Draft for comment

Important problems in information retrieval

Dagobert Soergel
College of Library and Information Services
University of Maryland
College Park, MD 20742

August 1989

(Most of the work on this paper was done during the author's stays as visiting professor at the Graduate Library School, University of Chicago)

Table of contents

Preface

This paper gives a panoramic overview of problems that arise in the design of information retrieval systems and in information retrieval research and puts the individual problems in perspective. It can serve as a framework for research and for curriculum development. I hope to give the experienced reader some new perspectives and relationships not commonly thought about, and to the newcomer a sense of the spirit of the field.

Overview

Problem areas 1 and 2 deal with the user, her problems, information needs, and information processing behavior. Problem areas 3 - 5 deal with the design of systems that can produce helpful information packages. Problem areas 6 and 7 are concerned with a closer coupling between information systems and users' daily work. Finally, problem area 8 returns to the user: Did the information provided have the desired effect in terms of improved task performance?

Introduction

The purpose of information retrieval - I am using this term very broadly - is to provide information that changes the knowledge state of a user so that this user is better able to perform a present task (solve a problem, make a decision), is better prepared to perform future tasks, is better able to assimilate information needed for a present or future task, is better entertained, more curious, more occupied with interesting things, is happier, or enjoys in other ways a better quality of life.

In this paper I will consider an information retrieval system as a problem solver's assistant. Such a system should assist with the three steps or functions in each "information cycle" executed during the problem solving process. These three functions are: determining information needs, finding information, and applying information. A problem solver carries out these functions - if perfunctorily - every time he applies information in the problem solving process (Figure 1).

1. Clarify and analyze the problem or a problem aspect.
   Define sub-problems.
   Determine information needs.

2. Find information.

3. Apply the information.

Figure 1. **Steps of applying information in problem solving**

These steps or functions overlap and are nested: Clarifying and analyzing the problem may require information; one often needs to find out what to find out. Finding information itself involves problems such as finding some documents and deciding which ones to read, or selecting a database to search and

devising a search strategy, or deciding which expert to believe - all problems that in turn require information. Applying information found often requires additional information to show **how** the information found can be applied.

All three functions are important for a successful information cycle. But most systems assist only in function 2, finding information once the need is known. Usually they do this in a rather straightforward manner that does not do justice to the decision problems involved. More often than not - depending on the match of the problem with her abilities and background - the user does need assistance with problem analysis as well. This is crucial, because everything else depends on a good perception of information needs.

Thus the first information retrieval problem:

Problem 1. Assisting the user in clarifying and analyzing the problem and determining information needs

Such assistance involves the steps shown in Figure 2. The challenge is to design and implement systems that can provide such assistance.

Clarifying and analyzing the problem.

Determining what part of the problem solution can be effected by the system and what part is left to the user.

Determining what knowledge the user requires for her part in the problem solution.

Determining what the user knows already.

Deduce what information is necessary to lead the user from her present knowledge state to the required knowledge state.

Figure 2. **Steps in analyzing the problem and information needs**

Assistance in problem analysis can be provided by intermediaries - the human components of ISAR systems. Intermediaries must be fluent in problem solving and in techniques for determining the user's state of knowledge, and they must know the subject area they are dealing with (Problem analysis requires a good deal of subject knowledge). These requirements should be considered in basic and continuing education.

The problem analysis function has important consequences for the need for intermediaries. Discussion on this need has focused on the end users' ability to negotiate search systems: If we could only build sufficiently user-friendly systems, so the argument goes, there would be no need for intermediaries. This argument misses the point that in many situations the most important function of the intermediary is assistance in clarifying the problem. Computer systems need to be far more than just easy to use to take over parts of this function.

Assistance in problem analysis can also be provided by computer systems. The system can provide the user with information helpful for problem analysis. This strategy can be implemented even in a bibliographic ISAR system by introducing descriptors for types of problems and indexing helpful documents with the appropriate problem descriptor. A list of these problem descriptors is displayed for the user; he selects the appropriate one and retrieves documents that can assist in problem analysis.

But the ultimate goal is a computer system that can engage the user in a problem-analysis dialog. Such a system needs a large knowledge base consisting of a list of problems, arranged in a hierarchy of problems and subproblems, and for each problem a list of types of helpful information, indicating for each type how it would be helpful for solving the problem. One may object that the number of possible problems is immense and that there is no hope of capturing them all in a database. The answer is two-fold: establish specialized databases and allow for user enhancement whereby a user having solved a problem contributes knowledge about it. One may also object that many problems are new and ill-structured and thus are - almost by definition - not amenable to this approach; these may be the most important problems. This point is well-taken. But it should not deter us from building systems that provide what assistance **is** possible considering the degree of difficulty and newness of the problem at hand. Furthermore, problem difficulty is a matter of degree, and some assistance is better than none at all.

Very similar to determining the information needed to solve a problem is the task of determining what information one should assimilate to reach an educational goal - the task of putting together an educational program, especially an individualized program. A database relating educational goals to specific topics could support this task.

In information systems design and operation there are two levels at which determination of information needs is required. Data on information needs of the user population in general are needed in the design of an information system. Data on the information needs of a specific user are needed to provide this user with suitable information.

Finding out about the user's present and required state of knowledge is not easy. It requires understanding of the user's task and determination of what information might be sufficiently helpful. Some tools are provided by cognitive psychology, especially theories of memory. Such theories are closely related to knowledge representation in computer systems (see below).

Understanding the problem at hand and the information requirements that follow is not sufficient for preparing a useful information package. One must also know how people in general and the user at hand in particular assimilate and use information in order to gear the information package to the user's mind set and thinking patterns.

Problem 2. Knowing how people use and process information

Problem 2.1. Relationship of information use to the problem-solving/decision-making process

Content and form of the information presented must be adapted to the stage in the decision-making process and in tune with the motivational state of the user. For example, if the user has already made a preliminary decision, information contravening that decision must be presented very forcefully to over-come the tendency to disregard such information (even then the user may not accept the information).

A similar situation exists in the provision of information for a research or development project. The type of information varies as the project progresses from the inception and planning phase, which requires general orientation, to the main work phase, which requires ideas on data collection techniques, instrumentation, or design and specific information, such as properties of materials, to the write-up phase, which requires verification of specifics but also general information for putting the results in context.

Problem 2.2. How do people make relevance judgments

If we know how people make relevance judgments, we can improve procedures for determining relevance in indexing and searching. We can also decide better what summary information should be presented to a user to support good relevance judgments. Bad relevance judgments due to insufficient information cause information loss, especially when these relevance judgments are used to modify the query formulation, as is done in interactive retrieval with feedback.

Existing studies of relevance judgments shed light on the information, the points of view, and the inference and weighting procedures that people use in making such judgments. Future work might add to our knowledge through an emphasis on the mental processes that occur as a user judges the relevance of a document or a piece of information **with respect to its contribution to problem solving** and assesses the nature and size of this contribution. One might also compare the user's judgment with usefulness as established by an expert. Such studies might reveal that some people do not have the background or mental ability to recognize relevance in cases where it is not immediately obvious.

An information retrieval system should assist users in recognizing relevance; otherwise the user might fail to recognize the significance of a piece of information. For example, document records might be enriched by notes of the form

> Useful for problem X because of Y.

Problem 2.3. How do people organize information in their minds, acquire it, process it for output

*****

Once the user's problem understood, the system should solve the problem or give the information needed to solve the problem or move it closer to a solution. Thus the overall problem of information retrieval:

Problem group	**Assembling a package of information that enables the user to come closer to a solution of his problem.**

This is the basic information retrieval problem. It entails assistance to the user in finding information and in applying information to solving the problem (see Figure 1). These two functions must be treated together, because they are often inextricably linked, as the example given in Figure 3 shows (example due to Don Swanson).

Query topic

Dietary treatment of Raynaud's syndrome.

There are no documents on this topic.

However, there are documents that show that

Raynaud's syndrome can be treated with a drug that causes certain changes in the blood.

There are furthermore documents that show that

Dietary fish oil causes the same changes in the blood.

Clearly these two sets of documents are relevant when combined with each other. Retrieving the two kinds of documents requires inference; the system must apply the information to the problem.

Figure 3. **Retrieval by inference.**

The example shows that retrieving helpful information often requires inference, and inference goes beyond mere retrieval to problem-solving application. Such inference is by no means simple and - if it is to be done automatically - requires that documents are indexed very explicitly using role indicators. A system like MEDLINE would have to be refined considerably in order for this to be possible through a simple search; as it is, making the connection requires considerable effort, including reading or skimming the actual articles.

The ISAR system power and the nature of the problem determine the distance between the information package delivered to the user and the problem solution. On one end of the scale, the information package

spells out the finished solution. On the other end, it consists merely of a list of documents from which the user must gather the information needed and then apply it to solve the problem.

The question, then, is how to assemble such an information package. To address this question, we must first lay out the requirements that such a package must meet. The package must give the needed information while recognizing that the user has a limited capacity for assimilating and processing information (see Figure 4).

Give all information needed to lead the user from his present knowledge state to the required knowledge state.

Minimize user time and other user resources needed to process the information by giving the information

in a format that is easily understood and/or suitable for further computer processing;

in the sequence best suited for understanding;

in the shortest space.

Omit pieces whose value is too small to warrant their cost.

Explain how a solution or partial solution was arrived at or how pieces are helpful in solving the problem.

Figure 4. **Criteria for information packages.**

All of these criteria must be evaluated from the point of view of the user at hand and his or her background knowledge, preferred form of presentation, and cognitive style. Different users, all needing the same information, often need it packaged and presented in different ways. If this sounds like a prescription for preparing educational materials, it is no accident. Information retrieval and education both have the objective of transferring information to people to change their state of knowledge; information retrieval and education have a lot to learn from each other.

In the following we discuss these criteria in more detail.

All information. Often the user does not have the background needed to understand information and apply it to the task. An information package must supply the needed background, perhaps through several steps. It must provide a ladder that takes the user from her present state to the required state of knowledge. In order to produce such a response, the system must have two kinds of data. It must have data - as complete as possible - about the user's present state of knowledge. And it must have data about the prerequisite ordering of topics or topic presentations, such as documents.

Format. The format must support assimilation and further processing of the information presented. The factors to be considered include: the use of language versus tables, graphs, and charts; the language to be used; vocabulary; style, and requirements for further processing: If the user wants to process data - text, numbers, or other data - by computer, the data should be made available in machine-manipulable form together with the tools for such manipulation (such as a program for required statistical manipulations), unless the user already has them.

Sequence. The sequence of presentation affects assimilation and understanding. Consider a system that retrieves individual paragraphs of text that describe just what them user needs to know. These paragraphs must be arranged in a sequence that makes for coherent reading. Sequencing is also important in bibliographic retrieval. The sequence in which documents (or document records) are presented affects relevance judgments. One may sequence document records in such a way that reading titles or abstracts prepares the user to better recognize the relevance of documents listed later. Or one can place the most relevant documents first, so that the user examining a document record later in the sequence can quickly see that the document is unlikely to add anything to the documents already seen.

Brevity. Some components of brevity are:

> **Specific selection**. The ideal system selects relevant chunks of information (for example three paragraphs from a document), so that the user need not assimilate irrelevant material and spend time on deciding that it is irrelevant.
>
> **Absence of unneeded redundancy**. The ideal system omits repetitive material, even if it is relevant, so that the user need not read the same thing twice. On the other hand, some pieces of information may have to be presented twice or more times, perhaps in different formats or treated from different viewpoints, before they are assimilated.
>
> **Degree of processing, presenting only the results**. The ideal system aggregates data and performs other statistical analyses, uses models and simulation, and draws logical inferences, presenting only the results when the raw data are not needed for solving the problem. The data from which the results were derived may be presented in an appendix or in explanations.

Value. Information may be helpful but not be helpful enough to warrant the resources (both the system's and the user's) needed to acquire and assimilate it. The question is how much improvement in task performance can be expected as a result of a chunk of information, and what increase in ultimate benefits can be expected from this improvement in task performance. The increase in ultimate benefits may be called the value of the chunk of information. This expected value must exceed some other value that could be acquired with the same resources. Ideally an information retrieval system should estimate the expected value of a chunk of information, its expected cost, and what else could be done with the resources before presenting a chunk of information. To make things more complicated, the contribution of a chunk of information to problem solution, and therefore its value, may well depend on what other information is available in the user's head or in the information package given to the user.

Explanation. Explanation is needed for several reasons. (1) Before acting on a conclusion drawn by an inferential retrieval system (such as a system for medical diagnosis), most users would want an explanation and justification for the conclusion. (2) The user may not see how certain information is

helpful in accomplishing a task; without an explanation the user may not be willing to spend the resources (time, money) to acquire and assimilate that information. (3) An information system may need to engage the help of the user in selection. In that case it should be able to present the user with a brief description or summary of a package of information (which may be a small chunk, a group of similar chunks, a whole document, or even a whole group of documents).

The ground is now prepared to talk about the design of systems that can produce such information packages. This problem - the retrieval problem - has many subproblems that will be treated individually in the rest of this paper.

How good an information system is at putting together appropriate information packages depends crucially on the structure and content of its database/information base/knowledge base. (These terms all mean the same thing, with knowledge base preferred in systems that do not simply retrieve information but that do apply it to solving the problem.) The central problem in information system design is, then,

As an — often preferred — alternative to the system providing an information package in response to a one-shot question, the system may support the user in collecting pieces of information through navigation (Marcia Bates' berry-picking), which provides more opportunity for the user to reformulate his need in response to what he learns. Figure 4a gives requirements for navigation support.

A system to support navigation must do the following:

Find a starting node.

Offer links to nodes that .

give information helpful for a solution of the problem
and/or
give information helpful in further navigation.

Offer specific links that enable specific selection of the next node to be visited.

Lead to a useful sequence of nodes visited.

From several versions of a node select the one most appropriate to the user's background

Figure 4a. **Criteria for navigation systems**

Problem 3. **Knowledge representation**

A knowledge/information/data base can be seen as an assembly of facts and rules (see Figure 5). (The division between facts and rules is by no means sharp.)

Sample facts

MEDLINE **is intended to inform** physicians
for the purpose of patient care

Document-325 **is about or relevant for** Pneumonia

Penicillin **cures** Pneumonia

John **has** Pneumonia

Sample rule

IF person X has disease Y **AND**
drug Z cures disease Y
THEN person X should-take drug Z

Figure 5. **Facts and rules as knowledge elements**.

The way this knowledge or information is represented inside the system determines what information can be stored, what operations can be performed on it, and how efficiently information can be stored and processed. The way knowledge or information is represented in output to a user or another system determines the effectiveness and efficiency of transmission.

The problem of knowledge representation occurs on four levels:

1 The general approach to knowledge representation.

2 The conceptual schema defining the nature of the data in the database through specifying entity types and relationship types.

3 Lists of entity values for each entity type.

4 The actual data/knowledge.

Each level guides the next lower level.

Problem 3.1. Choosing the general approach to knowledge representation.

Facts and rules can be represented in natural language, or in tables, charts, and graphs primarily for human comprehension, or in a more tightly structured way more amenable to computer processing but sometimes also useful for humans. The most important approaches to knowledge representation are shown in Figure 6.

The entity-relationship approach.

Predicate logic.

Semantic nets.

Rule-based approach.

Frames and scripts, case grammar, facets.

Records.

Tabular representation.

Graphical representation.

Grammar (on the level of clause, sentence, paragraph, entire text or discourse). Composition rules.

Figure 6. **Approaches to knowledge representation.**

These approaches can be combined; for example, a semantic net whose nodes are chunks of text is known as hypertext. Choosing the appropriate combination of representational approaches, considering the interplay of task and user characteristics, is the key to optimal storage or packaging of information.

The belief is gaining currency that many of the approaches shown in Figure 6 are notational variants of each other; that is, they can express the same content but do so in different form, which in turn is related to storage structure and efficiency of processing. For example, frames can be thought of as superstructures that group data items that are individually expressed as entity-relationship statements.

Some ideas on structuring and processing information are linked with particular approaches and - at present - supported primarily by software based on one approach. In particular, the ideas of default values, of using certain procedures to determine a given value, and to initiate a certain procedure when a certain value is entered (for example, a procedure to update another part of the database) are associated with frame representation. The ideas of hierarchical inheritance and of spreading activation are associated with semantic nets. But there is no reason why these ideas should not be used generally.

Some software tools now available provide for easy implementation of several of these approaches and include all the ideas on structuring and processing mentioned. These tools let the designer choose the approach most appropriate for the type of knowledge/information at hand.

Classification

Problem 3.2. Constructing a conceptual schema.

A general approach to knowledge representation does not provide the detailed structure needed for constructing a knowledge base. This detailed structure is provided by a conceptual schema that specifies the entity types and relationship types or the frame structures and frame interrelationships needed to represent the knowledge/information to be stored. Some sample entries are given in Figure 7.

atom-pair **bound-through** [bond-type,
orbital-of-A,
orbital-of-B,
bond-characteristics]

process **has-component-process** [process, sequence-number]

process **has-starting-product** [substance,
affected-substructure,
facilitating-substructure,
inhibiting-substructure,
phase, amount]

Figure 7a. **Conceptual schema for chemical data. Sample entries**

database **produced-by** legal-entity

database **implemented-as** [db-impl, document-form, legal-entity]

database **intended-for** [class-of-legal-entities, purpose]

database **gives-data-on** [rel, identifier-form, db]

(The following is an example of a fact recorded with the relationship **gives-data-on**:

MEDLINE **gives-data-on** [entity deals-with subject, term, MeSH])

Figure 7b. **Conceptual schema for data about databases. Sample entries**

text-1	**includes**	text-2
text-1	**is-continuation-of**	text-2
text-1	**has-prerequisite**	text-2
text-1	**is-summary-of**	text-2
text-1	**has-same-content-as**	text-2
text-1	**is-simplified-from**	text-2
text-1	**is-supported-by**	text-2
text-1	**is-corrected-by**	text-2
text-1	**is-criticized-by**	text-2
text	**is-relevant-for**	concept

Figure 7c. **Conceptual schema for data about pieces of text in a hypertext. system. Sample entries**

The conceptual schema, together with the lists of entity values to be discussed next, determines precisely what knowledge or information can be stored and how this information can be processed. For example, in the hypertext system, a user who does not understand a piece of text can follow a **has-prerequisite** link to find a piece of text to read first, or a user can follow a **is-criticized-by** link to find criticism. All of this depends on the design of the conceptual schema. If the designer forgot to introduce the relationship type **is-criticized-by**, the user is deprived of an avenue for retrieval.

A conceptual schema encompasses a good deal of highly formalized knowledge about the structure of the domain to be represented. Therefore, developing a conceptual schema requires a good grasp of this domain structure. This is true particularly for developing a conceptual schema to represent what I call substantive data (as opposed to bibliographic or pointer data).

A conceptual schema serves to organize data/information/ knowledge for the purpose of retrieving and processing information as an aid in solving problems. Therefore, developing a conceptual schema requires a good knowledge of the problems/requests to be expected and the kinds of information needed to compose suitable responses.

Developing conceptual schemata also requires a good methodology for extracting the needed relationship types, entity types, and entity values from the knowledge of domain structure and of problems/requests and for creating order and structure in the often bewildering picture that results.

Problem 3.3. Constructing a list of values for each entity type or rules for generating such values

Once a conceptual schema with its entity types and relationship types has been constructed, the designer must address the problem of entity values. Each entity type must have specifications as to the permissible entity values. Often this is done through enumerating these values in a list. I call the conceptual schema - the specification of entity and relationship types - augmented by the specification of permissible entity values the **extended conceptual schema**.

Much of the success of information retrieval systems - including expert systems - depends on properly chosen and defined entity values and their relationships as recorded in a thesaurus. Consider the concept **Dance** in an information retrieval system for referral. **Dance** should be recognized as a combination of **Movement** and **Music**. This aids retrieval for a client who, on his doctor's recommendation, is looking for a movement class. Due to the proper conceptual relationship a dance class is found. If the thesaurus builder had not recognized the relationship between **Movement** and **Dance**, the dance class would not be found and the client would miss a valuable opportunity. To give another example, an expert system for medical diagnosis needs excellent classifications of diseases and of infectious organisms, among others. As these examples show, the need for representing vast amounts of knowledge in expert systems gives new significance to thesaurus construction.

Further examples of lists of entity types are:

a list of subject descriptors,

a scheme of chemical bond types,

a scheme of crystal lattice types,

a system of chemical structures,

a list of purposes a database might serve.

Once the values for an entity type are defined, facts expressing hierarchical and associative relationships among them can be entered into the database. In a bibliographic database and in other contexts the set of values for the entity type subject together with the relationships among them are called the index language. If there are additional terms in the database with relationship facts linking them to subject values, one speaks of a thesaurus. While thesaurus data are part of the database, their status is somewhat different from facts describing documents or food products or persons or what have you. The reason is that thesaurus data are more stable - they require less updating - and that their use is much more pervasive than, for example, data describing a single document.

The entity values and the relationships between them must be displayed in a sensible and easily comprehended order, providing a framework to support knowledge acquisition/indexing as well as searching, especially browsing. Such a framework may also serve users for organizing information in their minds.

At the end of Section 3.2 we remarked on the need for subject knowledge and reliance on expected problems/requests in the construction of a conceptual schema. These remarks apply here as well.

Problem 3.4. Knowledge/data acquisition and assimilation

Data are acquired by adding facts and rules to the database. Data are assimilated by organizing the facts and rules in the database into a coherent, non-redundant whole. Data acquisitions involves transforming data from some other representation - often from the representation stored in someone's head - into the representation used in the database. This transformation is often done by human indexers or knowledge engineers so that the input to the system consists of data in the proper format. But sophisticated systems can do the transformation themselves, and input may consist of natural language text (see Problem 4.1 Transformations). Assimilation involves operations to remove redundancy by eliminating duplication, by restructuring the database in accordance with hierarchical inheritance, and by discovering generalizations from which many specific items of information can be deduced and can thus be removed from the database. Similarly, several specific rules can sometimes be replaced by one general rule. (See Problem 4.6.)

Knowledge/data acquisition and assimilation is the single most voluminous and resource-intensive task in developing an information system. Some modes of knowledge acquisition are shown in Figure 8. They must all be supported by cleverly designed procedures. Such procedures use the functions in information processing shown in Figure 9 and discussed in problem area 4. Such procedures often require large amounts of knowledge which must be acquired before the knowledge acquisition task at hand can begin; it takes knowledge to acquire knowledge.

Transforming information from natural language representation into an internal representation.

Picking an expert's brain.

Collecting data through automated measurement.

Figure 8. **Some modes of knowledge acquisition.**

Having discussed the four levels of knowledge representation, we now turn to two additional aspects, uncertainty and granularity.

Problem 3.5. Representing uncertainty

Many "facts" in a knowledge base are not known for certain but only with a known or estimated probability. Many inference rules are likewise probabilistic in nature. For example, rules leading from a set of symptoms to a disease often yield probabilistic rather than certain statements: patient X has disease Y with probability Z. This probability may furthermore depend on the strength of the symptoms. Likewise, the occurrence of a given term in a document may allow only a probabilistic statement at retrieval: document X is helpful for solving problem Y with probability Z; this probability may depend on the frequency with which the term occurs in the document. The knowledge representation mechanism must be able to represent this uncertainty so that it can be considered in searching.

Problem 3.6. Developing fine-grained information systems

Knowledge or data can pertain to many and varied types of entities. If they pertain to entities that are in turn used as sources of information, such as documents or persons consulted as experts, I speak of pointer (or directional) data, otherwise of substantive data. The user ultimately needs substantive data. If a database of substantive data is properly formatted, exactly the data needed can be selected; this provides a fine-grained approach to retrieval. Furthermore, inference can be used for retrieval, and the data retrieved are in a form suitable for further processing.

Documents are pre-made information packages which often contain a few pieces of information that are relevant to the problem at hand among many that are not; bibliographic retrieval is thus coarse-grained. More specific selection can be achieved by operating on the text of documents found and extracting the specific data needed. This can be done by a human analyst or by a sophisticated computer program. Another way to achieve a finer grain is providing access to smaller information packages (chapters or smaller sections, paragraphs, even sentences). The smaller the size of the pre-made packages, the finer is the grain of retrieval.

This leads to the idea variously known as automated encyclopedia, dynamic book, or hypertext. In such a system, pieces of text should be linked by a rich network of relationships. Some of the relationship types to be used are shown in Figure 7c.

A fine-grained information system needs a very elaborate conceptual schema.

Problem 4. Procedures for processing knowledge/information

We represent knowledge/information so that we are able to process it, operate on it. Processing takes place at input, during reorganization of the store, and at output.

Operations on knowledge/information involve one or more of the information processing functions shown in Figure 9. These functions overlap and are often used in conjunction with each other.

Transformation from one representation to another, possibly combined with data extraction.

Removing redundancy.

Summarization.

Computing statistics.

Deriving generalizations.

Drawing inferences.

Search and selection: predicting relevance of an item based on predictive clues.

Indexing: attaching predictive clues.

Figure 9. **Functions in information processing.**

These functions are often performed by humans, and many problems in information retrieval have to do with the analysis of the mental processes and other factors determining human performance. But much work is also done on devising computer systems that can - to some extent - perform these operations. Computers can deal with the large bulk of data and the often extensive processing needed for retrieval.

In the following we discuss these operations individually.

Problem 4.1. Transformation from one representation to another

Transformation from one representation to another is often needed between systems in order to transfer information from one system to another. Transformation may be needed within a system to arrive at a representation more suitable for certain operations.

A strict transformation leaves the basic content unchanged; however, subtle changes in content (meaning) are often unavoidable. Often, transformation goes hand in hand with **data extraction**: only selected data are transformed. One may also think of data extraction without transformation.

Since transformations must be done on large volumes of information, it is desirable to do them by computer. To do this, many problems in natural language processing must be solved.

Problem 4.1.1. Translation from one natural language to another.

The problem of translation between natural languages arises in systems that receive descriptors, abstracts, or full text in two or more languages. Translation is important for searching and for the presentation of search results. In searching, one should be able to search the entire database in one language or in any of the several languages; this can be accomplished either by translating each query into all languages found in the database, or by translating all texts at input into one common language and translating queries into the same language. In output, the presentation of text in the user's mother tongue or another language he can easily understand contributes to speedy comprehension. Translation for presentation may be done either ahead of time at input or ad-hoc once items have been retrieved.

Translation engenders costs for processing and possibly for storage. Human translation is very expensive and is usually done only on demand, except for items with high expected usage. However, human translation on demand introduces an extra wait for the user. For both reasons machine translation is receiving renewed attention, particularly in Europe, where economic integration makes the barriers stemming from language diversity more acute. The purposes of translations are best served by translation produced at input and stored. In addition to the high processing costs (considerable even for machine translation) this approach causes storage costs, particularly if each item is stored in several languages.

Problem 4.1.2. Translation from natural language to a formal representation

A large amount of information is available in the form of natural language text but is needed in other forms for efficient processing. Efficient translation into a formal representation is needed to overcome the "knowledge acquisition bottleneck". Such translation has been done primarily by humans; the development of computer systems for the task is urgent.

Consider the following examples:

> Translating the title pages of books or journal articles into bibliographic records.
>
> Translating the data given in medical textbooks into a form that can be used by a medical expert system.
>
> Translating the data in a very large handbook (perhaps several hundred volumes) into a form suitable for an online database. (Some retrieval operations may be possible simply on the machine-readable text, but this would not even begin to exploit the possibilities offered by a formal representation of the data.)

In the type of translation illustrated by these examples, the formal representation often does not cover all the information of the natural language text. In other words, systems often perform data or knowledge extraction rather than complete translation.

Translation into a formal representation is also important for the input of queries as discussed. **Where?** Finally, it might be useful at output from a full-text system when a diagrammatic representation of certain relationships discussed in a text may be helpful to the user; in this case the operation may again involve both selection and transformation or other kinds of summarization.

Problem 4.1.3. Translation from a formal representation to natural language

A system response should be easily understood by the user. The user may have difficulty with a response in the internal format for data representation or even with a list or table; translation into natural language is then desirable. Again, such translation can be done ad-hoc as the response is generated or ahead of time at input. Depending on the difficulty of translation and on the quality of a translation produced entirely by computer, one might store the entire contents or large parts of a database both in a formal representation and in text form. Such double storage might be useful for a handbook database that is to be made available for online searching as well as in print.

Problem 4.1.4. Translation from one formal representation into another

As more and more data become available in formatted databases, this problem takes on increasing significance. The examples shown in Figure 9a start with the trivial - but nevertheless important - problem of code conversion and end with more substantive translations.

Converting from one code to another, such as from ASCII to EBCDIC.

Converting from one text mark-up format to another; for example, from one word processor to another.

Converting from on index language to another.

Converting from one database format to another.

Convert a rule-based representation of data implemented as a Lisp program into a representation based on facts and rules implemented as a Prolog program.

Figure 9a. **Translation between formal representations**

Problem 4.1.5. Expression of data in tabular or graphic form adapted to the user's purpose

A mention of these two problems must suffice here. Computer graphics in particular is a burgeoning field of its own. It presents on of our best hopes to increase the human assimilation rate for information.

Problem 4.2. Removing redundancy

Removing unneeded redundancy is important for economy of storage and data integrity, for economy of transmission, and for economy of assimilation. This is one of the themes in data base management. Three kinds of redundancy are discussed below in order of increasing complexity.

There is straightforward redundancy, the simple repetition of data. For example, bibliographic union databases such as OCLC this often contain duplicate records for the same book cataloged by different

participating libraries. A bit less obvious is the repetition of data in separate records for several editions of a book. The problem can be solved by applying the principle of hierarchical inheritance to bibliographic databases: One introducers one record with the data in common for all editions and subordinate records with the data specific to each edition. A sophisticated system can recognize and fix these situations automatically.

More subtle cases of redundancy arise when two pieces of text (two paragraphs, two documents) overlap in the substantive data they present, or when a document could be rewritten to half the length without loss of information.

The most complex kind of redundancy arises when some data in a database can be inferred from other facts and rules in the database. In some cases it may be possible to reduce a database to a generating core, that is a set of general laws or rules, together with a set of facts such that all the other rules and facts in the database can be derived from the core by inference and thus be discarded without loss of information. This case is more complex still if the general laws are yet to be found.

Problem 4.3. Summarizing

Summaries can be a great help to a user in navigating through a data base. Often a summary can replace a piece of text or a body of data.

A summary or abstract of a text or other corpus of data can belong to one of three types or be a mixture of these:

- Selection summary or excerpt.

- Generalization summary.

- Indicative summary.

A selection summary or excerpt consists of specific chunks of information selected from the body of data being summarized; example are the conclusions of a scientific article or a commission report or the most salient facts from a news story. If the user needs only the chunks selected, the summary will do very nicely in lieu of the original. Summarization by selection is often used when a text serves as a source of data for a database, such as the knowledge base of an expert system; only selected data - not all the data represented in the text - are included in the database (data extraction, see above).

A generalization summary consists of general laws or principles derived from the body of data being summarized. If the problem at hand can be solved using the generalizations without recourse to the specific data, the summary can be used in preference to the original. Sometimes general laws and some specific data form a generating core from which the rest of the summarized data can be inferred, as discussed under removing redundancy.

An indicative summary consists of data that describe or indicate the kind of data to be found in the item summarized. The sole purpose of an indicative summary is to allow the user to make a judgment on whether to consult the body of data that is summarized.

Problem 4.4. Computing statistics

**

Problem 4.5. Deriving generalizations

Deriving generalizations is important for a parsimonious storage of knowledge in the mind, in computers, and in documents. We saw this already in the sections on removing redundancy and on summarization, where the link to hierarchical inheritance was also discussed. Just as important, generalization often allows for more powerful application of knowledge.

Generalization is usually done in the reorganization of a store of information/knowledge. However, the answer to a query may require on-the-spot generalization: The answer may not be available in the store explicitly, but it may be possible to derive the answer from the stored data through generalization.

Generalization is the business of science and is usually considered a task for the human investigator. However, computers have been programmed to derive generalizations, and this is an important problem. **Machine learning** *more*

Problem 4.6. Drawing inferences

Inference combines data (rules and facts) to arrive at new data, often about a specific entity. It is, in a sense, the inverse of generalization.

Drawing inferences allows for answering queries even if the answer is not stored explicitly in the database. It is the key to avoiding information overload, since answers produced through inference are much closer to the problem solution than answers that merely retrieve what is in the database. In other words, the capability to draw inferences opens a whole new realm of questions that can be asked.

The capability of drawing multi-step inferences employing many pieces of evidence distinguishes an expert system (a sophisticated information retrieval system) from a plain non-inferential information retrieval system. [** repeated in search** One must be careful, however. Plain retrieval operations, such as finding a disease based on a combination of symptoms, can be disguised as inference. (In a formal way, any retrieval prescription is an inference: IF document indexed by X, THEN document relevant. Here we talk about more complex inferences.)]

Inference is often not a yes-no proposition. The facts and rules used in drawing inferences are often probabilistic. Thus we need to use probabilistic reasoning; the problems encountered are very similar to the problems of probabilistic retrieval.

Problem 4.7. Search and selection

Operations on information culminate in search. A search produces an information package as described in the beginning of this paper.

4.7.1. Search as prediction based on retrieval clues

In a straightforward non-inferential search - such as in most bibliographic retrieval - one tries to predict the relevance of an item - such as a document - based on a variety of retrieval clues. A retrieval problem is exactly parallel to a research problem: The retrieval clues are independent variables which are used to predict the value of the dependent variable - the relevance of the item for the problem at hand.

As the researcher must carefully choose the independent variables and the formula which combines them into a predictor of the independent variable, so must the searcher carefully choose the retrieval clues and the formula for computing predicted relevance. But while the researcher generally deals with recurring situations and can therefore investigate the relationships between dependent and independent variables in one instance and then use the results for prediction in the next, the searcher often deals with one-of-a-kind query topics and must make a best guess as to which retrieval clues to use and how to combine them. In this situation the searcher can learn only from feedback within the search at hand, adjusting the retrieval clues in response to preliminary retrieval results (relevance feedback). There are also search programs that do this automatically: They ask the searcher for relevance judgments as the search proceeds, use the answers to derive conclusions on the usefulness of the various retrieval clues, and adjust the query formulation accordingly .

Retrieval clues for a given entity are established by the relationships of that entity to other entities. The problem of choosing retrieval clues presents itself on two levels:

- The extended conceptual schema level: Choice of the universe of retrieval clues available in the system (the index language).

- The knowledge acquisition level:

    - Choice of retrieval clues to be associated with an individual entity (indexing), an issue inextricably intertwined with searching.

    - Choice of retrieval clues for an individual query.

For documents and other entities represented as text or digitized data (such as a digitized image), retrieval clues can be chosen from characteristics inherent in the entity itself, such as words in the title, abstract or text, author name, journal, words in the journal title, publication year, etc. One can define the universe of available retrieval clues as the totality of all such characteristics, or one can select a subset of "good" retrieval clues to lower costs. The question of what makes a good retrieval clue is discussed below.

For any type of entity, one can assign retrieval clues by indexing. One must define the universe of retrieval cues from which the indexer or indexing process can choose and rules for assignment.

In both cases the question arises: What makes a retrieval clue "good", what makes it suitable for inclusion in the index language? One answer considers the usefulness in queries - but that judgment depends on the ability to anticipate future queries. Another answer considers the power of a retrieval clue in discriminating between items in the collection. This can be computed for the present collection, but will it hold for the future? And does good discrimination translate into usefulness in searching?

4.7.2 Search as a probabilistic problem

This section elaborates on a set of related ideas that are implied by the discussion in the previous section.

Many searches are by their very nature probabilistic in that the retrieval clues are rarely perfect predictors of relevance. The search problem is to find the predictors suitable for the query at hand and combine their predictive strengths into a single value of predicted relevance which is derived for each document. (There is a debate whether this value should be interpreted as the probability of a document being relevant or as the strength of the document's relevance. More generally, one should probably think in terms of predicting a most likely relevance value within a confidence interval.) The goal is to maximize predictive strength, i.e., the correlation between relevance predicted by the system and relevance ascertained in some more reliable way.

The best way for weighting individual retrieval clues or descriptors and then combining to arrive at a total figure for expected relevance is a problem of long standing. One can use any or all of three kinds of weight:

- The intrinsic "goodness" or retrieval power of a descriptor.

- The strength of association between a descriptor and the entity to which it is assigned (indexing weight).

- The importance of the descriptor for the query at hand (query weight).

Many formulas for combining retrieval clues have been used. Still others could be developed by considering the various ways in which independent variables are combined to predict values of the independent variables in research studies. Any formula must cope with the following problems:

- There is often a strong interdependence between descriptors. Regardless of the query configuration, this lessens the predictive power of a descriptor if a highly correlated descriptor has already been considered.

- There is often a very strong positive interaction effect. The relevance of an entity having both descriptors A and B is much greater than would be expected by using the predictive strengths of A and B individually, even when considering their interdependence. This is the basis for using Boolean AND.

- There is often what might be termed an "indifference effect": Descriptors A and B individually have equal predictive power as part of a query, but if both are present the predictive power is no greater. This is the basis for Boolean OR.

- There is the question whether the presence of specifically named descriptors or of any and all descriptors not specified in the query should lower the value of predicted relevance, as is the case with Boolean NOT and many prediction formulas (such as the often used cosine).

4.7.3. Search as inference

As we have seen, sophisticated searching requires inference either to recognize that a piece of information could be used for an inference in solving the problem or, even better, to draw the inference and present the solution.

One can take the point of view that all search is based on inference. Even simple Boolean retrieval can be expressed as the inference

IF document indexed by A AND by B, THEN retrieve document.

However, one usually talks about searching through inference only when several pieces of evidence are combined. The question is how much inference is used, how many different kinds of evidence are used, how far from stored data to the problem solution the system takes its user, and how much is left for the user to do. As a borderline example, consider an "inclusive" bibliographic search (such as EXPLODE in MEDLINE) that retrieves documents indexed by the search descriptor or any of its narrower terms. This search combines indexing evidence with thesaurus evidence.

In the course of a complex inference chain, ancillary simple retrieval operations may occur. In real systems the speed of such simple retrieval operations is of the essence; increasing the speed by an order of magnitude may well make possible complex searches that are qualitatively different. For example, a system written in Prolog may need access to facts that must be selected from thousands or hundreds of thousands like structured facts that together form a relation. Finding the needed facts through a sequential search is fine for small demonstration systems but impractical for working systems. Hence the necessity of either incorporating powerful database facilities into Prolog or linking Prolog with a database management systems. This problem of speedy retrieval access is a problem in the intersection of information retrieval and computer science; it concerns data structures and hardware.

Usually it is not practical to carry out all inference chains possible in a system to see whether any of them lead to a result that would be helpful for solving the problem. There needs to be some restraining heuristic. The knowledge base of problems proposed at the beginning of this paper could perhaps be used as a set of templates for inference chains to try for a given type of problem.

The need for thinking about chains of inference is by no means limited to searching substantive data. In bibliographic retrieval either the system or the user must think about possible inference chains and for each link in the chain find documents that contain helpful information, and users indeed do so now.

4.7.4 Concluding comments on searching

Above we put forth the idea that indexing, especially automated indexing, can be viewed as anticipatory searching (request-oriented indexing). Therefore, any solutions to the problem of searching apply also to request-oriented indexing.

There is still a lot of work to do on how to reason with uncertainty and how to reach final retrieval decisions. (Remember that a piece of information should be retrieved only if its expected value exceeds its opportunity cost.)

Search may also require generalization. For example, the query statement might be "What makes a good salesperson?". If there is no direct answer in the database, the system could do a regression analysis of how characteristics of salespeople and their territories relate to sales volume. Many research problems are of this nature.

**Sequencing and formatting output.

Problem 4.8. Indexing: attaching predictive clues

In a general sense, any kind of knowledge acquisition, such as recording facts as relationships between entities, can be called indexing. Here we consider description and indexing specifically from the point of view of retrieval; we consider establishing relationships between the entity indexed and other entities that might be useful as retrieval clues, that serve as descriptors. The descriptors given to an entity serve as the basis for deciding whether to examine the entity itself. How to construct suitable descriptions and descriptors is one of the long-standing problems in information retrieval. Descriptions must be short enough to save work yet complete enough to provide a sound basis for selection.

Descriptors must enable retrieval in response to requests that are not yet known. One answer is to learn as much as possible about problems and requests to be expected and employ this knowledge in problem- or request-oriented description and indexing. In this view, descriptors are anticipated requests (or components of requests) and indexing is anticipatory retrieval. This anticipatory retrieval can be done automatically, using any available clues (such as words in the title or abstract of a document) as independent variables that predict the relevance of the document for a descriptor. This is the problem discussed in Section 4.7.1 Search as prediction, except that now one can arrange for a training set of documents from which the system can learn about the association between given retrieval clues and descriptors to be assigned, and then use these associations to index new documents.

Indexing is a measuring operation. Each element of the index language is a variable, and the strength of association between a descriptor and the entity indexed is the measured value. Often this measured value is restricted to 0 or 1, but in indexing with weights more values are allowed. The reliability of indexing as a measurement operation is known as consistency of indexing. The validity of indexing, as judged by the usefulness of a descriptor assignment to an entity for the retrieval of that entity over all requests, is known as correctness of indexing.

Descriptors vary in their predictive power with respect to relevance for a given request. Title words or text words tend to have less predictive power than descriptors assigned by a knowledgeable indexer in the request-oriented mode, provided the selection criteria used in the request at hand were anticipated in the development of the index language.

The problem of how to organize a list of descriptors was mentioned above under knowledge representation. The problem of how to apply these descriptors in indexing is of great significance for retrieval performance. What does a human indexer need to know by the way of background and what instruction does he need to recognize the descriptors needed? What is the influence of index language display on the quality of indexing?

Indexing can be done by humans or by computers. Both processes pose many problems for research.

Problem 5. The human-computer interface

Problem 5.1. Functions in the human-computer interface

An information system user needs assistance in developing a query formulation that will produce a helpful information package in accordance with the specifications discussed earlier. Providing such assistance is the major function of the user-system interface. The interface must assist the user with problem clarification and with expressing the query in terms of the system and making best use of available system features. The ideal information system interface supports two modes of interaction. In mode 1 the information system initiates a dialog without any initial user input (through displaying a menu or a fill-in-the-blank form). In mode 2 the system interprets an initial natural language query statement, transforms it into a first query formulation, and uses that query formulation as a starting point for a dialog. The format of the query formulation depends on the system. For example, most bibliographic retrieval systems require a query formulation using AND, OR, ADJACENT, etc; a relational database system may require a query formulation in SQL (Structured Query Language).

The system should assist in database selection; it should include an intelligent guide to databases.

The system should be able to initiate searches in several databases and combine the results in one report. For example, MicroCSIN can access several databases with various data about chemical substances, retrieve data about a given substance, and integrate the data found into one nicely formatted report. The user does not need to be concerned with what databases were accessed or where the individual pieces of data come from.

These interface functions require considerable knowledge. Knowledge concerning problem clarification and suggestions as to what types of information might be helpful can be found in the database of problem templates mentioned earlier. Knowledge about the structure of the data and, therefore, about the structure of queries is given in the conceptual schema. The knowledge about entity types and relationship types or facets given in the conceptual schema supports facet analysis of the search topic. Facet analysis is helpful in figuring out the appropriate logical operators. It also uncovers facets not yet considered so that the system can ask the user whether she wants to specify descriptors from these facets. For example, in a search for food products, specifically milk, the system could ask whether the user wants to specify fat content.

At many occasions during the interaction the user is presented with options from which to choose. Often choices are arranged in a hierarchy; at a high level the user makes a broad choice among groups of options; at a low level the user selects a specific option from a group. The structure of such a hierarchy, the sequence of the options presented together, and the terms used to identify the options all influence user success. The problems of design are exactly the same as in the development and display of an index language or other structured list of entity values as discussed above. The study of index languages/ classifications has accumulated a considerable body of thought and experience in this area; this knowledge should be applied to menu construction.

Elaborate
Computerized systems have the advantage that such displays can be tailored to the user - his or her cognitive style down to the choice of typeface and color - and to the problem at hand.

Problem 5.2. Formal design of the human-computer interface

There is a need for a systematic layout of the parameters that define a design space in which menu, script/form, and command language in their pure form are just three of infinitely many points. Such systematic parametrization could guide thinking and empirical research on what combination of parameter values is best in a given situation. The same is true for the identification of other variables. To the extent possible the user should have control over the form of interaction.

Problem 6. Designing integrated workbench systems

Users - managers, scientists, scholars - can be assisted in many ways in their work through support in information retrieval and processing. A collection of computer tools that provide such assistance is often called a "workbench". An integrated workbench system provides for a division of labor in which human and computer each do what they can do best. The design of such a system requires a detailed analysis of the human's work to pinpoint tasks that could be performed or supported by computer. The software tools needed must then be assembled or constructed and integrated into a total workbench system in which they can be accessed through a common interface and in which they share information and thus are greatly enhanced in their usefulness. Software tools needed by most users are shown in Figure 10.

Document (text and graphics) processor with outlining capability

Database management system / expert system shell for all types of data (numeric data, formalized propositions, bibliographic data, text and graphics from the document processor).

Work management system (appointments, tasks)

Communication software that uses the document processor for the preparation of messages and that allows incoming messages to be processed as documents.

Program facilitating access to multiple outside databases.

Thesaurus used to check spelling, to assist in writing, and for internal and external retrieval.

Figure 10. **Software tools needed by most users.**

For specific applications, other tools are added.

Examples

Manager - decision support system

Chemist - needs support for the functions shown in Figure 11.

Automated measurement and data collection.

General data analysis and mathematical package.

Determination of the chemical structure of a substance based on its properties (for example, a mass spectrogram).

Determination of properties of a substance with known structure.

Modeling of molecules, crystals, solids, or multi-component systems.

Planning syntheses and chemical processes.

Input and editing of structure diagrams (a function needed in many contexts and linked to retrieving starting structures and building blocks).

Figure 11. **Functions of a chemist's workbench.**

Programs to perform these functions individually are available now. The problem is to package them and tie them together with a common interface, much like a statistical package.

Literary scholar - functions shown in Figure 12

Word frequency analysis, including analysis of word patterns, optionally qualified by context.

Structural analysis: syntactic analysis of sentences and paragraphs; text structure analysis, discourse analysis, story understanding, theme identification.

Content analysis.

Creation and maintenance of monolingual or multilingual information-rich dictionaries.

Figure 12. **Some functions of a literary scholar's workbench.**

Problem 7. Designing user-enhanced information systems.

The data/information/knowledge base of an information system can be much enhanced by users adding information as they use the system. The system of information transfer through the literature is user-produced in that the readers of the literature are also the contributors. However, there is scant opportunity for the reader of a document to react to it and to share with other readers the benefits of her insight. To be sure, there are reviews and letters to the editor, but these are somewhat cumbersome vehicles, and they are usually not known to the reader as he peruses a document. A hypertext system to which any user could contribute new pieces of text, linking them into the existing corpus, or just add new

links would serve the purpose well. An embryonic example is IRCS Medical Science, an on-line full-text journal that allows user comments which are stored in a separate data field of the article record.

Similar mechanisms for easy updating by users should be put into place for other databases. Some examples are given in Figure 13. Such user input - encouraged by the ease with which it can be given - can increase the usefulness of databases tremendously at moderate cost.

There is a problem with such user updating, namely, quality control. The first line of defense is the use of detailed source indications, perhaps coupled with a biographical database giving information about each contributor, so that users can make their own judgments on data quality based on trust in the source. The system might give the user the option to restrict his search to data from sources he trusts. The second line of defense is editing of user input and either indicating a mark of approval for data judged of good quality by the editors or including only approved data (depriving users of the opportunity to judge for themselves).

Learning thesaurus relationships from user queries

Example 1

Query:

(data OR information OR knowledge) AND acquisition

Inference by the system:

data, information, and knowledge belong in the same concept neighborhood.

Example 2

Query 1: knowledge AND acquisition

User needs more, enters an additional query.

Query 2: knowledge AND collection

Inference by the system: acquisition and collection belong in the same concept neighborhood.

The system also asks the user whether the concept relationship can be made more specific, e.g. BT/NT. The system accumulates information on the same term relationships from many users.

Figure 13. **Example of a system that learns**

User comments on documents or specific items of information/data can be stored and made accessible to other users.

As users analyze problems and determine types of information that might be helpful in solving the problem, they could update the problem database. On a rudimentary level, the users would just enter a description of the problem, and the system would capture the query formulations the user employed in searching and store them with the problem description. A specific example is retrieval in databases of social science survey results. Such databases are often accessible only through free-text searching of the questions in the survey instruments. The scientist who needs data for research on a theoretical construct must think of all the words that might occur in survey questions that would elicit responses with a bearing on the theoretical construct. If the scientists enter a term for the theoretical constructs they are after, this theoretical term can then be linked with the survey question words they used in searching, creating a database of linkages for the benefit of future searchers.

The relevance judgments that users make after receiving search results can be used for dynamic augmentation of indexing: Users are encouraged to enter query statements; these query statements, the user's query formulations, and lists of documents or other entities found relevant are stored. The query statements serve as additional descriptors, improving retrieval access.

The information that users produce as they update their individual workbench thesauri could be used to update a central thesaurus.

As users find indexing errors, e.g., when they retrieve irrelevant material or chance on relevant material not retrieved in a search, they could enter (or suggest) appropriate corrections.

Figure 14. **Examples of user enhancements to information systems.**

Problem 8. System evaluation

Comes back to problem 1. Importance of determining the impact of the information given on user achievement. Evaluation not tied to this not meaningful. Difficult.

Conclusion

** Implications. Requirements for human and technical resources.