Indexing and Retrieving Natural Language Using Ternary Expressions

by

Jimmy J. Lin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

Author .....
Department of Electrical Engineering and Computer Science
February 6, 2001

Certified by .....
Boris Katz
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Indexing and Retrieving Natural Language Using Ternary Expressions

by

Jimmy J. Lin

Submitted to the Department of Electrical Engineering and Computer Science
on February 6, 2001, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Traditional information retrieval systems based on the "bag-of-words" paradigm cannot completely capture the semantic content of documents. Yet it is impossible with current technology to build a practical information access system that fully analyzes and understands unrestricted natural language. However, if we avoid the most complex and processing-intensive natural language understanding techniques, we can construct a large-scale information access system which is capable of processing unrestricted text, largely understanding it, and answering natural language queries with high precision. We believe that ternary expressions are the most suitable representational structure for such a system; they are expressive enough for information retrieval purposes, yet amenable to rapid large-scale indexing.

Thesis Supervisor: Boris Katz
Title: Principal Research Scientist

Acknowledgments

Thank you, mom and dad, for your love and unconditional support.

Thank you, Boris, for giving me the dream—that mythical machine with human-level understanding and the infalliable memory of a computer. I hope this brings us one step closer.

Special thanks to Ally, Steen, and Cindy for the important roles they have played in my life.

Special shout goes out to my boyz. Keepin' it real and stickin' with the program.

This document originated from a paper co-authored with Boris Katz and Sue Felshin, submitted to SIGIR2001, under the same title. Thanks to Sue for her helpful suggestions and valuable comments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The explosive growth of information available electronically has given people potential access to more knowledge than they have ever had before. Massive repositories of information are available at everyone's fingertips. (The most prominent example is, of course, the World Wide Web.) However, much of this potential remains unrealized due to the lack of an effective information access method.

Simply stated, it is difficult to find the knowledge that one is looking for. There is an overwhelming amount of information available, and existing search services often do little to reduce the information overload. To further compound the problem, textual information exists in different forms, and is often varied and unorganized.

An ideal intelligent information access system would combine human-level understanding with the infallible memory of a computer. This document presents many ideas and an architecture for bringing us one step closer to that dream.

1.1 Currently Deployed Systems

Most deployed information access systems (e.g., Web search engines such as Google) have been built on the "bag-of-words" assumption, which equates a document's keyword statistics with its semantic content. Typically, a query returns documents that contain words similar to those found in the query itself. These traditional information retrieval systems do not take into account the plethora of relationships contained in

language between various entities in the same sentence or across sentences.

Obviously, a document is much more than the sum of its individual keywords, and matching documents based on keyword content is far from perfect. Although they may offer some indication of "meaning," keywords alone cannot capture the richness and expressiveness of natural language. Consider the following sets of sentences/phrases that have similar word content, but (dramatically) different meanings:

(1) The bird ate the young snake.
(1′) The snake ate the young bird.
(2) the meaning of life [1]
(2′) a meaningful life
(2″) life's meaning
(3) the bank of the river
(3′) the bank near the river
(3″) the river bank
(4) He interfaced the design...
(4′) He designed the interface...
(4″) the design's interface
(4‴) the interface's design

Due to the inability of keywords to capture the "meaning" of documents, a traditional information retrieval system (i.e., one using the "bag-of-words" paradigm) will suffer from poor precision in response to a user query accurately and precisely formulated in natural language.

Consider the question "Where do bears live?" The user is expecting an answer similar to the following:

(5) Brown bears live on Kodiak Island and in other parts of south-central and southeast Alaska.
(6) Polar bears live along the Arctic Coast.

If the user were using a keyword search engine, the system would most likely be looking for the keywords *bear* and *live*.[2] Unfortunately, the following unrelated

[1] Example taken from [26]

[2] We are assuming that the keyword search engine performs morphological stemming (a fairly standard technique).

sentence also contains the same two keywords.

(7) Mammals bear their young live.

Because a traditional keyword search engine does not have any understanding of document content, irrelevant and nonsensical results are often returned to the user.

1.2 Question Answering

We believe that question answering through natural language is the best information access mechanism for humans. It is intuitive, easy to use, rapidly deployable, and requires no specialized training. Unlike information retrieval, which returns a list of potentially relevant documents for the user to peruse, a question answering system returns an information segment (e.g., paragraph, sentence, or even phrase) that directly answers the user's natural language query. For example, the question "Who wrote the Declaration of Independence?" would be answered with "Thomas Jefferson wrote the Declaration of Independence in 1776" instead of a document about Independence Hall in Philadelphia wherein the information is stated. An example of such a system is START [14, 15], the first natural language system available for question answering on the World Wide Web.[3] Other examples of question answering systems include entries to the Text Retrieval Conference (TREC) [40], an annual conference that contains a track[4] to showcase and evaluate the latest question answering technology.

Despite the obvious advantages that question answering offers over traditional information retrieval, a robust information access system based on natural language in an unrestricted domain cannot be realistically expected any time soon. Language is simply too complex and full of ambiguities to be correctly analyzed by a computer system; numerous problems such as common sense implication, intersentential references, and discourse modeling (just to name a few) render full text understanding in

[3] http://www.ai.mit.edu/projects/infolab

[4] The QA Track was begun in TREC-8.

an unrestricted domain out of the reach of current technology. Otherwise, we would simply analyze textual knowledge and derive a knowledge base from it directly.

We believe that by simplifying the sophistication of natural language techniques applied to document analysis, a significant portion of semantic content can be captured while many of the intractable complexities of language can be ignored. Through simplified linguistic analysis, meaning may be distilled into ternary expressions, a simple but powerful representational structure that is amenable to large-scale indexing and retrieval.

The Sapere System is a prototype that implements many of our ideas. By using ternary expressions as the basic unit of indexing, the system can create a more accurate representation of document content (as compared to the "bag-of-words" approach). Sapere offers a modular architecture that will serve as a testbed of various natural language information retrieval techniques. Hopefully, the system will lay the foundation for a next-generation information access system.

1.3 Outline

The rest of this document is organized as follows:

- **Chapter 2** describes ternary expressions, the representational structure that we believe is most suitable for natural language information retrieval.
- **Chapter 3** describes transformational rules as a mechanism for handling linguistic variation.
- **Chapter 4** examines some previous work in natural language process techniques as applied to information retrieval.
- **Chapter 5** details the system architecture of the Sapere System.
- **Chapter 6** reports an evaluation of the Sapere System, comparing it against a traditional boolean keyword information retrieval system.

- **Chapter 7** explores many issues that were raised by the evaluation conducted in the previous chapter.
- **Chapter 8** outlines a roadmap for future developments and enhancements to the Sapere System.
- **Chapter 9** briefly summarizes this document.

Chapter 2

Indexing Ternary Expressions

Although sophisticated linguistic analysis may be performed on natural language text for the purpose of question answering, such techniques are not currently viable on a large scale. Yet, an indexing scheme that uses keywords as its basic unit cannot capture the semantic content of documents. If we avoid the most complex and processing-intensive natural language understanding techniques, and ignore some complexity and ambiguity in language, we can construct a large-scale information access system which is capable of processing unrestricted text, understanding it well enough for purposes of information retrieval, and answering natural language queries with high precision. A variety of representational structures ranging in linguistic sophistication have been attempted (to be discussed in Chapter 4), but all with limited success. We believe that ternary expressions are currently the most suitable solution.

Ternary (three-place) expressions [17, 18, 14] capture the syntactic relationships between various entities in text. They may be intuitively viewed as subject-relation-object triples, and can easily express many types of relations, e.g., subject-verb-object relations, possession relations, etc.[1] From a syntactic point of view, ternary expressions may be viewed as typed binary relations. Given the binary branching hypothesis of linguistic theory, ternary expressions are theoretically capable of expressing any

[1]Although most relationships between elements of a sentence involve two distinct entities, there are some some which do not, e.g., intransitive verbs. A sentence like "he slept" would be represented by a ternary expression with a null element, i.e., [he sleep -].

arbitrary tree—thus, they are compatible with linguistic theory. From a semantic point of view, ternary expressions may be viewed as two-place predicates, and can be manipulated using predicate logic. Finally, ternary expressions are highly amenable to rapid large-scale indexing, which is a necessary requisite of information retrieval systems.

Although other representational structures (e.g., trees or case frames) may be better adapted for some purposes, they are much more difficult to index and retrieve efficiently due to their size and complexity. Therefore, such representational structures are not viable for current information retrieval systems. Ternary expressions appear to be the most suitable solution. (Chapter 4 provides an in depth discussion of previous work.)

Since ternary expressions are merely three-place relations, they may be indexed and retrieved much in the same way as rows within the table of a relational database.[2] Relational databases are designed to handle large amounts of data, thus offering a scalable solution in terms of performance with minimal development effort. Furthermore, most operations with ternary expressions can be formulated as SQL queries (or sequences thereof). Thus, relational algebra can serve as the underlying foundation for relations indexing.

By analyzing and extracting relations between entities in natural language as ternary expressions, an information access system can capture the differences in meaning between the pairs of sentences and phrases given in Chapter 1.

(1) The bird ate the young snake.
< bird eat snake >
< young `mod` snake >

(1′) The snake ate the young bird.
< snake eat bird >
< young `mod` bird >

(2) the meaning of life
< meaning `possessive-relation` life >

[2]In fact, Sapere uses a SQL database to handle the indexing of relations (see Chapter 5).

(2′) a meaningful life
< meaningful `describes` life >

(2″) life's meaning
< meaning `possessive-relation` life >

(3) the bank of the river
< bank `possessive-relation` river >

(3′) the bank near the river
< bank `near-relation` river >

(3′) the river bank
< river `mod` bank > [3]

(4) He interfaced the design...
< he interface design >

(4′) He designed the interface...
< he design interface >

(4″) the design's interface
< interface `possessive-relation` design >

(4‴) the interface's design
< design `possessive-relation` interface >

The ability to extract subject-verb-object relations, e.g., (1), (1′), (4), and (4′), allows an information access system to distinguish between two very different statements. Similarly, a ternary expressions indexer can differentiate between prepositional phrases (2), adjectival modification (2′), and compounds (2″). Although the system does not have any notion of semantics (e.g., word sense), syntax may offer crucial clues to meaning in cases such as (3), which probably refers to both side of the river, and (3′), which probably refers to the financial institution in proximity to the river. Because "design" and "interface" can be both nouns and verbs, phrases containing those pairs of words can have very different meanings, i.e., (4) through (4‴).

[3] "The river bank" may also be interpreted as a lexical atom, i.e., `river_bank`.

Chapter 3

Transformational Rules

While a syntactically informed representational structure improves the precision of an information retrieval system due to its ability to identify the relationship between entities within a sentence, such representational structures face the problem of *linguistic variation*, the phenomenon in which similar semantic content may be expressed in different surface forms. Simply stated, there are many ways to say the same thing. As a result, recall may be negatively affected unless different variants that have the same meaning can be explicitly equated; a possible solution is to normalize different constructions into the same representational structure.

Consider the following sets of sentences that express the same meaning using different constructions:

(8) What is the capital of Taiwan?
(9) What's the capital city of Taiwan?
(10) What is Taiwan's capital?

Linguistic variations can occur at all levels of language; the examples above demonstrate lexical, morphological, and syntactic variations. Linguistic variations may sometimes be quite complicated, as in the following example, which demonstrates verb argument alternation.[1]

[1]Beth Levin [21] offers an excellent treatment on English verb classes and verb argument alternations.

(11) Whose declaration of guilt shocked the country?
(12) Who shocked the country with his declaration of guilt?

Transformational rules[2] provide a mechanism to explicitly equate alternate realizations of the same meaning at the level of ternary expressions.

As an example, Figure 3-1 shows a sample transformational rule for handling (11) and (12).[3] Thus, through application of this rule, question (11) can be equated with question (12).

[n_1 shock n_2]		[n_3 shock n_2]
[shock with n_3]	$\longleftrightarrow$	
[n_3 related-to n_1]		[n_3 related-to n_1]
where $n \in$ *Nouns*		where $n \in$ *Nouns*

Figure 3-1: Sample Transformational Rule

The relevant instantiation of the rule in this example is

[who shock country]		[declaration shock country]
[shock with declaration]	$\longleftrightarrow$	
[declaration related-to who]		[declaration related-to who]

A general observation about English verbs is their division into "classes," where verbs in the same class undergo the same alternations. For example, the verbs 'shock', 'surprise', 'excite', etc., participate in the alternation shown in (11) and (12) not by coincidence, but because they share certain semantic qualities. Although the transformational rule required to handle this alternation is very specific (in that it applies to a very specific pattern of ternary expression structure), the rule can nevertheless be generalized over all verbs in the same class by associating with the rule conditions that must be met for the rule to fire, i.e., *verb* $\in$ *emotional-reaction-verbs* $= \{$*shock, surprise, excite* ...$\}$; see Figure 3-2.

[2]Borrowed from START's "S-rules" [16, 14, 15].

[3]This rule is bidirectional in the sense that each side of the rule implies the other side. The rule is actually used in only one direction, so that we canonicalize the representation.

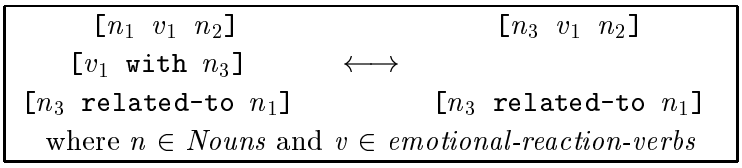

Figure 3-2: Sample Transformational Rule

Note that transformational rules can also encode semantic knowledge and even elements of common sense. For example, a rule can be written that equates a *selling* action with a *buying* action (with verb arguments in different positions). Or as another example, rules can even encode implicatures:[4]

$$[n_1\ v_1\ n_2] \ \rightarrow\ [n_1\ v_2\ n_2]$$
$$\text{where } n_1, n_2 \in Nouns,\ v_1 \in \{\text{kill}, \text{murder}, \ldots\}, v_2 \in \{\text{hurt}, \text{harm}, \ldots\}$$

Transformational rules can apply at the syntactic, semantic, or even pragmatic levels, and offer a expressive framework for handling linguistic variation.

In order for a question answering system to be successful and have adequate linguistic coverage, it must have a large number of these rules. A lexicon which classified verbs by argument alternation patterns would be a good start, but this is another resource lacking in the world today. Rules generally may be quite complex, and thus an efficient mechanism by which to create these rules is an issue that requires further research. An implementation of transformational rules is discussed in Chapter 5.

[4]Note the directionality of the rule.

Chapter 4

Previous Work

The concept of indexing more than simple keywords is not new. This chapter describes some of the previous attempts at applying natural language processing techniques to information retrieval.

4.1 Early Work

Nearly two decades ago, Katz [17, 18, 14] proposed a mechanism for indexing and retrieving natural language using semantic relation nets and relations. The system was capable of understanding fragments of text and retrieving analogous text segments stored within its knowledge base.

The idea of indexing (parts of) phrases is more than a decade old; Fagan [9] experimented with indexing two different types of phrases: syntactic phrases, which derived from natural language analysis, and non-syntactic phrases, which derived from statistical properties such as frequency, proximity, and co-occurrence of keywords. Syntactic phrases consisted of word pairs that formed phrase descriptors, which were generated by analyzing the parse output of the PLNLP System [11, 13], a programmable natural language text analysis tool.

As an example, the phrase "use of an automatic text analyzer in preparation of sdi profiles" generated the following phrase descriptors:

[automatic analyzers]
[text analyzers]
[preparation analyzer]
[profiles preparation]
[sdi profiles]

Fagan's work focused primarily on noun phrases and their attached prepositional phrases, and he described various techniques for handling language phenomena such as conjunction, disjunction, hyphenated forms, etc.

Experiments were performed by comparing the effects of indexing syntactic and non-syntactic phrases (pairs of words created using statistical techniques) in addition to normal single word terms in a vector space model. In order to convert phrase descriptors into a standard vector space model description, both words in the pair were morphologically stemmed and concatenated into a single lexical item. Thus, the above example would derive the following elements for indexing:[1]

automat analyz
text analyz
prepar analyz
profil prepar
sdi profil

Fagan's experiments showed that indexing non-syntactic phrases actually resulted in better precision improvements over the baseline (single term index) than indexing syntactic phrases. His work seemed to suggest that natural language processing techniques were not only slower, but also less effective than pure statistical techniques.

A limitation in Fagan's study arises from the conversion of phrase descriptors into atomic indexing terms, by simple morphological stemming and lexical concatenation. Although this technique simplifies the indexing process, it disregards the fundamental relationship between the component words in the phrase descriptor (e.g., adjective-noun, conjunction, etc.) This simplification limits the types of queries that may be formed. For example, treating adjective/noun pairs ([*adj.*, *noun*]) as lexical atoms

[1]Fagan's exact stemming algorithm was not described, and thus these lexical items were derived from similar examples.

renders it impossible to find the equivalent of "all *big* things," corresponding to the pair [big, *].

4.2 More Recent Work

More recently, several researchers have explored the effects of indexing word pairs derived from linguistic analysis.

Strazlkowski, et. al. [33] described an approach to applying natural language processing techniques to information retrieval that involved extracting and indexing head-modifier pairs. These pairs were generated using the Tagged Text Parser, a text processing system based on the Linguistic String Grammar [34]. Types of pairs that were considered included a head noun and its associated adjectives, a head noun and the head noun of its right adjunct, the main verb of a clause and head of its object and subject. These pairs were then merged with indexing terms generated using other techniques (mostly statistical). Although the system only reported "modest gains" [33] in precision, it demonstrated the viability of applying linguistic analysis to information retrieval.

Other authors [43, 2, 3] have also experimented with indexing word pairs that derive from head-modifier relationships. The performance improvements have been neither negligible nor dramatic, but despite the lack of any significant breakthroughs, the authors affirmed the potential value of linguistically-motivated indexing schemes and the advantages they may offer over traditional IR.

We believe that pairs are not expressive enough to capture the semantic content of natural language text for question answering purposes. For example, (3), (3′), and (3″), would all derive the same pair [bank river], despite differences in meaning. As another example, the pair [design interface] could either represent a noun-noun modification pair, a verb-object pair, or a subject-verb pair (see (4) through (4‴)).

(repeated from Chapter 1)
(3) the bank of the river
(3′) the bank near the river

(4) He interfaced the design...
(4′) He designed the interface...
(4″) the design's interface
(4‴) the interface's design

By typing pairs (i.e., ternary expressions), we can create a representational structures more suitable for capturing document content for information access.

4.3 More Sophisticated Indexing Schemes

The sophistication of linguistic techniques employed to analyze text for information retrieval purposes range from relatively simple strategies, such as head-modifier pairs, to much more complicated schemes, such as tree structures and case frames.

Indexing linguistic tree structures has been attempted [31], with very disappointing results: precision actually decreased due to the inability to handle variations in tree structure (i.e., the same semantic content could be expressed using different syntactic structures),[2] and to the poor quality of the full-text natural language parser, which was also rather slow. The full-text natural language parser Smeaton used was relatively relatively error-prone; indexing incorrect parse trees is a source of performance degradation.[3] Furthermore, matching trees and sub-trees is a computationally intensive task, especially since full linguistic parse trees may be relatively deep. Relations are easier to match because they are typically much simpler than parse trees. For example, the tree

[[shiny happy people] [of [Wonderland]]]

may be "flattened" into three relations:

[2]This issue is addressed by our use of transformational rules, which may offer a potential solution.
[3]Parsing technology has advanced in the last several years. It would be interesting to apply the same approach, but with a faster, more accurate, and more robust parser available today.

< shiny `describes` people >
< happy `describes` people >
< people `related-to` Wonderland >

Indexing case frames has also been attempted [7, 26], but with limited success. Full semantic analysis is still an open research problem, especially in the general domain. Since such analysis cannot be performed without full-text parsing, case frame analysis inherits the unreliability of current parsers. Furthermore, semantic analysis requires extensive knowledge in the lexicon, which is extremely time-intensive to construct. Finally, due to the complex structure of case frames, they are more difficult to store and retrieve than ternary expressions.

4.4 Indexing Relations

There have also been previous attempts at applying various kinds of relations to question answering.

The MURAX System [20] performs question answering using an online encyclopedia. It utilizes a boolean proximity-based keyword based approach guided by heuristics for narrowing and broadening the search parameters. A limited and small number of fixed phrase types are recognized in order to extract or verify hypothetical answers, e.g., IS-A relations, appositives, lists. In an informal evaluation of the system on seventy "Trivial Pursuit" questions, the first result returned by MURAX was correct 53% of the time, and the correct answer lies in the top five results 74% of the time.

Litkowski [25] described a question answering system for TREC-8 that utilized *semantic relation triples*; the focus of the work was on extracting semantic relations (rather than syntactic relations), which may involve additional computational lexicons. Because full semantic analysis is still currently intractable, the system settled on syntactic relations such as subject and object, and a small number of fixed, easily identifiable semantic relations (e.g., time and location). Information extraction techniques were used to identify many types of relations. The system received a mean reciprocal rank (see [40] for the methodology of the scoring system) of 0.281, which

was slightly below the average score for all TREC participants (0.331). Litkowski's system, while primarily focused toward use in TREC and as a lexicographer's workbench, seems similar to the work described here. It could be instructive to compare them on similar corpora and similar tasks.

The START System [14, 15] parses language and indexes it with ternary expressions in order to provide question answering. (In fact, the system was an inspiration for the work described here.) START performs a full analysis of each sentence it processes; since it cannot fully parse unrestricted text, it restricts itself to parsing queries and "annotations," which are simple, computer-parsable sentences and phrases that describe more complex text (or non-text content) which START cannot parse. Annotations, representative of the contents of the corpus, are then indexed to form a knowledge base. These annotations must be manually generated, and START is therefore not able to index very large corpora of unrestricted text. Nevertheless START's robust capabilities within its domains of knowledge demonstrate the usefulness of the ternary expression representation, and more generally, of natural language question answering.

Chapter 5

System Architecture

The Sapere System[1] provides the framework of a "smart" information access system that implements many of the ideas presented in earlier chapters. The system was designed with extensibility and flexibility in mind, with the hope that it will serve as a playground and testbed for future experiments in natural language processing techniques as applied to question answering.

The overall architecture of the Sapere System is shown in Figure 5-1. Sapere itself is written as a Perl module with a API (Application Programming Interface). The external dependencies of the system are Minipar, the parser that creates the relations, and mySQL, which handles the storage and indexing of relations. Sapere accesses the external dependencies through simple, well-defined interfaces, thus forming a modular, swappable architecture. Similarly, Sapere provides an interface through which question answering can actually be performed, e.g., though a form on a website.

Abstractly, Sapere can be viewed as a repository of information segments. The system indexes segments that are tagged with natural language annotations, similar to the model used by START [14, 15]. An information segment can be anything: text, images, video, even multimedia content. The annotations are natural language sentences that describe the information segments, and the questions they are capable of answering; these annotations are analyzed and converted into ternary expressions,

[1] *Sapere*, the Latin verb "to be wise."

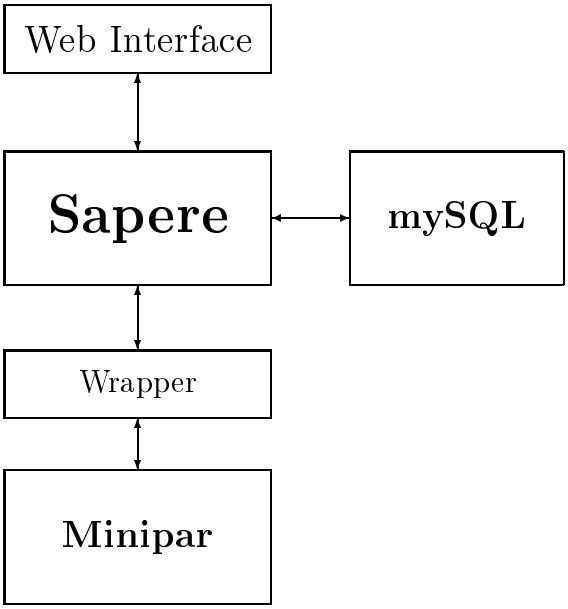

Figure 5-1: Architecture of Sapere

which are then indexed. Currently, the information segments and the annotations are exactly the same;[2] this system design leads to greater flexibility and future extensibility when non-textual segments are considered. One can imagine, far off into the future, mechanisms that will generate annotations automatically from non-textual sources, e.g., speech-recognition technology for spoken text, object-recognition technology for images, etc.

5.1 Generating Relations

The Sapere System employs the Minipar parser to generate relations from the corpus text. Minipar[3] is a principle-based minimalist parser. A principle is essentially a constraint over X-bar structure, and offers many advantages over rule-based and unification-based grammars [23, 22, 24]. A major advantage of Minipar is its sim-

[2]This is not exactly true. The segments contain some HTML tags so that they can be displayed properly.

[3]Minipar can be downloaded at `http://www.cs.alberta.ca/~lindek/`

plicity, which translates into fast parsing speeds; the system can achieve respectable speed performance without sacrificing coverage and accuracy. Minipar can handle a wide range of phenomena in English, such as passive voice, raising, super-raising, exceptional case marking, ditransitive verbs, displacement of expletives, *wh*-movement, *that*-trace, *wh*-island [24].

Within the Sapere System, the parse tree output of Minipar is postprocessed by a custom-built wrapper in order to convert it into ternary expressions. The wrapper traverses the parse tree and performs matching on local tree fragments. If a known fragment is recognized, it is converted into the equivalent ternary expression(s).

As an example, the Minipar output to the following sentence can be seen in Figure 5-2:

The big bad wolf prowled around the dark forest.

The above sentence derives the following relations:

[wolf prowl around]
[prowl around forest]
[big `mod` wolf]
[bad `mod` wolf]
[dark `mod` forest]

Currently, Sapere indexes the following types of relations: subject-verb-object, adjective-noun modification, noun-noun modification, possessive relations, predicate nominative, predicate adjectives, appositives, prepositional phrases, relative clauses.[4]

Although Minipar is currently the most suitable parser for extracting relations from natural language for question answering purposes, Sapere is not tied down to

[4]Under the current implementation, the system cannot cannot handle a sentence like "The president of Russia visited the president of China," because it would derive the following relations: [president of China], [president of Russia], [president visit president]. Because the system cannot distinguish between the two different presidents, we cannot determine who actually was visiting the other. A solution to this problem is to add indices to the nouns, e.g., `president-1` and `president-2`, but this has not been implemented yet.

```
(
E0      (()     fin C   *       )
1       (The    ~ Det   4       det     (gov wolf))
2       (big    ~ A     4       mod     (gov wolf))
3       (bad    ~ A     4       mod     (gov wolf))
4       (wolf   ~ N     5       s       (gov prowl))
5       (prowled        prowl V_N       E0      i       (gov fin))
E2      (()     wolf N  5       subj    (gov prowl)     (antecedent 4))
6       (around ~ Prep  5       mod     (gov prowl))
7       (the    ~ Det   9       det     (gov forest))
8       (dark   ~ A     9       mod     (gov forest))
9       (forest ~ N     6       pcomp-n (gov around))
)

[fin-E0 C:i:V_N prowl-5]
[prowl-5 V_N:s:N wolf-4]
[wolf-4 N:det:Det the-1]
[wolf-4 N:mod:A big-2]
[wolf-4 N:mod:A bad-3]
[prowl-5 V_N:subj:N wolf-E2]
[prowl-5 V_N:mod:Prep around-6]
[around-6 Prep:pcomp-n:N forest-9]
[forest-9 N:det:Det the-7]
[forest-9 N:mod:A dark-8]
```

Figure 5-2: Sample Minipar parse output
Minipar can output the parse tree in two different formats: as a linearized graph structure (above), and as a collection of triples (below).

any particular parser. Minipar can be substituted by any other text analysis system, provided that a similar wrapper is written.[5]

5.2 Indexing Ternary Expressions

Ternary expressions extracted from the text are indexed using mySQL[6], a popular Open Source RDBMS (relational database management system). Ternary expressions are three-place expressions, and hence they fit neatly into a relational model of data; they can be manipulated using SQL queries. Most operations on ternary expressions can be neatly translated into SQL.

The use of a RDBMS for indexing and storing relations vastly simplifies development efforts; relational databases are designed to handle vast amounts of data, and intelligent use of indices provides nearly instant access to any ternary expression stored within the database. Although a custom-built relations indexer would perhaps be faster, such a system would be costly to develop, and would probably lack the robustness of a production quality software package. The use of SQL databases provides a simple, fast, and scalable solution to the indexing and retrieval problem.

Sapere interfaces mySQL through the Perl DBI/DBD modules. This connection method is an industry standard, and drivers are available for most popular databases on the market: Oracle, DB2, Postgres, etc. Although mySQL is a lightweight database system, another RDBMS can be easily substituted should mySQL buckle under the larger loads involved with a larger corpus.

5.3 Matching Ternary Expressions

The ternary expressions matcher is the component of the Sapere System directly responsible for generating answers to user queries. This module directly interfaces with the front-end of the information access system (usually a Web form).

[5] Because every parser displays its parse trees in a different format, the wrapper must be rewritten.

[6] `http://wwe.mysql.com`

Ternary expressions are derived from the query in the same manner that corpus text is processed (i.e., using Minipar, mediated through the wrapper). *Wh*-words in the questions are substituted with ∗, which indicates a wildcard that matches any word, e.g., "Who loves Mary" derives

[∗ love Mary]

For some types of question, e.g., "what" questions, the use of the wildcard may produce a short, exact answer. If a ternary expression matching the above one is found in knowledge, the element that binds to ∗ (i.e., in the corresponding position), precisely answers the question.

Currently, the matcher employs a rather simplistic algorithm; information segments (sentences) are scored by the number of relations that they share with the query. Thus, a score of two is given to a sentence that has two ternary expressions in common with the query. All segments with a score greater than one are returned in a query result object; the front-end formats the output and displays the top results.

Obviously, the current matching algorithm is inadequate for many purposes (detailed in Chapter 8). Keeping in line with the philosophy of the system, the algorithm can be replaced with a more advanced version when it is written (Chapter 8 also offers a roadmap for future work on the system).

5.4 Transformational Rules

The Sapere System also implements a module for transformational rules. The system is essentially a forward chaining rule-based system, in which the antecedents and consequents of the rules are both ternary expressions.

The rule-based system attempts to unify relations[7] generated from text with the antecedents of the transformational rules. If unification is successful, the pattern of

[7]For a brief introduction to logic, unification, and rule-based systems, see [42].

ternary expressions indicated in the consequent is generated, with the proper bindings instantiated.

Transformational rules in Sapere can either be applied at the time of indexing or at the time of retrieval. If index-time rule application is chosen, all variations of the corpus text generated by the transformational rules will also be inserted into the index. This slows down the indexing process and results in a larger index size, but speeds up matching at retrieval time. If rule application is performed at retrieval time, the transformational rules are applied to the user query, which is then broadened to encompass all the variations generated by the rules; this reduces the index size at the expense of increasing retrieval time. Most likely, Sapere should utilize a combination of the two methods; frequently occurring transformations may benefit from index-time rule application, and index size can be reduced by applying infrequently encountered transformations at the time of retrieval.

In addition to the actual machinery for transformations, Sapere also provides a intuitive interface for adding new rules. Transformational rules are written in English, and are automatically compiled into patterns of ternary expressions. For example, a "shock" rule (see Figure 3-1) can be written as:

`X` shocked `Y` with `X`'s `Z` $\longleftrightarrow$ `X`'s `Z` shocked `Y`

Sapere would compile the above pseudo-English definition into patterns of ternary expressions with the proper free variables. The user has the option of reviewing the ternary expressions generated to check their validity, and can manually edit the rules if needed. Currently, there is no mechanism to attach conditions onto the rules (i.e., to handle verb classes).

5.5 Creating a Testbed

Although Sapere intends to demonstrate the potential improvements over current systems that may result from indexing ternary expressions, its primary purpose is to establish a extensible framework onto which future improvements can be grafted. We

hope that the system will function as a playground for testing new techniques aimed at building better information access systems. A roadmap for future developments is outlined in Chapter 8.

Chapter 6

System Evaluation

This chapter evaluates and compares two information access systems: Sapere, a prototype relations indexer, and a baseline keywords indexer. The experimental setup is described and the results are presented.

6.1 The Test Systems

A more detailed description of the two test systems is as follows:

- **Relations Indexer and Matcher.**[1]

  Every sentence in the corpus is parsed and the relations that derive from the parse trees are indexed. The matcher extracts relations from the user query in the same manner; sentences in the corpus are scored on the number of relations that they share with the query. The top scoring sentences within the corpus are returned to the user as the answers. Although the system implements a functional module for transformational rules, it was not utilized in the evaluation.

- **Keywords Indexer and Matcher.**

  The keywords indexer derives content words from the corpus via tokenization, stopword filtering, and morphological stemming. A standard inverted index is

[1] This is the Sapere System as described in the previous chapter.

built from these content words (keywords), which serves as the data structure used in matching and retrieval.

The matcher likewise extracts query terms from the user question through stop-word filtering and morphological stemming. The engine then attempts to match keywords in the query (conjoined with a logical `AND`) with keywords contained in the inverted index. Sentences are ranked in order of the number of keywords that they share with the query. The top scoring sentences are returned as the answer.

6.2 The Worldbook Corpus

The test corpus used for the experiments was an electronic version of the Worldbook Encyclopedia. Because the raw articles were originally packaged with SGML tags, significant preprocessing was required to prepare the corpus for indexing.

The encyclopedia contained approximately 20,000 articles that varied greatly in length, from a few sentences to several pages. The entire collection contained nearly six hundred thousand sentences and over four million words, which totaled approximately 50 megabytes (with all SGML tags removed).

6.3 The Test Run

The twenty questions listed below were submitted to both systems. These questions attempted to reflect typical questions that real users seeking knowledge would ask.

What countries has Japan invaded?
What do frogs eat?
What eats snakes?
What countries have invaded Russia?
What does Japan import?
When do lions hunt?
What is the capital of the Netherlands?
When did the Crusades start?

What is the capital of Russia?
Who founded Boston?
Who invented the electric light?
What are antibodies?
What does bonsai mean?
What is the largest country in the world?
What is the silk road?
What is interferon?
What is the largest planet?
Who defeated the Spanish Armada?
Where do ants live?
What started the civil rights movement?

The primary purpose of these tests was not to serve as a formal evaluation of the systems, but to highlight the advantages of indexing and retrieving text based on relations.

6.4 Results

The comparison between the keywords indexer and relations indexer can be seen in Table 6.1.

The keywords indexer achieved a precision of only 27.7%, while the relations indexer improved that precision figure to 85.1%. (This enormous rise in precision is due in part to the test corpus, which was designed to highlight the advantages of indexing relations—see Chapter 7 for more detail.)

In addition to achieving higher precision, the relations-based system returned far fewer results on average compared to the keyword-based system (4 results per query *vs.* 47.6 results per query). Although the keywords search engine was less precise, it returned a larger number of correct answers (per query) than the relations search engine (5.75 correct answers per query *vs.* 3.2 correct answers per query). A more in-depth discussion of the results will be presented in the next chapter.

	Relations			Keywords		
Question	n	i	p	n	i	p
What countries has Japan invaded?	5	5	1	241	7	0.029
What do frogs eat?	1	1	1	32	1	0.031
What eats snakes?	11	10	0.909	40	17	0.425
What countries have invaded Russia?	8	8	1	181	9	0.050
What does Japan import?	3	2	0.666	27	8	0.296
When do lions hunt?	4	2	0.5	17	2	0.118
What is the capital of the Netherlands?	2	2	1	6	3	0.5
When did the Crusades start?	1	1	1	5	1	0.2
What is the capital of Russia?	1	1	1	15	11	0.733
Who founded Boston?	5	4	0.8	22	4	0.182
Who invented the electric light?	1	1	1	2	1	0.5
What are antibodies?	10	9	0.9	211	9	0.043
What does bonsai mean?	1	1	1	11	1	0.091
What is the largest country in the world?	4	2	0.5	32	5	0.156
What is the silk road?	2	2	1	15	10	0.666
What is interferon?	1	1	1	12	1	0.083
What is the largest planet?	2	1	0.5	40	7	0.175
Who defeated the Spanish Armada?	1	1	1	3	1	0.333
Where do ants live?	14	8	0.571	35	15	0.428
What started the civil rights movement?	1	1	1	4	2	0.5
Average	4	3.2	0.851	47.6	5.75	0.277

Table 6.1: Comparison between relations and keyword indexing
n is the total number of sentences returned. i is the number of sentences that correctly answer the question. p is the precision.

6.5 Sample Output

The output from the keywords indexer in response to the question "what do frogs eat" is shown in Figure 6-2; the output for the same question from the relations indexer is shown in Figure 6-1.

After removing stopwords from the query, the keyword search engine returned 32 results that contain the keywords *frog* and *eat*. Of all the sentences returned, only (K13) and (K14) correctly answers the user query; the other results answer the question "What eats frogs?" (Apparently, our poor frog has more predators than prey.) A bag-of-words approach fundamentally cannot differentiate between a query in which the frog is the predator and a query in which the frog is the prey.

The relations search engine, because it correctly identifies the relation between "frog" and "eat" within the query, returns only the correct result (R1).[2] Using relations, it was able to filter out sentences that contain the same keywords, but in different relations. This ability to better capture the meaning of English is the reason that indexing relations leads to higher precision.

Although an information access system that indexes relations is able to obtain higher precision than a system that utilize keywords, the recall of such a system may be lower because answers to a user query may be phrased with a construction different from the one used in the query itself. This problem of linguistic variation is reflected in the fewer correct results returned by the relations indexer, when compared to the keywords indexer. The following chapter explores this issue further.

[2](K13) was not returned due to complications that arise from embedding. Compare the following constructions: "enabling the frog to eat," "allowing the frog to eat," and "forbidding the frog to eat." They all have the same syntactic structure, but have different meaning.

Question: What do frogs eat?
(R1) **Frog** Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders.

Figure 6-1: Relations search results for "What do frogs eat?"

Question: What do frogs eat?
(K1) **Alligator** Alligators eat many kinds of small animals on or near the water, including fish, snakes, frogs, turtles, small mammals, and birds.
(K2) **Bat** Some bats catch fish with their claws, and a few species eat lizards, rodents, small birds, tree frogs, and other bats.
(K3) **Bowfin** Bowfins eat mainly other fish, frogs, and crayfish.
(K4) **Civet** Most civets eat birds, frogs, insects, rodents, and small reptiles.
(K5) **Cobra** Most cobras eat many kinds of animals, such as frogs, fishes, and small mammals.
(K6) **Copperhead** Sometimes the snake eats insects and frogs.
(K7) **Crane** Cranes eat a variety of foods, including frogs, insects, snails, and plants.
(K8) **Tiger snake** The snakes eat frogs and a variety of small mammals.
(K9) **Electric eel** The electric eel eats frogs and smaller fish.
(K10) **Fer-de-lance** Young snakes eat lizards and frogs, and adults feed on small mammals.
(K11) **Fishing** Freshwater fish eat worms, insects, and frogs.
(K12) **Frog** Some tadpoles eat frog eggs and other tadpoles.
(K13) **Frog** The digestive system changes, enabling the frog that develops to eat live animals.
(K14) **Frog** Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders.
(K15) **Frog** In the United States, people mainly eat the legs of bullfrogs and green frogs.
(K16) **Garter snake** They catch and eat other cold-blooded animals, such as frogs and salamanders.
(K17) **Ibis** They eat various small animals, including crayfish, fish, frogs, and insects.
(K18) **Kingfisher** Kingfishers also eat crayfishes, frogs, tadpoles, salamanders, and insects.
(K19) **Kite** Most kites eat insects, frogs, lizards, snakes, and small mammals.
(K20) **Kookaburra** Kookaburras eat caterpillars, fish, frogs, insects, small mammals, snakes, worms, and even small birds.
(K21) **Lungfish** Lungfish eat mainly small fish and other water animals, such as frogs and snails.
(K22) **Mantid** They even seize and eat small tree frogs.
(K23) **Marabou** They also eat live prey, including frogs, fish, reptiles, and locusts.
(K24) **Otter** They also eat clams, frogs, insects, snails, snakes, and, occasionally, waterfowl.
(K25) **Peacock** These birds eat snails, frogs, and insects, as well as grain, juicy grasses, and bulbs.
(K26) **Racer** They also eat insects, frogs, small mammals such as mice, and other snakes.
(K27) **Snake** Most snakes eat birds, fish, frogs, lizards, and such small mammals as rabbits.
(K28) **Spider** Although spiders feed mostly on insects, some spiders eat small frogs, fish, and mice.
(K29) **Spider** Enemies of spiders include snakes, frogs, lizards, birds, and other animals that also eat insects.
(K30) **Tamarin** Tamarins eat fruit, insects, frogs, and tree gums.
(K31) **Turtle** Snapping turtles eat small water animals, such as fish, frogs, insects, and snails.
(K32) **Weasel** Weasels also eat earthworms, insects, frogs, lizards, rabbits, snakes, and birds.

Figure 6-2: Keyword search results for "What do frogs eat?"

Chapter 7

Discussion

One reason that we were able to obtain such dramatic improvements over standard keyword indexing schemes is that many of the sample test questions were designed to highlight the advantages of indexing relations. However, we believe that this does not negate the impact or significance of our research.[1] This chapter examines the results of our evaluation in greater detail.

7.1 Precision and Recall

The focus of our work has been on improving precision, possibly at the expense of recall. Due to the problem of linguistic variation, if the answer to a user query is stated using a different construction than the one used in the query, the answer may not be properly retrieved. Because ternary expressions represent document content at the syntactic level, semantically equivalent sentences differing in syntax will be considered different by the matcher. For example, the following is the only sentence in the Worldbook Encyclopedia that answers the question "What do ravens eat?"

(13) Ravens feed on insects, worms, young birds, frogs, and other small animals.

[1] We are interested in applying our techniques to the question answering track of the TREC Conferences in order to compare our system with other approaches against a standard corpus.

Since the relations derived from the above sentence differ from the relations derived from the question, this sentence will not be retrieved, and hence the question will not be correctly answered.

This problem may be alleviated with transformational rules (as described in Chapter 3). By encoding a specific rule that equates the "eats" construction with the "feeds on" construction (Figure 7-1), a relations indexer will be able to retrieve the correct answer even if the original question were formulated differently. However, these transformational rules must be coded manually; a better method to construct the resources required for such a task is an issue that requires further research.

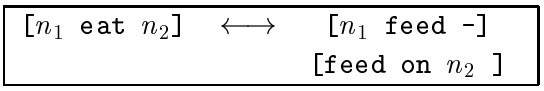

Figure 7-1: Transformational Rule for "eats"

Although retrieval based on relations may inadvertently omit some correct answers (e.g., if the answer is stated using a different construction), we believe that the increased precision is worth the tradeoff. In our sample tests, the keyword search engine did retrieve more correct answers than the relations search engine; however, those additional answers did not offer any new knowledge, but mostly repeated information already present or presented irrelevant facts.

For example, the following is the answer returned by the relations search engine in response to the question "What was the Silk Road?"

> (14) Silk Road was a group of ancient trade routes that connected China and Europe.

The keyword search engine provides additional results, some of which are listed below:

> (15) Along the most famous route, the Great Silk Road, Chinese silk and other products flowed into the Roman Empire.

(16) Kushan emperors opened and protected the Silk Road, a major trade route for caravans carrying silk and other luxury goods from China to India and the Middle East.
(17) At that time, the city was an important stop on the Silk Road, a caravan route to the West.
(18) Marco Polo and other traders traveled the Silk Road, carrying Chinese goods to Europe and western Asia.
(19) One such early system of roads was the Old Silk Trade Route which ran over 6,000 miles, connecting China with Rome and pre-Christian Europe.
(20) The northern route, called the Silk Road, cut from China across central Asia either to the east coast of the Mediterranean Sea or to the Black Sea and from there to Byzantium.
(21) Xinjiang was on the Silk Road, an old trade route that connected the Middle East and Europe with China.

Additional answers returned by the keyword search engine in this example provided extra information that was not directly relevant to the user query.

In general, an information access system that utilizes relations must carefully balance the linguistic sophistication of its document analysis techniques with their robustness. Design decisions may greatly affect the precision and recall measures, as well as system performance in terms of speed. For example, the precision of a system may be further increased by extracting and indexing more (linguistically) complex relations. However, such a strategy carried too far may actually decrease the performance (see Chapter 4), both in speed (from the additional indexing and retrieval overhead), and accuracy (from the failure to correctly identify overly complex relations). The optimal point of balance is an issue that warrants further study.

7.2 Parser Error

Although applying natural language processing techniques to information retrieval promises to improve the quality of a question answering system, such systems are often very brittle due to parser errors.

Because sentences in the corpus may be quite long and complex, the relations extracted from them may be incorrect. There are two sources of error in the relations

building process: the parser may have incorrectly constructed the parse tree, or the parser wrapper may have incorrectly converted the parse tree into relations. Both false positive errors (non-existent relations that are returned) and false negative errors (relations in the text that are missed) are encountered. Due to the complexities of natural language, these errors are unavoidable.[2]

Unfortunately, a relations-based approach to question answering depends critically on the quality of the relations. If the parser failed to correctly identify the relations (or identifies the wrong ones), the matching algorithm can easily be lead astray. As a result, both precision and recall could potentially suffer.

The situation can be alleviated by utilizing parsers that return partial analysis of text. When such a parser encounters a complex sentence that cannot be parsed, it will return a partial parse tree that may be incomplete. For the most part, these partial parse trees are relatively accurate because they represent self-contained linguistic units (e.g., a noun phrase). Thus, some ternary expressions may still be derived from the text even if the entire sentence cannot be analyzed.

Also, to anticipate future developments in natural language processing technology, the ternary expression extraction module in the Sapere System is a self-contained and well-abstracted component. It may be easily replaced with another module when more advanced parsers become available.

7.3 Question Types and the Benefits of Relations

As the corpus size increases, the benefits of indexing relations over indexing keywords will become more pronounced. A relations indexer effectively filters out sentences that contain the correct keywords, but in the wrong type of relation, whereas normal information retrieval techniques perform no such selection. With larger corpora, the probability that a sentence will have the correct keywords, but the wrong relation(s), increases, and thus a standard keyword retrieval system will decrease in performance.[3]

[2]More experiments are required to quantitatively assess the impact of parser error.

[3]Note that the Worldbook Encyclopedia is relatively small in size.

The following are a few classes of questions for which relations indexing offers the most advantages.

7.3.1 Semantically Similar Verbs

Verbs that subcategorize for arguments of the same semantic class can often confuse keyword-based systems. Many verbs take arguments of the same semantic type (e.g., visit, eat, import (to/from), see, etc.), and in most of these sentences, reordering the verb arguments drastically alters their meaning. A keyword indexer would suffer from poor precision when given a query that required the identification of relations centered on such verbs:

(22) When has Iran invaded Iraq?
(23) Where did John see Mary?
(24) Regarding what issue did the president of Russia criticize China?

For example, a keyword search engine would not be able to distinguish between a question regarding Iran invading Iran and Iraq invading Iran (22) because both queries have the same keyword content. Similarly, the keyword approach would be unable to determine who did the seeing (23), or who did the criticizing (24).

7.3.2 Modification relations

A keyword indexing system is unable to capture many types of modification relations and other special constructions. For example, in the sample question (24), above, was it the president of Russia or the president of China? In addition, consider the following questions:

(25) What is the largest country in the world?
(26) Are electronics the biggest export from Japan to the United States?

Many phrases may contain the correct keywords, but in the wrong modification relations. For example, both "largest country in the world" and the "largest man-made lake of any country in world" contain the keywords "country", "largest", and

"world." However, only one of those phrases may point to the correct answer. Furthermore, there are some constructions whose meaning critically depends on relations between the entities, e.g., (26), because "from `X` to `Y`" and "from `Y` to `X`" usually differ in meaning.

7.3.3 Queries involving Common Words

In many questions, all salient keywords are very commonly occurring words; thus, a keywords indexer may return an overwhelmingly large number of results. As an example, the only salient keyword in the question "what is interferon?" is the word "interferon"; all other words are stopwords that are usually discarded. Given such a query, a standard keyword search engine would return all sentences with the keyword "interferon." Unfortunately, most of these sentences will not answer the user question.

Chapter 8

Future Work

Applying natural language processing techniques shows potential to improve the quality of current information retrieval systems. The current Sapere System is merely a prototype which we hope will serve as the core of a "playground" devoted to the integration of natural language processing techniques and information retrieval. In this chapter, we outline a roadmap for future direction in building the next generation information access system.

8.1 Information Extraction

Many question answering systems ([32, 29, 5, 10], just to name a few) today utilize information extraction (IE) [1] techniques as an important component of their architecture.

In question answering, information extraction technology is used to match the semantic type expected by the user query with the semantic type of the answer. By analyzing the question, it is possible to anticipate the type of the answer, e.g., the answer to a "who" question is usually a person, the answer to a "where" question is usually a location. With this knowledge, a question answering system can then focus on retrieving the text fragments with the correct answer type. This technique is far from perfect, however, because many sentences contain multiple constructions of the same type, and it is difficult to distinguish among them, e.g., a sentence

that supposedly answers a "who" question may contain two or three different names. Furthermore, it is difficult to determine the semantic type of questions other than the standard "who," "where," "what," ones; for example, a "how" or "why" question may be answered with a variety of different constructions.[1]

Although information extraction techniques alone cannot determine the validity of an answer, integration of such technology into an information access system can nevertheless improve the precision of responses.

An information access system could also utilize information extraction technology to preprocess corpus text in order to generate more accurate relations. Natural language is full of named entities such as personal names, organization, dates, numbers, etc., that often cannot be handled by a linguistic parser. This often leads to an incorrect analysis of language; for example, Minipar is unable to parse the "VIII" in "King Henry VIII," leading to the construction of an incorrect parse tree. Information extraction technology can be applied to efficiently preprocess text, so that these constructions can be recognized and extracted. The integration of an information extraction system with a relations indexer will increase the accuracy of the parser, and also the relations derived from it.

Furthermore, information extraction techniques will enable a relations-based information access system to intelligently handle variations in named entities. For example, "President John Fitzgerald Kennedy" is exactly the same person as "John F. Kennedy," "President Kennedy," and many other forms.[2] Thus, a relation involving "President Kennedy" should match another relation that uses an alternate title for the same person. The same correspondence should also be established with locations (e.g., "Massachusetts" is the same as "MA"), companies (e.g., "International Business Machines" is the same as "IBM"), dates (e.g., "January 1st" is the

[1] Although it may be possible to look for textual markers such as "because . . . ," these techniques fail at least as often as they succeed because the answer to a "why" question often does not involve any easily identifiable textual markers. For example, the answer to "Why is the sky blue" might be "Sunlight scattering off gas molecules and dust particles in air give the sky a blue color."

[2] A system must be careful in equating different variants, and should take context into account. e.g., there are many "John Adams," and it would be wrong to equate "Mr. Adams" with every of occurrence "John Adams."

same as "first of January"), and other named entities. Variations with named entities could be handled by a combination of heuristic rules (e.g., rules for determining the equivalence of names) and dictionary lookups against lists of known variants.

8.2 More Sophisticated Query Processing

A system would benefit from more in-depth analysis of the user query in order to identify the relations that should be expected in the correct answer. For example, consider the following incorrect answers to the question "Where do ants live" retrieved by Sapere:

(27) Army ants live by hunting other insects.
(28) But except for the fact that they all live in groups, ants vary greatly in their ways of life.
(29) The worker ants will live for a month or two, and then die.

Currently, the matching algorithm converts any *wh*-word in the question to wildcards (see Chapter 5), but performs no additional processing based on the type of the *wh*-word; hence all the above sentences are returned as the correct answer because they contain *ants* and *live* in a subject–verb relation. More sophisticated query processing techniques, e.g., mapping "where" to prepositional phrases headed by "in" and other possible prepositions of location, could to some degree alleviate this problem, although such techniques could not distinguish between "in groups" and "in Africa."

8.3 More Sophisticated Matching Algorithm

A drawback with the standard boolean model for keyword-based information retrieval is the incorrect assumption that all keywords are equal in importance [6]. This has lead to the development of information retrieval models utilizing non-boolean models [35, 36] (e.g., fuzzy logic [4]), which are more flexible and achieve better performance.

The same approach can be utilized to increase the flexibility of an information access model based on relations; relations of different types are not equal in importance, and should be reflected in the matching strategy (and scoring algorithm) of the system. For example, subject-verb-object relations are generally more important than adjective-noun modification.

To illustrate this point, consider the question "Where do black bears live?" When a system is presented with this question, it should retrieve knowledge about the habitat of other types of bears (that may be similar) than random facts about black bears (e.g., how big they are, what they eat). Also, adjective-noun modification relations are much more frequent than subject-verb-object relations, and thus a strict match on the former will likely produce fewer results, preventing information overload. A matching algorithm that takes the relative importance of relations into account will lead to a system that better satisfies user information needs.

8.4 Coreference Resolution

A disadvantage of performing retrieval at the sentence level (as opposed to the document-level retrieval techniques used by traditional information retrieval systems) is the inability to handle intersentential anaphoric references. However, retrieval at the sentence level is a good basis for question answering because a sentence serves as a (relatively) compact, self-contained unit of information that can be easily digested by the user. (Although often the answer to a question may span several sentences, this problem can be alleviated by providing results in a surrounding context, e.g., the sentence preceding and following the result.) A successful information access system must handle intersentential anaphora and explicitly equate each referring expression to the entities it refers.

There is a substantial body of research on anaphora resolution, ranging from traditional syntactic/semantic approaches [12, 30, 41] to more recent corpus-based approaches [8, 19]. Coreference resolution has also been applied to question answering systems [28, 37], but has not been integrated with a relations-based approach.

We believe that resolving pronominal expressions and definite noun phrases will enhance the performance of question answering systems.

8.4.1 Pronominal Expressions

After the initial introduction of many named entities, subsequent references to the same entity may be accomplished using pronouns, e.g., "Albert Einstein was a brilliant physicist... he left Germany for the United States..." or "Today IBM announced... It plans to...." Use of pronouns as referring expressions is common in all types of writing, and should be handled by a sophisticated question answering system.

8.4.2 Definite Noun Phrases

In many situations, a definite noun phrase is used to refer to another entity introduced previously. Often, the head noun used in the referring expression is embedded in the complete name of the entity, e.g., "President Bush... the president...." Such instances are relatively easy to handle, because explicit links exist between the referring expression and the entity to which it refers. However, many definite noun phrase anaphors are much more difficult to handle, e.g., "Polly the parrot wants a cracker. The bird..." or "Taipei is the capital of Taiwan. The city...." Resolving the definite noun phrases in the previous two examples requires knowledge that a parrot is a bird, and a capital is also a city. Thus, a large semantic ontology of entities may be required to properly resolve many definite noun phrase anaphors.

Currently, no existing resource can be utilized to solve this problem. For example, WordNet [27] is ill-suited for this purpose because it is primarily a lexical database, not a semantic one. Furthermore, the classification scheme (hierarchy) of the semantic ontology must match the classification scheme implicit in the articles, or else the ontology may not be helpful.

Although corpus-specific ontologies for the purpose of resolving definite noun phrase references may be built by hand, they require too much time to construct. A potential technique that may simplify the ontology building process is to extract

ontological definitions from the corpus itself. For example, if the corpus contained the definition of a parrot, e.g., "the parrot is a colorful tropical bird," a hypernym/hyponym pair can be extracted automatically. Nevertheless, there are limits to this technique, because many relationship between entities are obvious to humans and will probably never be explicitly stated, e.g., an American is a person.

8.4.3 Significance

Referring expressions play an important role in language, and thus a coreference resolution module is an important component of a complete question answering system.

8.5 Lexical-semantic relations

Query expansion techniques have classically been used to ameliorate the difficulties that synonyms and homographs cause in text retrieval systems. However, research has shown that query expansion involving synonymy and other lexical-semantic relations is generally ineffective without accurate word-sense disambiguation [38, 39]. Although a relations indexer cannot distinguish between word senses, identification of syntactic relations may nevertheless be helpful in applying lexical-semantic relations to question answering; in many cases, syntax offers clues regarding semantics, e.g., in sentences (3) and (3′).

(repeated from Chapter 1)
(3) the bank of the river
(3′) the bank near the river

8.6 Transformational Rules

Although the system currently implements a module for transformational rules, this aspect of the system requires further attention. Transformational rules may be a promising solution to the problem of linguistic variation, provided that a general set of

broad coverage rules can be constructed. Naturally, a major challenge is the creation of such rules. Although transformational rules can be hand-coded, it is impractical to build a large knowledge base of rules solely by hand. Promising techniques include utilization of a distributed system for gathering linguistic knowledge, or application of statistic and machine learning techniques.

8.7 A Hybrid System

We believe that a successful question answering system should be a hybrid that combines a variety of techniques, ranging both in sophistication of the language analysis and the robustness of the technique.

A relations-based model of indexing may potentially decrease the recall of an information access system, but keyword and statistical techniques may be applied to boost recall to acceptable levels. Although information retrieval schemes based on keyword statistics are far from perfect, they are nevertheless robust. Therefore, a integrated question answering system should still utilize keyword-based techniques, either as a fallback mechanism or as a method of cross-validating results derived from other methods.

A complete information access system requires the integration of many different techniques and components; the synergistic effects of such a hybrid could bring us one step closer to the "smart" information access system of tomorrow.

Chapter 9

Conclusion

This document describes an architecture that integrates natural language processing techniques with information retrieval using ternary expressions as the fundamental representational structure. By striking a balance between the sophistication of natural language techniques and the efficiency of the indexing scheme, we may be able to retain many advantages of natural language processing, but expand it into the domain of unrestricted text. The Sapere System, an initial implementation of our ideas, shows dramatic improvements over existing systems in our relatively limited tests. We believe this architecture serves as a good foundation to a next-generation natural language question answering system.

Bibliography

[1] Douglas E. Appelt and David Israel. Introduction to information extraction technology. In *IJCAI-99 Tutorial*, 1999.

[2] Avi Arampatzis, Th.P. van der Weide, C.H.A. Koster, and P. van Bommel. Phrase-based information retrieval. *Information Processing and Management*, 34(6):693–707, December 1998.

[3] Avi Arampatzis, Th.P. van der Weide, C.H.A. Koster, and P. van Bommel. An evaluation of linguistically-motivated indexing schemes. In *Proceedings of BCS-IRSG 2000 Colloquium on IR Research*, 2000.

[4] A. Bookstein. Probability and fuzzy-set application to information retrieval. *Annual Review of Information Science and Technology*, 29:117–151, 1986.

[5] Eric Breck, John Burger, Lisa Ferro, David House, Marc Light, and Inderjeet Mani. A sys called quanda. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[6] William S. Cooper. Getting beyond boole. *Information Processing and Management*, 24:243–248, 1988.

[7] Bruce Croft and David D. Lewis. An approach to natural language processing for document retrieval. In *Proceedings of the 10th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-87)*, 1987.

[8] I. Dagan and A. Itai. Automatic processing of large corpora for the resolution of anaphora references. In *Proceedings of the 13th Annual International Conference on Computational Linguistics (COLING'90)*, 1990.

[9] Joel L. Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparisons of Syntactic and Non-Syntactic Methods.* PhD thesis, Cornell University, 1987.

[10] Robert Gaisauskas and Kevin Humphreys. A combined IR/NLP approach to question answering against large text collections. In *Proceedings of the 6th RIAO Conference on Content-Based Multimedia Information Access (RIAO '00)*, 2000.

[11] George E. Heidorn. Natural language inputs to a simulation programming systems. Technical Report NPS-55HD72101A, Naval Postgraduate School, 1972.

[12] J. Hobbs. Pronoun resolution. Technical Report 76-1, Department of Computer Science, City College, City University of New Work, 1976.

[13] Karen Jensen, George E. Heidorn, and Stephen D. Richardson, editors. *Natural Language Processing: The PLNLP Approach.* Kluwer Academic Publishers, 1993.

[14] Boris Katz. Using English for indexing and retrieving. In P.H. Winston and S.A. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers*, volume 1. MIT Press, 1990.

[15] Boris Katz. Annotating the World Wide Web using natural language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97)*, 1997.

[16] Boris Katz and Beth Levin. Exploiting lexical regularities in designing natural language systems. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING '88)*, 1988.

[17] Boris Katz and Patrick H. Winston. Parsing and generating english using commutative transformations. Technical Report 677, MIT Artificial Intelligence Laboratory, 1982.

[18] Boris Katz and Patrick H. Winston. A two-way natural language interface. In *Proceedings of the European Conference on Integrated Interactive Computing Systems (ECICS '82)*, 1982.

[19] C. Kennedy and B. Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th Annual International Conference on Computational Linguistics (COLING'96)*, 1996.

[20] Julian Kupiec. Murax: A robust linguistic approach for question answering using an an on-line encyclopedia. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

[21] Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago Press, 1993.

[22] Dekang Lin. PRINCIPAR—an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING '94)*, 1993.

[23] Dekang Lin. Principled-based parsing without overgeneration. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics (ACL'93)*, 1993.

[24] Dekang Lin. Minipar—a minimalist parser. In *Maryland Linguistics Colloquium*, University of Maryland, College Park, March 12, 1999.

[25] Kenneth C. Litkowski. Question-answering using semantic relation triples. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[26] Edward Loper. Applying semantic relation extraction to information retrieval. Master's thesis, Massachusetts Institute of Technology, 2000.

[27] George Miller. Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):49–51, 1995.

[28] Thomas S. Morton. Using coreference in question answering. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[29] John Prager, Dragomir Radev, Eric Brown, Anni Coden, and Valerie Samn. The use of predictive annotation for question answering in TREC8. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[30] C.S. Sidner. Towards a computational theory of definite anaphora comprehension in english discourse. Technical Report 537, MIT Artificial Intelligence Laboratory, 1976.

[31] Alan F. Smeaton, Ruairi O'Donnell, and Fergus Kelledy. Indexing structures derived from syntax in TREC-3: System description. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, 1994.

[32] Rohini Srihari and Wei Li. Information extraction supported question answering. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[33] Tomek Strzalkowski, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, Jose Perez-Carballo, Troy Straszheim, Jin Wang, and Jon Wilding. Natural language information retrieval: TREC-5 report. In *Proceedings of the 5th Text REtrieval Conference (TREC-5)*, 1996.

[34] Tomek Strzalkowski and Peter Scheyen. An evaluation of TTP parser: A preliminary report. In H. Blunt and M. Tomita, editors, *Recent Advances in Parsing Technology*, pages 202–220. Kluwer Academic Publishers, 1993.

[35] C.J. van Rijsbergen. A new theoretical framework for information retrieval. In *Proceedings of the 1986 ACM Conference on Research and Development in Information Retrieval*, 1986.

[36] C.J. van Rijsbergen. A non-classical logic for information retrieval. *Computer Journal*, 29:481–485, 1986.

[37] Jose L. Vicdeo and Antonio Ferrandez. Anaphora resolution in question answering systems. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, 2000.

[38] Ellen M. Voorhees. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-93)*, 1993.

[39] Ellen M. Voorhees. Query expansion using lexical-semantics relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94)*, 1994.

[40] Ellen M. Voorhees and Dawn M. Tice. The TREC-8 question answering track evaluation. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[41] B.L. Webber. *A Formal Approach to Discourse Anaphora*. Garland Publishing, London, 1999.

[42] Patrick H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, Massachusetts, third edition, 1992.

[43] Chengxiang Zhai, Xiang Tong, Natasa Milic-Frayling, and David A. Evans. Evaluation of syntactic phrase indexing – CLARIT NLP track report. In *Proceedings of the 5th Text REtrieval Conference (TREC-5)*, 1996.