Problem

Natural language (human language, such as English) is difficult for computers to interpret. Because sentences are difficult for computers to process, a new model for information storage is required.

Solution

I have created a model that graphically represents sentences in a format that can be understood by a computer. In a graphic form, information and be understood by a computer. When multiple graphs are combined, references to the same object or idea are combined, and several sentences can continue to add information. Queries can be executed on the graph. Information that is extracted from the graph is located based on the meaning than simple word matching.

Parameters

The model must satisfy the following:
Multiple graphs can be combined together.
Clear and efficient identification and association of phrases
Allow for the extraction of information by both computers and humans

Background

Natural Language's ambiguous and complex structure makes it difficult to process. Current algorithms for extraction of information from text use simple keyword extraction. Most information retrieval systems (IRS's) do a simple word for word search--they simply look for the presence of keywords, then return the relevant portion of the searched document. Others use probabilistic models and n-grams to try to make a more educated guess at the information that a user desires, but none yet looks at the meaning of the document or query sentence.

There are several theories to formalize English syntax. The two most popular are Dependency Grammars, and Constituency Grammars. Both system identify phrases in a sentence, but their definition of a phrase differs slightly. My graphic structure is more similar to the dependency grammar structure. It is possible to convert from a dependency relation to my graphic structure.

Link-grammar is a formal grammatical system developed at Carnegie Mellon University. It processes sentences, by following a set of rules to link words. These linkages can be used to identify the constituents (subject, object, verb, etc...) of the sentence. They also have several relevant properties. One is that the various pieces of a sentence are structurally identical across the sentence. These sub-components each can be things such as noun instances or verbs (Sleator et al. 1). The conditions that link grammar places on the formation of links also gives a starting point for some of the criteria for the design of my model.

In link grammar, words are given connectors that allow them to form links with other words. A parsed sentence can be generated by following a set of rules to link the words together. These rules are matching like links, making sure that all words link to another word, and that link should not cross. A visual representation of the links can be seen in the Figure 1.

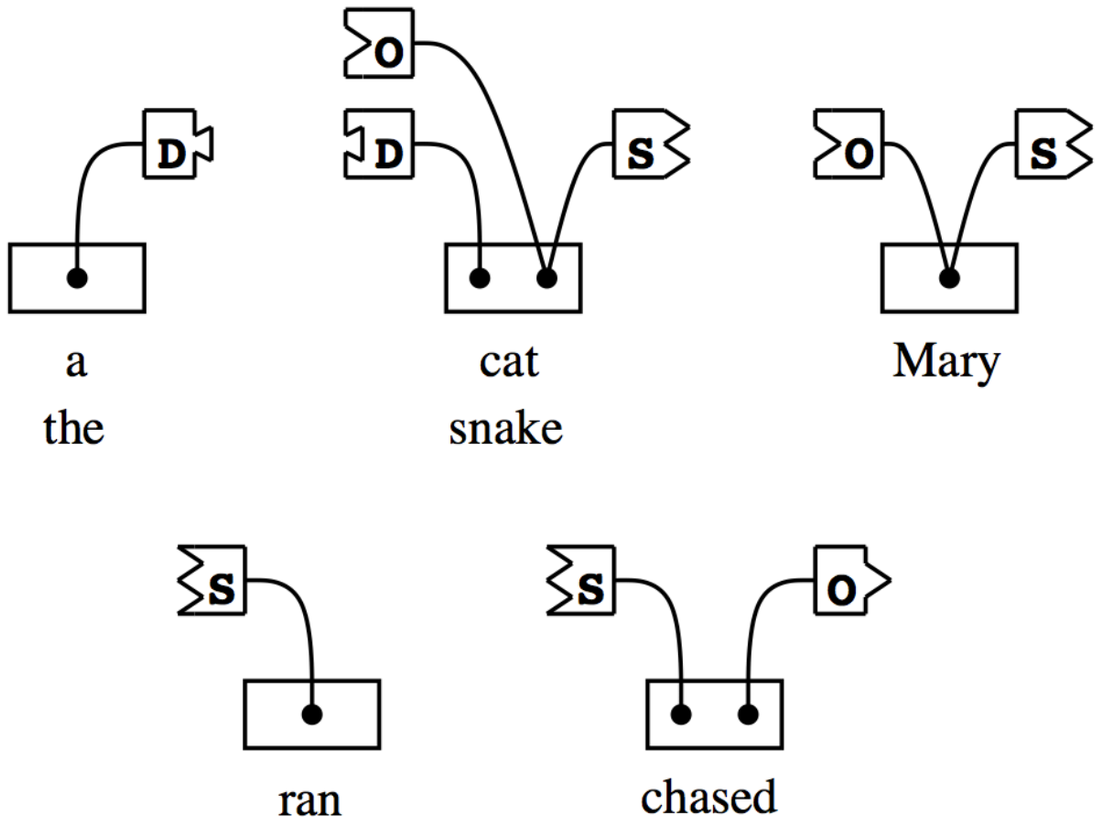

Figure 1: words have connectors. (Sleator et al. 2)

Graphs

I am modeling sentences using a directed graph. A graph in computer science is a structure that contains objects (often called nodes), that are connected together by links. These nodes can represent anything. The links show relationships between nodes. In a directed graph, links can have direction. The figure on the left, shows an visual representation of a directed graph of numbers. The green arrows are links, and the circles are nodes.

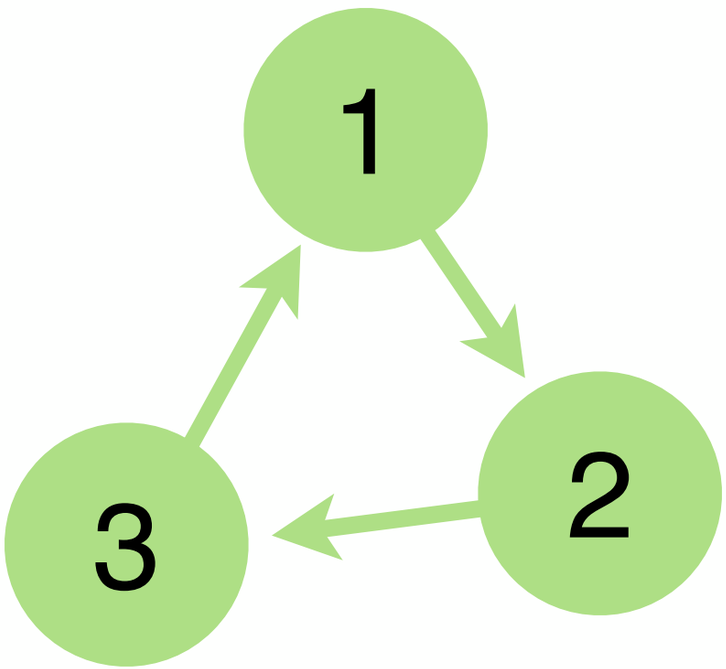

Figure 2: a directed graph

Representation of Sentences

The various entities in a sentence are represented by nodes, and the relationships between them shown as links. To convert a sentence into a graphical format, a parser follows a set of rules.

Dictionaries

Some information is always true. In the case of the english language, the properties of individual words is constant. This information can be stored in a dictionary graph. This graph will be merged into graphs that store specific information to help with querying. *Figure 3* shows an excerpt from a dictionary.

Modeling Method

To model a sentence, I represent each word as a node and add links between these nodes to give the sentence meaning. This can be seen in *Figure 4*. Figure 4 is the graph that would be generated on the input sentence: “The dog chases the the cat.”

Anonymous Nodes

In *figure 4,* some nodes can be seen that are prefixed with “@”. These are anonymous nodes. Anonymous nodes are used to represent unnamed entities in a sentence. While linking “dog” to “noun” says that “dog” is a “noun”, linking an anonymous node to “dog” says that the anonymous node represents a dog. Anonymous nodes take the form of

“*@identifier*” where “*identifier*” is a unique identifier to distinguish between anonymous nodes. These are used to represent instances of the words in a sentence. See *Instances of Words* and *Figure 4* for an example of their usage to model English phenomena.

Handling English Phenomena

The way in which various components of English sentences should be graphed is identified below.

Nouns and Verbs

At its root, every sentence in English is a noun instance performing a verb instance. The graph of this structure is the noun instance node with an link pointing at the verb instance node. This can be seen in Figure 5.

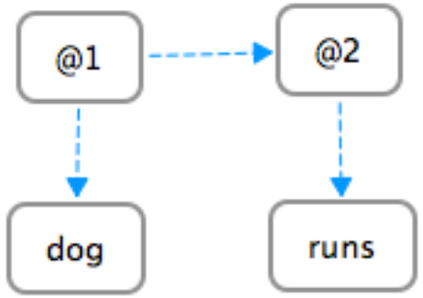

Figure 5. This is a graph of the sentence: “The dog runs.”

Adjectives

Adjective instances are pointed at by a link from the noun instance that they modify. If the adjective instance has adjective instances that modify it in turn, it points at those instances.

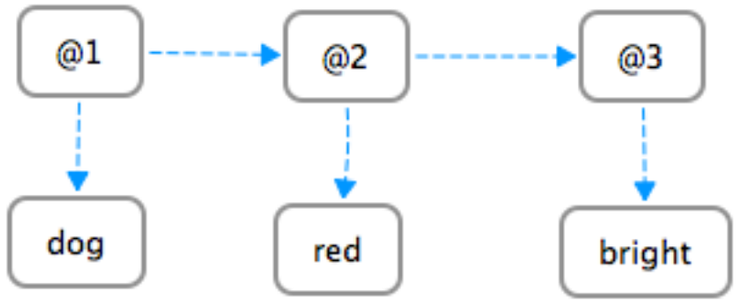

Figure 6. Nodes and links modeling that “bright” modifies “red” modifies “dog”.

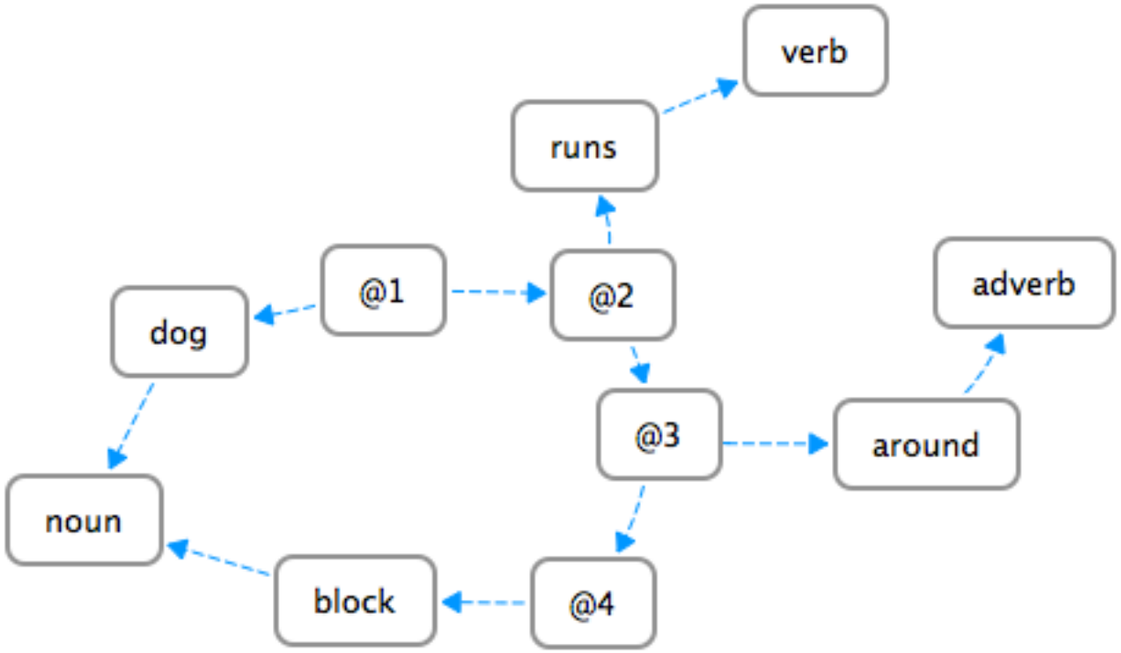

Figure 7. The graphical model generated by the sentence: "The red dog runs around the block."

In Figure 7, $@1$ represents an instance of the noun "dog", $@2$ an instance of the verb "run", $@3$ is an instance of the adverb "around", and $@4$ is an instance of the noun "block".

Base Information Storage

Information is stored in two graphs: the *dictionary graph* and the *environment graph*. The dictionary holds information about the English language and constitutes the lexicon of the model. An environment holds specific information. An environment may contain a story about a dog, the results of an experiment, or any other information.

Question Graphing

The graph of a question takes a similar form to the graph of a statement. The difference is that one or more of the nodes in a question graph are marked as unknown. To mark a node as unknown, a `?` is used in the place of the node, and is prefixed by an identifier. The resulting node takes the form of `*identifier*?`. The identifier allows the model to differentiate between different unknowns. An example of a question can be seen below in Figure 6.

Figure 6: An example of a question node used to query information from an environment.

Information Retrieval

To retrieve information from a source graph that results from the structure above, the model uses a *question graph*. The *question graph* can be compared to the *environment graph* which is where the information is stored. The results of this comparison will solve for the unknowns in the *question graph*. An proof-of-concept design can be seen in the *Query Program* section below. The form of the *question graph* is discussed above.

Query Program

To prove that the model is comprehensible by computers, I developed a “querying” program to extract information. The querying program is written in the Python programming language. The branching recursive design of my code, does not lend itself well to implementation as a traditional flow chart. However, a simplified flow from inputs to outputs of my code be seen in the flowchart below.

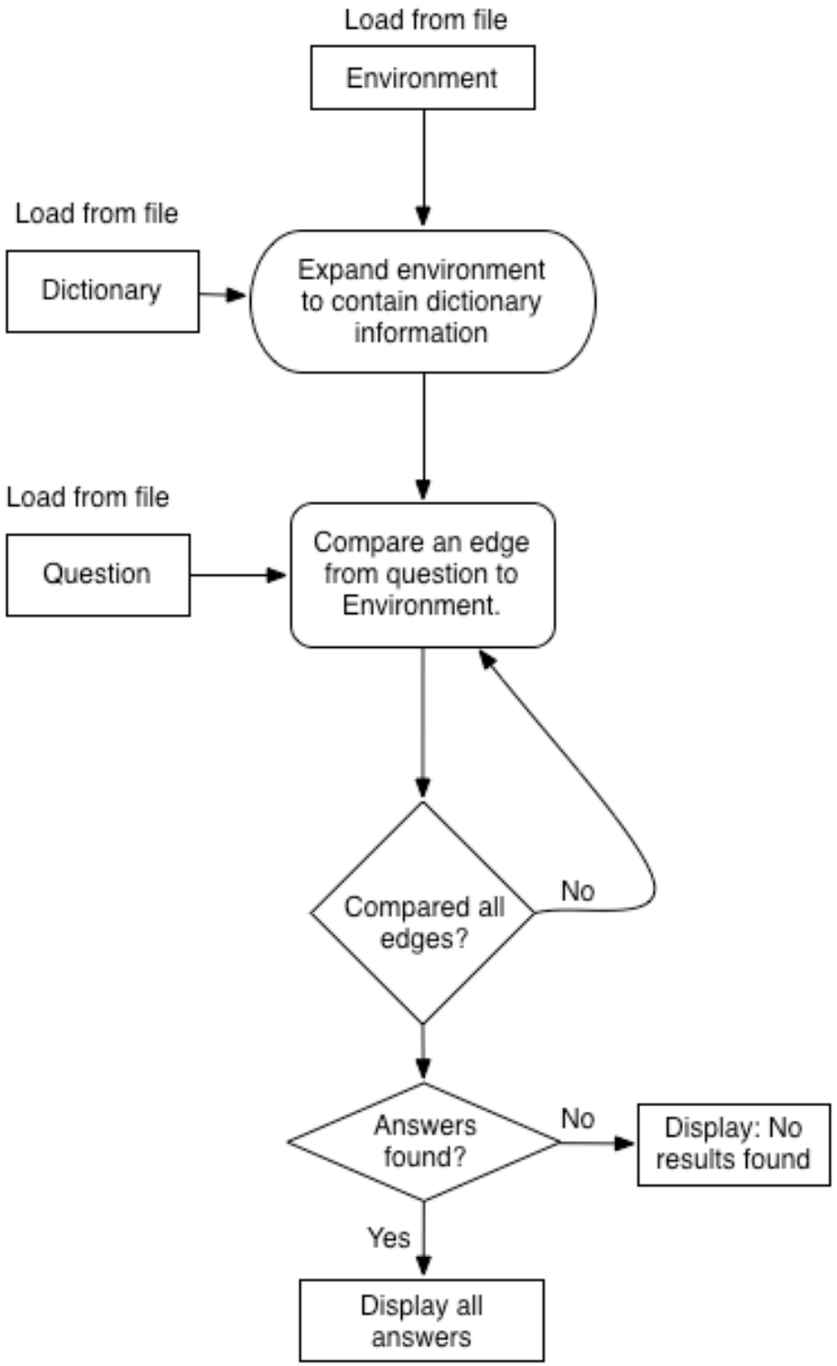
Load from file
Environment
Load from file
Dictionary
Expand environment to contain dictionary information
Load from file
Question
Compare an edge from question to Environment.
Compared all edges?
No
Answers found?
No
Display: No results found
Yes
Display all answers

Figure 7: Flowchart of the query algorithm.

The code begins by loading maps from the dictionary and environment. It then merges the dictionary into the environment. The program then loads the query file and begins comparing the links of the query to those in the environment. Once it has finished comparing links it displays the answers that are found.

more in-depth explanation

The Querying Algorithm

1. Get a list of links in the query.
2. Grab a link from the query, if there are no links left, return answer set.
3. Look for matches to the link in the environment. (If the link contains unknown nodes, more than one matching link may be found).
4. For each match, work through on the assumption that the match is valid:

make a new list of query links to reflect the match and go back to step 2
if link contains a query node, add match to answer set.

Results

I have developed a programmatic representation of the theoretical model. This representation can load graphs from a file format I developed or from a mindnode file using a parser I developed. I also developed a query program that utilizes my graph library. My program is capable of taking graph files and finding the answers to questions entered in a query graph.

*** Sheyne: Show pics and elaborate on user interface
*** Sheyne: possibly show the public interface for my library

Conclusion

I have proven that graphs are a feasible way to extract information from sentences. I have developed a functional graph implementation, and querying algorithm. Sentence input still needs to be implemented. This model proves the viability of graphs as a way to model English sentences.

Future Applications and Development

Extensive research using Google Scholar and the Weber State University academic database did not turn up any sign that others were attempting this approach. The novel

results that I have produced thus far indicate that graphs are a viable method for the storage of arbitrary natural language information.

I plan to develop an automatic conversion program, that would take English sentences and build their corresponding graph. Graphically modeling sentences would simplify and improve the accuracy of Information Retrieval Systems. Sentence graphs are also an efficient method for information storage.

Bibliography

*** Sheyne: add, define the search engines that I used.