Spontaneous Retrieval in a Conceptual Information System

Lisa F. Rau
Artificial Intelligence Branch
GE Company, Corporate R&D
Schenectady, NY 12301 USA

Abstract

A traditional paradigm for retrieval from a conceptual knowledge base is to gather up indices or features used to discriminate among or locate items in memory, and then perform a retrieval operation to obtain matching items. These items may then be evaluated for their degree of match against the input. This type of approach to retrieval has some problems. It requires one to look explicitly for items in memory whenever the possibility exists that there might be something of interest there. Also, this approach does not easily tolerate discrepancies or omissions in the input features or indices. In a question-answering system, a user may make incorrect assumptions about the contents of the knowledge base. This makes a tolerant retrieval method even more necessary. An alternative, two-stage model of conceptual information retrieval is proposed. The first stage is a spontaneous retrieval that operates by a simple marker-passing scheme. It is spontaneous because items are retrieved as a by-product of the input understanding process. The second stage is a graph matching process that filters and evaluates items retrieved by the first stage. This scheme has been implemented and validated in the SCISOR information retrieval system.

I. Introduction

The System for Conceptual Information Summarization, Organization and Retrieval (SCISOR) is an information retrieval system designed to analyze, answer questions about, and summarize short newspaper stories in natural language. Operating in the domain of corporate takeovers and finance, SCISOR is unique in its approach to retrieval of the complex conceptual events stored in its knowledge base. Most approaches to conceptual information retrieval can be broken down into the following four phases:

1. Retrieval decision: The system comes to a point in its processing when it desires some information from memory.
2. Retrieval setup: Indices or features are collected and put into a correct format for retrieval.
3. Retrieval: The retrieval process is performed.
4. Post-processing / Matching: The outcome of the retrieval process is examined and conditional actions may be taken.

The conceptual information retrieval performed in FRUMP [DeJong, 1979], CYRUS [Kolodner, 1984], COREL [DiBenigno *et* al., 1986] (which uses the PEARL [Deering *et* a/., 1981] AI package, and IPP [Lebowitz, 1983] can all be put into this framework, as could any system that performs deductive information retrieval in the style of Charniak, Riesbeck and McDermott [Charniak *et* a/., 1980]. An exact match restriction may be relaxed after the initial fetch returns a negative result. The model of finding items in memory here is an iterative one of generate and test. This model has certain problems when we consider how it could be used to perform certain desirable functions in an intelligent information retrieval system. In contrast to this model, in SCISOR, items are retrieved from memory automatically as a result of the instantiation of new input instances. When a user's question is instantiated, potential answers appear in a short-term buffer. When a new story is instantiated, any previously existing context for that story appears in the buffer. This automatic retrieval is implemented with a constrained form of marker-passing. Items spontaneously retrieved in this manner are then run through a more computation intensive matching process. Three problems with the model of retrieval initially described will be given, followed by a description of the SCISOR system and its solution to these problems.

II. Problem Description

A, Unanswered question retrieval

One capability the SCISOR system has is to retrieve automatically a user's previously posed but unanswered question when an answer to that question becomes known or refined. For example, consider the following scenario:

User: How much did ACE offer to take over ACME?
System: Figures for the deal have not yet been disclosed.
Intervening time...
System: *BEEP! Figures for the ACE-ACME takeover have just been released. ACE has offered $40 / share for all outstanding shares of ACME.*

In order to provide this capability to a system that performs retrieval as previously described, the system would

have to keep a list of unanswered questions present. When new stories were input to the system, it would poll the unanswered questions, asking "does this answer you?". A better approach might be to set up demons on each unanswered question that look for certain input features that might relate to the content of the question. In either case, however, the system is always looking for answers to its still outstanding questions. In SCISOR, if input features happen to relate to an unanswered question in memory, that question is spontaneously brought up for consideration. If the input does not relate to any unanswered questions, nothing happens.

B. Retrieval with partial information

SCISOR has another capability that would be awkward to implement in the model of retrieval previously described: to find events in memory even when a user's question contains only partial, or even incorrect information. For example, consider the following question, along with some independent potential states of the world that might be true at the time the user asked the question.

- Did ACE food company take over ACME hardware company?
  1. The ACME hardware company took over ACE food company.
  2. The ACE food company took over the BIG-ACME company, which owns the ACME hardware company.
  3. The ACE food company made an offer to the ACME hardware company, but has not yet succeeded in taking over the ACME company.
  4. ACE is a conglomerate and not simply a food company.

In each of these cases, the question asked cannot be well answered simply "yes" or "no". Consider what a deductive information retrieval mechanism such as that described in Chamiak, et. al. [Chamiak et al., 1980] might do to find the answers to the questions above. One possible method would be to retrieve all takeover events in which the company taking over another company was the ACE food company. The episodes found would then be checked to find ones in which ACE took over another company. The resulting events are examined to see whether the object of the takeover was the ACME hardware company. Such a procedure is incapable of detecting any of the scenarios described above without further augmentation.

One potential augmentation would be to incorporate heuristics such as "If user asks if X did Y, and nothing is found, check if Y did X". This would allow the system to find the correct scenario numbered (1) above. To answer (4), a heuristic that looks up and down the "isa" hierarchy of all the input features could be used. Note that many such ad hoc heuristics would be necessary to find relevant episodes, given only arbitrary partial or erroneous information. A second potential augmentation would be to try combinations of features instead of all features. For example, if one takes ACE, ACME, and TAKEOVER as three features of the question above, then the system could ask "ACME takeover who", "ACE takeover who" and "ACE, ACME what" to find the situations (2) and (3) above. A third possibility, used in the CYRUS program [Kolodner, 1984], is to generate plausible indices through reconstruction of what was likely to be present in the situation. Although this procedure has a certain cognitive appeal, it is not guaranteed to find events present in the memory. In fact, none of the augmentations described is guaranteed to find relevant situations; some are not particularity principled, and all involve substantial additional computation.

In SCISOR, finding partially matching scenarios is a by-product of the retrieval process because the first pass of the two-stage retrieval process SCISOR uses simply finds events with features that are the same as, or similar to, features in the input question. The validity of the relationships between the features is not considered until the second pass of the two-stage process is performed. Thus, any of the scenarios above would be retrieved given the input question as stated. The evaluation mechanism then determines what is the same as the user's question and what the differences are. These differences may then be expressed to the user. Note that when nothing closely matches what the user asked, no events in the system's knowledge base will have enough activation to exceed the threshold, and the system will respond that it doesn't know.

C. Previous story retrieval

Finally, the SCISOR system can find previous articles stored in memory when a new article is input that deals with the same situation. In the corporate takeover domain, events happen over time. For example, ACE may make an offer to ACME on Monday, and ACME may respond to the offer on Friday. The initial offer should be retrieved from memory when the response to the offer is input, so that this new piece of information can be properly integrated into the history of events. The way that this could be done with the traditional method of retrieval is similar to the case of retrieving unanswered questions when the answers are input. That is, checking after each new story to see if it is a continuation of a known story would result in substantial additional overhead. Also, dealing with the features present in a story continuation could have the same problem as dealing with missing or incorrect features in a user's input questions. For example, it would be difficult to find out that ACE was trying to take over ACME when subsequently ACME announced it was trying to take over ACE. Both of these concerns are elegantly addressed with SCISOR's method of retrieval.

Summary of Problems SCISOR Solves:

1. SCISOR addresses a user's previously unanswered questions when an answer comes along. Instead of always looking for answers that may or may not be present in incoming stories, incoming stories that hap-

pen to relate to unanswered questions activate only those questions, which can then be considered.

2. SCISOR is capable of locating relevant information in response to a user's questions even when that question contains misleading or partial information.
3. SCISOR retrieves previous events when updates of those events are read. This is done in the same efficient manner as retrieving an unanswered question, and with the same tolerance for partial or contradictory input as in the question-answering case.

The following exchange illustrates the kind of input-output behavior SCISOR was designed to exhibit:

User: Did ACE food company take over the ACME hardware company?

System: *Yes, ACE took over the BIG-ACME company, which owns the ACME company. ACE completed the $18.2 million friendly takeover last Friday.*

User: What was the sequence of events in the takeover?

System: *Rumors that BIG-ACME was to be taken over started May IS, 1985. The stock rose $5 a share to $65. On May 16, the ACE company announced that it had made an offer to BIG-ACME valued at $66 a share. The offer was a cash offer to purchase all t million outstanding shares of BIG-A CME. After a month of deliberation involving some potential conflict of interest matters, the deal was completed last Friday.*

User: What happened yesterday in the HAL Corporation bid for BIG?

System: *HAL Corporation increased its bid for the BIG company yesterday, up $4 to $70 a share. BIG has not commented on the new offer.*

These three hypothetical exchanges illustrate three capabilities difficult to obtain from traditional document retrieval technology [Salton and McGill, 1983]. The first illustrates the ease of obtaining *concise answers* to straightforward questions by posing the question to the system in natural language. The second illustrates obtaining a *summary* of an event-sequence whose description spanned multiple articles occurring over time. In the third exchange, a simple *update* of an ongoing story, presumably being followed by the question-asker, is obtained.

Full-text retrieval systems are incapable of relating articles to one another. Thus, it is impossible to ask such a system for a *summary* of a situation that unfolds over a period of time and probably involves multiple articles. The user must retrieve the entire series of articles and read each one to obtain an understanding of all that has occurred. Given that most articles consist of background information potentially known to the reader, simply restricting the information shown to the user to new information would be very helpful. The best scenario, however, is to give the user the ability to retrieve a preprocessed summary of events in any given situation, as SCISOR does.

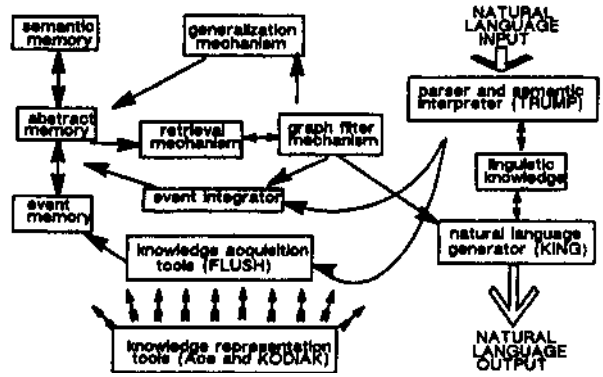

Figure 1: SCISOR System Architecture

This section has described some of the problems with both traditional methods of conceptual information retrieval and with the traditional paradigm of document retrieval. The next section describes the SCISOR system and in more detail how its approach to retrieval addresses the problems.

III. Implementation

SCISOR takes input in natural language, integrates new information into memory, and answers natural language questions in natural language. The natural language input is processed with the TRUMP parser and semantic interpreter [Jacobs, 1986]. New events are integrated as a continuation of an ongoing story (if present) by the event integrator, which also stores new events for retrieval. The FLUSH acquisition tools [Besemer and Jacobs, 1987] are aids to the acquisition of vocabulary and phrases in the system's phrasal lexicon. The events in SCISOR are represented using the KODIAK knowledge representation language [Wilensky, 1986], augmented with some scriptal knowledge [Schank and Abelson, 1977] of typical events in the domain. Linguistic knowledge is represented using the Ace linguistic knowledge representation framework [Jacobs and Rau, 1985]. Responses to the user are to be generated with the KING [Jacobs, Fall, 1987] natural language generator. Figure 1 illustrates the architecture of SCISOR.

A. Memory Organization

SCISOR manipulates conceptual structures represented in the KODIAK knowledge representation language. KODIAK can be viewed as a hybrid frame and semantic net-based language, similar in spirit to KRYPTON [Brachman *et a*/., 1983]. In SCISOR, knowledge stored is either specific, abstract, or semantic. An example of a specific memory is the memory of the ACE food company acquiring the ACME hardware company. A traditional episodic memory [Tulving, 1972] is composed of specific memories. An example of an abstract memory is the generalization across specific experiences we might have heard about where companies have acquired other companies. Abstract (or generalized episodic) memories are on the border between spe-

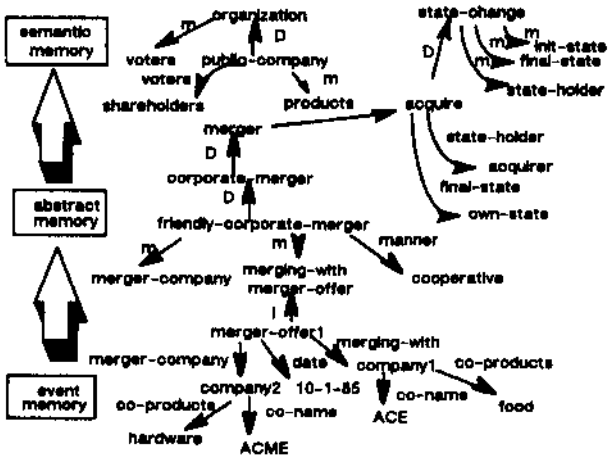

Figure 2: Structure of long-term memory

cific memories and semantic memory. Semantic memory is the memory or knowledge of what "companies" are and what "acquiring" is. Semantic memory is used in understanding and making inferences about the input to the system.

This tripartite division of memory approximates a continuum, where the three divisions represent the most specific level, the most general level, and the levels in between. At the most specific level are things that happen in the world that are composed of particular, unique instantiations of concepts. Specific memories may abstract through generalization by the generalization mechanism, and these abstractions may abstract. Figure 2 illustrates the structure of long-term memory with associated examples.

In addition to this tripartite division, another level of organization is superimposed on memory. Groups of related concepts in episodic or abstract memory are linked together through a common node, called a TAG. The TAG allows the system to detect quickly whenever multiple instantiated concepts appear in the same event or episode. This TAG is attached to concepts by the event integrator. The integrator simply takes all the instantiated concepts passed to it and assigns them a new TAG. Each TAG has a numerical threshold value, currently equal to a fraction of the number of concepts in the episode, currently one-third. Long articles may consist of TAGs that have TAGs as components. Figure 3 illustrates the kind of structure the integrator superimposes on memory.

B. The Retrieval Mechanism

Retrieval in SCISOR is a two-stage process. The first stage is a coarse search that finds events in memory likely to be relevant. Relevance is determined by the number of features present in an event in memory related to features in the input. After the most likely candidates have been isolated, a more computation-intensive matching process is performed. The operation of the retrieval mechanism,

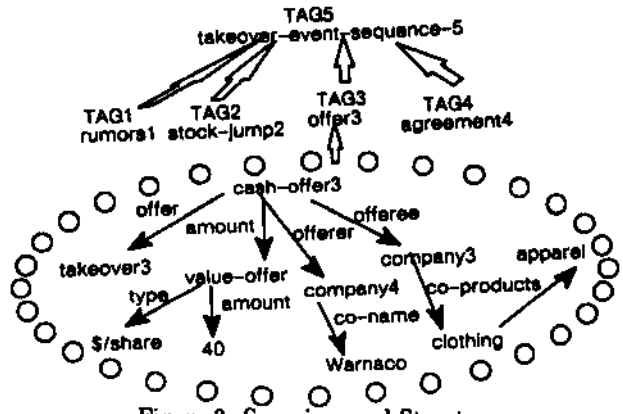

Figure 3: Superimposed Structure

and the SCISOR system in general, is duscussed in more detail in Rau [Rau, 1987].

The first stage of the retrieval process is performed by a process of priming or constrained spreading activation. As concepts are instantiated in the system, instances of concepts that are related via category membership links are marked (i.e., primed or activated). When a certain subset of the concepts in an event or episode is marked, the entire episode is put into the system's short-term memory buffer. This is the spontaneous retrieval phase. The events that have been spontaneously retrieved can then be run through the match filter to check the nature of the match between the input and the events retrieved. Periodically, all marks are deleted from the system. Marker-passing "waves" are propagated continuously as new concepts are instantiated. This mechanism retrieves answers to input questions, old unanswered questions to new input answers, and stories given related input stories in exactly the same manner.

In more detail, the retrieval mechanism operates in the following manner:

1. At the time of concept instantiation, related concepts are marked according to the "priming rules" (see next section). Every concept that is marked causes its containing episode or episodes (its TAGs) to have increased activation* The current and threshold levels of activation are simple properties of the TAG.
2. TAGs that have had their values increased check themselves to see whether their current activation levels exceed their threshold.
3. If the TAG threshold has been exceeded, an *intersection* has been detected, and the entire episode is put into the short-term memory buffer. The match filter is then called to refine the potential matches.
4. If a concept is instantiated that is included in an episode already in the memory buffer, an incremental match filter is called to match incrementally the new concepts.

•Actually, high-frequency, low-content concepts may not participate in this process.

5. All concepts in the memory are unmarked after every one or two sentences and after every question are input.

The rules that guide how levels of activation or marks are passed through the conceptual hierarchy are given below. These rules were formulated to decrease the possibility of retrieving memories that have limited predictive capacity or relevance to the current situation.

1. Priming Rules

The effect of the following rules is that all instances of concepts that are components of episodes and are children of the incoming parent of the parent of the incoming instance in the conceptual hierarchy, are marked. Also, all instances of concepts that are components of memories, and are *direct* instances of concepts that are parents of the incoming instance, are marked. This rule prevents everything in the hierarchy from being marked, and limits what is marked to a level of conceptual abstraction supported by the presence of a direct instance at that or a more general level.

For example if a user asked "What happened to the ACE company?", the event being asked about is at a fairly general level of conceptual abstraction. Any event involving the ACE company and more specific than this general "event" instance would be a valid answer. If no event were known, the unanswered question would be stored. In the future, *any* input events involving the ACE company (i.e., offers, rumors) would cause the unanswered question to be activated. Note that all concepts as they are instantiated cause any related instances to be marked, which allows any feature in the input to be a potentially useful index key into memory.

Rules:

1. Mark concepts that are components of specific or abstract events. These concepts are marked with TAGs.
2. Determine the categories to which the incoming instance belongs (Categories-of A).
3. Determine the concepts in the reflexive, transitive closure along category membership links of Categories-of (Categories-of A).
4. Mark the direct instances of concepts in this closure. Each marked concept increases the current activation value of each of its TAGs.

Additional refinements to this algorithm have been made to increase the system's efficiency. One such refinement has been to check the number of episodes containing a certain concept before passing markers to all those episodes. This check can be made easily by maintaining a count of the number of category members in each conceptual category. For example, if the system had read about events involving thousands of companies, this number would be stored at the parent COMPANY node. When this number is very large, the system suppresses the marker-passing operation. Events that are retrieved through the activation of more unique concepts then incorporate the high-frequency concept information in just those events. This simple refinement suppresses waves of activation unlikely to cause significant differentiation among events in memory, while still taking all features of the input into consideration.

After a set of events has been spontaneously retrieved, a match filtering operation is performed to ensure that the correct *relationship* exists between concepts within the events.

2. The Match Filter

The match filter takes spontaneously retrieved events and computes a final degree of match, based on the results of matching *relationships* between the concepts in the input and the concepts in the retrieved events. In the following discussion, a "component" refers to a constituent concept in an event.

Input to the match filter consists of an ordered list of the most highly activated episodes in the short-term memory buffer. Also present is information about what input components caused a retrieved episode's component to be activated. Output of the filter is a potentially reordered list of matching episodes, with additional information that relates the components of the input to the components of the retrieved episodes with more precision and certainty.

The match filter starts at one node in the retrieved episode and begins to construct a new graph that represents the matching elements of the input and the retrieved nodes. It performs this process by checking that values of the "slots" (called *aspectuals* in KODIAK) in the retrieved episode are also marked by values of the slots in the input. For example, consider the example illustrated in Figures 4 and 5.

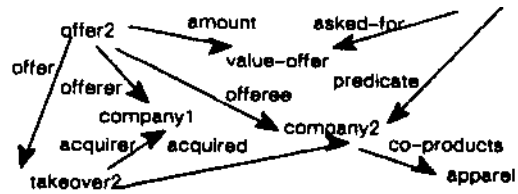

Figure 4: How much are takeover offers for apparel companies?

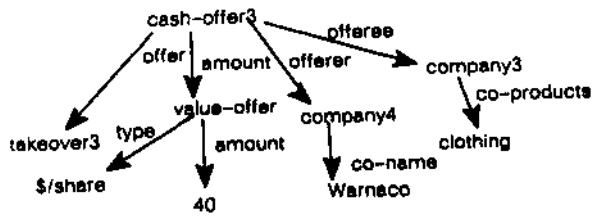

Figure 5: Part of a Story Episode

The story episode contains information about a kind of offer, a CASH-OFFER, made by Warnaco for a clothing company. The question episode requests the value of any

offers made by a company in a takeover attempt on an apparel company.

Due to the marker-passing, the instantiation of OFFER in the question episode causes CASH-OFFER to be marked. Similarly, CLOTHING marks APPAREL. The other features that contribute to the retrieval of the story episode are TAKEOVER, COMPANY and VALUE-OFFER. The match filter begins at a node that is marked only by one node from the input. OFFER2 is such a node, but note that COMPANY2 and COMPANY3 are both marked by COMPANY4 and COMPANY5.

The filter proceeds to check that the OFFER of CASH-OFFER3 (TAKEOVER3) is marked by the OFFER of OFFER2 (TAKEOVER2). Given that this is true, the OFFER2/CASH-OFFER3 match is added to a new graph that represents the matching elements of the input and the retrieved episodes. This process is repeated for every node in the input. In the case of answering a question, those nodes that do not correspond are added to a list of presuppositions to be expressed to the user. Also, any node that was more general than the input may be expressed to the user. For example, if the user was interested in what pet-food companies were being taken over, and the system only knew about food companies, this generalization is pointed out.

C. System Status

SCISOR is implemented in Common Lisp; it is used on VAX computers and Symbolics and SUN workstations. The TRUMP parser and semantic interpreter has not yet been tested with a large grammar or vocabulary but, in these early stages, it has been relatively easy to customize. On the SUN-3 it processes input at the rate of a few seconds per sentence, including the selection of candidate parse and semantic interpretation. The KING natural language generator was implemented in Franz Lisp, and at this writing has not yet been converted to Common Lisp to run with TRUMP.

The system has been tested with a dozen or so stories stored in the knowledge base. Hundreds of semantic concepts and domain vocabulary are also present. About a dozen questions are answered by the system.

IV. Related Research

A. Question Answering

In SCISOR, the processes that find the approximate location of an answer to a user's question and the processes that determine what the answer should be are separate. A great deal of work has been done on the second problem, most notably by Lehnert [Lehnert, 1978]. Determining an appropriate answer to a user's question, given that the context in which the user's question was posed is already known, is a separate process from the initial retrieval of a context in which to search for an answer. This initial retrieval of a context is the spontaneous retrieval this paper describes.

B. Memory-based Reasoning

Stanfill and Waltz [Stanfill and Waltz, 1986] provide excellent motivation for using episodic memory to perform various reasoning tasks. While it is hoped that SCISOR will eventually be used to perform reasoning of the type they describe, the method of retrieval of episodes used in the two systems is antithetical. While Stanfill and Waltz assume that it is impossible to find a best match from an event-based memory without examining every item in the knowlege base, this is in fact exactly what SCISOR does. SCISOR accomplishes this by only activating events with features related to features in the input. Thus search is focused entirely on best-match candidates. Events with no features related to features in the input are never touched.

Also, SCISOR uses its semantic knowledge to find close matches, where closeness is a measure of semantic distance. Stanfill and Waltz's proposed lack of knowledge of the semantic domain of system operation will prevent their system from finding truly best matches.

C. Partial Parsing

Recently, there have been some limited successes in the development of AI systems to parse partially and to understand short texts in constrained domains [Dejong, 1979, Lebowitz, 1983, Kolodner, 1984, Young and Hayes, 1985]. However, work in effectively *accessing* these knowledge bases has not been as successful. For example in Young and Hayes [Young and Hayes, 1985], the information cannot be accessed after it has been understood except through a traditional database front-end, or by direct examination of the conceptual, frame-like representation. In CYRUS, a natural language question-answering component allows queries of the knowledge base. However, the system is not guaranteed to answer correctly. As a cognitive model, its memory failures are understandable and interesting, but when accuracy and reliability of information are important, such a model is not viable. SCISOR demonstrates some of the uses that can be made with automatically extracted conceptual information from text when that conceptual information is combined with a powerful and robust method of spontaneous retrieval.

D. Cognitive Effects

In systems such as SCISOR that use the same parsing mechanism for both question parsing and story parsing, interesting effects occur. For example, in BORIS [Dyer, 1983], the system would occasionally take as true information present in a user's question not previously known to the system. Although this has been shown to be a cognitively valid effect [Loftus, 1975], and does increase the system's opportunities to learn, it is not a desirable effect for an information retrieval system. In the SCISOR system, a strict difference is maintained between information read in articles, and information present in user's questions in that all information present in questions and not previously known to the system is flagged as potentially

unreliable. Thus the system believes nothing it is told, but everything it reads in the paper.

The model of retrieval discussed has a certain cognitive appeal. It exhibits the same kind of spontaneous recall ,*M* as people do, as is discussed in Schank's work on reminding [Schank, 1982]. Also, the answer to a user's question may be retrieved before the user has finished asking the question, an interesting effect also first achieved in Dyer's BORIS system.

V. Summary

SCISOR performs retrieval in a two-stage process. The first step, a spontaneous and coarse search, can be summarized as follows:

1. New inputs are instantiated: As the system reads new stories, or users ask the system questions, concepts are instantiated.
2. Marker passing occurs: Markers are passed to other instances related to the input instances through semantic category ("isa") links.
3. Items are spontaneously retrieved: When enough instances in an event have been marked, the event is retrieved from long-term memory.
4. Items are evaluated: The system must evaluate items retrieved for relevance in the processing at hand.

The second stage is a kind of graph matching process that performs a syntactic matching function on the likely candidates retrieved by the first process. This spontaneous method of retrieval elegantly solves some retrieval problems found in other systems:

1. SCISOR efficiently addresses a user's previously unanswered questions if an answer becomes known. Instead of always looking for answers that may or may not be present in incoming stories, incoming stories that relate to unanswered questions activate only those questions, which may then be considered.
2. SCISOR can locate relevant information in response to a user's questions, even when that question contains misleading or partial information. This capability is a by-product of the method of retrieval used.
3. SCISOR retrieves previous events when updates of those events are read. This is done in the same efficient manner as retrieving an unanswered question, and with the same tolerance for partial or contradictory input as in the question-answering case.

VI. Problems

Currently, matching of features can be relaxed only by the spreading of markers through the "isa", or abstraction hierarchy. An example of a more difficult problem is mentioned in a discussion of matching in KRL [Bobrow and Winograd, 1977]. That is, it might be nice to retrieve answers to questions like "Does John own a dog" given that we have in memory that John owns a dog license. This type of reasoning and deduction could be performed with the SCISOR system by an iterative spontaneous retrieval. John's owning a dog could potentially retrieve the information that dog owners have dog licenses, which could then activate John's owning of the dog license. The effects on processing time and efficiency of such iterative retrieval are unknown, they could be substantial.

Another limitation SCISOR has is in the kind of questions the system can answer. Currently the SCISOR system is capable of answering only questions about information explicitly stored in the knowledge base. Any information that potentially could be *reconstructed* or *inferred* from information stored in the knowledge base is not available. The line between what is explicit in a story and what can be deduced from that story is not sharp, because some amount of "figuring" must go on to obtain any reasonable understanding of the story. To obtain this understanding, SCISOR computes something similar to a maximally complete inference set [Cullingford, 1986] as the set of information present explicitly in articles and inferred from the context and other world knowledge. Anything in that understanding can be directly retrieved.

For example, SCISOR is able to answer the question "What company was sold for $3 billion?" without preindexing a story containing that information by AMOUNT-OF-SALE. However the system cannot answer the question "Which companies have been taken over more times than they have taken over other companies?" for example, because an answer would require counting all the times a company has been taken over and has taken over other companies, comparing these two numbers, and repeating the process for every other company in the knowledge base.

Another problem that ultimately must be addressed is the speed and memory requirements necessary in a viable information system. The effects of such a large knowledge base on the speed and accuracy of the retrieval mechanism are unknown.

VII. Conclusions

SCISOR is an experiment in the usefulness of a spontaneous, marker-passing approach to conceptual information retrieval. In its current implementation, it has demonstrated some promising results. The marker-passing scheme, combined with the knowledge representation used, seems to produce an effective contents-addressable, distributed representation. Retrieval occurs spontaneously when features in the input are related to features of events in memory. SCISOR can find answers to input questions even in fight of missing or misleading input information.

The system has not yet been tested on a large number of documents. However so far, the tests that have been performed are quite promising. Before any definitive claims can be made about the ultimate usefulness of this type of system, it must be tested with a large sample of documents in real IR tasks. The next stage of the project will include such tests.

References

[Besemer and Jacobs, 1987] D. Besemer and P. Jacobs. FLUSH: a flexible lexicon design. In Proceedings of the 25th Meeting of the Association for Computational Linguistics, Palo Alto, California, 1987. Forthcoming.

[Bobrow and Winograd, 1977] D. Bobrow and T. Winograd. An overview of KRL, a knowledge representation language. Cognitive Science, 1(1):3-46, 1977.

[Brachman et al, 1983] R. Brachman, R. Fikes, and H. Levesque. Krypton: integrating terminology and assertion. In Proceedings of the National Conference on Artificial Intelligence, pages 31-35, Kaufmann, Los Altos, CA, August 1983.

[Charniak et al, 1980] E. Charniak, C. Riesbeck, and D. McDermott. Artificial Intelligence programming. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

[Cullingford, 1986] R. E. Cullingford. Natural Language Processing: A Knowledge-Engineering Approach. Rowman and Littlefield, Totowa, NJ, 1986.

[Deering et al, 1981] M. Deering, J. Faletti, and R. Wilensky. PEARL: an efficient language for artificial intelligence programming. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia, 1981.

[DeJong, 1979] G. DeJong. Skimming Stories in Real Time: An Experiment in Integrated Understanding. Research Report 158, Department of Computer Science, Yale University, 1979.

[DiBenigno et al, 1986] M. DiBenigno, G. Cross, and C. DeBessonet. COREL - A Conceptual Retrieval System. Technical Report CS-86-147, Computer Science Department, Washington State University, 1986.

[Dyer, 1983] M.G. Dyer. In-Depth Understanding. MIT Press, Cambridge, MA, 1983.

[Jacobs, 1986] P. Jacobs. Language analysis in not-so-limited domains. In Proceedings of the Fall Joint Computer Conference, pages 247-252, IEEE Computer Society Press, Washington, DC, November 1986.

[Jacobs, Fall, 1987] P. Jacobs. Knowledge-intensive natural language generation. Artificial Intelligence, 31, Fall, 1987. Forthcoming.

[Jacobs and Rau, 1985] P. Jacobs and L. Rau. Ace: associating language with meaning. In T. O'Shea, editor, Advances in Artificial Intelligence, pages 295-304, North Holland, Amsterdam, 1985.

[Kolodner, 1984] J. Kolodner. Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model. Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[Lebowitz, 1983] M. Lebowitz. Generalization from natural language text. Cognitive Science, 7(1):1-40, 1983.

[Lehnert, 1978] W. G. Lehnert. The Process of Question Answering: Computer Simulation of Cognition. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[Loftus, 1975] E. F. Loftus. Leading questions and the eyewitness report. Cognitive Psychology, 7, 1975.

[Rau, 1987] L.F. Rau. Knowledge organization and access in a conceptual information system. Information Processing and Management, Special Issue on Artificial Intelligence for Information Retrieval, Forthcoming(Summer), 1987.

[Salton and McGill, 1983] G. Salton and M. McGill. An Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983.

[Schank, 1982] R.C. Schank. Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press, Cambridge, 1982.

[Schank and Abelson, 1977] R. C. Schank and R. P. Abelson. Scripts, Plans, Goals, and Understanding. Lawrence Erlbaum Associates, Halsted, NJ, 1977.

[Stanfill and Waltz, 1986] C. Stanfill and D. Waltz. Toward memory-based reasoning. Communications of the Association for Computing Machinery, 29(12):1213-1228, 1986.

[Tulving, 1972] E. Tulving. Episodic and semantic memory. In E. Tulving and W. Donaldson, editors, Organization and Memory, pages 381-403, Academic Press, New York, 1972.

[Wilensky, 1986] R. Wilensky. Knowledge Representation - A Critique and a Proposal. In J. Kolodner and C. Riesbeck, editors, Experience, Memory, and Reasoning, pages 15-28, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Young and Hayes, 1985] S. Young and P. Hayes. Automatic classification and summarization of banking telexes. In The Second Conference on Artificial Intelligence Applications, pages 402-208, IEEE Press, 1985.