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. It stores the information in the form of a directed graph. 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 rather than simple word matching.

Parameters

The model must satisfy the following:
Multiple graphs can be combined together.
Phrases can be clearly and efficiently identified.
Information can be extracted 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 basic 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 study common sequences of words to identify words roles in sentences. Neither looks to model the meaning of a sentence directly.

There are several approaches to formalizing English syntax. The two most popular are Dependency Grammars, and Constituency Grammars (Covington, 2001). Both systems identify phrases in a sentence, but their definition of a phrase differs slightly. My graphic structure is similar to the dependency grammar structure and 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--see figure 1. 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 can each 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: form links between two connectors of the same type, make sure that all words link to another word, and ensure that links do not cross. A visual representation of the links can be seen in the Figure 1.

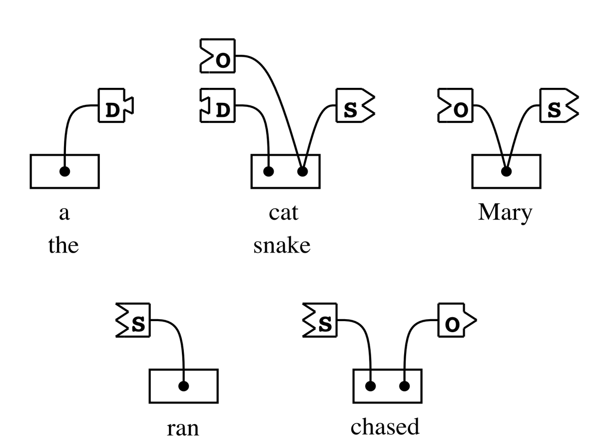

Figure 1. In link grammar, words have connectors. Connectors with the same label (symbolized by connector shape in the diagram) fit together. Figure by Sleator et al.

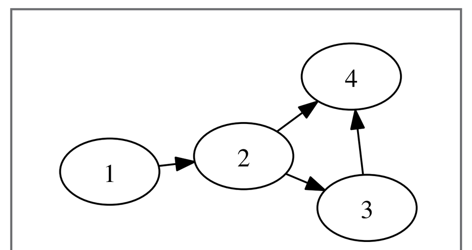

Figure 2. A visual representation of a directed graph of numbers. The arrows are links, and the circles are nodes.

Graphs

I am modeling sentences using a directed graph. A graph in computer science is a structure that contains nodes (the things that the graph relates together). Nodes are connected together by links (Figure 2) which show relationships between nodes. In a directed graph, links have direction.

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 to generate links and nodes.

Dictionaries

Some information is defined by a language. This information is things like the meanings of words. In the case of the English language, the properties of individual words are constant. This information can be stored in a dictionary graph. This graph will be merged into graphs that store specific information. Because the dictionary is merged in, the parsing algorithm can make use of it to determine the meanings of words. Figure 3 shows an excerpt from a dictionary.

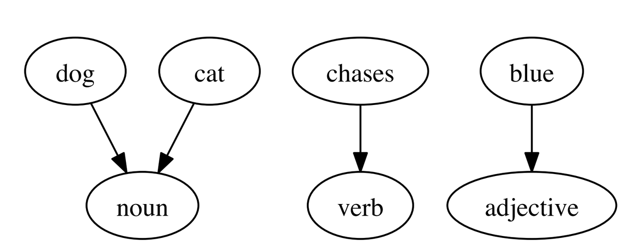

Figure 3. Example dictionary, here words are connected by links to show relationships.

Modeling Method

To model a sentence, I represent each word as a node and add links between these nodes to give the sentence meaning. Figure 4 is the graph that would be generated on the input sentence: "The dog chases 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 Figures 4, 5, and 6 which all show how anonymous nodes can be used 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. To graph this, a noun instance (represented by an anonymous node) links to a verb instance (also an anonymous node) as demonstrated in Figure 5.

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, it points at those instances.

Figure 6 shows how adjectives can modify nouns and other adjectives. "Dog" is modified by "happy" and "red". "Red" is modified by "bright".

Environment Graph

Information is stored in the environment graph. This graph begins as a copy of a dictionary graph, which contains the definitions of English words. As additional information is gathered, by user input or other means, the environment graph grows. Each sentence that is added is merged into the environment graph.

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 in Figure 7.

Information Retrieval

To retrieve information described by a query graph, the query graph is compared to the environment graph. The results of this comparison are the values for the unknowns in the question graph. A proof-of-concept design can be seen in the Query Program section. The form of the question graph is discussed above.

Parsing

To convert an English sentence into my graphical model, I programmed a parsing tool. The tool splits the sentence into component words, and looks up a function to perform for each word. These functions can define values. They can also request values that have already been defined, or the values that will be defined. An example of the flow of parsing can be seen in Figure 8.

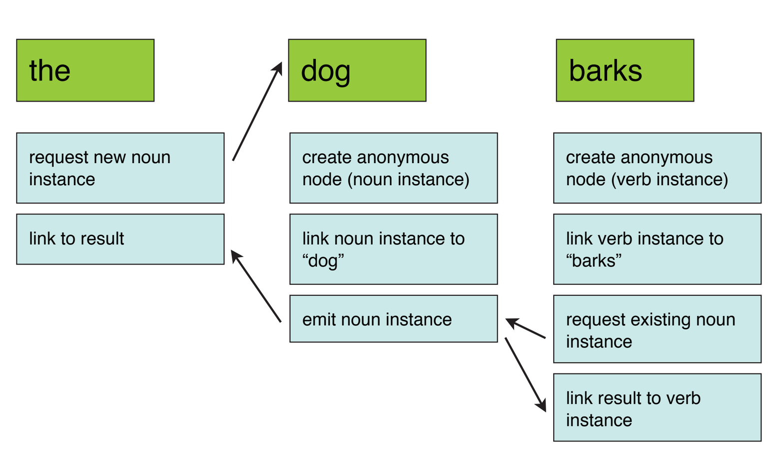

Figure 8. Example of parsing for sentence: "The dog barks". Execution flows down from each word unless redirected by a request.

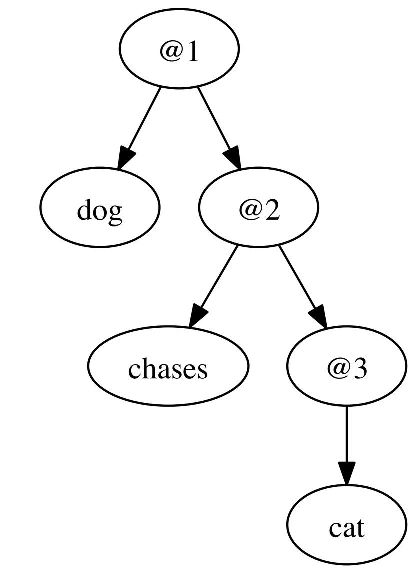

Figure 4. Graph of sentence: "The dog chases the cat." The nodes beginning with "@" are anonymous nodes. These are nodes that represent nameless things. The number allows them to be unique.

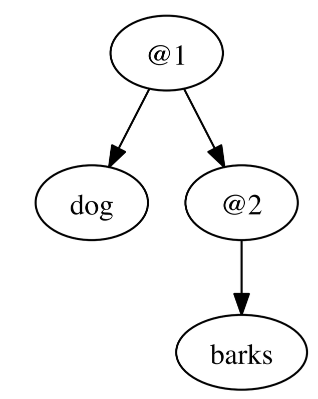

Figure 5. Graph of sentence: "The dog barks. "

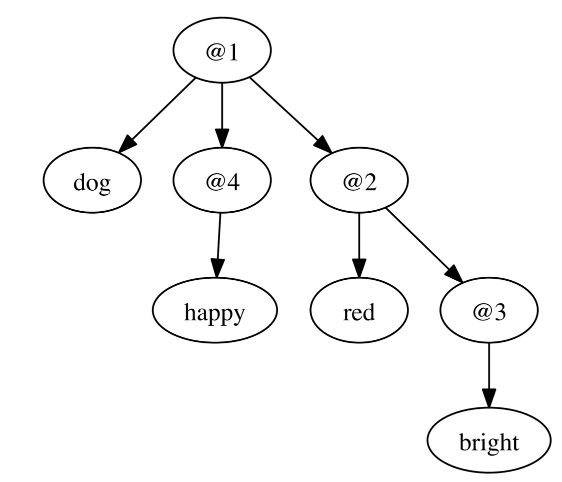

Figure 6. Graph of: "The bright red dog is happy."

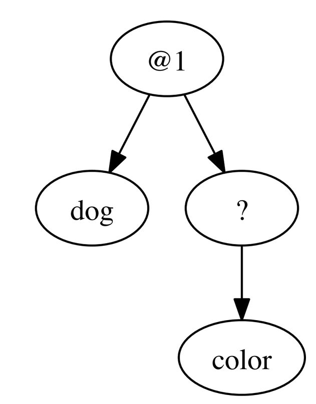

Figure 7. An example of a question node used to query information from an environment graph. This graph asks the question, "What color is the dog?"

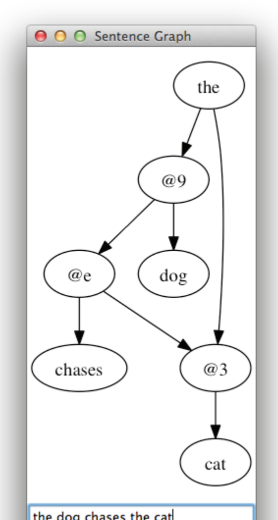

Figure 10. Screenshot of the grapher program running.

Query Program

To prove that the model is comprehensible by computers, I developed a "querying program" to extract information from the sentence. The querying program is written in the Python Programming Language. The branching recursive design of my program does not allow for the creation of an exact flowchart. I created a simplified flowchart (Figure 9).

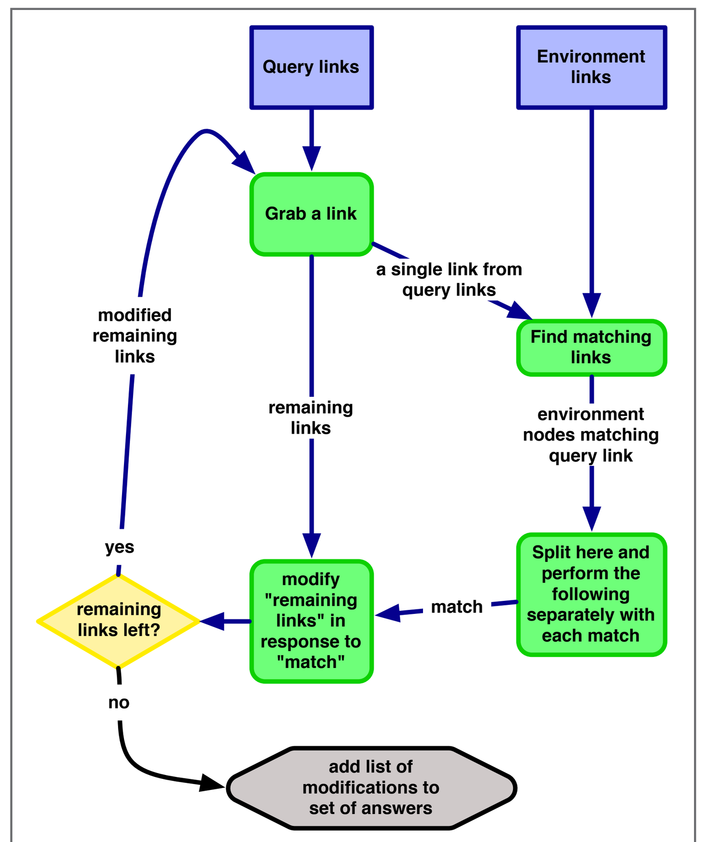

Figure 9. Details of the functionality of the "query algorithm."

Query Algorithm

1. Get the set of links from query.
2. Pick a link from the retrieved set, if there are no links left go to step 5.
3. Look for matches to the link in the Environment Graph. (More than one matching link may be found if one or more of the nodes in the link pair are unknowns.)
4. For each match, assume that the match is valid:
   a. If link contains a query node, add the node and it's replacement value to the answer set.
   b. Modify query link list to reflect match and go to step 2.
5. Return answer set.

Results

I have developed a programmatic representation of the theoretical model. This representation takes the form of a Graph class. Instances of the class can be queried for information with a query method, have new links added to them with their add method, and merge with other graphs with their combine method.

Conclusion

I have proven that graphs are a feasible way model and facilitate for the interpretation of information from natural language 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 natural language sentences. Interpretation of graphs allows for the extraction of information with improved ease and relevance than keyword extraction on input sentences.

Future Applications and Development

Extensive research using Google Scholar and the university library academic database did not turn up any sign that other people were attempting this approach. The results that I have produced thus far indicate that graphs are a viable method for the storage of arbitrary natural language information in a form that demonstrates the conceptual meaning.

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. Graphs of sentences are also an efficient method for information storage.

I plan to improve the range of sentence type this can handle, as well as designing a way of parsing questions to simplify querying. I also plan to improve the efficiency of the various components of the code.

My model and code can be used to improve search engines, artificial intelligence programs, speech to text software, and human-computer interfacing.

Bibliography

Callan, J., & Croft, W. (1992). The INQUERY retrieval system. of the third international conference.

Covington, M. (2001). A fundamental algorithm for dependency parsing. Proceedings of the 39th annual ACM southeast.

Damashek, M. (1995). Gauging Similarity with n-Grams: Language-Independent Categorization of Text. Science (New York, N.Y.), 267(5199), 843-8. doi:10.1126/-science.267.5199.843

Ifrim, G., & Weikum, G. (2008). Fast logistic regression for text categorization with variable-length n-grams. Learning, 354–362. ACM Press. doi:10.1145/1401890.1401936

Kübler, S., McDonald, R., & Nivre, J. (2009). Dependency Parsing. (G. Hirst, Ed.)Synthesis Lectures on Human Language Technologies, 2(1), 1-127. Morgan and Claypool Publishers. doi:10.2200/S00169ED1V01Y200901HLT002

Levesque, K., Librera, S., Nelson, D., & Riley, R. W. (2000). February 2000 Vocational Education in the United States : Toward the Year 2000, (February).

Lewis, D. D. (1993). Natural language processing for information retrieval 1 Abstract 2 Introduction 3 Document retrieval. Communications of the ACM, 39(1), 1-22.

Maybury, M. T., & Pa, M. (2004). New Directions in Question Answering. (M. T. Maybury, Ed.)-Computational Linguistics (Vol. 9, pp. 383–386). AAAI Press.

Mihalcea, R., & Radev, D. R. (2006). Graph-based algorithms for natural language processing and information retrieval. Computational Linguistics, (June), 303-304. Association for Computational Linguistics. doi:10.3115/1614101.1614104

Sleator, D. D. K., & Temperley, D. (1995). Parsing English with a Link Grammar. Third International Workshop on Parsing Technologies, 64(October), 91. ACL/SIGPARSE.