An Enhanced Noun Definition Algorithm

Scott Napieralski
stn2@cse.buffalo.edu

August 26, 2002

Abstract

As part of research into contextual vocabulary acquisition, improvements have been made to Karen Ehrlich's algorithm for defining unknown nouns based on the context in which they appear. The code that implements the algorithm was improved by adding a tracing facility and comments, among other changes that make it easier for a newcomer to the project understand the algorithm. The algorithm itself was enhanced by adding several new categories of information to the definition generated by the algorithm and removing one redundant category. Finally, the problem of storing the definition of a previously unknown word in the knowledge base after it has been defined was explored.

1 Introduction

The work described in this paper was performed as part of the contextual vocabulary acquisition project. Researchers from the University at Buffalo Departments of Computer Science & Engineering and Learning & Instruction are cooperating to develop a greater understanding of the human process of learning new vocabulary from context clues. In order to improve our understanding and test our theories, we have made use of a computational algorithm that attempts to model the human process of defining an unknown noun based on context.

The goal of the summer's work was to improve the computational algorithm for defining nouns with respect to both the amount of resources used and quality of the definition. The version of the noun definition algorithm that I attempted to improve was Marc Broklawski's modification [Broklawski 2002] of Karen Ehrlich's algorithm [Ehrlich 1995].

Ehrlich's algorithm examines a SNePS [Shapiro 1999] network for information pertaining to some unknown input word. The information that is found is then filtered and assembled into a definition. This definition should be dictionary-like and also resemble the definition that a person might give an interviewer. Specifically, the algorithm searches for and defines the unknown word in terms of information that falls into the following categories: class inclusions, structure, functions, actions, properties, owners, synonyms, and individuals. If some categories contain information, other categories may not be reported as part of the definition, depending on the relative importance that Ehrlich attached to each category. For example, if the word being defined is "brachet", and if Cassie (a software agent consisting of the SNePS knowledge-representation and reasoning system plus all the knowledge entered into this system) knows that "Fido is a brachet" and that "brachets are dogs", then the algorithm would find "class inclusions: dog" and "individuals: Fido," but would only report "class inclusions: dog." By encountering the unknown word repeatedly in context and filtering out irrelevant information, Ehrlich's algorithm refines the definition toward a more dictionary-like definition.

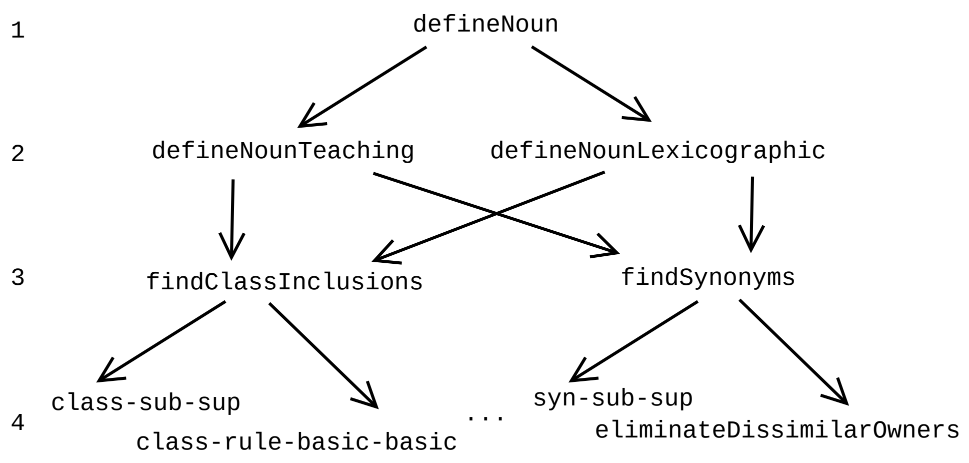

Figure 1: The hierarchy of functions and the trace levels they occupy. An arrow from one function to another indicates that the higher function calls the lower function. Levels 3 and 4 include more functions than are pictured, since level 3 includes about 10 functions and level 4 includes approximately 50 functions.

The motivation to change Ehrlich's algorithm stemmed from a need to reduce the computational resources consumed by the program and a desire to utilize advances made since Ehrlich completed her dissertation. Most of the work that was intended to reduce the running time of the algorithm was performed by Broklawski and required only minor modifications on my part; therefore, I will not describe it in detail here. New work that needed to be integrated into the Ehrlich algorithm included updated case frames for representing knowledge, a new system for default reasoning, and a method for storing generated definitions in the knowledge base.

The changes to the algorithm fall into two categories. First, the structure of the Lisp code that implements Ehrlich's algorithm was significantly changed in order to facilitate the work of anyone wishing to make changes in the future. Second, new categories of information and new ways of recognizing knowledge were added to the algorithm.

2 Changes to Structure

2.1 Comments

In order to make the code that implements Ehrlich's algorithm easier to understand, many comments were added. The purpose of these comments is to explain important and obscure parts of the code. Each function now has a documentation string that explains its intended purpose. In addition, major sections of each function are pointed out or explained using in-line comments.

2.2 Tracing

In order to facilitate future additions to the algorithm and aid in the diagnosis of errors, a tracing facility was added. There are five different levels of tracing (Figure 1), labeled level 0 through level 4. Trace level 0 provides no tracing at all; this level is used to turn off tracing after another trace level has been used. For trace levels 1 through 4, each level includes all of the functions traced at the previous level. Trace level 1 provides a description of the inputs and outputs of the top-level function defineNoun. Trace level 2 provides a description of the inputs and outputs of the functions

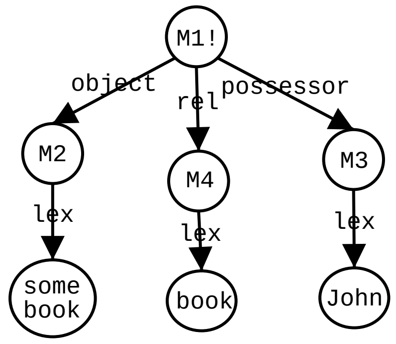

Figure 2: An example of the object-rel-possessor case frame. Semantics: [[M1]] = Some book is John's book.

that are directly called by defineNoun. These include functions to generate the two possible types of definitions, functions to store definitions in the SNePS network, and functions to retrieve definitions from the network. Trace level 3 provides a description of the inputs and outputs of all functions called by the functions included in a level 2 trace. These functions search for information in each of the categories that constitute a definition. For example, one of the functions traced at level 3 is findStructure, which finds information that is reported in the "structural features" slot in the definition. Trace level 4 provides a description of the inputs and outputs of all functions involved in generating and storing definitions.

There are some functions in the code that are not traced. These functions implement the tracing facility itself and format the definition for output. They were not included, since they do not alter the definition, and they would have created unnecessary complexity in the output of the tracing facility.

2.3 Updated Case Frames

Since Ehrlich finished her work on the algorithm, several of the case frames that she used to represent information in the SNePS network have fallen out of common usage and been replaced by other case frames [Shapiro 1996]. In order to make it easy for future researchers to encode passages for analysis by the algorithm, it has been changed to accommodate the case frames that are currently in common use.

In Ehrlich's version of the algorithm, structural features were represented using the object-rel-possessor case frame (Figure 2). While this scheme made sense from a linguistic viewpoint (one can say "This is John's nose") it did not adequately distinguish "John's nose", something that is a part of John, from "John's book", something that John owns. Therefore, the new version of the algorithm uses the part-whole case frame (Figure 3) to represent structural information, such as "John's nose." For representing ownership information such as "John's book," the new algorithm retains the object-rel-possessor case frame.

Ehrlich also made use of the agent-act-object case frame (Figure 4), that was in common use at the time, to represent information about actions. In place of that case frame, the new algorithm uses agent-act-action-object (Figure 5), that has replaced the former case frame.

In order to represent collections, Ehrlich devised a simple scheme, using an objects1-rel-object2 case frame. However, this case frame was implemented inconsistently in the algorithm. Some sections of the code searched for it, while others completely ignored it. Since a suitable theory of collections in SNePS remains an area for future research [Cho 1992], all references to "objects1"

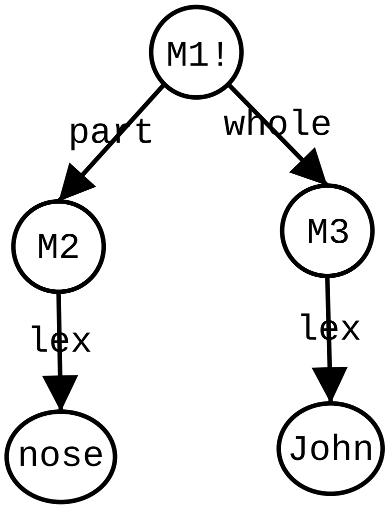

Figure 3: An example of the part-whole case frame. Syntax: If M2 and M3 are individual nodes and M1 is an identifier not previously used, then this figure is a network and M1 is a structured proposition node. Semantics: [[M1]] is the proposition that [[M2]] is a part of the object [[M3]]; in this case, a nose is part of John.

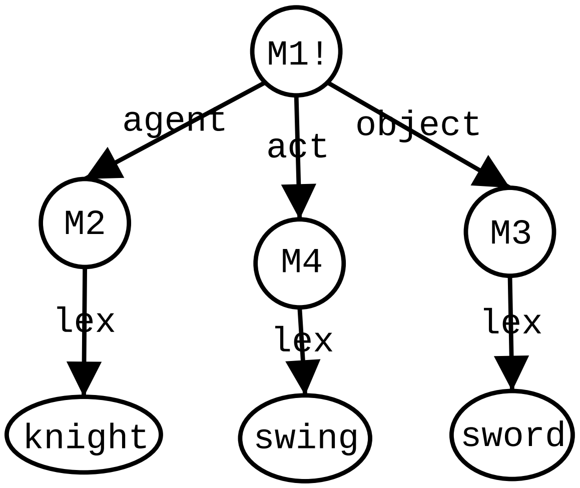

Figure 4: An example of the agent-act-object case frame. Semantics: [[M1]] = A knight swings a sword.

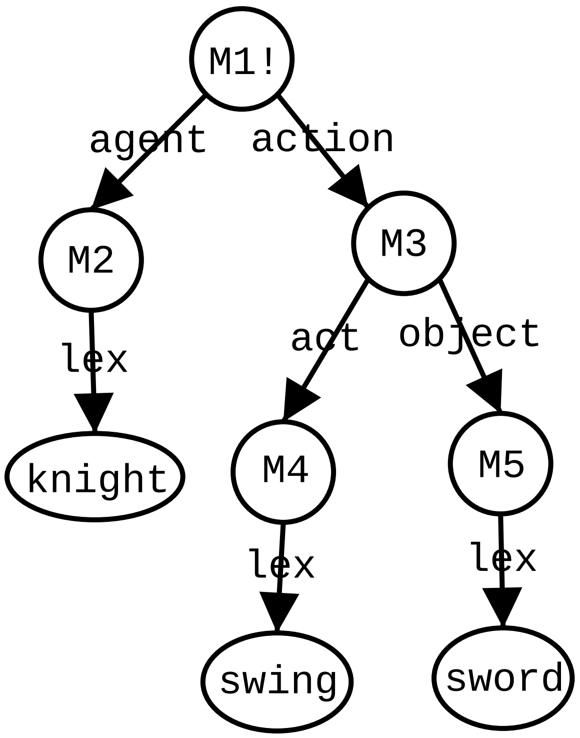

Figure 5: An example of the agent-act-action-object case frame. Semantics: [[M1]] = A knight swings a sword. [[M3]] = A sword is swung.

have been removed from the algorithm in favor of the less expressive but more standard construct "object1". This change does not significantly diminish one's ability to express ideas in SNePS, but does reduce the possible number of paths that must be searched to fill each category in the definition.

2.4 Format of Results

The Ehrlich algorithm used a list of lists to represent and report the definition. This scheme had several deficiencies. First, in order to label information, extra elements (such as "possible class inclusions=") were inserted into the beginning of each sub-list. This led to complications when processing the list. Any time that a category of information needed to be examined, extra code had to be added to ignore the first two or three elements of the list since they were really a label, not members of the category. Second, it was difficult to extract a specific category from the definition. Since different categories were included in the output list based on the available information, there was no way to tell what position a specific category occupied in the list. In one definition, that included class inclusions, properties could occupy the third position in the list of categories. In another definition, that did not include class inclusions, properties might occupy the second position in the list of categories. This left the code without a general method of extracting a specific category from the definition.

In order to remedy these problems, the new algorithm has abandoned the list of categories in favor of a Lisp structure. This has the advantage that categories do not need to be labeled, since each field in the structure is given a name. In addition, each category can be accessed using its name, making it simple to extract information about properties from the whole definition without

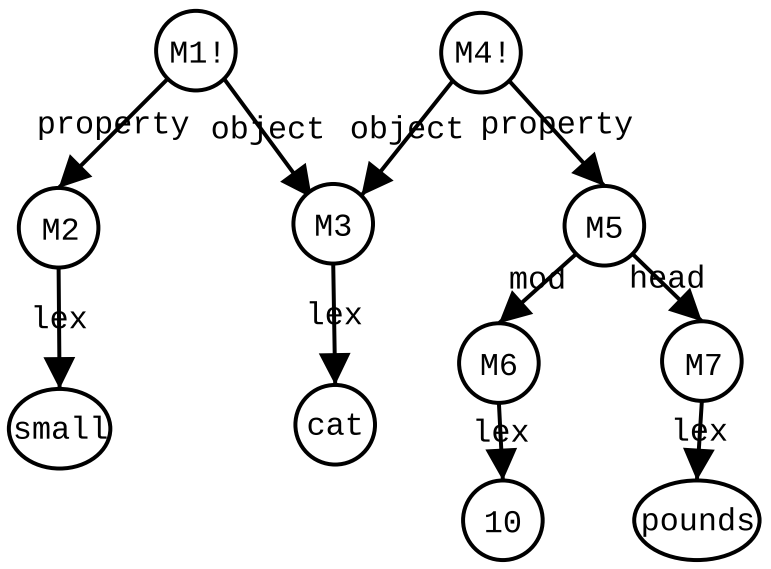

Figure 6: "10 pounds" would not have been reported by Ehrlich's algorithm

knowing or caring that position in the structure is occupied by that category.

In order to simplify the output and regain the readability that Ehrlich's scheme of category labels offered the end user, the new algorithm includes a reporting function. This function prints the name of each category followed by the information in that category in a human-readable format.

2.5 Structure of Search Paths

The inclusion of a dedicated reporting function allowed the search paths used to find information in the network to be simplified by eliminating some arcs that were common to most search paths. These arcs mainly show the words that are used to represent a concept, so their proper place is in a function that generates human-readable output. For each category of information in the definition, the reporting function calls the function lexicalize. This function examines each member of the list it is given and attempts to translate it into a human readable form. The lexicalize function can handle SNePS nodes that are at the head of lex arcs or of a mod-head case frame, and nodes that are the class in a member-class case frame or an object1-rel ISA-object2 case frame.

Once lex and mod-head were searched for in the reporting function, they became unnecessary in functions that construct the definition. Therefore, the `lex-` arc that had previously started each search path in the definition construction functions was removed. This gives us added flexibility to represent the words that denote concepts in different ways, including mod-head, without needing to repeat search paths. Figure 6 provides an example of information that would have been ignored under the old scheme, but can now be found.

In addition, the `class- member` and `object2- object1` paths that had previously started many of the search paths were moved to the reporting function. Since these paths were in the reporting function, they became unnecessary in each of the search functions. When these paths were removed, many of the search functions became duplicates of one another and were consequently removed from the algorithm. Searching for `class- member` and `object2- object1` paths in the reporting function has the benefit of increased flexibility as described above. It also simplifies the task of writing search functions for new categories of information, because authors of these modules will have fewer possible search paths to consider. It is no longer necessary to write three different search paths to handle the cases where the information you are searching for is at the end of a lex arc, at the end of a `class- member` path or at the end of a `object2- object1` path. One simple search path will find all three of these possibilities.

This scheme only addresses the problem of different representations of the information that belongs in the definition. It does not address the problem of different representations of the noun being defined. Just as the information that we are searching for can be represented in several different ways, the noun we are defining can be represented differently. If the noun is basic-level, the representation will use the member-class case frame. If the noun is non-basic-level, the representation will use the object1-rel ISA-object2 case frame. The word we are defining could be at the end of a lex arc or it could be part of a mod-head case frame. Currently, each of these different possibilities has a separate function to handle it. It would be more efficient to handle all of these cases with a single function, as we did with the reporting function. However, doing this for the noun being defined is a more difficult problem, because the relevant parts of the search path are at the end of the path, rather than at the beginning as was the case for the paths that were moved to the reporting function. Condensing the different possible paths leading to the noun being defined into a single search function is a subject for future work.

3 Modifications to the Algorithm

3.1 Categories of Nouns

Ehrlich's version of the noun-definition algorithm had different methods of generating a definition based on which of several categories the noun fell into. A definition containing one set of information would be generated if the noun were "basic level", and another set of information would constitute the definition if the noun were "abstract" or a member of some other category. In order to distinguish between nouns of different types, Ehrlich's algorithm required that each noun being defined should be explicitly labeled with one of her categories. Beside creating an unnecessary burden on people, and a potentially insurmountable problem for automatic parsers generating input for the algorithm, this technique was probably psychologically inappropriate. When a person creates a definition for a noun, it seems unlikely that they first consider whether the noun is basic, non-basic, abstract, physical, or some other type.

Upon close examination of the different definitions for each of the types of nouns, it became apparent that there were no large differences between information included in any of the definitions. The only significant difference was that some information would be excluded from the definition when it was not appropriate. In the case of an abstract noun, actions would not be reported, because something that is not a physical object cannot perform any actions. However, excluding the category from the definition is no different than filling the category with nil, which is exactly what would happen if the algorithm attempted to find actions for an abstract noun.

Since keeping the different types of definitions would cause unnecessary complication, and the information that they reported was similar, I decided to consolidate all of the different definition types into a single type. The new definition includes all of the information reported by each of Ehrlich's definition types. If some category does not have any information, that category is filled with a nil, and the reporting function filters it out of the definition that is displayed on the screen.

3.2 Definition Modes

Although all of Ehrlich's definition types have been consolidated into one, the new algorithm introduces two new definition modes. These modes are not dependent on the word being defined;

rather, the definition mode is selected by the user when he/she invokes the algorithm.

The default mode is known as "teaching mode"; in this mode, all information that can be found is included in the definition of the noun. This mode is intended to be used for testing the system and for use as an interactive learning aid for students. In its role as an interactive learning aid, the program would provide the student with all the information it can find about an unknown word with the assumption that the student would be able to distinguish between relevant and irrelevant information.

The other mode is "lexicographic mode". This mode follows Ehrlich's original theory of which information should be included in a definition and which information should be disregarded as irrelevant. In this mode, definite information is preferred to probable or possible information, and probable information is preferred to possible information. Therefore, whenever definite information is present, probable and possible information is not reported. Similarly, whenever probable information is present, possible information is not reported. When filling the categories with information, basic-level information is preferred to non-basic-level information, as in Ehrlich's version of the algorithm.

3.3 New Categories of Information

In order to improve the quality of the definition that the algorithm generates, several new categories of information have been added, and some existing categories have been enhanced. Ehrlich's algorithm recognized one-argument properties, such as "red ball" or "small dog," but the algorithm disregarded two-argument relations. Two-argument relations, such as the one represented by node M1 in Figure 7, can be used to represent such information as "freshwater and saltwater mix" and "estuaries deserve protection". The new version of the algorithm now recognizes relations that are represented using the object1-rel-object2 case frame and reports them as a part of the properties, probable-properties, or possible-properties categories. The algorithm still does not recognize relations with more than two arguments. In the future, a method for representing and recognizing relations with three or more arguments should be explored.

Spatial information is an entirely new category of information that has been incorporated into the algorithm. Sternberg (1983), whose categories Ehrlich used as a basis for her own, includes spatial information in his list of types of context clues, but Ehrlich did not include it in her algorithm. Spatial information tells the reader that the unknown word is a place where some action occurs or some characteristic holds true. For example, "an estuary is a place where ducks live" and "an estuary is a place where fresh water and salt water mix" both give the reader spatial information concerning estuaries. Spatial information is represented using the object-location case frame (Figure 7).

It should be noted that the algorithm is only capable of handling information about absolute locations such as an estuary or a house. The algorithm is not capable of finding information about relative locations such as "next to the table" or "above the rooftops". This limitation should certainly be examined and overcome in the future. In addition, the algorithm currently does not contain either probable-location or possible-location slots. All information about location is returned in the single location slot. If it should become obvious that probable and possible spatial information must be separated from definite spatial information, this could be the subject of future work [Yuhan 1995].

The old algorithm included the categories "class inclusions" and "probable class inclusions", but it did not include the category "possible class inclusions". Following the format set up by all of the other types of information with definite and probable slots, a "possible class inclusions" category

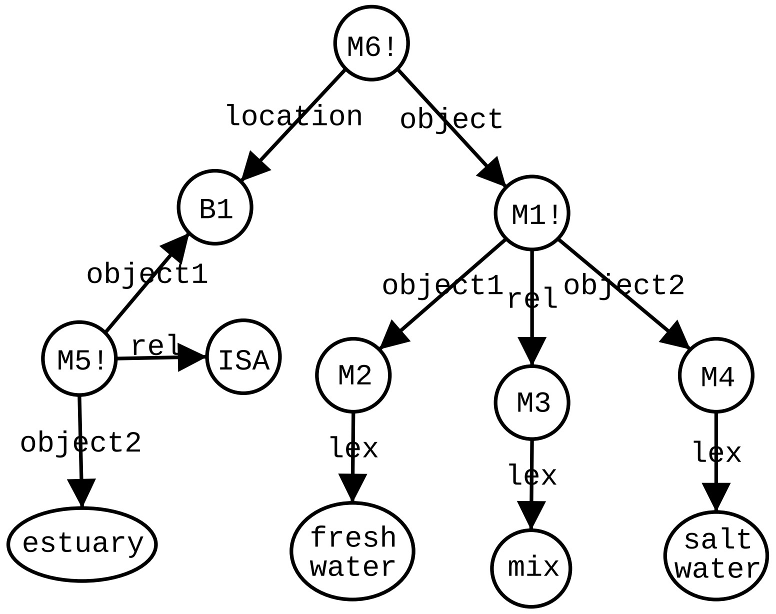

Figure 7: Representation of "In an estuary, fresh water mixes with saltwater" using the object-location case frame. Syntax: If B1 and M1 are individual nodes and M6 is an identifier not previously used, then this figure is a network and M6 is a structured proposition node. Semantics: [[M6]] is the proposition that [[M1]] occurs in or is true in [[B1]]; in this case, fresh water mixed with saltwater in an estuary.

was added. This category contains information about the other class inclusions of individuals who are members of the class we are trying to define. For example, when defining "estuary", if Cassie knows "the Chesapeake Bay is an estuary" and "the Chesapeake Bay is a body of water" (Figure 8), then the algorithm will report "body of water" in the possible class inclusions category.

The information included in "possible class inclusions," like many of the other "possible" categories, is very unreliable. Since this information is based on only one individual, it may be the case that this individual is a member of the several classes, but those classes do not bear any relationship to one another. For example, there could be a person who is both a student and a daughter, but not all daughters are students, and not all students are daughters. Even if the two classes are related, we do not have a way to determine which class is the superclass and which is the subclass (if there had been an explicit subclass-superclass relation, it would have been picked up in the definite-class-inclusions section). To return to our "estuary" example, it might be the case that estuaries are bodies of water (subclass-estuary-superclass-body of water), but it could just as easily be the case that bodies of water are estuaries (subclass-body of water-superclass-estuary).

Despite the uncertainties and possibly misleading information that can be included in the "possible class inclusions" category, it can provide some extra clues when no other information is available.

3.4 Storing the Definition

Ehrlich's original algorithm did not include a mechanism for storing the definition in the knowledge base. Each time the algorithm was asked to define a word, it would consult the knowledge base and build a definition from scratch, as though it had never defined that word before. However, there

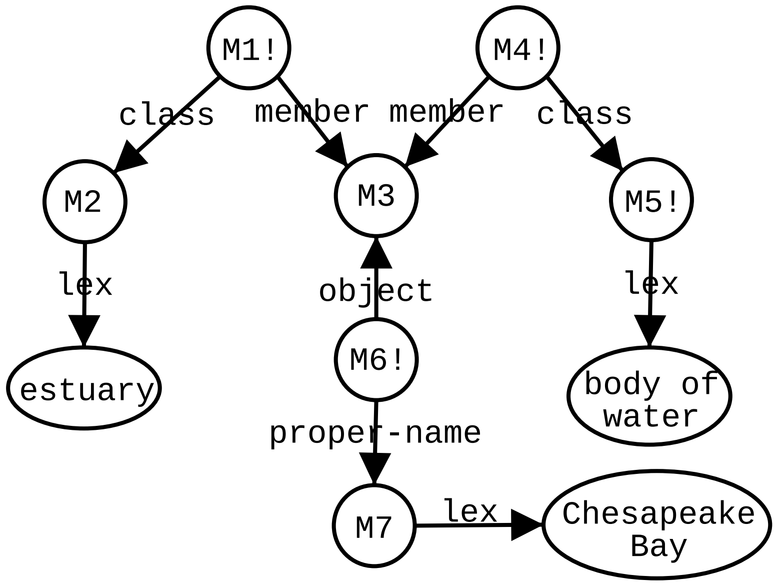

Figure 8: Knowledge that would cause the algorithm to report "body of water" as a possible class inclusion when defining "estuary". [[M1]] = The Chesapeake Bay is an estuary. [[M4]] = The Chesapeake Bay is a body of water.

do seem to be times when humans store definitions rather than reconstructing them from scratch. If someone were to ask you to define "estuary", you would probably need to think for a while to construct a definition, but, if you were asked again a short time later, you would probably be able to repeat your previous definition.

Motivated by the observation that humans do store some definitions, we attempted to implement a mechanism that stored definitions in the knowledge base. Ideally, the stored version of the definition would be reported until some new information would cause the definition to change, at which point a new definition would be generated and stored.

A mechanism for storing definitions and retrieving stored definitions was implemented. Whenever the algorithm was invoked to define a word, it first checked the knowledge base to see if a definition was stored there. If the definition was stored, then the stored definition was reported. If the definition was not stored, then a definition was generated and stored in the knowledge base.

Unfortunately, this procedure did not allow any new information to become part of the definition. The first definition that was generated and stored would always be reported when the algorithm was asked for a definition. This was obviously unacceptable, because the first encounter with a word rarely provides sufficient information to construct an adequate definition. Therefore, we considered several methods for overcoming this problem.

First, we considered generating the definition each time the algorithm was invoked and then comparing the generated definition to the stored definition. If the two definitions were different, the generated definition would be reported and would replace the stored definition in the knowledge base. However, using this procedure would not accurately model the human behavior that we observed. If humans do store definitions, there should be no need to think about them after they have been stored. Similarly, there should be no need for Cassie to "think" about the definition once it has been stored in her knowledge base.

Second, we considered storing the definition for a short period and then removing it from the knowledge base, thereby "forgetting" it. This would more closely model human short-term memory and is the path that we currently consider to be most promising. However, this approach has several shortcomings. If new information is added to the knowledge base, and the algorithm is asked to define a word before the old definition is forgotten, the new information will not be incorporated into the definition. In addition, we do not currently know of any theory of how forgetting takes place. When should a definition be forgotten, and when should it move into long-term memory? Until a reasonable theory of forgetting has been developed, this work will probably not make significant progress.

Currently, work is being done to explore definitions generated after the story that provided context has been forgotten. The method we are using involves running a demo and then removing all the information that was directly given in the story from the knowledge base, leaving any knowledge that was inferred while reading the story intact. Our theory is that even after the story has been "forgotten", a definition can be generated. We believe this is possible because information which is most important to the definition is rarely given directly in the story; the most important information is usually inferred from information in the story and background knowledge.

The solution to the problem of storing definitions, indeed the determination of what is the correct solution, has not yet been resolved. This problem is left to future work.

3.5 Other Changes

In addition to the changes described above, a number of small changes were also made to the algorithm. Ehrlich's algorithm only recognized class inclusions represented using the "subclass-superclass" case frame. The new algorithm also recognizes class inclusions represented using rules of the form "for all x, if x is a member of the class y, then x is a member of the class z". This addition was partially based on Matt Sweeney's work on the "estuary" passage [Sweeney 2002].

The new version of the function findSynonyms does not take probable or possible information into account when determining if two words are synonyms. This change is a result of the difference between the information-retrieval functions of the old and new versions of the algorithm. If future work demonstrates that probable and possible information are vital to accurately finding synonyms, they should be reintegrated into the search functions called by findSynonyms.

Ehrlich's algorithm included categories for function information. The information reported in these categories were actions that were the primary purpose of the noun to perform or to have performed on it. These categories were removed from the new version of the algorithm. Since the information reported by the functions categories was always an action of some type, it should be picked up by either the actions categories or the "agents who act on a noun" category, making the listing under functions redundant.

The majority of the search paths now include checks for asserted nodes in more locations than they did in the original algorithm. This change was made to prevent the algorithm from reporting information that Cassie has considered while performing inference, but ended up not believing. This change should not have any effect on the output of existing demos.

4 Future Work

The immediate next step for this project is to reintroduce automatic belief revision. Ehrlich's demos used arcs labeled "kn_cat" to assign confidence values to each piece of information in the network. Then, when a contradiction was detected, the system automatically chose to keep the belief with the highest confidence level and remove the conflicting beliefs from the knowledge base. Ehrlich used the now outdated SNePSwD system to perform the automatic belief revision. Since SNePSwD is not known to work with the most recent revision of SNePS, and it is no longer actively maintained, the noun-definition algorithm should make use of Fran Johnson's new system for automatic belief revision [Johnson 2000]. It is my understanding that Johnson's system will require the confidence labels ("kn_cat") in the demos to be changed into a new format, but it should not require any changes to the noun-definition algorithm itself.

If this program is to be used as a teaching aide for students, an easily understandable explanation facility must be developed. This explanation facility would probably build upon the already completed tracing facility to develop a simple, step-by-step explanation of how the algorithm helped Cassie to arrive at her definition of the unknown word.

In the long term, more research into storing definitions should be performed. It might be beneficial to develop an experiment with human subjects that explores the conditions that lead people to store definitions and whether there are different ways of storing definitions (in short term memory versus long-term memory, for example). Since SNePS currently has no distinction between short-term and long-term memory and no theory of "forgetting", something that would be essential to correctly model short-term memory, this work is best left to the distant future.

A Running the Algorithm

The following instructions for running SNePS and the noun-definition algorithm assume that the commands are being executed on a computer at the University at Buffalo Department of Computer Science and Engineering with access to the /projects filesystem.

- Type `composer` to start Allegro Common Lisp (or in XEmacs type `M-x run-acl`).
- At the lisp prompt type `(load "/projects/snwiz/bin/sneps")`. Note: This algorithm has been tested with SNePS 2.6.0. The algorithm may or may not be compatible with previous versions of SNePS.
- Type `(load "/projects/stn2/CVA/defun_noun.cl")` to load the noun-definition algorithm.
- Type `(sneps)` to start SNePS.
- Type `(demo "/projects/stn2/CVA/demos/some-demo.demo")` to run a demo. Note: you should replace "some-demo" with the name of the demo you want to run.

B Location of Files

/projects/stn2/CVA/defun_noun.cl
The noun definition algorithm.

/projects/stn2/CVA/demos
Contains all of the demos and supporting files.

References

[Broklawski 2002] Broklawski, M. (2002), "Computational Contextual Vocabulary Acquisition: A Noun Algorithm" [http://www.cse.buffalo.edu/~mkb3/docs/cva.wrapup.pdf].

[Cho 1992] Cho, S. (1992), “Representations of Collections in a Propositional Semantic Network”, *Working Notes of the AAAI 1992 Spring Symposium on Propositional Knowledge Representation.* AAAI.

[Ehrlich 1995] Ehrlich, K. (1995), “Automatic Vocabulary Expansion through Narrative Context”, *SUNY Buffalo Computer Science Technical Report 95-09.*

[Johnson 2000] Johnson, F., (2000), “Automatic Belief Revision is SNePS” [http://www.cse.buffalo.edu/∼flj/kr2000/autoBR-in-sneps.ps]

[Shapiro 1996] Shapiro, S., (1996), “A Dictionary of SNePS Case Frames” [http://www.cse.buffalo.edu/sneps/Manuals/dictionary.pdf].

[Shapiro 1999] Shapiro, S. (1999), “SNePS 2.5 User’s Manual” [http://www.cse.buffalo.edu/sneps/Manuals/manual25.ps].

[Sternberg 1983] Sternberg, R.J.; Powell, J.S.; & Kaye, D.B. (1983), “Teaching Vocabulary-Building Skills: A Contextual Approach”, in A.C. Wilkinson (ed.), *Classrom Computers and Cognitive Science* (Academic): 121-143.

[Sweeney 2002] Sweeney, M. (2002), “Maximizing the Accuracy of the Retrieved Definition of ’Estuary’ from Contextual Information”, [http://www.cse.buffalo.edu/ sweeney4/papers/FinalReport_estuary_spring2002.doc].

[Yuhan 1995] Yuhan, A. and Shapiro, S., (1995) “Computational Representation of Space”, In J.F. Duchan, G.A. Bruder, and L.E. Hewitt (eds.), *Deixis in Narrative: A Cognitive Science Perspective*, pp. 191-225. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.