Retrieval of Passages for Information Reduction *

Jody J. Daniels
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA

Phone: (413) 545-1985
Email: daniels@cs.umass.edu

July 19, 1996

Abstract

Information Retrieval (IR) typically retrieves entire documents in response to a user's information need. However, many times a user would prefer to examine smaller portions of a document. One example of this is when building a frame-based representation of a text. The user would like to read all and only those portions of the text that are about predefined important features.

This research addresses the problem of automatically locating text about these features, where the important features are those defined for use by a case-based reasoning (CBR) system in the form of slots and fillers.

We propose to use a small set of "annotations", textual segments, that we saved when creating our original case-base to generate queries and retrieve relevant passages. Annotations are associated with the slot about which they provide information. Using a case-base of annotations for each slot we generate and pose a query to an IR system that is aimed at the retrieval of passages within a relevant document. By locating passages for display to the user, we winnow a text down to sets of several sentences, greatly reducing the time and effort expended searching through each text for important features.

1 Introduction

There currently exists a bottleneck in extracting information from pre-existing texts to generate a symbolic representation of the text that can be used by a case-based reasoner. Symbolic case representations are used in legal and medical domains, among others. Finding similar cases in the legal domain is crucial because of the importance precedents play when arguing a case. Further, by examining the features of previous cases and sentences imposed, a legal reasoner (e.g., judge, advocate) can decide how to handle a current problem. In the medical domain, remembering or

*This research was supported by NSF Grant no. EEC-9209623, State/Industry/University Cooperative Research on Intelligent Information Retrieval, Digital Equipment Corporation and the National Center for Automated Information Research.

finding cases similar to the current patient may be key to making a correct diagnosis. Previous cases may provide insight as to how an illness should be treated and which treatments may prove to be the most effective.

Our goal is to save a user from reading an entire text in order to locate those areas of text containing information relevant to filling in a frame-based representation of the text. If a user must read through ten pages of text in order to fill one case-frame, then there will be a huge expenditure of time, particularly if the user must do this for fifty or more texts. Alternatively, we could save an automated information extraction system from processing an entire text by focusing the system down to those portions of the text most likely to contain the desired information. Therefore, we would like to automatically and expedientially locate predefined important features from within novel texts. The important features are those defined for use by a case-based reasoning (CBR) system in the form of slots and fills and are the frame-based representation of a text or case. This research aims to find those portions of a text that that can:

- be used as a slot fill,
- designate a value for a slot, or
- provide the information necessary to decide if a particular slot applies.

Our approach is to use a small selective set of textual annotations that we derive from those texts whose cases are already in the case base. From these annotations we construct a query to retrieve the most highly matching passages in a new text. This research explores some of the various means of composing queries from these annotations for use in probing new texts for passages that provide information about a desired slot. Our system will return the top ranked passages to the user for examination.

1.1 Example of the Problem

To show the distinctions in the above categories and illustrate the magnitude of the problem, we provide the following example. Suppose an individual is evaluating the possible courses of action relative to claiming bankruptcy and filing a repayment plan for her creditors. The user is concerned with whether her plan will be considered as being proposed in "good faith" and wishes to compare her situation with that of others who have already been to court over the same issue.

One of the good faith considerations is the debtor's amount of monthly income, which combined with information about expenses will determine the available surplus. Figure 1 provides a partial overview of a sample case opinion, the *Easley*[1] case. It contains eight of the opinion's nine pages. In this particular opinion, text describing the amount of monthly income can be found near the top of the second column on the second page. The actual sentence containing the pertinent information is "Debtor's amended budget commits \$30 to the plan from a weekly take-home pay of \$262.30." This sentence does not contain the actual slot fill, but it does provide enough information for a user to be able to deduce the value for the slot, $262.5 \times 52 \div 12 = 1136.63$. Nowhere in the document is the value \$1136 ever associated with the debtor's *monthly* income.

Other text segments within the opinion expound on the debtor's income: "The debtor's original budget reflected slightly higher income...", "...income was slightly reduced by lost overtime.", "...amendments were explained by changes in income and expenses including the loss of overtime.", and "...given the debtor's small income,...". These all provide insight to the reader that the value

[1] *In re Easley*, 72 B. R. 948, 950 (Bankr. M.D. Tenn. 1987)

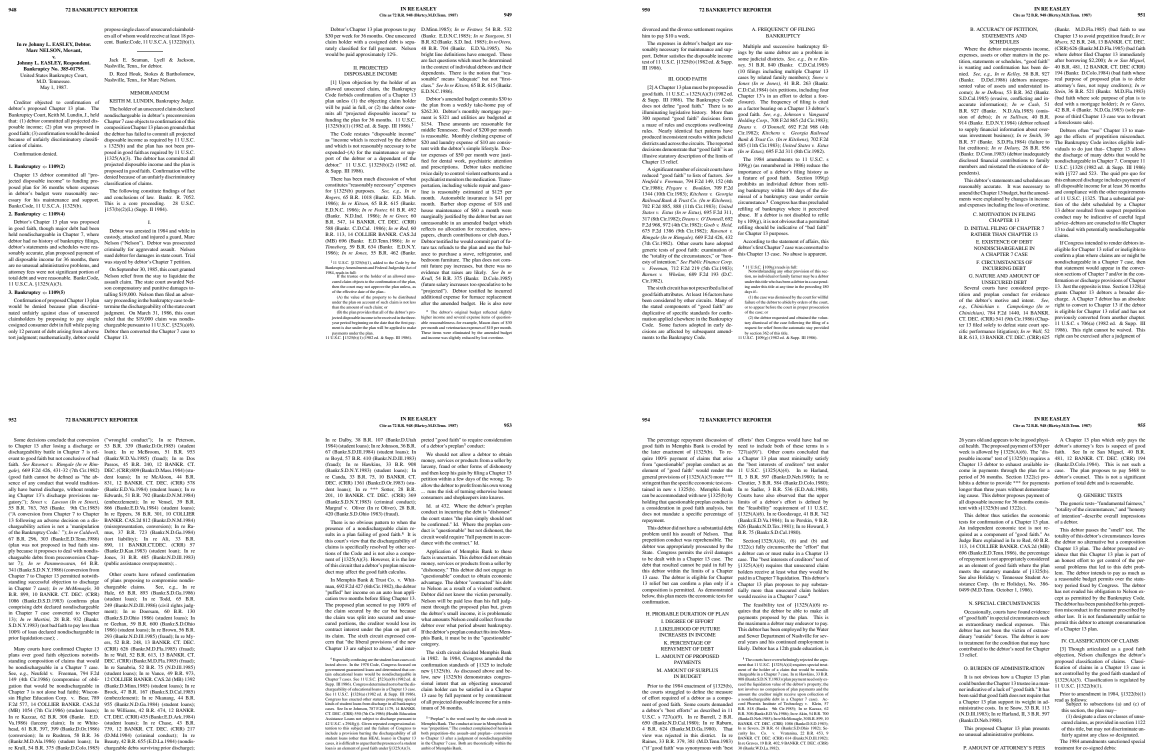

Figure 1: Replica of the first eight pages from the *Easley* opinion. The top row is the first four pages; the bottom row the next four pages.

of the monthly income slot has been previously discussed and is important to this case without ever stating any amounts.

These last four segments are spread throughout the opinion. The first two appear in footnote two on the second page, while the third is in the middle of the first column on the fourth page, and the last does not appear until the sixth page. Fortunately for the reader of this opinion, the segment containing enough information to provide a slot fill appears first. Unfortunately, the characterization of the income as being "small" does not appear until the sixth page and is buried in a discussion about the debtor's pre-plan conduct.

This highlights the magnitude of the problem facing the knowledge engineer who desires to use a complex case representation. If the user must read through the entire document, and must do so multiple times because of the numbers of slots that are in the case's representation, then any large-scale or even moderate reduction of the length of the text that must be read provides a tremendous savings.

1.2 Current Case-Based Reasoning Methods

There tends to be a large disparity between the size of the case bases and the size of the cases Among current case-based reasoning systems. Those systems with large-sized cases have small case bases and those with smaller-sized cases *may* have a larger case base. There are few CBR systems with large case bases and even fewer yet with both large-sized cases and a large case base.

One of the reasons for this is that the level of representation for the CBR case tends to be deep.

In general, CBR systems tend to focus on the automatic evaluation, selection, and adaptation of prior cases to solve a current problem or to justify and explain an interpretation of a case. Cases are hand-coded into structures that support this process.

Sadly, most CBR systems require the manual input of cases; they are not automatically generated from text or other sources (e.g., episodes in Kolodner's CYRUS [Kol84] and Bain's JUDGE [Bai86] used a conceptual dependency representation for their textual input.) A human makes the decisions about how to structure a case and what indexing structures to use. A case's input form is extracted from text or created from events, and information is put into either an intermediate or final form. A case's actual indices may be automatically generated (e.g., CYRUS dynamically decides what features to use as indices and may reorganize memory if better features are later discerned. Ashley's HYPO [RA87, Ash90] automatically generates an intermediate case representation, an interpretation frame, and the indices, called "dimensions", for each case.)

Because of the knowledge representation expense, there are few CBR systems with large case bases, that is, a thousand or more cases. In fact, there are few systems employing even several hundred cases. A rather interesting byproduct of this manual input characteristic is that those systems that do contain a large number of cases, tend to have very small or simplistic cases or case structures (e.g., MBRtalk used letters for the word pronunciation problem [SW86] while Anapron used word segments [Gol91], and PRO used word segments and associated phonemes [Leh87a]. Veloso was able to derive cases directly from the problem sets in her planning domain [Vel92], and Lehnert used 8-puzzle problems [Leh87b, BL88].)

CBR case structures typically represent more than one level of abstraction. Indices can be formed from these various levels so that a case can be retrieved and reasoned about from multiple perspectives and levels of abstraction. For this reason, one of the ways to merge CBR and IR is to try to automatically form text indices that have a higher level of abstraction. This would allow case-based reasoners to perform more complex reasoning about the relationships among these cases as represented by texts. It is because we would like to be able to more easily expand the size of our case-bases that we turn to automated techniques for representing texts and for extracting information from or locating information in them.

1.3 Current Document Retrieval Methods

We can currently use the surface-level features of texts to retrieve documents from a corpus of full-length texts. This is what is done in "Information Retrieval" (IR). We index each document based on the words it contains. To automatically convert a document into a set of word-based indices, we remove predefined "stop" words, that is, high frequency words that do not represent content and add little value for discrimination between documents (e.g., *and*, *but*, *the*, *a*). Then we "stem" the remaining words, that is, remove suffixes, to get at the root form of a word. What remains in a document constitutes the "terms" that are used as the (inverted) indices for it.

Although we may index the document based on its individual words, we can also store information about the location of each word. Using this location data we can base searches on pairs, triples, or larger sets of words that are within a stated proximity to each other. Using proximity information allows the retrieval engine to search for "phrases", words found within a close proximity to each other, while words found further apart may be treated as being in the same sentence, paragraph, or larger document element.

Combining statistics based on both the corpus as a whole and on the individual texts, we evaluate each document relative to a user's query. From this evaluation we decide which documents to retrieve. We frequently measure the quality of the retrieval via "precision" and "recall". Precision compares the number of relevant items retrieved against the total number of retrieved items. It measures accuracy. Recall compares the number of items that should have been retrieved by a query to the total number of items that actually were. It measures coverage.

Information retrieval obtains for us a set of documents statistically related to a query. The query represents an information need and the retrieved documents are believed to be relevant to the stated information need. However, all we have retrieved is a set of documents. What if the user desires to research specific facets of the text's content or if the document is lengthy and the user is only interested in a subportion(s) of the document? We have not shown the user where within the document the pertinent text resides. This is crucial if the user's information need is more specialized than a global examination of the document or if there are limitations on the available resources, such as time or money.

Current technologies allow us to display for the user those locations within the document where there is a match between the user's query and the various terms or phrases in the document. We can also highlight the passage that contains the best matches for the query. Problematically, if we employ the use of a thesaurus or other query expansion techniques, it becomes more difficult to explain to the user how each document is relevant to the query and to their stated information need.

Another problem arises when the user is interested in multiple facets of a document, that is, the user desires to find documents that are generally similar (i.e., they are about the same broad subject), yet additionally share multiple specific features in common (i.e., they discuss particular aspects of the subject). Current IR systems do not signify which text segments relate to which feature.

1.4 Current Information Extraction Methods

In some domains it is possible to use information extraction (IE) technologies to convert textual items found in a relevant text into a frame-based representation. Many examples of current message understanding systems and the domains they run in can be found in [MUC92, MUC93]. However, these are specialized domains and the techniques employed rely on the ability to find particular syntactic patterns associated with "trigger" or key words, usually verbs. The association of noun phrases with key verbs found in recognizable syntactic patterns allows for the extraction of pertinent information. Unfortunately, these techniques currently rely on significant numbers of manually annotated texts (e.g., on the order of 1000 or more texts[2]), to provide training for the extraction system.

The training documents for generating the linguistic patterns are domain specific and are therefore not generalizable to other extraction domains. Another important feature of these domains is that the information extracted for a slot is directly found within the text, that is, the actual verbiage found in the text is used as the slot fill. This is generally not true for most CBR case representations, and in particular for our domains. Frequently a human must draw some inference from the text to garner the slot fill. The definition for inference that we will use throughout this document is that a human reader must make an inference on the text in order to provide the slot filler.

[2] Although this may be changing. This will be discussed more in Section 3.4.

The cost of annotating such large numbers of texts can be prohibitively exorbitant in most domains, particularly if the information need is not recurrent, but sporadic. Similarly, the creation of large numbers of extensive frame-based case representations is a daunting task and prohibitively expensive in domains requiring extensive expertise.

In summary, existing information extraction techniques:

- require a large, annotated, training corpus,
- are domain specific, and
- directly extract their slot fills, verbatim, from the text.

1.5 What are Slots and Fills?

Given that information retrieval systems locate entire documents and information extraction systems are expensive to train, this research aims at being able to automatically *locate* the relevant portions of a text and associate them with a slot in a frame, without the use of a large, annotated, training corpus. The first issue this raises is whether the slot fills for our cases can be found verbatim within the case texts: *Are the slot values embedded in the text and, if so, can we automatically locate the relevant portions of text containing the values?*

For any particular frame-based representation of a domain, there may be various types of slots and types of fills that may be the slot's value. By examining the representations found in two legal domains, we have identified several different types of slots and fills.

The slots broke down into seven general type classes: boolean, category, numeric, range, set, date, and proper name. We give the definition of each slot type along with example slots and representative fills where necessary below. (This is not intended to be an exclusive list of all the possible types of slots in a case-frame, merely an enumeration of those encountered in these two case representations.)

Types of slots:

- Boolean:
  - Fills: A yes or no value.
  - Examples: special-circumstances-occurred, substantiality.
- Category:
  - Fills: One and only one from a given set.
  - Examples: decision-for (plaintiff, defendant), employment-history (poor, neutral, good), earnings-potential ((small poor) (medium neutral) (large good)).
- Numeric:
  - Fills: Single numeric value, either integer or real.
  - Examples: monthly-income, percent-surplus-income, amount-unsecured-claims.
- Range:
  - Fills: Numeric values with an upper and lower bound.
  - Examples: hours-per-week-in-home-office, room-temperature.
- Set:
  - Fills: One or more values from a given class of items.
  - Examples: furniture-in-home-office (desk, chair, telephone, etc.), debt-type (educational, taxes, judgment-debt, fraud, other)
- Date:

  - Fills: A calendar date
  - Examples: plan-filing-date, loan-due-date

- Proper Name:
  - Fills: Name of an individual or company
  - Examples: judge, plaintiff, defendant

- Free Text:
  - Fills: Free text
  - Examples: case-summary, case-citation

- Formatted Text:
  - Fills: Stylized text or symbols
  - Examples: case-citation

Because there is such a diversity of slot types, there is an assortment of available means to locate the slot fills. There is no direct correlation between slot types and a means of locating a fill. The various means may work well for slots of differing types. Below are some of the means of finding fills. (Again, this is not meant to be an exclusive list, merely representative of those techniques currently in practice.)

Ways to find fills:

1. Locating a word from a set that we might be able to enumerate – (ruling: affirmed, overturned), (furniture in home office: chair, desk, telephone).
2. Keywords/phrases that closely link a slot to its fill – These might be similar to the "triggers" for "concept nodes" [Ril93, RS95a]. When you see the word/phrase, you expect the next or preceding text to be the fill. (occupation: employed by, works for; ruling: judgment for, judgment against; monthly amount: proposes(ed) to pay, excess available).
3. Locating words/phrases that you might expect to be found in proximity to the fill – (plan duration: week(s), month(s), year(s): payments made: paid back, regular payments.)
4. Inferencing based on a segment of text – The reader or system must know some background information in order to make the association between the slot and the fill. (monthly income: given a weekly salary) (decision for: overturned – ruling value and knowledge about the level of this case (and previous decisions) is needed in order to know the outcome of this decision.)
5. Concept recognizers – These are methods for identifying closely related ideas or patterns such as: proper names, dates, monetary values, and foreign countries. (monthly surplus: monetary value; loan-due-date: date) (See Section 3.6.)
6. Synonym expansion through use of a thesaurus, an association thesaurus, [JC94], or a cooccurrence thesaurus, [SP94].

We observe that for most slots, there will be multiple ways in which to describe the value for a slot fill. These different descriptors will vary both within and across texts. For example, when trying to find the value for the "surplus" income available for payments of debts, you may directly locate the phrase "surplus of", which you would expect to be followed by the amount of the surplus. Alternatively, you might have to do some inferencing with such such expressions as: "did not have more than $50 per month for debt repayment" and "leaving him with approximately $100 per month to finance his plan". While these are quite descriptive of a surplus, it is likely that the average user will not be able to create a query capable of matching these expressions.

As a second example, consider the slot describing the number of "hours spent in home office". You might find "per week" and "spent" in proximity to an actual value, however, the expressions "on weekends" and "in the evenings" do not directly derive a value and, again, require inferencing on the part of the user. Thus, the usual extraction strategy of locating import syntactic or linguistic patterns in conjunction with key terms will not suffice here.

1.6 Sample Search Problem

Here is a hypothetical example scenario in which a user is searching through a text trying to locate the value of the plaintiff's monthly income in a bankruptcy case. The user would probably first request a string search trying to match "monthly income". Failing this, the user might proceed on searching for instances of "salary", "earnings", or just "income" alone. If this was unfruitful, the next attempt might be to look for "per month", "per week", or "per year". Other synonymous terms for monthly income might come to mind leading the user to try expressions such as "minimum wage", "stipend", "grant", or even "unemployment" or "unemployed". Finally, the user might just give up and either assume that there is no description of the plaintiff's monthly salary, or resort to reading the entire case opinion.

This scenario points out several problems with searching through natural language:

1. String search may not yield a result due to its requirement for an exact match. The user would have to understand stemming and when it would be appropriate to do so. For example, both "earnings" and "earned" might be appropriate for locating a value for the "annual income" slot. In which case, stemming to "earn" would work for both searches.
2. Slots that have a time factor, such as "monthly", may be expressed in multiple ways. The user must recognize that "per week", "per year", "every two weeks", and "annually", among others, may serve as valid locators for the value of the "monthly income" slot.
3. All the synonyms for a particular expression may not come to mind. An additional phrase describing "monthly income" is "take-home pay". In one text it was "take home pay", further complicating attempts at string matching.
4. Typographical errors will complicate matching. Misspellings will cause matches not to occur.

Information retrieval search engines are able to ameliorate the string search problem by using word-based rather than character-based approaches. They also have available such techniques as stemming and automatic query expansion via a thesaurus or association thesaurus. Nevertheless, the typical IR system is not designed to allow the user to conduct interactive searches over single documents; there is usually a tight loop between posing a query over an entire collection and examining the results of that query. This becomes problematic when the document of interest is not retrieved by a particular query or is much lower in the ranking and must be relocated after each query.

The above scenario points to the basic problem that language is ambiguous. This is even true even when the relevant vocabulary is small.

1.7 Summary

We examined the case representations with their slots and values in two domains, and found that, regrettably, many of the slot fills differ in their representation between the text and the case-frame. Current information extraction technologies can not directly extract the relevant information from within the text. IE methodologies retrieve values directly from a text for use as a slot value.

Additionally, IE techniques generally draw on a large set of training texts in order to learn where to find slot fills. These training texts are domain specific as is the resulting knowledge about what to extract from new novel texts. There is the further limitation that our domains have small case-bases (one consists of 55 texts, the other only 25) for use in training. It is infeasible to create training data of a larger magnitude because of the expense associated with generating each new case representation. This precludes our ability to use prevalent information extraction methods.

Furthermore, our CBR system requires a more knowledge-based level of representation (as opposed the text itself) than that currently achievable by IR techniques. Hence, we must turn to some other means of generating our case representations. This lead us to the possibility of automatically locating the text that the human would have examined to determine the slot's value. *Can we associate text pieces with their inferred representation?*

We know that a link exists (although possibly weak) between certain text pieces and a slot fill because a human has made this link. For every slot that has a value, we know that a human was able to transform the information contained in the text into the frame-based representation. Therefore, we know that there must have been some level of association between text pieces and a slot. This gives rise to the issue: *can we automatically locate those portions of a text affiliated with a particular slot?* If so, this will greatly reduce the time spent by a human searching through the text looking for information about each slot.

Since we already have a set of texts and their frame-based representation, we can go back and affiliate portions of each text with their associated slots and resulting fillers. When we do this, we end up with a *case-base of annotations* for each slot or feature. We propose to use these annotations to generate queries and to pose the queries against novel texts. We will assume that the novel texts are relevant to the problem at hand. We decompose each new text into a series of passages with the hope of retrieving the best passage(s) relative to a given slot. The best passage(s) will be presented to a user, who will then fill in the slot with the appropriate information for this text's representation. If the user marks the passage from which the new fill is derived, we can add it to our case-base of annotations for use in future queries.

In the next section we describe the system and Section 3 explains in more detail the technologies we will employ. Section 4 gives a review of related work and Section 5 covers the issues inherent in this research.

2 System Description

This section provides a broad overview of the system followed by a more detailed explanation of the processing of documents. It finishes with an example problem case, starting with the input of the problem, showing retrieval of relevant new documents, and finishing with an example of the passages retrieved in response to a particular slot query.

2.1 System Overview

We assume that we have available a small set of texts, their frame-based representation, and a set of annotations associating text with slots. Using a CBR system in conjunction with an IR engine, we use these cases and their texts to retrieve an additional set of texts believed to be relevant to a current problem situation (one possible technique is described in [DR95] and [RD95].) The next step is to see how closely the situations in the retrieved documents match our current problem. To be able to do this automatically, we must convert our newly retrieved texts into a representation with which the case-based reasoner can work. Therefore, we now focus on how we propose to make this conversion.

In general terms, we gather all the annotations from which our original case knowledge base (CKB) was derived. We use these as the basis of a new query, with which we retrieve passages from novel, but relevant, texts. The top ranked passages are presented to and reviewed by a user who will extract or infer slot values for each novel text. In this way, we have added new cases to our CKB at an expense to the user that is lower than the cost of reading the entire text.

2.2 Detailed System Description

To provide a more detailed explanation of how the proposed system will operate, we must go back to the start of the case-generation process. The first step to creating a case-frame representation of the new texts is to associate portions of the original case texts with the slots about which they provide information. Based on domain knowledge and the task at hand, a domain expert creates a case-frame representation to assist with the annotation and location of information from case texts. This representation will be exploited by the case-based reasoner to perform the desired type of reasoning. Utilizing the case-frame representation and a "text marker", the domain expert marks and extracts information from a small number of texts (on the order of 10-50). All of this data, the slots and their values, the associated annotations from within the original text, and the start location of each annotation, is stored in the CKB.

A new case is presented, called the "current fact situation" (CFS), in the form of a case-frame to the case-based reasoning component. The CBR component determines, using one of its similarity metrics, those cases most important to the CFS. The original case texts associated with these "best" cases are then passed over to the information retrieval system. Using a modified form of relevance feedback, the IR system generates and submits a query to retrieve additional related documents from a larger document collection. (See Figure 2.)

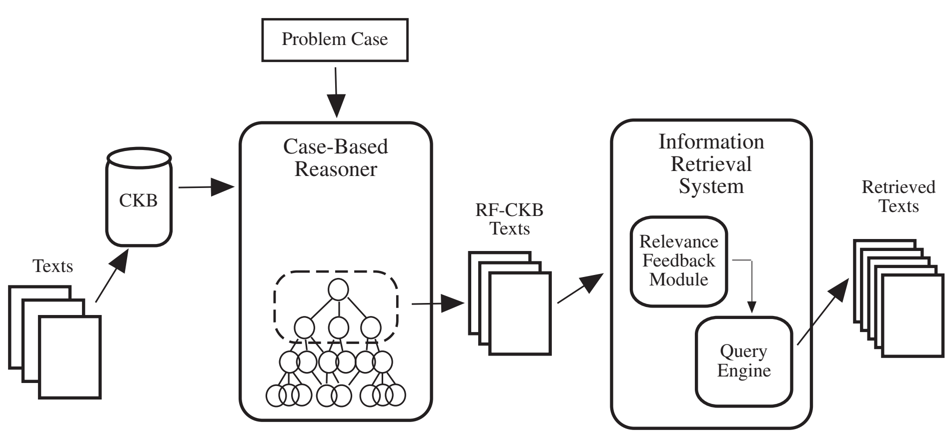

Figure 2: Retrieval process for novel documents.

The new retrieved texts are, unfortunately, not in a format that the reasoner can use. Consequently, we must find a means to convert them. If the system is capable of automatically extracting fills for any of the representation's slots, it will do so at this point. For most of the slots though, automatic extraction will not be possible. We now take advantage of the annotations that the domain expert has provided when creating the original case-frame representations. One slot at a time, the system passes the respective annotations over to the IR system. (See Figure 3.)

Instead of treating each new document as a single item, we propose to divide it into smaller elements. The IR system divides a relevant document into passages and treats each as a separate entity. There are various ways to segment a text: using user-defined boundaries such as sentences, paragraphs, or sections; semantically or thematically, or by indiscriminately dividing the text into windows of a particular (or varying) size. (In Section 3.5 we elaborate on the various options available and in Section 5.3 discuss which method we will use.)

The IR system next generates a query from the case-base of annotations to locate the relevant passages rather than relevant documents. The passage query is posed against the document's passages and the top ones presented to the user. The user will either extract or infer a slot value from the presented material. In this way the user adds new cases to the CKB. The user may also add passages or subportions of them as additional annotations to be used by later passage queries. Once the new texts have been converted and added to the CKB, the CBR component may reason about them relative to the CFS and the original task.

2.3 Example Problem Case

To better illustrate the approach of our system we run through the following scenario based on a real personal bankruptcy case, the *Rasmussen*[3] case. Suppose a client, Mr. Rasmussen, approaches a lawyer about his attempt to file a personal bankruptcy plan. The Bankruptcy Court has denied approval of the plan because it failed to meet the "good faith" requirement. However, Mr. Rasmussen believes that he does satisfy the requirement and wants to appeal the court's decision. He tells the lawyer various facts concerning his problem case. The lawyer inputs these facts to the CBR-IR system.

Having practiced in this area of law, the lawyer has knowledge of a set of past bankruptcy good faith cases and their outcomes. Assume she has represented these in her own in-house case-base, which is used by the CBR portion of the system. The system begins by performing an analysis of her client's problem case, with respect to this in-house case-base. In this instance, the CBR module uses a HYPO-style reasoner that uses a claim lattice to determine similarity [RA87, Ash90]. (See Section 3.1 for more details on CBR systems.)

Next, the CBR module selects a small set of special texts on which to employ relevance feedback to generate a query. We call this set of texts the "RF-CKB". One good choice of texts would be those associated with the top layer or top two layers of the claim lattice. The cases in the top layer are those most highly similar to the problem case, based on the reasoner's particular similarity metric. They are referred to as the "most on-point" cases (mopc's).

With *Rasmussen* as the problem case and a small corpus containing 43 hand-coded bankruptcy cases as the CKB, Figure 4 shows the top portion of the resulting claim lattice. All of the dimensions of the *Chura* case overlap with the problem case, hence, *Chura* is the only most on-point case. This is depicted by the single node coming from the root. There are three nodes in the second layer of the lattice and these nodes encompass four additional cases.

The case opinions associated with the mopc's or the top two or three layers of the claim lattice are then used as the set of marked, relevant documents on which the IR engine will perform relevance

[3] *In re Rasmussen*, 888 F.2d 703

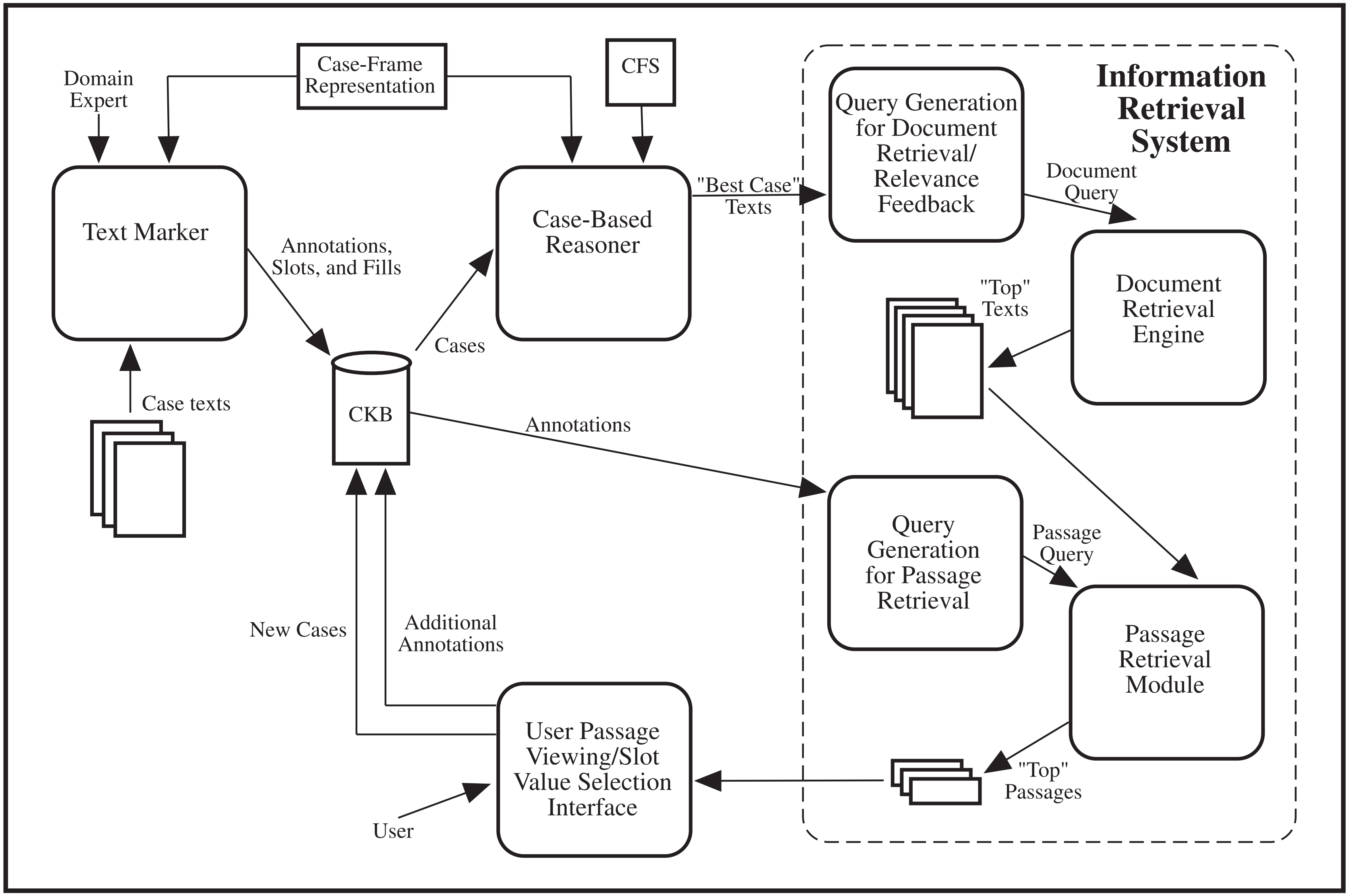

Figure 3: CBR-IR System Overview

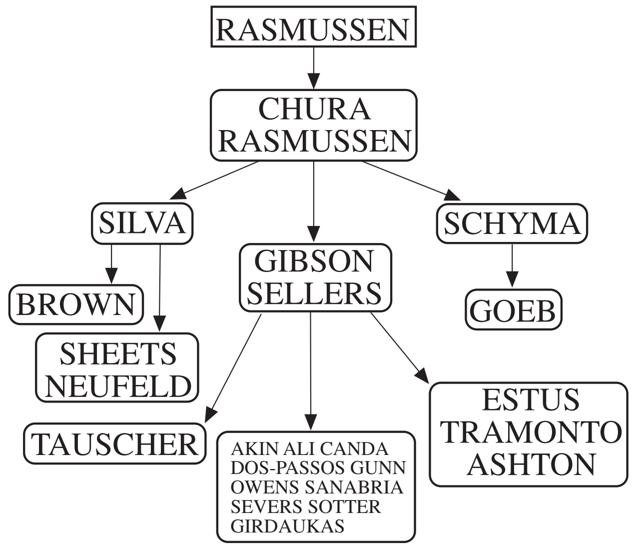

Figure 4: Partial claim lattice for the *Rasmussen* case.

feedback in a modified fashion to generate a query. To enable this, the CBR module passes the indices for these documents over to the relevance feedback module within the INQUERY system.

The relevance feedback module then selects and weights the top terms or pairs of terms from within these CBR-provided texts and forms a query. INQUERY acts on the query in the usual way to return a set of relevant documents from a larger collection, say the WestLaw®Federal Taxation Case Law collection. The system returns to the lawyer this set of highly relevant documents, some of which she already knows about since they were in her own personal CKB, to use in her research on Mr. Rasmussen's legal problem. Below is a sample query for this case, using the top 15 terms and the top three layers of the claim lattice:

```
#WSUM(1.000000  0.772051 d.c.cir. 1.524504 mislead 1.888424 likelihood 0.957335
 marlow 1.330305 sincer 1.523974 liber 1.345248 frequenc 1.426169 accurac
 1.744932 minim 1.136436 eighth 1.117896 inaccurac 0.891818 gen 1.441973 colleg
 1.247028 inordin 1.248914 preferent)
```

The lawyer now has a larger set of relevant documents for her research on Mr. Rasmussen's problem. The system has located new cases, previously unknown to the CBR module.

Now, we take these top documents and send them, one at a time, through the next portion of the system. Since the *Easley* case, partially depicted in Figure 1, shows up as a highly ranked document by many of the possible RF-generated queries, we will use it to illustrate the next step of inputting a text to the passage retrieval portion of the system.

For each slot about which we are concerned, we generate a query over the passages in a novel text. Consider, once again, the monthly income slot. If we were to use the sample terms and phrases previously discussed in Section 1.6, and manually form a query, it might look like:

```
#PARSUMN20(#SUM (#phrase(monthly income) salary earnings #phrase(per month)
#phrase(per week) #phrase(per year) #phrase(minimum wage) stipend grant
unemployment))
```

The above query searches for the given words and phrases among passages of size 20. Using this query[4] on the *Easley* text, the listing of the top passages looks like:

Psg Strt	Belief
4010	0.408031
4020	0.408031
1170	0.406361
1180	0.406361
2430	0.403198
160	0.403198
2420	0.403198
80	0.403198
1230	0.403198
1250	0.403198

The first two passages can be found in the second full paragraph, second column, on page six. The text in the general area is shown below. Embedded within the text are word counts so that the retrieved passages are identifiable. Both passages contain the words "income", "minimum", and "months", which end up giving these passages the highest score.

> As discussed above and below, *3990* new s 1325(b) demonstrates congressional intent that an objecting unsecured *4000* claim holder can be satisfied in a Chapter 13 case *4010* by full payment or by commitment of all projected disposable *4020* **income** for a **minimum** of 36 **months**.
>
> The percentage repayment *4030* discussion of good faith in Memphis Bank is eroded by *4040* the later enactment of s 1325(b).

Unfortunately, these passages do not provide any insight into Mr. Easley's monthly income. The second pair of passages is closer in that they address pay increases, yet no discussion of an actual salary is mentioned. These passages are found in the first full paragraph, second column, page two. Their text matches "salary" from the query. Below is the text surrounding these passages:

> The plan does *1160* not commit future pay increases, but there was no evidence *1170* that raises are likely. See In re Krull, 54 B.R. *1180* 375 (Bankr. D.Colo.1985) (future **salary** increases too speculative to *1190* be "projected"). Debtor testified he incurred additional expense for furnace *1200* replacement after the amended budget.

The next set of passages all contain both "income" and "months". The last passage in the list includes: "income was slightly reduced by lost overtime", which is the closest piece of information about monthly income within the entire set.[5]

Instead of manually forming a query we propose to use our case-base of annotations to generate one. Our query may treat these annotations either as separate entities or may merge all of the terms into one long natural language query. For the monthly income slot, our case-base of annotations includes those discussed below in Section 5.1.

[4] All passage queries were run under version 3.0 of INQUERY and the collection was both stopped and stemmed.

[5] Ties are not resolved. If there is a tie that would cause the listing to go beyond the top ten passages, those that make it into the top ten are arbitrarily assigned. This will be changed in the future so that all passages that are equivalent within the top ten places are returned.

For this next query, we treat each annotation in our case-base as a separate entity by surrounding each with an *#or* operator. Again using a passage size of 20, the retrieval results are given below.

Psg Strt	Belief
990	0.404301
1180	0.404116
1170	0.404116
150	0.403834
4820	0.403407
4810	0.403407
80	0.403407
600	0.403407
4010	0.403406
2420	0.403406

The returned set of passages are much improved over those from the previous query. The top passage is, in fact, the best one in this opinion. The pertinent text, including word counts and matching terms in boldface, is:

> "**Debtor's** amended **budget** commits $30 *990* to the **plan** from a **weekly take-home pay** of $262.30."

The second and third ranked passages, 1180 and 1170, are discussed above, and the fourth includes both "income" and "months", as well as "statement". It additionally includes "disposable", which is found in one of the annotations. Below is the relevant text surround the fourth passage with matching words in boldface.

> Debtor's Chapter 13 plan was proposed in good faith, though major debt had been held *140* nondischargeable in Chapter 7, where debtor had no history of bankruptcy filings, debtor's **statements** and schedules were reasonably accurate,**plan** *160* proposed **payment** of all **disposable income** for 36 **months,** there are no unusual administrative problems, and attorney fees were not *180* significant portion of total debt and were reasonable.

By using the case-base of annotations we were able to generate a query with which to locate a good passage about our slot. Once the user has read the first passage, they can easily calculate the value for the monthly-income slot. If the user so chooses, they may add text from this retrieval to the case-base for use in future queries. This may prove particularly fruitful if an additional means of expressing monthly income becomes apparent.

2.4 Summary

This example shows how we can leverage the strengths of two different systems for mutual benefit. The CBR system with its highly defined sense of relevance passes salient cases to the IR system. The IR system uses its knowledge of word distributions and the CBR provided relevant documents to create suitable queries. We hypothesize that we can use this same style of interaction to retrieve germane passages over a variety of types of slots and fillers.

3 Background

This research touches on the work done in many different methodologies. We start with a case-based reasoner and a frame-based case representation, transition over to an information retrieval system with full-length texts, and end up with short text segments from novel documents being affiliated with case-frame slots. Along the way we use techniques from natural language processing for the task of information extraction and we take advantage of previous work on passage retrieval and concept recognition.

3.1 Case-Based Reasoning

CBR systems focus on the automatic evaluation, selection, and adaptation of prior cases. There are two basic types of CBR systems; those that solve problems and those that are precedent-based. Problem-solving CBR systems retrieve solutions to previous problems or subproblems and merge or adapt these solutions to yield a new solution, preferably optimal. Precedent-based or interpretive CBR systems will explain or justify the optimum answers based on comparisons with prior outcomes.

As previously mentioned, most CBR systems require the manual input of cases. (Although there are systems with automatically generated case-bases: [Vel92, Gol91].) The case and indexing structures are decided upon by a human. Each case's input form is extracted from text or created from events, and data is placed into the appropriate representation. The actual indices for a case may be automatically generated based on the input data. Known cases are stored in a case-knowledge base (ckb).

To reason about a new case or problem, a current fact situation (cfs) must be input to the system. It is generally in the same format as the cases in the ckb. Once input, the cfs is compared to the ckb using any one of a variety of similarity metrics. (See Figure 5.) The comparison yields a set of documents or cases deemed to have some level of similarity to the cfs. In some CBR systems it is sufficient to find a "close enough" case and not search the entire case base for the best matches [Kot88]. Problem-solving CBR systems frequently use some form of nearest-neighbor metric, while precedent-based systems frequently use a "claim lattice". This means of measuring similarity is based on Ashley's HYPO system [RA87, Ash90].

A claim lattice is a partial ordering of the cases similar to the cfs. The ckb is sorted based on the intersection of each case's "dimensions" (or features) with those applicable in the cfs; cases with no shared dimensions are not considered since they are not deemed relevant [Ash90, RA87]. Dimensions address important aspects of cases and are used both to index and compare cases.

In this sorting, *Case A* is considered more on-point than *Case B* if the set of applicable dimensions it shares with the cfs properly contains those shared by *B* and the cfs. Maximal cases in this ordering are called "most on-point cases". The result of sorting the cases is shown in the claim lattice. It contains only the subset of cases from the ckb that are considered relevant to the problem case at hand. (See Figure 4 for an example claim lattice.) Those cases on the top level of the lattice are the mopc's. The cfs is the root node.

The set of retrieved cases may or may not be weighted, providing discrimination within the set as to their utility in satisfying the current need. For many CBR applications, particularly in planning or designing, it is desirable to work with a single case or at least a small number

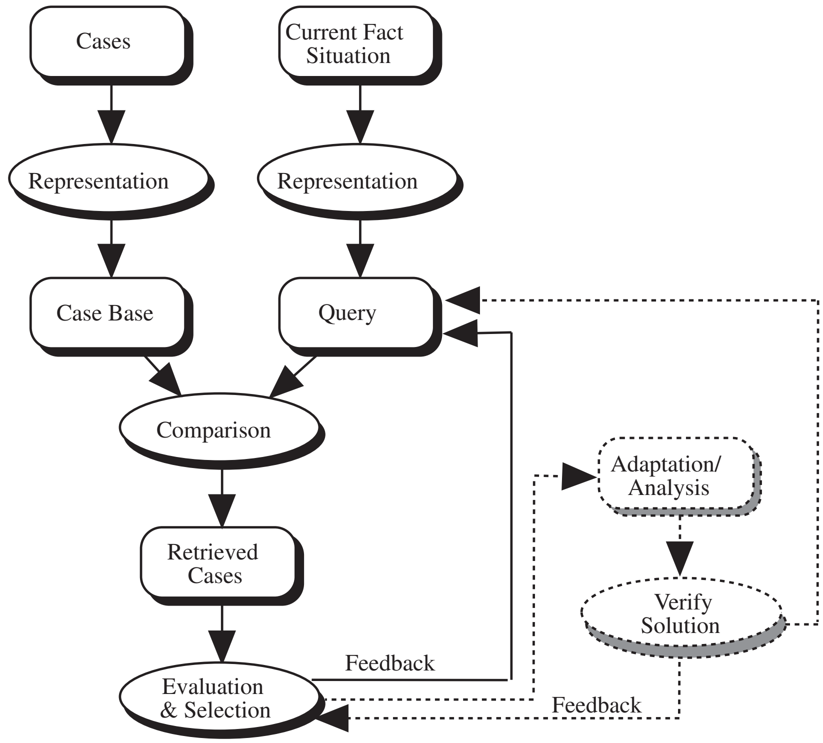

Figure 5: Overview of the CBR process. Optional elements are dashed and in grey.

of cases for adaptation (or possibly a case for each problem component.) Therefore, problem-solving CBR systems generally have some means of weighting the relevant cases. (Kolodner's PARADYME [Kol89] uses preference heuristics, Hammond's CHEF [Ham87] uses a discrimination net that hierarchically orders features in terms of their relative importance, and Stanfill and Waltz applied a variety of distance metrics to arrive at the "best case" in MBRtalk [SW86].)

Interpretive CBR systems do not necessarily assign numerical values to relevant cases, but find other means for generating at least a partial ordering among the retrieved cases. (HYPO and Rissland and Skalak's CABARET [RS91] generate a claim lattice and Goodman's Battle Planner [Goo89] uses "case prototypes" or conjuncts of indices to select the "most on-point" cases for analysis.)

Once a solution or evaluation is generated, the CBR system may wish to validate the solution. Additionally, the CBR system has the option of storing information about the problem-solving or interpretive episode, such as knowledge about failures, adaptations, and the final outcome.

To summarize, the strengths of CBR include the following:

- CBR is a developing methodology that accords a great deal of flexibility to individual systems, such as deciding on case structure, the indices and/or their structure, similarity metrics, adaptation or evaluation techniques, retrieval model, and the storage mechanisms to use.
- Methodology flexibility allows CBR systems to take advantage of domain knowledge and do a single job well (e.g., generate plans, develop interpretations.)
- Cases may have multiple levels of, or complex structures for, indexing. Having multifarious indices enables a system to view a case from differing levels of abstraction and thereby allows for a variety of means of reasoning.

- Because of the indexing methods used, case retrieval is done with a high level of relevancy.
- Problem-solving CBR systems can adapt a solution from a previous experience to solve a new problem, thereby saving the time of re-solving all or a portion of a current problem. Interpretive CBR systems can analyze old experiences to predict an outcome for a current situation or relate/distinguish the current situation to/from previous events.
- CBR systems can learn by storing solutions to, or analyses of, new situations.
- CBR systems can learn from failures by storing these events and the indices that predict them, thereby avoiding repeating failures.

Even though CBR is a strong paradigm for reasoning at multiple levels of abstraction, there are certain weaknesses found in many systems:

- Devising case representations, indexing structures, and extracting cases for use in some domains is too manually intensive.
- Many of the currently used storage and retrieval mechanisms will not scale-up for larger case bases.

Overcoming or reducing the first of these weaknesses, so that we may take advantage of the strengths of CBR in reasoning from multiple perspectives and levels of abstraction, is the focus of this research. Hence the integration with information retrieval which can rapidly index its "cases".

3.2 Information Retrieval

Over 25 years ago Dyke summarized the IR process and its goals:

> The ideal data retrieval system will take a user's question, interpret it into the language of the system, search all of the available information and data, and supply the optimum answers in minimum time at a reasonable cost. [Dyk69] page 6.

These are still the same goals as today: automatic indexing of documents (or other data items), with little or no manual input, and then retrieving those documents that match an information need stated in the form of a query as effectively and efficiently possible. The internal representation of a document may contain information about the document's words, their locations, their frequencies, whether they have been identified as being part of a concept, and information about adjacent words. As described in Section 1.3, when indexing each document, words may be stemmed and stop words may be removed.

The most common models for representing documents include the vector space model [Sal89], as implemented in SMART [Buc85], the probabilistic model, and inference networks as implemented in INQUERY [CCH92]. This research will be done using the inference net model as implemented in INQUERY, but there is no apparent reason why it could not be done within another framework.

INQUERY uses an inference network model [TC91], [TC92] specifically, a Bayesian probabilistic inference net, to represent texts and queries. It uses a directed acyclic graph with a information need at the base, document nodes at the leaves, and a layer of query nodes, query concept nodes, content representation nodes, and text representation nodes in between. (See Figure 6; copied from [TC91].) Nodes that represent complex query operators can be included between the query and query concept nodes. The INQUERY model allows for the combination of multiple sources of evidence (beliefs) to retrieve relevant documents.

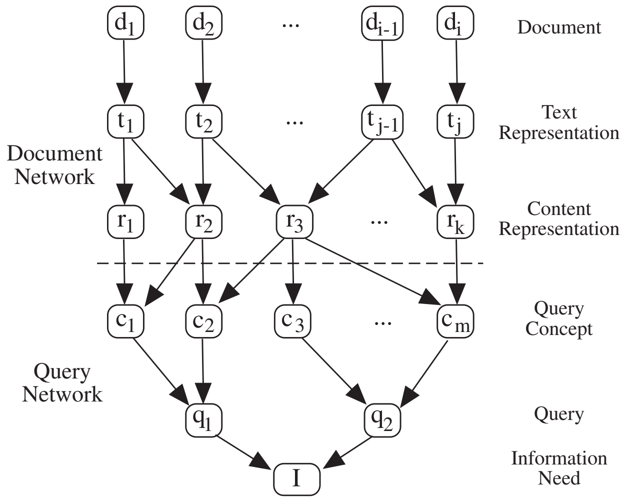

Figure 6: Sample inference network for document retrieval.

Query formulation for all types of representations consists of taking an information need and turning it into a form that can be matched against that of each document. The comparison yields a set of documents deemed to have some high level of similarity to the stated query. The retrieved document set is usually ranked, providing discrimination among the set as to their utility in satisfying the query. Frequently IR systems implement a mechanism for the user to judge the results of the query and to request a new query that incorporates these judgments. The system automatically generates a modified query in a process known as "relevance feedback" [SB90].

Because there has been much research and advancement in the IR process over the past 30 years, current IR systems are robust and exhibit the following strengths:

- IR is a well-worked out methodology, that accords flexibility to individual systems, such as deciding on a morphological rule set, stop word list, index set, query model, and the storage mechanisms to use.
- English language characteristics allow IR systems to entirely automate the traditional index structure creation process, thereby being applicable across domains.
- IR systems can access and retrieve data from very large collections or "case-bases".
- Accessing and retrieving from these large collections can be done very quickly.
- IR systems can automatically add new data items to their collections (modulo data entry.)
- IR systems can utilize learning in the form of relevance feedback to iteratively improve retrieval during a query session.

Even though there has been a great deal of research, Information Retrieval does have some weaknesses and there are still many challenging areas within the field:

- Only limited reasoning about the retrieved documents can be done. Similarity assessments may be made, but current indexing techniques do not enable their indices to be used for problem-solving or analysis of documents.
- Formulating good queries can be difficult. However, the recent addition of new query operators has helped ameliorate this problem.
- IR systems can only achieve high recall at the cost of lowered precision and *vice versa*. However, reasonable results are achievable at intermediate values.

- IR systems do not learn from their failures over time. For example, suppose a user poses a query and then provides relevance feedback. At a later time, that same initial query will produce the same initial results. (Although this is an active area of research.)

3.3 Merging CBR and IR

As previously mentioned, CBR systems have an information bottleneck problem and IR systems have the problem of not being able reason abstractly about their collections. Conversely, CBR systems have the ability to reason abstractly about their cases while IR systems can easily and rapidly transform documents into an internal representation. It seems logical that we should want to combine these two paradigms to tap the strengths of each in support of the other.

We have already been able to exploit the reasoning ability of a CBR system working in concert with the indexing and retrieval abilities of an IR system [DR95, RD95]. The CBR system identifies those cases most similar to a current fact situation. The documents associated with these most similar cases can then be presented to the IR engine. The IR engine treats these documents as though they were hand-marked by a user as being relevant and performs a modified version of relevance feedback to generate a query against a much larger corpus of texts. Our experiments showed that we were able to retrieve texts relevant to the current problem and were able to do so with a high rate of confidence.

There is another system the merges IR and CBR techniques, but it is used for the classification task. Prism is a system for classifying bank telexes for further distribution and routing [Goo91]. It uses a lexical pattern matcher to generate retrieval indices. These indices are used to select cases from a case base of over 9600 sample telexes. It then adapts the best matching cases to find classifications solutions for the new telex.

3.4 Information Extraction

The objective of an information extraction system is to locate and extract those pieces of information considered to be significant from within a text. The information is placed in a frame-based representation. The slots summarize information about an event and include such items as the actor, date, location, type of event, important object, outcomes, etc. Compiling the extracted data, one could easily create a summary the significant events or data items found in the text. Domains used for the Message Understanding Conferences (MUC) over the years include naval operations reports and terrorist newswires [MUC92, MUC93]. TIPSTER evaluations used business joint venture and micro-electronics stories [Pro93].

Frequently, information extraction systems are paired with a filtering system. The filtering system tries to separate the relevant from the non-relevant texts so that the extraction system only has to expend effort on apposite texts.

Some extraction systems look for important phrases by applying heuristics such as: look for proper nouns to be the names of individuals, companies, or countries, look for a word that is all capitals surrounded by parenthesis, (which would denote an acronym,) look for the use of italics or underlining to signify important concepts, and find multiply repeated compound noun phrases (with the belief that some of these will be domain specific.) [KB95] (These techniques are related to concept recognizers, which are discussed in Section 3.6.)

One system filters a text stream based on keywords, then uses the ODIE information extraction system to do extraction on the remaining texts [SHH95]. Therefore, only those texts that have met predefined characteristics are considered suitable for examination by the extraction system. (We do a similar thing in that we use our relevance feedback generated query to act as a filter on the collection and locate relevant texts.) Additionally, they further reduce the amount of semantic processing by only examining those sentences in which the keywords appear [Huf95].

AutoSlog-TS [RS95a] shows great promise in overcoming one of the limitations found with most natural language processing systems – that of needing large quantities of training data to learn concepts and construct dictionaries. Primarily this training data has consisted of a domain-specific manually-annotated corpora. These corpora represent a huge investment each time a new domain is to be explored. AutoSlog [Ril93], used for dictionary construction with several information extraction domains, also suffered from this limitation. This new approach, AutoSlog-TS, relies on linguistic rules and statistics collected over the new corpus to generate a dictionary without the need for an annotated corpus.

AutoSlog-TS has been tested for two NLP tasks, extraction and text classification. It performed comparably to its counterpart for the classification task. For extraction, AutoSlog-TS was able to produce concepts that would not otherwise have been generated because it examines statistics and patterns throughout the texts, not just the annotated portions. There remains a need for a human to review the generated concepts, but the need for an annotated corpus is potentially dissipating.

Even though we may be able to remove the need for a large training corpus, IE still relies on linguistic patterns or rules associated with keywords to locate slot fills and extractions that come directly out of the text. We have tried to use the natural language processing extraction system at the University of Massachusetts, Amherst on our domains because this system has done quite well over the MUC domains. Unfortunately, our domains come with little training data and the linguistic patterns were sufficiently different that they did not readily apply. Limited automatic extraction was obtained and with a lower than acceptable precision and recall rates.

An additional problem with IE systems is that there appears to be an implicit or even explicit [Huf95] assumption that the information being extracted is about an "event". These systems further assume that the constituent elements being found fill "roles" within an event. This assumption does not hold for the domains of this research; the sentences containing the desired information generally use non-descriptive passive or infinitive verb forms.

IE systems extract facts, but their underlying purpose is closely aligned with the summarization task. By default, they generate information that encompasses broad coverage of a topic. Some of their techniques are directly applicable for some of our slots. Unfortunately due the magnitude of training data generally required and the nature of the information captured we have not been able to transfer this technology to our domains.

3.5 Passage Retrieval

Previous work on retrieval of textual subelements is limited. Primarily the work falls into three areas: imagery analysis to locate document subelements, examining passages or other small document element (sentence or paragraph) to aid in document retrieval, and retrieval of document subelements. The second and third areas are more closely aligned to this work than the first.

The first line of research into breaking a document into smaller components primarily focuses on imagery analysis and gets down to the pixel level to aid in distinguishing document subelements.

Partitioning a document into smaller structure-based elements is reported in [RS95b, RS95c]. They use agents to search for "layout-based abstractions" such as tables, figures, and paragraphs, and "content-based abstractions" such as theorems, lemmas, and examples. Once located, smaller objects are further examined to see if they meet the information need. An example of their technique is to segment a newspaper, filter for relevant sections, segment the sections into paragraphs, then detect tables and graphs. The resultant table and graph objects are examined for specific information, such as stock data on a particular company.

This refinement of search toward progressively smaller levels of granularity is similar to what we propose, except that we do not rely on locating structurally delineated objects for retrieval, nor do we search at the pixel level.[6] (See [RS95b] for cites of other structure-based approaches that use image analysis techniques to locate document subelements.)

In the next category, various types of passages are examined to aid with document retrieval. Generally, the value of the best passage is combined with the similarity score of the document in some fashion.

This process may be motivated by Ro, who compared the results of document retrievals when queries were placed over full-text, paragraphs, abstracts, and a controlled vocabulary field [Ro88]. Not surprisingly, precision went, from best to worst: controlled vocabulary, abstracts, paragraphs, and full-text. Recall was almost the inverse, with the order being: full-text, paragraphs, controlled vocabulary, and abstracts.

We may not have an abstract or controlled vocabulary list associated with each document, so the next best object for improving precision with full-text retrievals may be the paragraph. By combining the score for the best matching paragraph with the scores of the individual terms, it may be possible to achieve a higher level of precision without sacrificing the recall of full-text retrieval alone.

Callan provides a comprehensive study of the relative merits of different types of passages when dealing with different types of collections and documents [Cal94]. This work uses the inference net model and in all cases, the belief value for the best passage within a document was added to the belief for the document as a whole. Comparison was done among discourse and window passages. Discourse passages were based on user-defined breaks such as sentence and paragraph breaks. Window passages looked at a fixed-size amount of text, say 100 or 200 words.

Because discourse passages may vary significantly in length, which will have an impact on term frequency statistics, he also examined fixed-size window passages. Fixed-size windows have the counter problem of possibly breaking related text across windows. This was alleviated by having passages overlap at intervals of half the window size. To ameliorate the size problem with discourse passages, bounded paragraphs, where small paragraphs were merged and large paragraphs were split, were also tested. His results indicated that fixed-size window passages did best over an assortment of collections that range in the size of their documents. Stanfill and Waltz discuss a commercial system, CMDRS, that uses the scores from fixed-size passages to retrieve documents and has been in operation since 1989 [SW89].

Another approach that incorporates both local and global evaluations to determine similarity among encyclopedia articles is by Salton and Buckley, [SB91a, SB91b]. They start with a global similarity metric and then use the top retrieved articles as subsequent queries. This multistage search showed additional improvement when requiring a minimum paragraph-to-paragraph or sentence-to-sentence similarity threshold.

[6] We may, however, use layout-based abstractions for the presentation of results.

Rather than using sentences or paragraphs, Moffat *et al.* tried section and "page" divisions [MSDWZ94]. They made page divisions based on the number of bytes in consecutive paragraphs; this would approximately normalize the length of each page. Again, though, their task was to retrieve entire documents.

Several systems actually retrieve passages in response to a query. In one case, the query set is restricted.

More recent work by Salton, Allan, and Buckley executed single-stage searches, computed global similarity, and then those documents exceeding a threshold had a local similarity computed at the sentence level [SAB93]. This allowed for sections and paragraphs in addition to entire texts to be retrieved in response to a query.

While this is in the same spirit as our work, in order to compute a sentence level similarity, they start with queries that are articles themselves. We have much smaller individual elements available as initial queries; our annotations are generally a sentence or shorter in length.

Hahn argued for analyzing and indexing the conceptual or thematic structure of a text and implemented his ideas in the TOPIC system [Hah90]. TOPIC was tested over short texts describing computer systems. Hahn asserts that storing a text's conceptual structure makes possible three types of retrieval operations: abstracting or summarization, fact retrieval, and passage retrieval. Summarization is discussed below in Section 4.5. His techniques for fact retrieval are closely aligned with those used by current information extraction systems: use of a domain-specific knowledge base, generally in the form of a dictionary with information about the semantic and syntactic constituents that may fill each term's representation. More on information extraction can be found in Section 3.4.

His final type of retrieval, passage retrieval, is made possible by linking passages to their representative in the concept index. This is a true instance of "passage retrieval", with the limitation that the only way to retrieve a passage is if it has been linked into the concept index. Passages are not retrieved in response to general queries, only in response to a concept.

Another technique based on a document's thematic structure is TextTiling [Hea93, HP93]. With TextTiling, coherent subtopic discussions determine document indices. Subtopic discussion breaks are ascertained by evaluating the frequencies of the terms found in proximity to one another. They compare small blocks of text (i.e. three to five sentence units,) to adjacent blocks to measure similarity and merge those that are thematically related. Subtopics or "tiles" may span multiple paragraphs. High value terms from within each tile form the subtopic's label. Assembling the labels into a listing effectively represents a table of contents for the document.

The query language for this system allows users to request related documents based on either or both overall document similarity and similarity among tiles. Document retrieval is based on accumulating the scores of the tiles. This type of query allows for searches where a topic is specifically given as being subordinate to a main topic. This is similar to what we wish to do, however we do not decide *a priori* which subtopics are discussed within a text. Since the discussion of our slots is usually on the order of one or two sentences long and may be scattered throughout the text, we are not able to take advantage of the TextTiling methods.

One final piece of research that should be mentioned is that of O'Connor [O'C70, O'C73, O'C75, O'C80]. He experimented with the retrieval of "answer indicative" or "answer reporting" passages.

The first experiments were on a corpus of 82 full-length texts on information retrieval. He created the second corpus by having medical librarians search for documents that answered specific medical questions and subquestions. To be added to the collection, a document had to be an "answer-paper", that is, one that contained either an "answer indicative" or "answer reporting" passage.

Two individuals without biomedical knowledge created sets of search terms for each question. They spent between 2.5 and 3.0 hours, on average, developing the query terms for a single question. (There was a 4.0 hour maximum per question.)

The collected documents were then manually searched for sets of consecutive sentences containing minimum numbers of matching query words. The results of this simulated retrieval were quite good with recall being either 72% or 67% depending on the set of search words.

This work is quite relevant to what we are trying to accomplish. Their simulations have shown that it is quite reasonable to expect that we can retrieve"answer indicative" or "answer reporting" passages with our queries. We have the advantage of being able to automatically run our queries and of generating our passage queries as a by-product of another process, rather than spending hours manually encoding sets of terms.

3.6 Concept Recognizers

Controlled vocabularies allow for higher-level reasoning about a document than reasoning with just using the words found in the document. They provide a second level of representation for a text.

If we consider that 1) slots represent concepts, and 2) indexing a document based on the concepts it contains enables a second level of reasoning about a text, it would seem a natural match to try to exploit concept recognizers for the passage retrieval task.

Concept recognizers come in various forms including table lookup, synonym expansion, and pattern recognition. They support various natural language processing and information retrieval tasks by merging multiple means of expressing a concept into a controlled vocabulary or into a single canonical form. Concept recognizers enable NLP systems in their extraction task by locating important patterns or terms from which information can be drawn. They enable IR systems to index and utilize a second and more abstract representation of each text [CC93, RJ91].

Table lookup can be used when the items in the concept can be enumerated, such as foreign countries, U.S. cities, or pieces of furniture. Pattern recognition, such as a series of capitalized words may identify company and other proper names. Other patterns that can be easily recognized include telephone numbers, dates, and social security numbers [Mau89].

FLEXICON, now Flexlaw, a legal text management tool, relies extensively on concept recognizers for its searches [SGM$^+$95]. The system displays four quadrants to the user, each containing a different type of knowledge. Three of these quadrants use concept recognizers. The first, in the form of table look-up, matches concepts from a manually encoded Legal Concept Dictionary to the text of each case opinion. The dictionary entries are highly technical terms and phrases as well as synonymous expressions. The next two types of conceptual knowledge are based on statutory and case citations. They rely on pattern matching and capitalization for identification.

The final quadrant contains the "facts" of the case. It basically consists of the words and phrases that are left over when the first three types of knowledge, along with stop words, are removed. Some reduction of the search space is done by removing low frequency terms and lengthy phrases.

4 Related Work

There are many aspects of this research that are currently the subject of much study. There is on-going research into the posing of multiple queries and the merging of results from multiple databases. Visualizing retrieval results is a recent field of exploration due to advances in display technology. Text classification or categorization, with the related are of document clustering, along with summarization, have been investigated for well over 30 years by both the IR and NLP communities. We now provide a brief overview of these areas.

4.1 Posing Multiple Queries and Merging of Results

There are two different categories of query merging. The first is when the same query is posed over multiple databases and each of their results must somehow being combined to present one overall retrieval. The second category of merging takes place when multiple representations of the same information need are posed against a single collection. The former is the "data fusion" problem while the latter is the "query combination" problem.

There is currently a great deal of on-going research into how to best combine the results of a query performed over multiple databases [VGJL95, CLC95, BKFS95]. (See [Har94, Har95] for additional references.) Here they must deal with the problems of normalizing statistics across the various collections, how to decide which collections might provide the most fruitful results, deciding how many documents to retrieve from each source, and how to actually combine the results from each source. Since our retrievals remain within the bounds of one collection and we examine a single document at a time, data fusion is not an issue in this research.

Query combination becomes an issue if we treat each annotation as a separate query. We now encounter the dilemma of how to merge the results so that we can identify a ranking of passages. Research into query combination has shown that using multiple representations for the same information need and merging the results generally improves retrieval performance [SK88, BCCC93, BCB94, BKFS95]. The issues here are how to generate more than one representation and the costs associated with doing so and how to effectively combine the results at query time.

As we have an available set of annotations for each slot, we can treat each of these as a different representation of the same information need. This research will be examining several means of query merging, utilizing our case-base of annotations.

4.2 Visualization of Retrieval Results

Once we have retrieved the passages most highly similar to the query, we will want to somehow display them to the user. For some of our queries we can learn, by observation of manually annotating sample case texts, that location is important for extracting certain slot fills. For example, we learn that the header information such as defendant, plaintiff, judge, hearing date, court, etc. are located at the beginning of the opinion. We also learn that "decision for" may be found either early (within the first few paragraphs) or at the very end of the text (within the last paragraph – unless there are dissenting opinions). There may be other instances of slots, where, having knowledge of the document's structure, and being able to see the links between related passages, will lead the user more quickly to the important passage(s).

We would like to take advantage of the knowledge we have learned. One means of doing so would be by displaying the document as an arc with the related passages annotated, we can quickly go to the passage that is most likely to contain the information we desire (at least hypothetically, by "selecting" the passage we wish to examine). This would take advantage of the work done by Salton and Allan on visualization of document structure [SA93, All96].

Other means of visualization include: cone trees [RMC91], a perspective wall [MRC91], a document lens [RM93], TileBars [Hea95], and NavigationCones [HKW94]. (See [RPH+95] for additional visualization technique references.) While these are all interesting options for the visualization of our results, this aspect will be left for future exploration.

4.3 Text Classification/Categorization

Suppose that we are unable to directly locate slot fills from a text, or even to locate the best passage(s) for a slot fill. One alternative would be to see if we could determine with a high degree of belief whether or not a text contains any information about a slot or possibly even a dimension (using a HYPO-style definition [RA87, Ash90]). Being able to more generally claim that we believe a document does or does not contain information about a slot would allow the case-based reasoner to reason, at a gross level, about the new texts. If we can do this, then potentially, we would not need to bother with generating extensive case representations for the majority of texts; we simply generate a case representation that lists those factors believed to be present in each text.

If we can build such a general representation based on the presence or absence of factors, then our reasoner can assign similarity ratings on this more general basis (using a nearest neighbor algorithm or by constructing a claim lattice.) Once we know which cases have the most potential to play a role in analyzing the current situation, we can concern ourselves with constructing a more in-depth case representation for this restricted set.

To make this entire scenario plausible though, it is essential that we believe that reasoning with a more general representation of the dimensions or features is sufficient for our needs. Assuming that it is, how would we go about determining that each case does or does not contain information about each slot? One possible method of making such an assertion would be classification techniques.

We may not have enough data on which to build a set of classifiers as these systems generally use much larger amounts of training data than we have available. The large size of the training set is necessary to gather sufficient statistics or other information with which to be able to make distinctions between documents.

Frequently, these systems use documents that are typically a page or less in length for training. Our documents are much longer, making it more difficult for a system to identify the important terms for a concept. One set of experiments on classifying medical record encounter notes used approximately 1000 training documents, where documents rarely exceeded one page in length [ASP+95]. Another medical classification task, that of assigning codes to discharge summaries, employed more than 10,000 training documents averaging 4.43 codes per document [LC95]. Discharge summaries average one fifth the length of our documents. There have been various experiments on classifying newswire stories, where again there is a large training set and the documents are much shorter than ours [Lew92, LG94, Yan94, MG95]. Another system categorized part of the MEDLINE collection where documents average 168 words and the training set contained 586 documents [Yan94].

4.4 Document Clustering

This area of research is closely related to document categorization. Clustering techniques group documents together based on either the number of features they share or some other metric that measures similarity. Features may be the words, phrases, or concepts that compose the text or other values such as author, document length, date of publication, etc.

Some clustering techniques are directed at the results of a query [CPKT92, CKP93], while others use clustering to locate highly correlated text sections within documents [SB91b]. Unfortunately, the passages that this research aims to locate may not be highly correlated to other passages within the document or to the set of annotations as a group. In order to take advantage of clustering techniques, we would have to find a statistical means of representing our sets of annotations as a cluster. This might prove difficult, if not impossible, to do since it would have to be done *ad hoc*. If we could establish our annotation sets as clusters, then they could be tested against the passages of each document for correlations.

4.5 Text Summarization

Automatic summarization done by extraction is the attempt to locate those portions of a text deemed to be the most representative of it. (See [SAB93] and its references for recent work on summarization.) The user doesn't have specific portions of the text in mind when requesting a summary; they merely would like the subportion(s) of the text that best represent the content of the entire document. The user may request that the summary be restricted to some percentage of the original text, e.g., 30 or 50 percent.

Paice and Jones built a system to do summarization by examining the structural features of technical papers [PJ93]. They created case-frames based on stylistic constructs found in a collection of crop agriculture technical papers. They gathered patterns indicative of particular types of information useful for generating an abstract. Their system automatically fills slots when their patterns match the new text. This approach is similar to that of many information extraction systems with the exception that the extracted information is then used to generate a document's abstract.

Most summarization systems do not fill case-frames, nor do they know *a priori* what sort of information the user desires. These methods are generally a means of creating a second, more compact representation of a text. This work does not attempt to create narrative summaries. Data derived from the texts is to be used in analyzing the situation described in the text and in reasoning about the text.

5 Issues

We do not propose to find all of the documents within the corpus that might contain a particular feature. If we were to do so, potentially many of the retrieved documents would not be relevant to our overall situation and would be later discarded. Rather, we prefer to focus on finding each feature's occurrence within a predefined subset of the corpus. This subset is that which is believed to be relevant to the overall problem situation because it was retrieved in response to a query generated by the overall fact situation. We have already achieved good results in retrieving documents relevant to the current fact situation [DR95, RD95].

It is possible that we missed domain-specific, relevant documents when retrieving based on our original query; however, that is not the concern of this research. It is also possible that there exist documents that were not retrieved in response to the original query that contain this feature. Again, we are not concerned with these documents since they have not satisfied the original problem-specific relevance criterion.

The process of gathering annotations, composing queries, retrieving passages, and evaluating results, gives rise to many complex issues. In the next sections we traverse through the passage retrieval process enumerating and elaborating on the issues that arise during each step.

5.1 Case-Base Construction

The case-base of annotations becomes crucial in the process of retrieving passages. Care must be taken that annotations are of reasonably good quality to ensure that they will enable the retrieval of similar or related passages. This is especially important when we consider that the case-base will be derived from a small set of texts, less than 25. To allow for the best possible case-base, annotations must meet the following criteria: the annotation explicitly gives a slot fill or the text provides sufficient information such that by reading it, a human could infer the slot fill.

Since the case-base will be manually created, we need to provide guidelines on the amount and type of information that should be included in an annotation. Pertinent issues are:

- how long should the annotations be, and
- how much information should be annotated to denote that some text provides information about a slot?

To illustrate these problems and staying with our example of monthly income, below are several example sentences.

1. "In an amended family budget filed June 27, 1983 the debtors list total income of $1,468.00 per month and total expenses of $1,350.00."
2. "The bankruptcy court found that the Kitchens' net disposable monthly income for 1979 averaged $1,624.82, $1,800 when federal and state income taxes are included, and that their estimated future monthly income was $1,479."
3. "Her monthly salary is $1,068.00."
4. "He is currently employed by a firm in a sales capacity, earning approximately $15,642.62 per year."
5. "Debtors' sworn Chapter 13 statement shows a combined monthly spendable income of $1,010.50, and monthly expenses of $900.00.

Should the annotations be complete sentences, just the slot filler, the smallest set of words that delineates the filler, or some intermediate level of text? One advantage to using complete sentences is that we could try to take advantage of linguistic context. However, we will not attempt to do so in this research. One disadvantage of using entire sentences is that if the information is only a small portion of a lengthy sentence, then we may have many terms in our case-base that provide no bearing on finding relevant passages.

In the first two examples, it seems likely that we will want only a portion of the text to be included in the annotation case-base. For the first example, we will save "the debtors list total income of $1,468.00 per month" and in the second, "net disposable monthly income for 1979 averaged $1,624.82".[7]

[7] If we were examining these as solution passages, we would accept any portion of the second example.

The converse problem of lengthy sentences arises if we only select the fill as our annotation when the fill in and of itself is meaningless for future queries. For example, if we only select the fill from the third sentence, then we will be searching future texts for the number "1,068". A more meaningful annotation would include larger phrases or even the entire sentence. We would select the entire third example sentence into our annotation case-base.

It is less obvious as to how much of the fourth and fifth examples to use. We would probably enter the entire fourth sentence. In the fifth case, we will chop off text from both the beginning and the end to leave "statement shows a combined monthly spendable income of $1,010.50". We remove "Debtors' sworn Chapter 13" since every case in the corpus is a Chapter 13 case and hence, this information would be useless for retrieval about income. Similar to the first example, we also remove the final clause, as it pertains to expenses and not income.

The final examples we give are when the information necessary for a slot fill spans multiple sentences. The following is from the *Easley* case and provides the information necessary to determine the monthly expenses:

> "Food of $200 per month is reasonable. Monthly clothing expense of $20 and laundry expense of $10 are consistent with the debtor's simple lifestyle. Doctor expenses of $50 per month were justified for dental work, psychiatric attention and prescriptions. Debtor takes medicine twice daily to control violent outbursts and a psychiatrist monitors the medication. Transportation, including vehicle repair and gasoline is reasonably estimated at $125 per month. Automobile insurance is $41 per month. Barber shop expense of $18 and house maintenance of $60 a month were marginally justified by the debtor but are not unreasonable in an amended budget which reflects no allocation for recreation, newspapers, church contributions or club dues."

A shorter example where the information is not contained in one sentence comes from the slot for the unsecured amount of the debt. In this case, the first bit of text provides the background to affiliate the values in the second with the slot. Interestingly, the first bit of text actually appears well after the second.

1. "her two unsecured student loan creditors"
2. "...the NYSHESC and SUNY obligations were Debtor's only scheduled debts, with SUNY due approximately $501.58 as the result of a National Direct Student Loan, and NYSHESC due approximately $7,600.80 as the result of two Guaranteed Student Loans..."

These examples show that we want our annotations to be descriptive of the slot, yet not contain excessive verbiage, nor be too terse. Our first attempt to satisfy these criteria will result in a case-base that contains both partial and complete sentences. In summary, the following are issues when constructing the annotation case-base:

- How long should an annotation be?
- Need it be a complete sentence? multiple sentences? or, will smaller segments such as phrases be sufficient?
- What text needs to be included?
- How much expertise is needed and how much care taken when creating the annotation case-base?

Initially we will assume that not a lot of expertise is necessary when gathering the annotations. If our results indicate otherwise, we will reexamine this position.

5.2 Query Formation

Assuming a set of annotations, all believed to be relevant to a given slot, how can we take advantage of this information? We would like to automatically transform this information into a query to locate "good" passages from within a novel text. While there are any number of specialized operators we can wrap around the text in the annotations, we must also decide which ones make the most sense.

To help us decide how our queries should represent the knowledge found in the annotations, our first task is to see if we can observe any patterns among them. If we do observe patterns, we should test if they are regular enough to describe within our set of query operators. We may wish to add additional operators or "concept recognizers" to accommodate these patterns.

Potential concept recognizers that we may wish to employ for these domains are dollar amounts and time periods. Specific dollar amounts show up in many of our slots and actual fills are frequently found in the text. Example slots with dollar amounts are monthly payments (toward either a bankruptcy plan or a loan), monthly income, monthly expense, amount of a debt, income from home office, and others. This recognizer would simply scan for instances of the dollar symbol followed by a numeric value. The concept of time periods, such as weekly, monthly, annually, and yearly, likewise show up in our domains with some regularity. Besides those descriptors just mentioned, the phrases "per week", "per month", and "per year" would additionally be a part of this time period concept.

Second, we should see if we can manually generate a query (using this information) that will retrieve the desired passages. These queries should initially be tested on the texts deriving the annotations, the CKB. After those texts, we should expand to novel ones.

When we start to build our queries, we must decide if we are going to turn the annotations into one query or to create multiple queries. If we wish to pose a single query, then we must decide how to coherently merge all or a subset of the annotations together. If we are to pose multiple queries then we must find a means for merging the retrieval results. In either case, we must create a method for converting the raw annotations into our query language.

The next decision is what operators should we use? Natural language queries with INQUERY currently use a *#sum* operator wrapped around the terms [CCH92]. This seems a bit too simplistic, but it might work perfectly well for this problem. It may be reasonable to collect proximity information about the terms and add this data to the queries. There already exists an INQUERY tool that makes this possible, even when our queries are not complete sentences. Were we to use proximity information, we will have to examine what size windows are reasonable and whether or not the terms must be ordered.

Many IR systems stop and stem by default. We must decide if this makes sense for this application. Riloff would argue against doing them for the information extraction task [Ril95] and Liddy also found that different discourse elements in abstracts were distinguishable by the form of the verb used [Lid91]. We must examine our annotations and decide which seems the most reasonable.

Our hypothesis is that we may want to stem but not stop. We base this hypothesis on examples found in the annotation case-base. Returning to the monthly income slot, many of the annotations use variations on the expression "payments of" followed by a dollar amount. In this domain, "payment" occurs quite frequently, while instances of "payments of" occur much less often. In this particular case, "of" provides a valuable indicator for a following dollar amount, highly indicative of the monthly income slot. It is conceivable that we may wish to modify the stop word list so that it is a minimal set, but this will have to be tested.

It is possible that we may not want to stem either, but we have not yet explored the annotations database enough to see if there are any concrete examples where this would make a difference. There are intuitive reasons for not stemming documents when doing retrieval of passages. Stemming is a valuable device for saving space when dealing with large collections by conflating multiple variants of the same word. This is problematic when stemming conflates words that have different meanings or if the task is such that the usages of the related words helps to provide discrimination among collection elements. In our case, we are no longer concerned with full-text document retrievals. We hope to make finer grain distinctions by retrieving smaller text elements, passages. Using the original words with their inherent meanings may be more appropriate at this level of retrieval.

Since we are not certain whether either or stopping or stemming makes sense in this environment, we will run some preliminary tests and examine those results. Based on those results, we will build the appropriate database for further testing.

Another query formation issue is how to deal with redundant parses, (i.e., two annotations may be reduced to the same stopped and stemmed set of terms – either in the same order or not.) If we decide that the fact that an item exists in more than one annotation is valuable, how do we incorporate this information? How do we handle terms, or phrases, that are identical across annotations, or even entire annotations that are identical?

Operationally, keeping redundant items is simple; if making a single query, multiple instances of an item can be reflected in the term weights. This is what is currently done with natural language queries by INQUERY [CCH92]. If there are multiple queries, then we simply run each of them.

Removal of redundant items is slightly more difficult. If a single query is formed, then it is simple to allow INQUERY to merge the items under the *#wsum* operator, but then go back and reweigh all the terms so that they share equal weights.

If there are multiple queries, then the issue of redundancy is more complex. We must decide on a definition of redundant and what it means to be either partially or fully redundant. It is possible for two annotations to stop and stem to the same set of terms, however if proximity or phrasal operators are used, then we must decide if order and distance matter when measuring similarity. If a pair of terms occur within one annotation in one order but in another annotation in a different order, will this matter? What if all but one term differs across two queries? What if two terms differ? We must define how similar two queries must be in order to eliminate or alter one of them.

We also have the option of expanding queries. We could try using a thesaurus to add new terms. Alternatively, we could further refine our expansion operation to take advantage of domain knowledge by building an association thesaurus [JC94]. We have several choices of database on which to base our association thesaurus. We could use either the entire collection or just those documents related to the problem case as found in the CKB. Another possibility would be to built the thesaurus from an intermediate collection that is comprised of the documents retrieved by the document-based query. Thesauri built from the second and third collections would be highly domain- and problem-specific.

In summary, there are numerous options available to us during query formation. This research seeks to address and resolve the following questions :

- Which operators will we use?
- Should we stop and/or stem?

- What will we do with redundant items?
- Do we pose one or multiple queries?
- Should we use concept recognizers?
- Should we expand queries?

5.3 Retrieval

Another issue is the length of a passage: How large or small should a retrieved passage be? Do we use writer-defined structures (e.g. sentence, paragraph, section), subject delineated segments (ala TextTiling,) or bounded-size windows? If we use bounded-size windows or discourse-based elements, how large a segment should each be? Should the retrieval size be the same for every slot, or should it depend on the size of the user-provided annotations?

Since the best results for retrieving documents when incorporating information about passages comes when using fixed-size passages, we will try this method first [Cal94]. While our task is not the same, we would like to retrieve the best passage instead of the best document, so we will utilize their evaluation technique to locate potentially good passages within each novel document.

We run the risk of using an incorrect window size for retrieval. Problems could arise if there are too many query terms for too small a passage size. When this is the case, there will be too many positive hits within many passages and we will be unable to distinguish relevant from non-relevant passages. This argues for either ensuring the retrieved passage size stays proportional to the number of query terms, or to pose multiple queries. Another consideration will be when adjacent passages receive high belief scores. In this case we will consider merging them into one passage.

During passage retrieval we should consider whether a document and all of its passages are to be considered as a single collection – generating its own statistics – or whether we use statistics from the total collection. We will have to run experiments to see how treating a document as a separate collection of passages affects retrieval of individual passages.

A final issue we must address when retrieving passages is whether we will require a minimum number of matching terms within the passage before allowing it to be retrieved. This was the case in [SAB93], where minimum numbers of terms had to be matched at the sentence level. Initially we will not require minimums. Should it be the case that passages with fewer matching terms rank higher than those with greater numbers of matching terms, and that the rank ordering is poor, we will consider imposing minimums as a means of boosting performance.

5.4 Merging Query Results

If we pose multiple queries against the same set of passages, we must somehow merge the results in order to attain an overall "best" passage or ranked set of passages. How best to merge is a very complex question. Available to us for consideration are such factors as the number of queries posed, the depth of the rankings each query provides, belief scores for individual passages, and the rank score of a passage.

If we are to merge query results, then we must either accept all passages that return a belief value, or set a cutoff, based on either rank or a threshold belief. If we utilize either a rank or belief cutoff, there should be some theoretical or logical justification for this value. Additionally, when using a rank cutoff, consideration should be given to dealing with ties and to queries where the total number of passages returned is less than the cutoff.

We have multiple options available to us for merging query results:
- Sum the belief scores for each passage across all queries.
- Calculate the average belief score for each passages across all of the queries.
- Examine the ranks of the top n passages from all of the queries - try to do something with this ordering.

None of these options has any intuitive appeal. Therefore, as an alternative to trying to merge the document beliefs across multiple queries, we can do query combination where we merge all of the query results as a part of the retrieval process. We can do this via a boolean *or* operator that we wrap around all of the individual annotation queries. Consequently, we will explore various ways to form natural language queries using the annotations either as individual queries that are merged via an *or* operator, or by combining all of the terms across the annotations into a single natural language query. A third possibility is to test the merging of the results from the two different types of queries.

5.5 Presentation of Passages to the User

There are numerous interface issues when dealing with the presentation of results to a user. For example,
- How many retrieved passages should be shown to a user?
- In what format should they be shown?
- Do we present a passage, a sentence, a paragraph, or some kind of listing with the highest ranked results?

Additionally, there is the ability to selectively highlight portions of a text within a display. What portion of the text should be displayed and what subportion should we highlight? Do we display only the best passage, or do we include additional text to possibly provide additional context? Do we expand the passage in each direction so that complete sentences are shown? Do we further expand so that the user sees only complete paragraphs? Or, finally, do we just bring up the entire document and then highlight a subportion of it?

Assuming that we use highlighting to focus the user's attention, what text do we emphasize? From less to more highlighting, some of our options are: emphasizing starting at the first matching word within a passage, the entire retrieved passage, the sentence(s) surrounding and containing the passage, or the paragraph(s) that do the same. One final consideration is the possibility of only emphasizing those words that matched query terms.

Presentation of query results and user interfaces for information retrieval systems is a large body of research. While this thesis poses many new aspects of interaction with a user, and these issues will have to be addressed in future work, they are not the primary focus of this research.

5.6 Solution Keys

To test this approach, we require knowledge about which portions of a document contain text relevant to the slot at issue. The passages that comprise the solution keys should probably be more lenient in their description of the slot than the annotations found in the case-base. Here we must decide whether to accept passages that provide information not directly applicable to the case at hand or not directly providing a slot fill. Both these passage types provide information "about" the slot but do not directly contribute to the filling-in of the current case-frame. Therefore, we must answer the following questions to determine what our Solution Keys will be:

- What do we consider a solution?
- Will there be a single "best" passage or will there be multiple "good" passages?
- Will a slot require that information be drawn from several sources within the text and all of these passages must be retrieved?

These simple questions are, in fact, much more complex than they appear and must be resolved in order to evaluate the success of the system.

The answer to the first question, "Which passages will be in the solution set?", somewhat answers the second question of whether there is a single best solution. It should be obvious that we will include in our solution any passage that expressly contains the slot's fill. Furthermore, we should also include any passage from which a human can infer the slot's value.

What is not so obvious is what to do with passages that provide a slot fill, but do so for another case that is being used for illustrative purposes or is being analogized to assist with a particular point of discussion. Making a distinction between a passage that provides a slot fill for this particular case and one that does so for another, at this point, is not reasonable. The state of the art is not sufficiently advanced to be able to provide enough context to discriminate amongst these passages. This type of passage should be considered acceptable and included as a solution.

What about passages that discuss the slot, but do not contain enough information to provide a fill? At the present time, for many slots, the state of the art does not allow us the ability to discriminate between these passages and those that actually contain a fill. If we have an available means of directly extracting a value or of discriminating between these types of passages, then we will do so. For most of the slots about which we are concerned, this will not be the case. Even more difficult a task would be to automatically discern differences between passages that are "about" a slot from passages that have enough information to derive a fill. We will include passages that discuss or are "about" a slot in its set of acceptable solutions.

In answering the next question, that of whether there is a "best" passage, it is important for us to recognize that there are multiple passages within a document that each, independently, provides enough information to fill the same slot. When there are multiple "best" passages we would like to give our system credit for finding any one of the solutions and not receive reduced credit for not finding <u>all</u> of them. We defer discussion of how we will do this until the section on Evaluation, 5.8.

We now turn to the last question, that of the need for multiple passages to discern a slot fill. This type of solution is beyond the scope of this research. Future work will examine these solutions and their associated slots for possible means of generating multiple queries and retrievals and other methods to notify the user that the solution spans more than one passage. It may be that the passages are in close proximity to one another. Were that the case, it may be a simple matter of either expanding the passage size or increasing the amount of the presented material. If there are many slots that fall into this category, we may try these simple approaches.

In summary, to be considered an acceptable solution, a passage must:

- explicitly state a slot fill for either this or another case,
- inferentially provide a slot fill, or
- provide discussion about the slot without giving a fill for any particular case.

Our objective is to reduce the amount of text a user must read in order to derive a slot fill. If we can focus the available material to be read down to information "about" a slot, and within that set include passages containing the data for the slot fill, then we have substantially reduced the total volume of text that the user must read to create a case-frame representation for a text.

5.7 Deriving Solution Keys

It is well-known that individuals will provide differing relevance judgments for the document retrieval task. This is due to different interpretations of the stated information need, different interpretations of the meaning of the document in question, order of document presentation, personal knowledge of the domain, etc.

We should, therefore, expect that there will be disagreements between individuals marking text passages that are "about" a topic. We should also expect there to be some level of overlap. We will examine the results of an initial marking to see if we have a sufficiently high level of agreement to proceed.

We will use an undergraduate to mark segments of texts that are "about" a particular slot. The student will be presented with texts to mark and a working definition of the slot. Initially, the student will denote selected segments by marking with a highlighter on paper copies of case opinions, but later we might use a computer screen with specialized marking software. We will periodically and randomly screen the student's marking so that we may test for consistency. All readers will be asked to categorize the marked segments as belonging to one of three types:

- the segment contains a value for the slot either for this case or another,
- the segment provides enough information for you to ascertain the value of the slot, either for this case or another, or
- the segment gives information "about" the slot's value.

These categories go from more specific to more vague. Because of that, we would expect there to be more agreement across individual markers, the higher a segment is in this categorization. We will test this hypothesis when examining the markings. Our initial concern is for the totality of marked passages and not the categorization of each.

From these markings we will ascertain into which passages each falls. If a marking crosses passage boundaries, then all shall be considered as a part of the solution.

We have experienced several problems with having one reader. These include:

1. missing relevant passages,
2. marking a segment in one text and not marking an identical (or nearly identical) one in another, and
3. marking larger segments than are necessary to describe the slot.

The first and second problems are understandable considering the length of the texts and the type of information that may be relevant. Some of the slots are difficult to describe, therefore increasing the level of subjectivity allowed the reader. The third problem merely allow for makes liberal positive judgments. We would prefer that the system not be given too much leniency in finding relevant passages.

Since finding all of the text about a particular topic has proved to be a difficult task in practice, we will have a second reader that provides relevance judgments based on the retrieval results. The top 10 passages from each attempted query will be merged into one large set and placed in ascending order. This reader will judge each passage as relevant or not.

(We did have a second reader who also marked the relevant portions of each document. This reader was unable to consistently mark the relevant passages. Hence we opted to have this reader make judgments on given passages rather than reading the entire document.)

5.8 Evaluation

We will not be evaluating the system on the basis of the time spent by the system in computing the top passages. System retrieval time is negligible when compared with a user's expenditure of time in reading an entire document. We are assuming that there is no monetary cost to the user. Therefore, the primary evaluation concern will be that of the actual retrieval effectiveness.

Should we use a multi-valued approach to relevance judgments, rather than a standard binary approach? The standard binary approach rates documents as either relevant or non-relevant. For many collections there will be non-rated documents relative to a query. These non-rated texts can be treated as either always relevant or always non-relevant, to produce best-case and worst-case scenarios. Since we will be able to provide relevant/non-relevant judgments for all of the passages in a given text relative to a particular topic, this does not pose a problem for us. However, there may be gradations of correctness for this problem.

If we decide that there are gradations of solutions, and that we wish to differentiate among them, then we must devise a scheme to allocate differing values to the various levels of acceptable answers. We must decide how much credit to extend to each of the different solution levels. But first, we must be able to consistently distinguish among the actual solutions and be able to categorize them.

If we strive for a more differentiated definition of a correct retrieval, we will need to examine the solution passages to ascertain if there are stand-alone solutions, (i.e., getting any one out of the bunch is sufficient.) It may be that multiple passages need to be retrieved and read by the user in order to find the slot value. This research will only make a cursory examination of situations where multiple passages are required. If it turns out that the solution passages are found in close proximity, then it may be worthwhile to expand the size of the passage, either during retrieval or as a post-processing step.

We need to consider what metrics we will use for evaluation. There are a variety of available metrics, primarily used by the IR community to evaluate the effectiveness of document retrieval. These include:

1. Recall – compares the number of items that should have been retrieved by a query to those that actually were.
2. Precision – compares the number of relevant items retrieved against the total number of retrieved items.
3. Average precision – precision averaged over some set number of recall points, typically 3 or 11.
4. The E measure – combines recall and precision values at a given retrieval cutoff, ignoring rank. The user provides the proportional value to give to each of recall and precision [Rij79].
5. Total number of relevant documents retrieved by a given cutoff.
6. Total number of queries with no relevant documents among the top n [Cro83].
7. Mean square error
8. Statistical methods for comparing ranks - Kendall, Spearman - but what does this tell you overall?
9. Probability of getting a relevant document as rank 1, probability of getting a relevant document within the top 2 returned, etc.
10. Some kind of weighting scheme that rewards higher ranked solutions i.e. 1st gets 5 points, 2nd gets 3, and 3rd gets 1. We can even use bins where any score over 5 is gratuitous.

We must decide which of these, or possibly some other, provides the most insight into how well our system is performing.

Before we decide on our evaluation metric, we might wish to consider how many "solutions" there are for each slot. If we pose exactly one query per slot, then the evaluation of the retrieved top passages is less complex than if multiple queries are posed. Additionally, we should consider how many solutions there are on average per text. If there are only a few, then we probably do not want to look at precision, recall, and average precision.

Within the realm of document retrieval, if there are only a small number of relevant documents that will satisfy the information need, then the query is "unstable". This is due to wild swings in the recall and precision statistics because of minor changes in the retrieval results. Since we expect that there will be only a few relevant items, passages, for each of our information needs, a slot query, within our collection, a document, we will have to contend with the problem of unstable queries if we use precision, recall, and average precision.

Further, our task does not require that every applicable passage be retrieved. If there are multiple good passages, then merely locating one or two of them may be sufficient for the user to be able to fill in the case-frame representation. Therefore, in this environment, it is acceptable for recall to be low while requiring precision to be high. Hence, if we were to use the E measure, we would weight precision much more heavily than recall. It is also not unreasonable that we might not use recall at all and rely on the other measures for assessment. Similarly, not using recall would rule out using average precision to evaluate system effectiveness.

We can take advantage of the fact that our retrieval engine generates a rank ordering among the retrieved passages.

Assuming we are primarily interested in looking at the top rated passages, we must decide on a reasonable value for n: How far down the ranked list will we look? Do we look at the top 3, 5, 10, or more, passages to see how many of the "solutions" we have retrieved? Will this depth vary based on the importance of a slot?

Since we strongly desire to reduce the total amount of text presented to a user, we would like to evaluate our system with the most stringent possible metric. We might expect that a user would generally be willing to read the top three passages, and occasionally as deep as the top five. Reading beyond the top five passages would be unreasonable to expect unless done very rarely or if the slot were particularly important.

We will, therefore, evaluate our system over the top three and five passages. At this point we will not make any distinctions in importance among the slots; all slots will be evaluated equally. Further, we will be concerned with slots and texts in which no solution appears in the top set and do a post-retrieval analysis to see if we can ascertain any reasons for individual failures.

We are interested in a metric that heavily credits those rankings where good passages are found early, similar in nature to how cross-country race scores are computed. This is a stronger measure than 5. above, where we simply counted the number of good passages in the top n. We should also look at 6., where we see if there are queries in which we have not satisfied the requirement to find any of the acceptable passages.

One measure that incorporates these features is Expected Search Length (ESL) [Coo68]. It measures the amount of effort wasted when trying to find a particular number of relevant items while searching through a ranked list. From ESL we can compute the average ESL to compare values across queries when the collection and retrieval engine remain stable. Further, we can derive the Expected Random Search Length (ESLR) and combining it with ESL we can compute the Expected Search Length Reduction factor. This enables us to compare results across queries and collections.

5.9 Learning New Annotations

To incorporate learning to the CBR component, our system will add new annotations to the case-base. The user denotes these new annotations when examining passages. However, in order to add them, our interface must be sufficiently robust. The interface's primary role is to facilitate the extraction of information about slot fills. Adding the ability to designate annotations must not detract from this goal. Therefore, our first consideration is whether we even allow the user to add new annotations to the case-base. Assuming we do, and we assemble an appropriate interface, we must theoretically and operationally incorporate these new annotations into the case-base.

Designating and adding new annotations will not be one of the goals of this work. It is sufficiently far through the information reduction cycle, that it will be a goal of future work.

5.10 Inadequate Slot Descriptors

One limitation of our approach is that we may not be able to determine when there is no passage within a novel document that will satisfy our information need. This is currently true of all information retrieval systems and is a side-effect of using language. Language is inherently ambiguous; there are many different ways of relaying the same meaning. Therefore, can we determine that there is no text within a novel document that will satisfy our information need? No. This is a weakness. But this weakness or limitation is true of all systems that are incapable of learning or generating novel criterion. Unfortunately, this is true of all current systems and not a reflection on this work.

If we are able to rank order text pieces, then we are stating that the top-most piece is the most likely to contain information relative to the examples or queries we have presented to the system. If the requested information is not present in the top ranked pieces, and we believe our retrieval algorithm to be sound, then we would like to be able to say that the information is not present. However, we are unable to know if information is, in fact, present, yet in an unfamiliar form. It may well be that the information exists in the document in a form with which we are unacquainted; our exemplars do not cover this particular situation. The best we can do at this point is to state that the information, if in a form we can recognize, based on information derived from our set of exemplars, we believe to be most probably found in the returned passages.

6 Summary

In this document we have expounded upon the myriad issues facing this research project and presented possible solutions for many of them. Below we summarize those issues that this research will address. The bulk of the work will focus on the issues of stopping and stemming, and then trying to identify a simple, efficient method for constructing effective passage queries.

- Case-Base/Annotations –
  - There currently exists a case-base of annotations for the bankruptcy (Chapter 13, good faith) domain. (The texts for this work were previously used for the first portion of the CBR-IR processing cycle and are reported in [DR95, RD95].) There are 13 cases containing approximately 60 slots. We derived annotations for approximately 55 slots. Some of the slots are very similar and therefore the entire case-base needs a scrubbing to determine how many unique slots we have.

  - Besides doing a scrubbing of the slots, we will also conduct a pilot study to decide whether stopping and/or stemming seems to make a difference. If it appears that one or the other will make a difference in certain cases, we will try to classify those cases.
  - Later experiments will look for instances where concept recognizers might prove beneficial. Since it seems likely that money and time period recognizers will be useful, we will build these.
- Query Formation – We will first look at using the annotations as one large natural language query. We will test both weighted and unweighted queries to garner insight into the redundancy question. We will also look at using each of the annotations singly and combining them with a boolean *or*. We will look for ways to utilize INQUERY's structure operators and to incorporate recognizers into queries. Use of InFinder or other thesauri techniques will be done last.
- Retrieval – Not an issue for this work. We will retrieve passages of fixed-size windows. The size will initially be set to 20 terms, representing an approximately two to three sentence segment.
- Merging Retrieval Results – Not an issue for this work. All queries will result in a single listing of the top ranked passages.
- Presentation Issues – Not an issue for this work. For comparative purposes we will generate listings of the top ranked passages with their starting token position and their belief value.
- Solution Keys and Deriving Solution Keys – Solution Keys need to be derived for whichever texts, both novel and training, and slots we test. The preferred method for marking texts is to have individuals highlight text segments as relevant in such a manner that we can easily derive solution keys. We will have individuals denote into which of the three categories each segment falls (as described in Section 5.7), but we will initially lump all the segments together for evaluation purposes. Each individual will mark approximately n texts per slot, and there should be at least two individuals that perform the same task.
- Evaluation – We will compare across the various queries to see which of the query formation methods work best.
- New Annotations – Not an issue for this work. We will not yet try to designate and/or incorporate new annotations.

A Sample case-frame from the bankruptcy "good faith" domain

The table below gives each slot and it's corresponding type from Section 1.5. Examples are given for the slots of type set and category. The first entry after a line is the name of the object class that contains the slot; it is signified with a pair of parentheses.

Slot Name	Slot Type
legal-case ( )	
citation	free text
year	numeric
level	category: :bankruptcy-court :appeals-court
judge	proper name
summary	free text
procedural-status	category: 'plan-confirmation 'plan-filing
decision-for	category: 'debtor 'creditor
citation-links	not implemented
factual-prototype-link	category: 'student-loan 'farm
alternative-factual-prototype-link	same as above
legal-prototype-link	category: 'Estus 'Kitchens
legal-theory-link	set: 'memphis-theory 'estus-theory

bankruptcy-case ( )	
chapter	numeric
plan-confirmed	boolean
past-filings	boolean
chapter-7-filing-date	date
plan-filing-date	date
unfair-manipulation-of-code	boolean
attempts-to-pay	boolean

estus-factors ( )	
duration-of-plan	numeric
preferential-creditor-treatment	boolean
secured-claims-modified	boolean
special-circumstances	boolean
frequency-relief-sought	numeric
trustee-burden	boolean

estus-factor-payments-and-surplus ( )	
proposed-payments	numeric
surplus	numeric
estus-factor-employment-history-prospects ( )	
employment-history	category: :poor :neutral :good
earnings-potential	category: :small :medium :large
likelihood-income-increase	category: :unlikely :likely
estus-factor-plan-accuracy ( )	
plan-accuracy	boolean
inaccuracies-to-mislead-court	boolean
estus-factor-debt-type-and-discharge ( )	
debt-type	set: 'auto-loan 'fraud
nondischarge-7	boolean or category: :after-5-years.
estus-factor-motivation-sincerity ()	
motivation	category: :discharge-educational-debt
sincerity	boolean

plan-payment ( )	
substantiality	boolean
monthly-income	numeric
monthly-expenses	numeric
amount	numeric
surplus	numeric
percent-of-surplus-income	numeric
percent-repayment-unsecured-debt	numeric
payments-already-made	not implemented

debt ( )	
debt-type-amount	set of pairs (debt-type amount) (category: numeric)
secured-amount	numeric
unsecured-amount	numeric
total-amount	numeric
percent-secured	numeric
percent-unsecured	numeric
percent-educational	numeric
otherwise-dischargeable	boolean
date-repayment-obligation-commences	date
student-loan-case ( )	
profession	set: 'dentist 'teacher
dropout	boolean
change-in-field	boolean
loan-due-date	date

makarchuk-factors ( )	
relative-timing	category: :before :after
relative-total-payment-amount	category: :greater :equal :less-than
relative-monthly-payment-amount	category: :greater :equal :less-than
use-of-skills-gained	boolean
attempts-to-pay	boolean
relative-educational-loan-debt	numeric
de-minimis-payments	boolean
other-relevant-consideration	set: :stress :medical-expenses
honest-debtor-case ( )	
profession	set: 'dentist 'teacher
judgment-debtor-case ( )	
judgment-type	not Implemented

Three slots were redundant as they had the same values, but were contained within different objects: Proposed-Payments, Profession, and surplus. Proposed-Payments can be found to be Payments and held the same value as Amount. Profession is in both the Honest-Debtor-Case and Student-Loan-Case and Attempts-To-Pay is in Bankruptcy-Case and Makarchuk-Factors.

There were also three slots that were not implemented: Citation-Links in the Legal-Case, Payments-Already-Made in Plan-Payment, and Judgment-Type in Judgment-Debtor-Case. In summary, there 67 slots, of which three are redundant and three were not implemented, leaving 61 for further study.

References

[All96] James Allan. Automatic Hypertext Link Typing. In *Hypertext 96*, Washington, D.C., March 1996. ACM. To be published.

[Ash90] Kevin D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* M.I.T. Press, Cambridge, MA, 1990.

[ASP+95] D. B. Aronow, S. Soderland, J.M. Ponte, Feng F., W.B. Croft, and W. G. Lehnert. Automated Classificatin of Encounter Notes in a Computer Based Medical Record. In *Proceedings of the Eight World Congress on Medical Infomatics*, pages 8–12, Vancouver, Canada, July 1995.

[Bai86] W. M. Bain. *Case-Based Reasoning: A Computer Model of Subjective Assessment.* PhD thesis, Yale University, New Haven, CT, 1986.

[BCB94] Brian T. Bartell, Garrison W. Cotrell, and Richard K. Belew. Automatic Combination of Multiple Ranked Retrieval Systems. In *Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, Dublin, Ireland, July 1994. ACM.

[BCCC93] N. J. Belkin, C. Cool, W. B. Croft, and J. P. Callan. The Effect of Multiple Query Representations on Information Retrieval System Performance. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, Pittsburgh, PA, June 1993. ACM.

[BKFS95] N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Information Processing and Management*, 31(3):431–448, 1995.

[BL88] Steven Bradtke and Wendy G. Lehnert. Some Experiments with Case-Based Search. In *Proceedings, Seventh National Conference on Artificial Intelligence*, volume 1, pages 133–138, St. Paul, MN, July 1988. AAAI.

[Buc85] Chris Buckley. Implementation of the SMART Information Retrieval System. Technical report, Computer Science Department, Cornell University, Ithica, NY, May 1985. 85-686.

[Cal94] James P. Callan. Passage-Level Evidence in Document Retrieval. In *Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, Dublin, Ireland, July 1994. ACM.

[CC93] James P. Callan and W. Bruce Croft. An Approach to Incorporating CBR Concepts in IR Systems. In *Working Notes of the AAAI Spring Symposium Series: Case-Based Reasoning and Information Retrieval – Exploring the Opportunities for Technology Sharing*, pages 28–34, Stanford, CA, March 1993. AAAI.

[CCH92] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY Retrieval System. In A. M. Tjoa and I. Ramos, editors, *Database and Expert Systems Applications: Proceedings of the International Conference in Valencia, Spain*, pages 78–83, Valencia, Spain, 1992. Springer Verlag, NY.

[CKP93] Douglass R. Cutting, David R. Karger, and Jan O. Pedersen. Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 126–134, Pittsburgh, PA, June 1993. ACM.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the 18th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, WA, July 1995. ACM.

[Coo68] William S. Cooper. Expected Search Length: A Single Measure of Retrieval Effectiveness Based on the Weak Ordering Action of Retrieval Systems. *American Documentation*, 19:30–41, 1968.

[CPKT92] Douglass R. Cutting, Jan O. Pedersen, David R. Karger, and J. W. Tukey. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, Copenhagen, Denmark, June 1992. ACM.

[Cro83] W. B. Croft. Experiments with Representation in a Document Retrieval System. *Information Technology: Research and Development.*, 2(1):1–21, January 1983.

[DR95] Jody J. Daniels and Edwina L. Rissland. A Case-Based Approach to Intelligent Information Retrieval. In *Proceedings of the 18th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245, Seattle, WA, July 1995. ACM.

[Dyk69] Freeman H. Dyke, Jr. *A Manual on Methods for Retrieving and Correlating Technical Data.* American Society for Testing and Materials, 1969.

[Gol91] Andrew R. Golding. *Pronouncing Names by a Combination of Rule-Based and Case-Based Reasoning.* PhD thesis, Stanford University, Stanford, CA, October 1991.

[Goo89] Marc Goodman. CBR in Battle Planning. In *Proceedings, Case-Based Reasoning Workshop*, pages 264–269, Pensacola Beach, FL, May 1989. DARPA.

[Goo91] Marc Goodman. Prism: A Case-Based Telex Classifier. In Alain Rappaport and Reid Smith, editors, *Innovative Applications of Artificial Intelligence - 2.*, pages 25–37. AAAI Press, Menlo Park, CA, 1991.

[Hah90] Udo Hahn. Topic Parsing: Accounting for Text Macro Structures in Full-Text Analysis. *Information Processing and Management*, 26(1):135–170, 1990.

[Ham87] Kristian J. Hammond. Explaining and Repairing Plans that Fail. In *Proceedings, Tenth International Joint Conference on Artificial Intelligence*, volume 1, pages 109–114, Milan Italy, August 1987. IJCAI.

[Har94] Donna K. Harman, editor. *The Second Text REtrieval Conference (TREC-2).* National Institute of Standards and Technology, Gaithersburg, MD, 1994. Special Publication 500-215.

[Har95] Donna K. Harman, editor. *The Third Text REtrieval Conference (TREC-3).* National Institute of Standards and Technology, Gaithersburg, MD, 1995. Special Publication 500-225.

[Hea93] Marti A. Hearst. Cases as Structured Indexes for Full-Length Documents. In *Working Notes of the AAAI Spring Symposium Series: Case-Based Reasoning and Information Retrieval – Exploring the Opportunities for Technology Sharing*, pages 140–145, Stanford, CA, March 1993. AAAI.

[Hea95] Marti A. Hearst. TileBars: Visualization of Term Distribution Information in Full Text Information Access. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Denver, CO, May 1995. ACM.

[HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willett. LyberWorld – a Visualization User Interface Supporting Fulltext Retrieval. In *Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 249–259, Dublin, Ireland, July 1994. ACM.

[HP93] Marti A. Hearst and Christian Plaunt. Subtopic Structuring for Full-Length Document Access. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 59–68, Pittsburgh, PA, June 1993. ACM.

[Huf95] Scott B. Huffman. Learning information extraction patterns from examples. In *Working Notes of the IJCAI Workshop on New Approaches to Learning for Natural Language Processing*, pages 127–134, Montreal, Canada, August 1995. AAAI.

[JC94] Yufeng Jing and W. Bruce Croft. An Association Thesaurus for Information Retrieval. In *Intelligent Multimedia Information Retrieval Systems and Management, RIAO '94*, pages 146–160, New York, NY, October 1994.

[KB95] Bruce Krulwich and Chad Burkley. ContactFinder: Extracting indications of expertise and answering questions with referrals. In *Working Notes of the AAAI Fall Symposium Series: AI Applications in Knowledge Navigation and Retrieval*, pages 85–91, Cambridge, MA, November 1995. AAAI.

[Kol84] Janet L. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Erlbaum Associates, 1984.

[Kol89] Janet L. Kolodner. Judging which is the "Best" Case for a Case-Based Reasoner. In *Proceedings, Case-Based Reasoning Workshop*, pages 77–81, Pensacola Beach, FL, May 1989. DARPA.

[Kot88] Phyllis Koton. *Using Experience in Learning and Problem Solving.* PhD thesis, Massachusetts Institute of Technology, Boston, MA, May 1988.

[LC95] Leah S. Larkey and W. Bruce Croft. Automatic Assignment of ICD9 Codes to Discharge Summaries. Technical report, University of Massachusetts at Amherst, Amherst, MA, 1995.

[Leh87a] Wendy G. Lehnert. Case-Based Problem Solving with a Large Knowledge Base of Learned Cases. In *Proceedings, Sixth National Conference on Artificial Intelligence*, volume 1, pages 301–306, Seattle, WA, July 1987. AAAI.

[Leh87b] Wendy G. Lehnert. Case-Based Reasoning as a Paradigm for Heuristic Search. Technical report, University of Massachusetts at Amherst, Amherst, MA, October 1987.

[Lew92] David D. Lewis. An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. In *Proceedings of the 15th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50, Copenhagen, Denmark, June 1992. ACM.

[LG94] David D. Lewis and William A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, Ireland, July 1994. ACM.

[Lid91] Elizabeth DuRoss Liddy. The Discourse-Level Structure of Empirical Abstracts: An Exploratory Study. *Information Processing and Management*, 27(1):55–81, 1991.

[Mau89] Michael Mauldin. *Information Retreival by Text Skimming.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, August 1989.

[MG95] Isabelle Moulinier and Jean-Gabriel Ganascia. Confronting an existing Machine Learning Algorithm to the Text Categorizatino Task. In *Working Notes of the IJCAI Workshop on New Approaches to Learning for Natural Language Processing*, pages 176–181, Montreal, Canada, August 1995. AAAI.

[MRC91] J. D. Mackinlay, G.G. Robertson, and S. K Card. The Perspective Wall: Detail and Context Smoothly Integrated. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 189–194, New Orleans, LA, April 1991. ACM.

[MSDWZ94] Alistair Moffat, Ron Sacks-Davis, Ross Wilkinson, and Justin Zobel. Retreival of Partial Documents. In D. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 181–190, Pittsburgh, PA, 1994. National Institute of Standards and Technology.

[MUC92] MUC-4. *Proceedings of the Fourth Message Understanding Conference*. Morgan Kaufmann, San Mateo, CA, 1992.

[MUC93] MUC-5. *Proceedings of the Fifth Message Understanding Conference*. Morgan Kaufmann, San Mateo, CA, 1993.

[O'C70] John O'Connor. Answer-Providing Documents: Some Inference Descriptions and Text-Searching Retreival Results. *Journal of the American Society for Information Science*, 21(6):406–414, 1970.

[O'C73] John O'Connor. Text-Searching Retreival of Answer-Sentences and Other Answer-Passages. *Journal of the American Society for Information Science*, 24(4):445–460, 1973.

[O'C75] John O'Connor. Retreival of Answer-Sentences and Answer-Figures from Papers by Text Searching. *Information Processing and Management*, 11:155–164, 1975.

[O'C80] John O'Connor. Answer Passage Retreival by Text Searching. *Journal of the American Society for Information Science*, 31(4):73–78, 1980.

[PJ93] Chris D. Paice and Paul A. Jones. The Identification of Important Concepts in Highly Structured Technical Papers. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 69–78, Pittsburgh, PA, June 1993. ACM.

[Pro93] TIPSTER Text Program. *Proceedings of the TIPSTER Text Program (Phase I)*. Morgan Kaufmann, San Franscico, CA, September 1993.

[RA87] Edwina L. Rissland and Kevin D. Ashley. A Case-Based System for Trade Secrets Law. In *Proceedings, First International Conference on Artificial Intelligence and Law*. ACM, ACM Press, May 1987.

[RD95] Edwina L. Rissland and Jody J. Daniels. The Synergistic Application of CBR to IR. *Artificial Intelligence Review*, 1995. Accepted.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[Ril93] Ellen Riloff. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings, The 11th National Conference on Artificial Intelligence*, pages 811–816, Washington D.C., July 1993. AAAI, AAAI Press/The MIT Press.

[Ril95] Ellen Riloff. Little Words can make a Big Difference for Text Classification. In *Proceedings of the 18th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 130–136, Seattle, WA, July 1995. ACM.

[RJ91] Lisa F. Rau and Paul S. Jacobs. Creating Segmented Databases from Free Text for Text Retrieval. In *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 337–346, Chicago, IL, October 1991. ACM.

[RM93] G. Robertson and J. D. Mackinlay. The Document Lens. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, November 1993.

[RMC91] G.G. Robertson, J. D. Mackinlay, and S. K Card. Cone Trees: Animated 3-D Visualization of Hierarchical Information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 189–194, New Orleans, LA, April 1991. ACM.

[Ro88] Jung Soon Ro. An Evaluation of the Applicability of Ranking Algorithms to Improve the Effectiveness of Full-Text Retrieval. *Journal of the American Society for Information Science*, 39(2):73–78, 1988.

[RPH$^+$95] Ramana Rao, Jan O. Pedersen, Marti A. Hearst, Jock D. Mackinlay, Stuart K. Card, Larry Masinter, Per-Kristian Halvorsen, and George G Robertson. Rich Interaction in the Digital Library. *Communications of the ACM*, 88(4):29–39, April 1995.

[RS91] Edwina L. Rissland and David B. Skalak. CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 34:839–887, 1991.

[RS95a] Ellen Riloff and Jay Shoen. Automatically Acquiring Conceptual Patterns Without an Annotated Corpus. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 148–161, Boston, MA, July 1995.

[RS95b] Daniela Rus and Devika Subramanian. Customizing Information Capture and Access. *ACM Transactions on Information Systems*, 1995. Submitted.

[RS95c] Daniela Rus and Kristen Summers. Using White Space for Automated Document Structuring. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Digital Libraries: Current Issues. Lecture Notes in Computer Science 916.*, pages 129–162. Springer-Verlag, 1995.

[SA93] Gerard Salton and James Allan. Selective Text Utilization and Text Traversal. In *Hypertext '93*, pages 131–144. ACM, November 1993.

[SAB93] Gerard Salton, James Allan, and Chris Buckley. Approaches to Passage Retrieval in Full Text Information Systems. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58, Pittsburgh, PA, June 1993. ACM.

[Sal89] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[SB90] Gerard Salton and Chris Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[SB91a] Gerard Salton and Chris Buckley. Automatic Text Structuring and Retrieval – Experiments in Automatic Encyclopedia Searching. In *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 21–30, Chicago, IL, October 1991. ACM.

[SB91b] Gerard Salton and Chris Buckley. Global Text Matching for Information Retreival. *Science*, 253:1012–1015, 1991.

[SGM+95] J. C. Smith, Daphne Gelbart, Keith MacCrimmon, Bruce Atherton, John McClean, Michelle Shinehoft, and Lincoln Quintana. Artificial Intelligence and Legal Discourse: The Flexlaw Legal Text Management System. *Artificial Intelligence and Law*, 3(2):55–95, 1995.

[SHH95] David Steier, Scott B. Huffman, and Walter C. Hamscher. Meta-Information for Knowledge Navigation and Retrieval: What's in There. In *Working Notes of the AAAI Fall Symposium Series: AI Applications in Knowledge Navigation and Retrieval*, pages 123–126, Cambridge, MA, November 1995. AAAI.

[SK88] T. Saracevic and P. Kantor. A Study of Information Seeking and Retrieving. *Journal of the American Society for Information Science*, 39(3):187–222, 1988.

[SP94] Hinrich Schütze and Jan O. Pedersen. A Cooccurrence-Based Thesaurus and Applications to Information Retrieval. In *Intelligent Multimedia Information Retrieval Systems and Management, RIAO '94*, pages 266–274, New York, NY, October 1994.

[SW86] Craig Stanfill and David Waltz. Toward Memory-Based Reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.

[SW89] Craig Stanfill and David L. Waltz. Text-Based Intelligent Systems. In Paul S. Jacobs, editor, *Statistical Methods, Artificial Intelligence, and Information Retrieval.* Addison-Wesley, 1989.

[TC91] H. R. Turtle and W. B. Croft. Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

[TC92] H. R. Turtle and W. B. Croft. A Comparison of Text Retrieval Models. *Computer Journal*, 35(3):279–290, 1992.

[Vel92] Manuela M. Veloso. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, August 1992.

[VGJL95] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning Collection Fusion Strategies. In *Proceedings of the 18th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179, Seattle, WA, July 1995. ACM.

[Yan94] Yiming Yang. Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval. In *Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22, Dublin, Ireland, July 1994. ACM.