Progress of the Noun Definition Algorithm During the Fall 2002 Semester

Scott Napieralski
stn2@cse.buffalo.edu

December 17, 2002

Abstract

This document describes work on the Contextual Vocabulary Acquisition project that was conducted during the Fall semester of 2002 under the supervision of Dr. William Rapaport. A detailed description of changes that were made to the noun definition algorithm is given. Some modifications to the demonstration programs and the motivation for these changes is also discussed.

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 work that was performed during the fall semester was to ensure that all of the demonstration programs developed by Karen Ehrlich for her dissertation [Ehrlich 1995] produced at least as much information using the new noun definition algorithm [Napieralski 2002] as they produced using the original noun definition algorithm.

The noun definition algorithm implements a theory of how an unknown word can be learned based on the passages of text in which the unknown word occurs. The information that is contained in a passage is first encoded as a SNePS network [Shapiro 1999]. Then, when asked to provide a definition for the unknown word the algorithm searches the SNePS network for certain types of information. When all the relevant information in the network has been found, the algorithm reports it as a definition for the unknown word.

In addition to the large effort of updating the demos, some changes were also made to the noun definition algorithm itself. The majority of these changes were of a minor nature and were intended to fix errors discovered while testing the demos. There were, however, several changes to the algorithm implemented new features or changed the way that information was retrieved from the network.

2 Changes to the Algorithm

The most significant changes to the algorithm involve the type of information that is retrieved from the SNePS network. The original algorithm searched for both basic level information (represented using the "member-class" case frame) and nonbasic level information (represented using the "object1-rel ISA-object2" case frame). These two different categories were useful for distinguishing information that should be reported in a definition from information

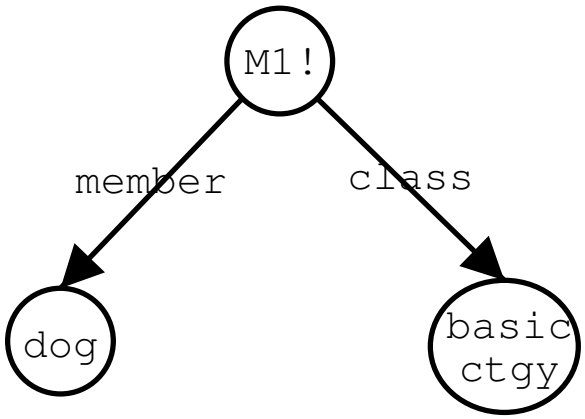

Figure 1: Representation of "Dog is a basic level category" in the current version of the algorithm.

that was either too general or too specific to report as part of a definition. However, having two different ways to represent class membership caused an explosion in the number of search paths that needed to be considered when designing the algorithm. In order to make the code more manageable, we decided to eliminate the "object1-rel ISA-object2" case frame and use "member-class" to represent both basic level and nonbasic level information.

Although the nonbasic case frame was removed from the algorithm it is still necessary to distinguish between information that is basic level and information that is nonbasic level. To solve this problem, we created a "basic ctgy" node and label basic level categories as members of the class "basic ctgy" (Figure 1). In this version of the algorithm, anything that is a member of the class "basic ctgy" is a basic level category. Anything that is not explicitly labeled in this way is assumed to be a nonbasic level category.

This scheme does have some drawbacks. Most significantly, it implies that the mind being modeled by our system is aware of the term "basic level category". Further, the person being modeled by our system is able to identify which categories are basic level and which categories are not. It may also imply that when teaching our method to a human, the

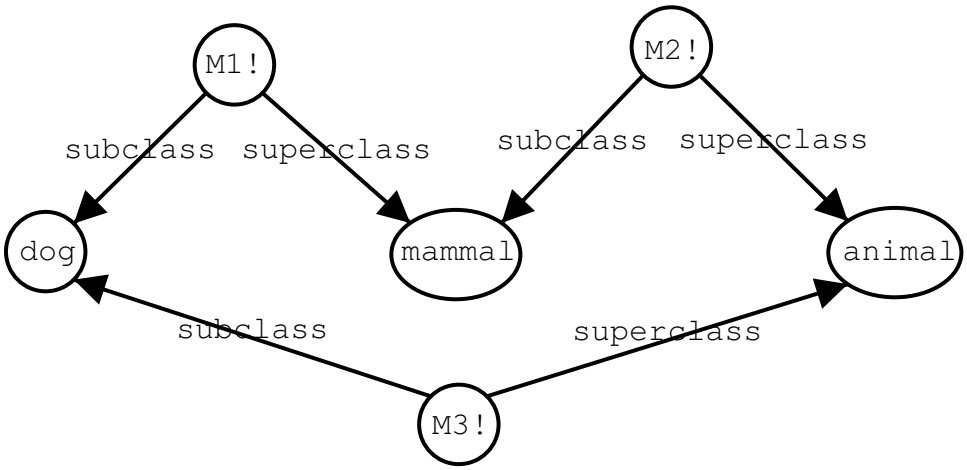

Figure 2: When defining "dog" class_filter eliminates "animal" and reports "mammal".

concept of a basic level category must also to be taught.

If the CVA team decides that it is no longer desirable to use the structure described above, an alternative has been proposed. Stuart Shapiro suggested that it might be possible to use a path of some sort to identify basic level categories. This proposal has not yet been seriously investigated.

After the algorithm was changed to use only "member-class" for class membership, a related change was made to the class_filter function. The purpose of this function is to remove all but the most significant class inclusions from the definition. In the original version of the algorithm, the function worked by searching for the most specific class inclusions and only reporting those inclusions. For example if the unknown noun was "dog" and the network contained the knowledge that "dogs" are "mammals" and "animals" and that "mammals" are "animals" then class_filter would remove "animal" from the list of class inclusions, leaving only "mammal" (Figure 2).

In the new version of the algorithm we retain this behavior, but we add an additional check. The class_filter function first determines whether the unknown noun is a member of

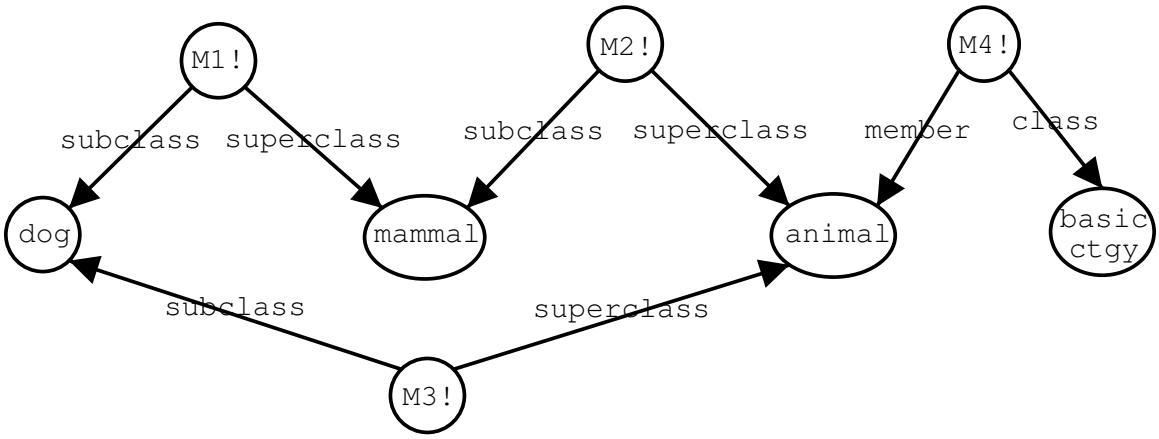

Figure 3: In this case, class_filter would eliminate "mammal" and report "animal" because "animal" is labeled as a basic level category.

any basic level category. If it is determined that the unknown word is a member of some basic category then that category is reported and any other class memberships are not reported (Figure 3). If the unknown word is not a member of any basic level category then class_filter uses its original behavior.

The way that the class_filter function is invoked has also been changed in this version of the algorithm. In the original program, class_filter was called from inside the functions findClassInclusions and findProbableClassInclusions so that whenever this information was used, the filtered form was returned. Unfortunately, this resulted in some unnecessary code duplication, since the section of code that searches for synonyms made use of the same logic but worked with the unfiltered class inclusions list. In order to simplify the code and reuse the logic in findClassInclusions and findProbableClassInclusions, the call to class_filter was moved outside the the class inclusion functions and into the definition function, defineNounTeaching. This scheme allows the algorithm to work with the full list of class inclusions internally while still displaying only the filtered list to the user.

The rest of the changes to the algorithm took place in the section that is responsible for

finding synonyms of the unknown word. This section consisted of two distinct parts. The first part searched for information that was explicitly labeled as a synonym of the unknown word. The second part attempted to find information that shared enough common traits with the unknown word to be labeled a "possible synonym". Since some members of the research group noted that the second type of information was really not a synonym at all, this section was given its own function, findPossibleSynonyms, and its own line in the output, "Possibly similar items".

In order to make the output generated by the findPossibleSynonyms function more closely match the original output, several changes were made. Most importantly, probable and possible information is now being taken into account when searching for possibly similar items. This information was considered by Ehrlich's original version of the algorithm, but it was removed in the initial revision of the code [Napieralski 2002] due to changes in the way information was represented internally in the algorithm. Unfortunately, when probable and possible information was not considered, the findPossibleSynonyms returned incorrect results, therefore it has been reintroduced.

3 Changes to the Demos

The majority of the semester was spent working on the "brachet" demo. The goal of this work was to alter the demo so that it would produce the same output when run with the new algorithm as it had when run with the original algorithm.

The first task that needed to be accomplished was to convert all of the class membership information in the demo from the "object1-rel ISA-object2" case frame to the "member-class"

case frame. When this was accomplished, many of the rules in the background knowledge for the demo became duplicates. The existence of these duplicate rules significantly increased the running time of the demo, so the demo was carefully searched and all duplicate rules were removed.

The brachet demo was finally coerced into giving the correct output by a combination of three factors. First, the background knowledge, which was previously added to the SNePS network using the assert keyword, is now added using the add keyword. This has the effect of triggering forward inference whenever a new rule is added to the background knowledge, rather than when the first piece of information in the demo is added. Second, the algorithm was slightly altered. Originally, when compiling the lists of class inclusions and actions, the network was simply searched for information using find. In the new version of the algorithm, SNePS is first asked whether it can infer any new class inclusions or actions using deduce and then the information is retrieved from the network as in the original algorithm. Finally, the order that information is retrieved from the SNePS network was changed. Actions are now the last piece of information retrieved from the network rather than the second. This change allows the brachet demo to correctly report that brachets hunt after it learns that brachets bay. Unfortunately, it is still not clear why this change makes a difference, determining the reason behind it could be an area of further investigation in the future.

Although the brachet demo does produce correct output, some of the results for the "Possibly Similar Items" category may seem strange. For much of the middle part of the brachet demo this category reports that the possibly similar items for "brachet" are "mammal" and "pony". On first inspection, this does not seem correct, however it does conform with

Ehrlich's theory of what should be included in her "Possible Synonyms" category. According to Ehrlich's theory two nouns are possibly similar if they share two or more class inclusions and they have more class inclusions in common than class inclusions that are different. These rules in combination with the particular set of background knowledge in the brachet demo result in the strange list of possibly similar items described above. This situation could be remedied by adding more class inclusion information for each of the elements in the list so that they would have less in common with brachet.

During the process of modifying the brachet demo, a problem with inferred subclass/superclass relationships was discovered. In the background knowledge for the brachet demo, some combination of rules and paths was leading SNePS to infer "animal is a subclass of animal", "mammal is a subclass of mammal", and so on. This problem was never completely resolved, but it was ameliorated by making some the rules in the background knowledge more specific. This same problem has cropped up in the cat demo. After the system is told "Frisky is a cat", it infers "animal is a subclass of animal". This inference leads the noun definition algorithm (specifically the class_filter function) to eliminate animal as a class inclusion for cat. Unfortunately, there was insufficient time in the semester to investigate this problem further.

4 Future Work

The immediate next step that should be performed by the next researcher to work on this project is to examine the "cat" demo to discover why "animal is a subclass of animal" is being inferred. Once the cause of this error is identified, it should be corrected and the demo

should be tested to determine whether the change has resulted in correct output. Assuming that the problem can be found and fixed, other existing demos should be carefully checked to see if they exhibit the same problem.

Once the "cat" demo has been modified so that it produces the correct output, Ehrlich's other noun demos should be examined. There are two demos, the "hackney" demo and the "tomato" demo, that have not yet been throughly examined.

In the more distant future, researchers might want to consider the possibility of adding the functions category back into the noun definition algorithm. In the most recent version of the algorithm, the functions category, which was present in the original version, was removed because we believed that all the information in that category should also be picked up by the action categories [Napieralski 2002]. However, Stuart Shapiro and J.P. Koenig suggested that there might be some cases where a noun serves a specific function that is not an action. A specific example of this problem is "key". The function of a key is to unlock something, but it is the person holding the key, not the key itself, who performs the action of unlocking. This problem will need to be addressed at some point in the future, but since there are relatively few words that exhibit the problematic property this task should not be given a high priority.

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 Emacs 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.

Online:
http://www.cse.buffalo.edu/ stn2/cva/index.html

References

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

[Napieralski 2002] Napieralski, S. (2002), "An Enhanced Noun Definition Algorithm" [http://www.cse.buffalo.edu/ stn2/cva/summer02/summer-report.pdf].

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