A Measure of Semantic Complexity for Natural Language Systems

Shannon Pollard*and Alan W. Biermann
Department of Computer Science, Duke University
Box 90129, D224, LSRC, Durham, NC 27708-0129
office: (919)660-6583 fax: (919)660-6519
e-mail: shannon@cs.duke.edu

Abstract

This paper will describe a way to organize the salient objects, their attributes, and relationships between the objects in a given domain. This organization allows us to assign an information value to each collection, and to the domain as a whole, which corresponds to the number of things to "talk about" in the domain. This number gives a measure of semantic complexity; that is, it will correspond to the number of objects, attributes, and relationships in the domain, but not to the level of syntactic diversity allowed when conveying these meanings.

Defining a measure of semantic complexity for a dialog system domain will give an insight towards making a complexity measurement standard. With such a standard, natural language programmers can measure the feasibility of making a natural language interface, compare different language processors' ability to handle more and more complex domains, and quantify the abilities of the current state of the art in natural language processors.

1 Introduction

Quantification of task difficulty has been applied to many areas in artificial intelligence, including information retrieval (Bagga, 1997) (Bagga and Biermann, 1997), machine learning (Niyogi, 1996), parsing and grammatical formalisms(G. Edward Barton et al., 1987), and language learning in general (Ristad, 1993). In addition to providing a way of comparing systems, these measures quantify task complexity before a system is built. The goal of this paper is to measure the complexity of domains for dialog processing. With a standard measure of complexity, domains can be compared and analyzed without having to build the dialog system first. This measure would be an indication of the cost, amount of code, accuracy, reliability, and execution time of the finished dialog system specified by the domain. The hope is to have a single number or pair of numbers that correlates strongly with these standard measures.

* Supported by the Defense Advanced Research Projects Agency, CPoF project, Grant F30602-99-C-0060

Specifically, if domain D_1 has complexity C_1 and domain D_2 has complexity C_2 where $C_2 > C_1$, then we would expect D_2 to have a greater cost of software, more lines of code, less accuracy, less reliability, and longer execution time.

Section 2 will describe the difference in semantic and syntactic complexity and explain why we consider each separately. In section 3 we define the terms in the complexity analysis, which is explained in section 4. Sections 5 and 6 discuss how to compute information measures that are needed in the complexity analysis, and in sections 7 and 8 we present future work and conclude.

2 Semantic vs. Syntactic complexity

The complexity measurement described above must be one that takes into account both the semantic and syntactic complexity of the domain. Semantic complexity is the number of "things" that we can talk about in the domain. This will include all the objects in the domain, the attributes of those objects to which one might refer, and the relationships between the objects that the user can express. Syntactic complexity refers to the variety of ways that the user will be allowed to refer to an object, attribute, or relationship. For example, a domain could include only two boys but if the user is allowed to refer to them in many ways (e.g., "Bob", "Jim", "he", "they", "the two boys next to the water cooler at the back of the room"), then the domain is semantically simple but syntactically complex. Likewise a domain with 100 objects that are each referred to only as Object1, Object2, etc... is semantically complex but syntactically simple.

Semantic and syntactic complexities form a tradeoff when it comes to building a language processor for a domain. To build a reliable and accurate processor, the domain must be sufficiently restrained. The more syntactic variety allowed the user, the fewer objects allowed in the domain. So, the more objects in the world, the more restricted the user's grammar and vocabulary. This leads to a tendency to consider the two fronts separately, and then consider a complete complexity measure as a combina-

tion of both. Having measures of syntactic and semantic complexity separately will help to find where the best compromise lies.

This paper addresses semantic complexity only. It therefore does not completely define the complexity measure described in the introduction, but hopefully takes a step toward defining such a measure. Syntactic complexity measures such as grammar perplexity (Cole and Zue, 1995) should augment this semantic measure to give a full complexity measure.

3 Domain Terms

To analyze a domain's complexity, the domain expert must first specify the domain in which the system will work by determining the objects in the domain, each object's attributes, and the relationships between objects. Consider as an example the small domain of a simple army map, where there are a few objects on the map and the user can display, move, and show or set attributes of them. This example will be used to show how to define a domain using the following terms:

Objects are the types of salient things in the domain. They correspond roughly to the subjects and objects of sentences used in the dialog. In the army display domain, the objects will be tanks, troops, bridges, forests, and hills. Notice that a type of object only needs to be specified once at this high level. Bridge is one object in our world, even though the actual program is able to distinguish many different bridges.

Attributes of an object are the things that the program needs to know about the object in order to use it in the domain. They correspond roughly to adjectives that describe the object, or things that distinguish one of the objects from the others of that type. In our example, the domain requires the name and position of the bridge and the material of which the bridge is made. These three pieces of information include everything the system needs to know about any bridge. In the following figure, the attributes of an object are listed underneath each object type.

Classes are objects, attributes, predicates, or other classes that are grouped together. A class can act as an object in the sense that it can have a name and have relationships with other objects. In our example domain, we will want to distinguish objects that can move from those that cannot, i.e., a MobileObject class as a grouping of Tanks and Troops. There are always three trivial classes: the class of all objects, all attributes (of all objects), and all predicates.

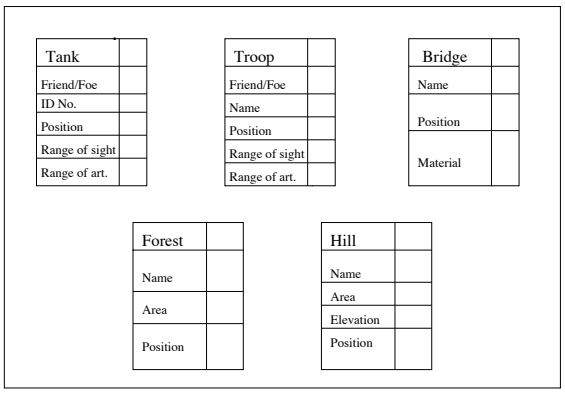

Figure 1: Example Domain Objects and Attributes

Predicates are the relationships between the objects in the world. Any meaning that the user can convey using one or more of the objects should be represented by a predicate. They correspond to the relationship words, like the verbs and prepositions in a sentence, and one can usually find the predicates needed from looking at the allowed operations. For the example domain, the following is the list of allowable predicates, in a typical programming language format to distinguish predicates from arguments.

Display(Object) ["Display the tanks"]
Move(MobileObject,Object) ["Move Troop at position 100, 400 to the hill"]
Show(Attribute,Object) ["Show the range of sight of Tank 434"]
Set(Object,Attribute,Attribute) ["The forest has an area of 100 square yards."]

Notice that classes can be written as predicate arguments to mean that any object in the class can be an argument. Specifically, the Object type refers to all objects, MobileObject refers to either Tank or Troop, and Attribute refers to any object's attribute.

4 Complexity Formulas

Now that the domain is specified, we can analyze its semantics by estimating the number of bits of information conveyed by referring to each different aspect of the domain. This is common in information theory (Ash, 1965); that is, when the user makes a statement, it must be encoded, and the number of bits needed to encode the statement is a measure of its information content. Since the number of bits required to encode a statement in a given domain corresponds directly to the number of salient objects, this information measurement is useful in assigning a semantic complexity measurement.

To get a complexity measure for an entire do-

main, we begin at the lowest level and make counts corresponding to the information content described above. The counts from lower levels are combined to give a higher level count. Specifically, first each attribute value for a specific object is computed, then attribute values are combined to give object values, which are combined to give class values, and so forth until a value for the entire domain is computed.

Define $B(X)$ to be the number of bits conveyed by an instance of random variable X, and $|X|$ to be the number of possible values of X. (Possible ways of computing $B(X)$ will be given in the next sections.) The random variable will represent different events, depending on where we are in the complexity analysis, but in general, the variable will represent the specification of possible attributes, objects, classes, or predicates.

We start by defining the complexity of a single attribute for a single object. We give the formulas for computing the different levels of complexity (attribute level, object level, etc) and then work through the example domain.

The complexity of attribute i for object j, denoted AC_{att_i,obj_j} is

$$AC_{att_i,obj_j} = B(A)$$

where A is the specification of an attribute value.

The object complexity of object j is the sum of all its attributes' complexities:

$$OC_{obj_j} = \sum_i AC_{att_i,obj_j}$$

A simple sum is used because identifying one object uniquely corresponds to knowing each of its attributes. Therefore, the sum of the attribute information is the same as the complete object information.

Since objects can be grouped together into classes, a class complexity is the number of bits conveyed by distinguishing one type of object from that class, plus the maximum object complexity that occurs in that class:

$$CC_{class} = B(O) + \max_{obj \epsilon class}(OC_{obj})$$

where O is the specification of an object in *class*.

When a member of a class is specified, the amount of information conveyed is equal to the information in the object type specification ($B(O)$), plus the information conveyed by the actual object itself. The most that can be is the maximum object complexity in the class. Classes of predicates and attributes are defined in the same way.

For each predicate, the complexity is the sum of the complexities of its arguments:

$$PC_{pred} = \sum_{class \epsilon arg} CC_{class}$$

This is the same as the object complexity as a sum of the complexities of its attributes.

In general, predicate arguments will be classes. If a single object is the only possibility for an argument rather than a class of objects, then the object complexity can be used. This would be the same as making a class of one object: the class complexity of one object is equal to the complexity of the one member of the class.

The entire domain's semantic complexity is then the same as the complexity of the class of all predicates defined for the domain. Specifically, for a domain with a set of predicates P, the semantic complexity $SEMC$ is

$$SEMC = B(P) + \max_{pred \epsilon P} PC_{pred}$$

where P is the specification of a predicate in the domain.

Any statement that the user can make should correspond to some predicate in the domain model. The information given in the sentence is the information given by the predicate specification ($B(P)$) plus the information given in the arguments to the predicate, which is as much as the greatest predicate complexity.

5 Using Equal Probability Assumptions

Now we find a formula for $B(X)$, the bits of information conveyed when referring to certain parts of the domain. For the army map example, we assume that all objects are equally likely to be referred to, and all attributes, classes, and relationships are also equally likely. So a troop is as likely to be referred to as a tank, or as a forest, etc. Also, a tank on the map is equally likely to be friend, foe, or unknown. Every value for the attributes will be equally likely.

Under this assumption, the number of bits of information conveyed by referring to one entity out of v possible entities is $log_2 v$. That is, for the equally probable case, $B(X) = log_2|X|$.

Now we fill in the table from Figure 1, beginning with attribute values. A domain expert would decide how many different values are allowed for each attribute. In this example, we will specify that Tank's Friend/Foe value is either friend, foe, or unknown - three possibilities.

$AC_{Friend/Foe,Tank} = log_2 3 \approx 2$

Assuming that there are 128 ID number possibilities, 65,000 positions, and 1,000 possible ranges, and assuming equal probability, we take the *log* of each number and fill in the complexity beside each attribute for that object. Following the hierarchy, we now add the attribute complexities to get the complexity of the tank object.

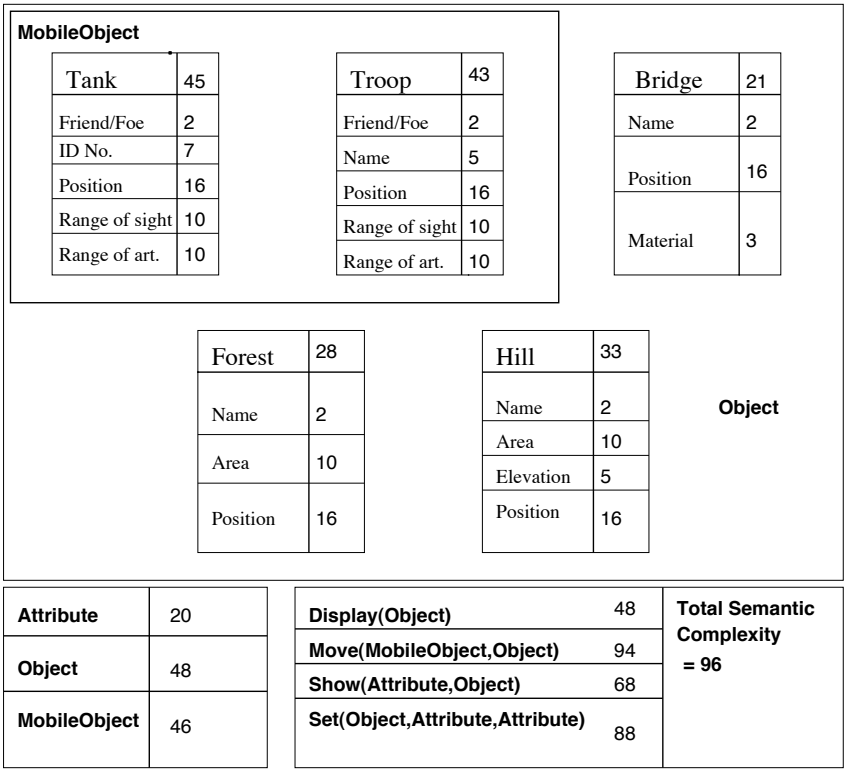

Figure 2: Map Domain Complexity Analysis under the Equal Probability Assumption

Now we have $OC_{tank} = 45$ and let's say in like manner we get $OC_{troop} = 43$. These two types of objects comprise the MobileObject class, so now we can compute this complexity:

$$\begin{aligned} CC_{MobileObject} &= \log_2 2 + \max_{obj \epsilon MobileObject}(OC_{obj}) \\ &= 1 + OC_{tank} \\ &= 46 \end{aligned}$$

Similar formulas are used for predicate and complete domain complexity measurements, and the rest of the example should be obvious from Figure 2.

6 More General Information Measurement

In most cases, the equal probability assumption will not hold. For example, the bridges in the domain can be made of any of eight materials, but if all of the visible bridges are made of wood, then the Material attribute for Bridge will probably be wood most of the time. In this case, referring to the "wooden bridge" on the map doesn't give much more information than just "bridge." For this more general case, define $B(X)$ to be $B(X_1, X_2, ...X_n)$ where each X_i is a possible value of X. Also define $p_1, p_2, ...p_n$ to be their associated probabilities. Then

$$B(X_1, X_2, ...X_n) = -\sum_{i=1}^{n} p_i \log p_i$$

These probabilities can be determined using frequency counts from sample dialogs, or estimated based on domain knowledge.

7 Future Work

The next step in this research is to obtain several domains that have been built into a dialog system and analyze them. The Circuit Fix-It Shoppe(Smith and D.R.Hipp, 1994) has been analyzed, but the results will only be interesting in comparison to other real domains. This comparison will not only help us verify the correctness of the analyses, but also bring up possible situations that the analysis may not cover.

Next, we will want to identify a measure of syntactic complexity. This could be related to grammar perplexity. It should take into account vocabulary size, grammar constraints, and the amount of ambiguity in the grammar. We would like to be able to analyze the domains with both the semantic complexity and the syntactic complexity, and see that the results match our intuitions of complexity and the standards of lines of code, reliability, cost of software, and execution time. We would also be

interested in observing the correlation between the syntactic and semantic complexities.

8 Conclusion

This paper describes a way to organize the objects, attributes, classes, and relationships in a domain and to use these classifications to define a semantic domain complexity. This measurement, along with a syntactic complexity measurement, will give natural language programmers a way to quantify the complexity of a given domain in terms of real-world costs: cost of software, reliability, accuracy, and execution time. After defining a syntactic complexity measure, domains can be analyzed against these real costs to be sure that the measure is accurate. Such a measure will allow natural language systems programmers a way to analyze domains and estimate the costs of building a natural language system beforehand, based on the domain's semantic and syntactic constraints. A standard complexity measure will also allow a comparison of different language processors' ability to handle more and more complex domains and quantify the abilities of the current state of the art in natural language processors.

References

Robert B. Ash. 1965. *Information Theory.* Interscience Publishers.

Amit Bagga and Alan W. Biermann. 1997. Analyzing the complexity of a domain with respect to an information extraction task. *Proceedings of the tenth International Conference on Research on Computational Linguistics (ROCLING X)*, pages 175—94, August.

Amit Bagga. 1997. Analyzing the performance of message understanding systems. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS '97)*, pages 637—40, December.

Ron Cole and Victor Zue. 1995. Survey of the state of the art in human language technology, November.

Jr G. Edward Barton, Robert C. Berwick, and Eric Sven Ristad. 1987. *Computational Complexity and Natural Language.* The MIT Press, Cambridge, Massachusetts.

Partha Niyogi. 1996. *The Informational Complexity of Learning from Examples.* Ph.D. thesis, MIT.

Eric Sven Ristad. 1993. *The Language Complexity Game.* MIT Press.

R.W. Smith and D.R.Hipp. 1994. *Spoken Natural Language Dialog Systems: A Practical Approach.* Oxford University Press, New York.