In Proceedings of the ACL 2000 Workshop on Natural
Language Processing and Information Retrieval (NLP&IR),
October 2000, Hong Kong, China

REXTOR: A System for Generating Relations from Natural Language

Boris Katz and **Jimmy Lin**
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139 USA
{boris, jimmylin}@ai.mit.edu

Abstract

This paper argues that a finite-state language model with a ternary expression representation is currently the most practical and suitable bridge between natural language processing and information retrieval. Despite the theoretical computational inadequacies of finite-state grammars, they are very cost effective (in time and space requirements) and adequate for practical purposes. The ternary expressions that we use are not only linguistically-motivated, but also amenable to rapid large-scale indexing. Rextor (Relations EXtracTOR) is an implementation of this model; in one uniform framework, the system provides two separate grammars for extracting arbitrary patterns of text and building ternary expressions from them. These content representational structures serve as the input to our ternary expressions indexer. This approach to natural language information retrieval promises to significantly raise the performance of current systems.

1 Introduction

Traditional information retrieval (IR) has been built on the “bag-of-words” assumption, which equates the weighted component keywords of a document with its semantic content. Obviously, a document is much more than the sum of its individual keywords. Although keywords may offer some indication of “meaning,” they 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]

(1) The big man ate the dog.
(1′) The big dog ate the man.
(2) The meaning of life
(2′) A meaningful life
(3) The bank of the river
(3′) The bank near the river

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.

The application of natural language processing (NLP) techniques to information retrieval promises to generate representational structures that better capture the semantic content of documents. In particular, syntactic analysis can highlight the relationships between various terms and phrases in a sentence, which will allow us to distinguish between the example pairs given above and answer queries with higher precision than traditional IR systems.

However, a syntactically-informed representational structure faces the problem of *linguistic variations*, the phenomenon in which similar semantic content may be expressed in different surface forms. Consider the following sets of sentences that express the same meaning using different constructions:

(4) What is Bill Gates’ net worth?
(4′)What is the net worth of Bill Gates?
(5) John gave the book to Mary.
(5′) John gave Mary the book.
(5″) Mary was given the book by John.
(6) The president surprised the country with his actions.
(6′) The president’s actions surprised the country.
(7) Over 22 million people live in Taiwan.
(7′) The population of Taiwan is 22 million.

An effective linguistically-motivated information retrieval system must not only handle rel-

[1] Examples taken from (Loper, 2000)

atively simple syntactic variations (e.g., (4) and (5)), alternate realization of verb arguments (e.g., (6) and (6′)), but also more complicated semantic variations (e.g., (7) and (7′)). This can be accomplished by *linguistic normalization*, a process by which linguistic variants that contain the same semantic content are mapped onto the same representational structure.

The precision of information retrieval systems can be dramatically improved if they index not only single terms, but normalized representational structures derived from language. However, the optimal structure of this representation and the efficient generation of these structures remains an open research problem.

This paper argues that, for the purposes of information retrieval systems, the most suitable representational structure of document content is ternary expressions (compared to, for example, keywords, trees or case frames). Ternary (three-place) expressions may be thought of as typed binary relations (e.g., subject-relation-object) or two-place predicates (e.g., transitive verbs like 'hit'); they are linguistically-motivated and efficient to index. Also, for information retrieval, a finite-state grammar is the most practical and cost effective method by which to extract these ternary expressions from documents. Combined together, a finite-state language model and ternary expression representation provide a convenient and powerful framework for integrating natural language processing with information retrieval.

Rextor (Relations EXtracTOR) is a document content analysis system designed to unify and generalize many previous natural language information retrieval techniques into one single framework. The system provides two separate grammars: one for extracting arbitrary entities from documents, and the other for building relations from the extracted items. Rextor also provides a playground and testbed for future experimentation in linguistically-motivated indexing schemes.

2 Motivation

We believe that, for humans, natural language is the best mechanism for information access. It is intuitive, easy to use, rapidly deployable, and requires no specialized training,

The Rextor System builds on the experience of Start (SynTactic Analysis using Reversible Transformations), a natural language system available for question answering on the World Wide Web.[2] Since December, 1993, when it first came online, Start has engaged in millions of exchanges with hundreds of thousands of people all over the world, supplying users with knowledge regarding geography, weather, movies, and many many other areas. Despite the successes of Start in serving actual users, its domain of knowledge is relatively small and expanding its knowledge base is a time-consuming task. The goal of Rextor is to overcome this bottleneck and to provide a general framework for natural-language information retrieval. Rextor not only draws its inspiration from Start (in providing question answering capabilities), but also borrows a simplified form of its representational structures (Katz, 1980; Katz, 1990).

The Start System (Katz, 1990; Katz, 1997) analyzes English text and builds a knowledge base from information found in the text. The knowledge is expressed in the form of embedded ternary expressions (T-expressions) — subject-relation-object triples where the subject and object can themselves be ternary expressions. For example, "The population of Zimbabwe is $11,044,147$" would be represented as two ternary expressions:

```
[POPULATION-1 IS 11044147]
[POPULATION-1 RELATED-TO ZIMBABWE]
```

Experience from Start has shown that a robust full-text natural language question-answering system cannot be realistically expected any time soon. Numerous problems such as intersentential reference, paraphrasing, summarization, common sense implication, and many more, will take a long time to solve satisfactorily. In order to bypass intractable complexities of language, Start uses computer-analyzable natural language annotations, which consist of simplified English sentences and phrases, to describe various information segments (which may be text, images, or even video and other multimedia content). These natural language annotations serve as metadata and inform Start regarding the type of questions that a particular information segment is capable of answering (Katz, 1997). By performing retrieval on natural language annotations, the system is able to provide knowledge that it may not be able to analyze itself (either language that is too complex or non-textual segments). Because these annotations must be manually generated, expanding Start's knowledge base is relatively time-intensive.

Rextor attempts to eliminate the need for human involvement during content analysis, and also aims to serve as the foundation of a natural language information retrieval system. Ultimately,

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

we hope that Rextor will serve as a stepping stone towards a comprehensive system capable of providing users with “just the right information” to queries posed in natural language.

3 Previous Work

The concept of indexing more than simple keywords is not new; the idea of indexing (parts of) phrases, for example, is more than a decade old (Fagan, 1987). Arampatzis (1998) introduced the phrase retrieval hypothesis, which asserted that phrases are a better indication of document content than keywords. Several researchers have also explored different techniques of linguistic normalization for information retrieval (Strzalkowski et al., 1996; Zhai et al., 1996; Arampatzis et al., 2000). The performance improvements were 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 offer over traditional IR.

Previous research in linguistically motivated information retrieval concentrated primarily on noun phrases and their attached prepositional phrases. Techniques that involve head/modifier relations have been tried, e.g., indexing adjective/noun and noun/right adjunct pairs (which normalizes variants such as “information retrieval” and “retrieval of information”). However, there has been little experimentation with other types of linguistic relations, e.g., appositives, predicate nominatives (i.e., the `is-a` relation), predicate adjectives (i.e., the `has-property` relation), etc. Furthermore, indexing of word pairs and phrases in many previous systems was accomplished by converting those representations into lexical items and atomic terms, indexed in the same manner as single words. The treatment of these representational structures using a restrictive bag-of-words paradigm limits the type of queries that may be formulated. For example, treating adjective/noun pairs ([*adj.*, *noun*]) as lexical atoms renders it impossible to find the equivalent of “all *big* things,” corresponding to the pair [big, `*`].

The extraction of these relations from documents has been relatively inefficient and unsystematic. One approach is to first parse the document using a full-text parser, and then extract interesting relations from the resulting parse tree (Fagan, 1987; Grishman and Sterling, 1993; Loper, 2000). This approach is slow and inefficient because full-text parsing is very time-intensive. Due to current limitations of computational technology, only a small fraction of the information gathered by a full parser can be efficiently indexed. For the most part, relations that can be effectively utilized for information retrieval purposes only occupy a few nodes of a (possibly dense) parse tree; thus, most of the knowledge gathered by the parser is thrown away. Also, extracting non-linguistic relations from parse trees is very difficult; many interesting relations (from an IR point of view) have no linguistic foundation, e.g., adjacent word pairs. The other approach to extracting relations from text is to build simple filters for every new relation. This approach is unsystematic, and does not allow for rapid addition of new relations to a system.

The Rextor System utilizes an integrated model to systematically extract arbitrary textual patterns and relations (ternary expressions) from documents. The concept of coupling structure-building actions with parsing originated with augmented transition networks (ATNs)(Thorne et al., 1968; Woods, 1970). Similarly, PLNLP (Heidorn, 1972; Jensen et al., 1993) is a programming language for writing phrase structure rules that include specific conditions under which the rule can be applied. These rules may also be augmented by structure-building actions that are to be taken when the rule is applied. However, these systems that attempt full-text parsing are less efficient for information retrieval applications due to the long time necessary to generate full linguistic parse trees. Rextor was designed with a simple language model and an equally simple, yet expressive, representation of “meaning.”

4 Bridging Natural Language and Information Retrieval

In order to bridge the gap between natural language and information retrieval, natural language text must be distilled into a representational structure that is amenable to fast, large-scale indexing. We argue that a finite-state model of natural language with ternary expressions is currently the most suitable combination for this task.

4.1 Finite-State Language Model

Despite its limitations, a finite-state grammar seems to provide the best natural language model for information retrieval purposes. One of the most notable computational inadequacies of the finite-state model is the absence of a pushdown mechanism to suspend the processing of a constituent at a given level while using the same grammar to process an embedded constituent (Woods, 1970). Due to this inadequacy, certain

English constructions, such as center embedding, cannot be described by any finite-state grammar (Chomsky, 1959a; Chomsky, 1959b). However, Church (1980) demonstrated that the finite-state language model is adequate to describe a performance model of language (i.e., constrained by memory, attention, and other realistic limitations) that approximates competence (i.e., language ability under optimal conditions without resource constraints). Many phenomena that cannot be handled by finite-state grammars are awkward from a psycholinguistic point of view, and hence rarely seen. More recently, Pereira and Wright (1991) developed formal methods of approximating context-free grammars with finite-state grammars.[3] Thus, for practical purposes, computationally simple finite-state grammars can be utilized to adequately model natural language.

Empirically, the effectiveness of the finite-state language model has been demonstrated in the *Message Understanding Conferences (MUCs)*, which evaluated information extraction (IE) systems on a variety of domain-specific tasks. The conferences have shown that superficial parsing using finite-state grammars performs better than deep parsing using context-free grammars (at least under the current constraints of technology). The NYU team switched over from a system that performed full parsing (PROTEUS) in MUC-5 (Grishman and Sterling, 1993) to a regular expression matching parser in MUC-6 (Grishman, 1995). Full parsing was slow and error-prone, and the process of building a full syntactic analysis involved relatively unconstrained search which consumed large amounts of both time and space. The longer debug-cycles that resulted from this translated into fewer iterations with which to tune the system within a given amount of time. Furthermore, the complexity of a full context-free grammar contributed to maintenance problems; complex interactions within the grammar prevented rapid updating of the system to handle new constructions.

Finite-state grammars have been used to extract entities such as proper nouns, names, locations, etc., with relatively high precision. To a lesser extent, these grammars have proven to be effective in identifying syntactic constructions such as noun phrases and verb phrases. Fastus (Hobbs et al., 1996), the most notable of these systems, is modeled after cascaded, nondeterministic finite-state automata. The finite-state transducers are "cascaded" in that they are arranged in series; each one maps the output structures from the previous transducer into structures that comprise the input to the next transducer.

There are many similarities between information extraction and building effective representational structures for information retrieval. Both tasks involve identifying entities (e.g., phrases) and the relationships between those entities. Thus, the application of proven information extraction techniques (i.e., finite-state technology) to information retrieval offers promise in raising the performance of IR systems.

4.2 Ternary Expressions

Ternary (three-place) expressions currently appear to be the most suitable representational structure for meaning extracted from 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. 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 arbitrary tree — thus, ternary expressions 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 prerequisite 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.

In fact, indexing linguistic tree structures has been attempted (Smeaton et al., 1994), 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), and to the poor quality of the full-text natural language parser, which was also rather slow. Despite recent advances, full-text natural language parsers are still relatively error-prone; indexing incorrect parse trees is a source of performance degradation. 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]]]

[3] However, these approximations overgenerate, although in predictable, systematic ways.

may be "flattened" into three relations:

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

Indexing case frames has also been attempted (Croft and Lewis, 1987; Loper, 2000), but with limited success. Full semantic analysis is still an open research problem, especially in the general domain. Since full semantic 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 index than ternary expressions.

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;[4] hence, well-known optimizations for databases may be applied for extremely high performance.

Previous linguistically-motivated indexing schemes may easily be reformulated using ternary expressions. For example, indexing adjacent word pairs consists of indexing adjacent words with the `adjacent` relation. In fact, all pairs (e.g., adjective-noun, head-modifier) can be reformulated as ternary expressions by assigning a type to the pair. This finer granularity allows the capture of more intricate relations between words in a document.

5 The REXTOR System

Using its finite-state language model, the Rextor System generates a set of ternary expressions that correspond to content of a part-of-speech-tagged input document. Currently, the Brill Tagger (Brill, 1992) (with minor postprocessing) is used for the part-of-speech (POS) tagging. The relations construction process consists of two distinct processes, each guided by its own externally specified grammar file. *Extraction rules* are applied to match arbitrary patterns of text, based either on one of thirty-nine POS tags or on exact words. Whenever an item is extracted, a corresponding *relation rule* is triggered, which handles the actual generation of the ternary expressions (relations).

5.1 Extraction Rules

Extraction rules are used to extract arbitrary patterns of text according to a grammar specification. The Rextor grammar is written as regular expression rules, which are computationally equivalent to finite-state automata.[5] Writing grammar rules in this fashion allows for perspicuity, the property whereby permitted types of constructions are readily apparent from the rules. Such a human-readable formulation simplifies maintenance of the grammar.

The extraction stage of the Rextor System performs a no-lookahead left-to-right scan of every input sentence, identifies the longest matching pattern (from any grammar rule), reduces the input sequence based on the matched rule, and continues with the next unmatched word. If a word cannot be included in any grammar rule, it is skipped.

An extraction rule takes the following form:

```
EntityType := template;
```

The rule can be read as `EntityType` *is defined as* `template`. A successful match of the pattern in `template` signifies a successfully extracted entity. The `template` consists of a series of legal tokens, which are shown in Table 1. In addition, token modifiers (also in Table 1) can alter the meaning of the immediately preceding token. Tokens surrounded by curly braces ({}) are saved as *bound variables*, which can be later utilized to build relations (ternary expressions). These variables are referenced numerically starting at zero (e.g., the 0th bound variable).

5.2 Relation Rules

A relation rule is triggered by the successful extraction of a particular entity (`EntityType`). The relations grammar directs the construction of the actual ternary expression. A relation rule takes the following form:

```
EntityType :=> <atom1 atom2 atom3>;
```

The `EntityType` is the trigger for the relation, i.e., the rule is applied whenever a string of that type is extracted. The right hand side of the relation rule is the ternary expression to be generated, which is a triple composed of three atoms. Valid atoms are shown in Table 2. They are either string literals or they manipulate the bound variables saved from the extraction process in some manner.

[4] In fact, our first implementation of a ternary expressions indexer used a SQL database.

[5] For an algorithm converting regular expressions to nondeterministic finite-state automata, please refer to (Aho et al., 1988), Chapter 3.

Token	*Description*
`POS`	This matches any word tagged as the part-of-speech `POS`.
`POS[`*string*`]`	This matches a specific word (*string*) of a specific part-of-speech (`POS`).
`EntityType`	This matches any extracted string of type `EntityType`.
(token$_0$|token$_1$|...)	This expression matches any one of the alternative tokens given within the parentheses. Matches are attempted in the order in which they are written, e.g., the first token is tried first.
Token Modifier	*Description*
`*`	This modifier matches zero or more occurrences of the previous token.
`?`	This modifier matches zero or one occurrence of the previous token.
`+`	This modifier matches one or more occurrences of the previous token.

Table 1: Valid tokens and token modifiers for extraction rules.

Modifier	*Description*
[*n*]	Evaluates to the *n*th bound variable of the trigger `EntityType`, interpreted as a string.
{*n*}	Evaluates to the *n*th bound variable of the trigger `EntityType`, interpreted as a list of strings. The extraction rule token inside the bound variable is stripped of its outermost `*` or `+`, and the bound variable is broken into a list according to this pattern. For example, {`JJX*`} is interpreted as a list of `JJX`, or adjectives.
[*i*],EntityType$_1$[*j*],...	This expression extracts a bound variable nested inside other bound variables. The *i*th bound variable of trigger `EntityType` is extracted; if this item is of type EntityType$_1$, then the *j*th bound variable is extracted (the expression returns `false` if the entity types do not match); each comma separated unit is interpreted in this manner, up to an arbitrary depth.
(*alternative*$_1$|*alternative*$_2$|...)	This compound expression evaluates to the disjunction of an arbitrary number of valid atoms (as defined in this table). Each alternative is evaluated in a left to right order; the disjunction evaluates to the first alternative that returns a non-empty string.
'*string*'	A literal string.

Table 2: Valid atoms for the relation rules.

```
Extraction Rules:  NounGroup := (PRPZ|DT)? {JJX*} {(NNPX|NNX|NNPS|NNS)+};
                   PrepositionalPhrase := IN {NounGroup};
                   ComplexNounGroup := {NounGroup} {PrepositionalPhrase};
Relation Rules:    NounGroup :=> <{0} 'describes' [1]>;
                   ComplexNounGroup :=>
                            <[0],NounGroup[1]
                            'related-to'
                            [1],PrepositionalPhrase[0],NounGroup[1]>;
```

Figure 1: Example of relation and extraction rules. (`PRPZ` is the part-of-speech tag for possessive pronouns, `DT` for determiners, `JJX` for adjectives, `JJR` for comparative adjectives, `JJS` for superlative adjectives, `NNX` for singular or mass nouns, `NNS` for plural nouns, `NNPX` for singular proper nouns, `NNPS` for plural proper nouns, `IN` for prepositions.)

5.3 Examples

A few extraction and relation rules are given in Figure 1. The first extraction rule defines a `NounGroup` as a sequence consisting of: an optional possessive pronoun or determiner, any number of adjectives, one or more nouns (of any type). Also, the sequence of adjectives is saved as the 0th bound variable, and the sequence of nouns is saved as the 1st bound variable. The rules for `PrepositionalPhrase` and `ComplexNounGroup` can be interpreted similarly.

Consider the following noun phrase:

the big, bad wolf of the dark forest

Rextor recognizes two `NounGroups` in the above phrase: *the big, bad wolf* and *the dark forest.* The corresponding relation rule triggers, and generates the following relations:

< (big, bad) `describes` wolf >
< (dark) `describes` forest >

Note that the first bound variable in `NounGroup` is interpreted as a list; thus, the above two relations expand into three distinct relations when completely enumerated:

< big `describes` wolf >
< bad `describes` wolf >
< dark `describes` forest >

The ability to interpret bound variables as a list of strings allows for easy manipulation of repeated structure, like textual lists or enumerations.

In addition, the entire noun phrase *the big, bad wolf of the dark forest* will be recognized as a `ComplexNounGroup`. This will result in the following relation:

< wolf `related-to` forest >

The relation rule associated with `ComplexNounGroup` involves extracting nested bound variables. The first atom evaluates to the 1th bound variable (a `NounGroup`) inside the 0th bound variable inside the trigger item `ComplexNounGroup`. The third atom is similarly evaluated.

6 Discussion

Informal analysis of documents using Rextor reveals that it can potentially serve as an effective framework for extracting "meaning" from documents. In particular, the system is capable of identifying the following types of linguistic constructions and generating relations from them:

- **Simple sentences** can be extracted by noting a simple `NounGroup VerbGroup NounGroup` pattern. From this, subject-verb-object (SVO) relations can be derived.

- **Predicative nominatives** can be recognized by identifying the "be" verb and the `NounGroup` directly following it. These constructions may be useful in establishing ontological hierarchies, i.e., `is-a` trees.

- **Predicative adjectives** can be recognized by the "be" verb and a succession of one or more adjectives (or adjectival phrase). They may provide addition information regarding the attributes of entities, e.g., `has-property`.

- **Appositives** are characteristically offset by commas and usually contain a single noun phrase; thus, they can be recognized relatively easily. Common in prose, appositives offer a wealth of additional information regarding various entities, e.g., location of sites, age or position of people, etc.

- **Prepositional phrases** are relatively easy to extract, and may supply valuable relations that increase the precision of information retrieval systems. Ternary expressions allow for a better representation of prepositional phrases (compared to pairs) because they allow the preposition to more specifically determine the type of relation (thus, examples like "boat *by* the water" and "boat *under* the water," which have completely different meanings, may be indexed separately and distinctly).

  However, the prepositional phrase attachment problem (in the general-domain case) is still an open research topic, and thus poses some problems to content analysis. Regardless, for the purposes of information retrieval, it may be acceptable to err on the side of overgeneration in considering attachment, i.e., enumerate all possible relations. This will no doubt generate a large number of (possibly incorrect) relations, and more research is required to determine effective methods of controlling this explosion.

- **Relative clauses** of some types can be identified by a finite-state language model. They may supply additional useful SVO relations for indexing purposes.

We believe that future breakthroughs in natural language information retrieval will occur in the generation of meaningful relations. Although the finite-state language model of Rextor is powerful enough to extract many linguistically interesting constructions, the approach is not fundamentally new. What differentiates our system from previous work such as Fastus (Hobbs et al., 1996) is that Rextor not only provides a mechanism for extraction, but also introduces the paradigm of ternary expressions to capture document content for information retrieval. The relations view of natural language documents is highly amenable to integration with information retrieval systems.

Through a relations representation, Rextor is able to distinguish the subtle differences in meaning between the pairs of sentences and phrases given in the introduction:

(1) The man ate the dog.
< man is-subject-of eat >
< dog is-object-of eat >

(1′) The dog ate the man.
< man is-object-of eat >
< dog is-subject-of eat >

(2) The meaning of life
< meaning possessive-relation life >

(2′) A meaningful life
< meaningful describes life >

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

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

The ability to extract subject-verb-object relations, e.g., (1) and (1′), allows an IR system to distinguish between two very different statements. Similarly, Rextor can differentiate between prepositional phrases (2) and adjectival modification (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) and (3′).

Similarly, Rextor is capable of performing linguistic normalization at the syntactic and morphological levels. Consider these sets of examples originally presented in the introduction:

(4) What is Bill Gates' net worth?
(4′) What is the net worth of Bill Gates?
< "net worth" related-to "Bill Gates" >

(5) John gave the book to Mary.
(5′) John gave Mary the book.
(5″) Mary was given the book by John.
< John is-subject-of give >
< book is-direct-object-of give >
< Mary is-indirect-object-of give >

(6) The president surprised the country with his actions.
< president is-subject-of surprise >
< country is-object-of surprise >
< surprise with actions >

(6′) The president's actions surprised his country.
< actions related-to president >
< actions is-subject-of surprise >
< country is-object-of surprise >

(7) Over 22 million people live in Taiwan.
< "22 million" is-quantity-of people >
< people is-subject-of live >
< live in Taiwan >

(7′) The population of Taiwan is 22 million.
< population is "22 million" >
< population related-to Taiwan >

With relations, different surface forms of expressing the "possession relation" may be normalized into the same structure, e.g., (4) and (4′). Similarly, alternative surface realization of the same verb-headed relation can be recognized and equated with each other by writing different extraction rules that generate the same relations, e.g., (5), (5′), and (5″). The process of normalization will hopefully lead to greater recall in information retrieval systems. Note that (6) and (6′) demonstrate a limitation of Rextor, namely its inability to deal with alternative realizations of verb arguments. Also, the system does not have any notion of semantics, and thus is unable to equate two sentences that have the same meaning, e.g., (7) and (7′). Although it is certainly possible to manually encode such semantic knowledge as extraction and relation rules, this solution is far from elegant.

A potential solution to this semantic variations problem is to borrow the solution employed by Start. A ternary expression representation of natural language mimics its syntactic organization, and hence sentences that differ in surface form but are close in meaning will not map into the same structure. In order to solve this problem, Start deploys "S-rules" (Katz and Levin, 1988), which are reversible syntactic/semantic transformational rules that render explicit the relationship between alternate realizations of the same meaning. For example, a buy expression is semantically equivalent to a sell expression, except the subject and indirect objects are exchanged. Because many verbs can undergo the same alternations, they can in fact be grouped into verb classes, and hence governed by the same S-rules. Thus, S-rules can be viewed as metarules applied over ternary expressions. A similar technique for handling both syntactic and semantic variations can be found in (Grishman, 1995; Jacquemin et al., 1997). Both utilize metarules (e.g., for passive/active transformation) applied over textual patterns in order to generate and handle variations.

Below we present a concrete example of how Rextor could potentially improve the performance of existing keyword search engines dramatically. We indexed an electronic version of the Worldbook Encyclopedia at the sentence level using the following two techniques:

1. A simple inverted keyword index. All stopwords are thrown out, and all content words are stemmed. Retrieval was performed by matching content words in the query with content words in the encyclopedia articles.

2. A ternary expressions index using the relations generated by Rextor. The grammar was written to extract possessive relations, description relations (adjective-noun modification), prepositional relations, subject-verb relations, and verb-object relations. Retrieval was performed by matching ternary expressions from the query (extracted using a separate grammar) with ternary expressions extracted from the encyclopedia articles.

The following shows the results of the keyword search engine:

Question: What do frogs eat?

Answer:

(R1) Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders.

(R2) Bowfins eat mainly other fish, frogs, and crayfish.

(R3) Most cobras eat many kinds of animals, such as frogs, fishes, birds, and various small mammals.

(R4) One group of South American frogs feeds mainly on other frogs.

(R5) Cranes eat a variety of foods, including frogs, fishes, birds, and various small mammals.

(R6) Frogs eat many other animals, including spiders, flies, and worms.

(R7) . . .

After removing stopwords from the query, our simple keyword search engine returned 33 results that contain the keywords *frog* and *eat*. However, only (R1), (R4), and (R6) correctly answer the user query; the other results answer the question "What eats frogs?" or otherwise coincidentally contain those two terms. (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 in the subject position and a query in which the frog is in the object position. However, by parsing subject-verb-object relations using Rextor, a ternary expressions indexer can effectively filter out irrelevant results, returning the three correct responses. While indexing relations may potentially lower recall, due to unanticipated constructions, it has a tremendous potential in increasing precision.

Furthermore, consider the following queries, in which Rextor would outperform traditional keyword engines:

(8) How many South Koreans were recently allowed to visit their North Korean relatives?

(9) Where did John see Mary?

(10) Regarding what issue did the president of Russia criticize China?

(11) Are electronics the biggest export from Japan to the United States?

A traditional search engine using the bag-of-words approach would suffer from poor precision when faced with the above queries. Many verbs take arguments of the same semantic type, and in most of these sentences, reordering the verb arguments drastically alters their meaning. For example, a keyword search engine would not be able to distinguish between a question regarding South Koreans visiting North Korea and North Koreans visiting South Korea (8) because both queries have the same keyword content. Similarly, the keyword approach would be unable to determine who did the seeing (9), or who did the criticizing (10). Modification relations also pose difficulties to the bag-of-words paradigm, e.g., was it the North Korean or South Korean relatives (8)? Was it the president of Russia or the president of China (10)? Furthermore, there are some constructions whose meaning critically depends on relations between the entities, e.g., (11), because "from `X` to `Y`" and "from `Y` to `X`" usually differ in meaning.

The current version of Rextor is merely a prototype; thus, we have made minimal attempts to optimize its processing speed. On a Pentium III 933 MHz Linux system with 512 megabytes of RAM,[6] analyzing a sentence in the Worldbook Encyclopedia required 0.0378 seconds on average. This translates into a content analysis rate of roughly 340 words a second, or approximately 11.4 megabytes of text per hour. Although the system composed of Rextor and the ternary expressions indexer is slower than the simple keyword indexer, we believe that the potential to dramatically increase precision offsets the longer processing time.

[6] However, Rextor is not a memory-intensive system; RAM utilization during trial runs was rather low.

This paper presents only the first stage of an linguistically-motivated information retrieval system. Although we have presented the results of a preliminary investigation into the effectiveness of this approach, we cannot draw any conclusions until more comprehensive tests have been conducted. However, many prior techniques used in natural language information retrieval (e.g., head/modifier pairs) can be expressed within the Rextor framework, and furthermore the system provides a playground for experimenting with new techniques. Thus, we believe that our approach shows great promise in moving towards higher performance information retrieval systems.

7 Conclusion

This paper presented a scheme for integrating natural language processing and information retrieval by adopting a finite-state model of language and a ternary expression representation of document content. We provided justification for our language model and representational structures in both linguistic and empirical terms. Rextor is an implementation of our ideas — it not only integrates many previous natural language indexing techniques, but also provides a sufficiently general framework for much future experimentation. Although we have not yet conducted comprehensive tests, the extraction of "meaning" from documents using Rextor promises to better fulfill users' information needs.

8 Acknowledgments

We would like to thank Sue Felshin for her insightful comments in reviewing drafts of this paper.

References

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1988. *Compilers—Principles, Techniques, and Tools*. Addison-Wesley.

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

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

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*.

Noam Chomsky. 1959a. A note on phrase structure grammars. *Information and Control*, 2:393–395.

Noam Chomsky. 1959b. On certain formal properties of grammars. *Information and Control*, 2:137–167.

Kenneth W. Church. 1980. On memory limitations in natural language processing. Technical Report TR-245, MIT Laboratory for Computer Science.

Bruce Croft and David D. Lewis. 1987. 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)*.

Joel L. Fagan. 1987. Experiments in automatic phrase indexing for document retrieval: A comparison of syntactic and non-syntactic methods. Technical Report TR87-868, Cornell University, Department of Computer Science, September.

Ralph Grishman and John Sterling. 1993. New York University: Description of the PROTEUS system as used for MUC-5. In *Proceedings of the 5th Message Understanding Conference (MUC-5)*.

Ralph Grishman. 1995. The NYU system for MUC-6 or where's the syntax. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*.

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

Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Roche and Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, Massachusetts.

Christian Jacquemin, Judith L. Klavans, and Evelyne Tzoukermann. 1997. Expansion of multi-word terms for indexing and retrieval using morphology and syntax. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*.

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

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

Boris Katz. 1980. A three-step procedure for language generation. AI Memo 599, MIT Artificial Intelligence Laboratory.

Boris Katz. 1990. 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.

Boris Katz. 1997. 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)*.

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

Fernando Pereira and Rebecca Wright. 1991. Finite-state approximation of phrase structure grammars. In *Proceedings of the 29th Meeting of the ACL*.

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

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

J. Thorne, P. Bratley, and H. Dewar. 1968. The syntactic analysis of English by machine. In Donald Michie, editor, *Machine Intelligence 3*. Edinburgh University Press.

William A. Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10).

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