Generating Relations from Natural Language Using REXTOR

Boris Katz, Jimmy Lin & Sue Felshin

Artificial Intelligence Laboratory
Massachusetts Institue Of Technology
Cambridge, Massachusetts 02139

http://www.ai.mit.edu

The Problem: Everyone wants to be able to find information rapidly and conveniently. Ideally, we would like to have a system with the understanding of a human being and the perfect memory of a computer. A user should be able to pose a question in plain English—for example, "what do frogs eat"—and get a sensible answer, such as "Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders."

A full natural language processing (NLP) system should have the language comprehension of a human, but NLP has not yet advanced to that level. When applied to information retrieval (IR), NLP systems suffer from poor recall because data cannot be parsed quickly or accurately enough, so that too little knowledge is correctly indexed.

The field of IR has traditionally sidestepped NLP and relied on simple, fast techniques, primarily the "bag-of-words" approach, which equates the weighted component keywords of a document with its semantic content. Keyword-based IR systems suffer from poor precision because they will retrieve documents containing query keywords regardlesss of whether the keywords are related to the sense of the query. For example, such a system might erroneously answer "what do frogs eat" with "Bowfins eat mainly other fish, frogs, and crayfish", because both query and answer contain the words 'frogs' and 'eat'.

Motivation: Keyword-based IR is clearly useful to a degree, as demonstrated by the popularity of Web search engines, yet too many searches return a large number of irrelevant links. A solution to the full NLP problem, which would yield high precision *and* high recall, could still be years away. In the interim, we would like to build a system that combines the high recall of IR with the high precision of current NLP.

Previous Work: For a more detailed description of REXTOR, see [2].

Approach: We wish to bridge the gap between natural language and information retrieval by distilling natural language text into a representational structure that is amenable to fast, large-scale indexing. We believe that a finite-state model of natural language with ternary expression representation is currently the most suitable combination for this task.

Two major problems in producing representations with full NLP systems are that language is so complex that it currently takes too long to parse large amounts of text (and much text can not be parsed at all without human intervention) and that when we do manage to parse, the resulting representations are so detailed that they overwhelm current indexing and retrieval schemes with too much data.

While a finite state parser cannot perfectly model theoretical human linguistic capabilities, its computational power is good enough for practical purposes; that is, a finite state parser can capture most of the structure of the language which humans actually produce.[3] By substituting a finite state parser for a fully capable parser, we can parse rapidly and still find most of the linguistic relationships within language. We can customize our grammar to parse more or fewer relationships, so as to find and index only as many relationships as our indexer can efficiently index and retrieve.

Rather than representing linguistic relationships as syntactic parse trees or semantic case frames, we represent them as ternary expressions. Ternary expressions are two place predicates of the form {subject relation object}, for example {frog eat insect} or {animal include earthworm}. For 20 years, they have been successfully used by the START system [1] to store knowledge and answer queries. Ternary expressions fail to capture some subtleties of language, but they represent the basic sense of language, and because they are so simple in form, they can be indexed and retrieved rapidly.

Our finite state parser uses a grammar of extraction rules to find syntactic structures within text and relation rules to construct ternary expressions from syntactic structure.

Given the noun phrase *the big, bad wolf of the dark forest*, REXTOR uses extraction rules to recognize two "noun groups": *the big, bad wolf* and *the dark forest*. The corresponding relation rule triggers, and generates the following relations (ternary expressions): < big `describes` wolf >, < bad `describes` wolf >, < dark `describes` forest >. In addition, the entire noun phrase *the big, bad wolf of the dark forest* will be recognized as a complex noun group. This will result in the following relation: < wolf `related-to` forest >.

The relations are then indexed, and queries are answered by matching ternary expressions from the query with ternary expressions extracted from the indexed text.

Difficulty: 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. More notably, when making use of the linguistic relations in text, it is necessary to detect when structures have similar meanings but differ in form, e.g., "eat" should match "feed on"; "the frog's diet is X" should match "the frog eats X". Otherwise the system's recall will suffer as it fails to match queries to text that was indexed differently. The ideal full NLP system would completely understand not just the structure but also the meaning of all text. REXTOR, on the other hand, is capable of equating minor variations in language, such as synonyms, active/passive, etc., but it cannot yet recognize greater variations.

Impact: By using simplified NLP techiques which are rapid yet retain much of the intelligence of full NLP, and applying them to the domain of IR, we should be able to improve on keyword-based IR algorithms without suffering the drawbacks of full NLP systems, as shown in Figure 1. Here, a standard keyword-based IR system retrieved 32 answers from an encyclopedia, two of which were correct, while REXTOR returned only the two correct answers.

Question: What do frogs eat?

Answer:

Keyword-based IR system:	**REXTOR:**
(R1) Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders.	(R1) Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders.
(R2) Bowfins eat mainly other fish, frogs, and crayfish.	(R2) Frogs eat many other animals, including spiders, flies, and worms.
(R3) Most cobras eat many kinds of animals, such as frogs, fishes, birds, and various small mammals.	
(R4) Cranes eat a variety of foods, including frogs, fishes, birds, and various small mammals.	
(R5) Frogs eat many other animals, including spiders, flies, and worms.	
. . .	
(R32)	

Figure 1: Example of keyword-based IR vs. REXTOR.

Future Work: The current version of REXTOR is a prototype; we need to extend the grammar, index large amounts of text, and test a variety of queries to vet the effectiveness of this approach. We would like to explore combined approaches to resolve some of the speed and low recall issues with REXTOR; perhaps we could use keyword-based IR as a first pass and REXTOR as a second pass.

Research Support: This research is funded by DARPA under contract number F30602-00-1-0545 and administered by the Air Force Research Laboratory.

References:

[1] B. Katz. Using English for Indexing and Retrieving. in P. H. Winston and S. A. Shellard (eds.), *Artificial Intelligence at MIT: Expanding Frontiers,* vol. 1, MIT Press, 1990.

[2] B. Katz and J. Lin. REXTOR: A System for Generating Relations from Natural Language. (to appear in *Proceedings of ACL 2000 Workshop on Recent Advances in Natural Langauge Processing and Information Retrieval*).

[3] K. Church. On memory limitations in natural language processing. *Technical Report TR-245*, MIT Laboratory for Computer Science, 1980.