGIR Conference, 1991, pp. 347-355.

Salton, G., Buckley, C., Improving Retrieval Performance by Relevance Feedback. J. of ASIS. 41:288-297. 1990.

Quinlan, J.R.,Induction of decision trees. Machine Learning 1(1):81-106 1986.

Tong, M. R., Appelbaum, A., and Askman V. "A Knowledge Representation for Conceptual Information Retrieval", International Journal of Intelligent Systems. vol. 4, 259-283, 1989.