

The Case of the Unknown Word:

Imposing Syntactic Constraints on Words Missing from the Lexicon

Stan C. Kwasny and Barry L. Kalman

Center for Intelligent Computer Systems[†]
Department of Computer Science
Washington University
St. Louis, Missouri 63130
sck@wucs1.wustl.edu
barry@wucs1.wustl.edu

1. Introduction

Natural Language Processing (NLP) is not an easy problem. This is true even when all the words in the sentence being processed are expressed in carefully typed text with a lexical entry for each word. But no lexicon could be expected to contain entries for every possible English word, given the dynamic nature of language and the creativity of humans. Nonetheless, researchers continue to build language processors that assume the existence of such an exhaustive collection. Using conventional means and under the best of circumstances, sentence processing becomes virtually an impossible task without access to part-of-speech and other information typically provided by the lexicon. Even to produce a simple syntactic analysis of a sentence, a conventional parser requires knowledge of syntactic categories and features of the words.

While parsing techniques vary in the degree to which they rely upon a lexicon, the lexicon provides, at a minimum, much of the knowledge required to determine proper categorization and usage of the words during sentence processing. Words in closed categories could all be included quite easily due to the bounded nature of the collection. Closed categories are also called structural, formal, or grammatical categories and include prepositions, auxiliary verbs, determiners, etc. However, words in open categories could not be included as completely. Open categories are also called lexical or material categories and include nouns, verbs, adjectives, etc. Thus, even a very large lexicon will not overcome this problem.

This research investigates to what extent categorical and other lexical properties can be derived from the syntactic context of an unknown word during parsing. That is, can a parser be constructed that is robust enough in its ability to process sentences that it can overcome the ambiguity caused by a single occurrence of an unknown word and produce a reasonable parse structure? As a by-product, can the unknown word be classified according to the role it plays in the structure and, in the process, can the skeleton of a new lexical entry be formulated?

We are investigating a hybrid deterministic parser to answer these and other questions. In our model, the rules of deterministic parsing have been captured in a neural network. The model is quite capable of following the rules to construct a parse in any situation in which the rules would perform similarly. If the input sentence form does not obey the rules precisely, however, the neural network can overcome some modest variations.

[†] The sponsors of the Center are McDonnell Douglas Corporation and Southwestern Bell Telephone Company.

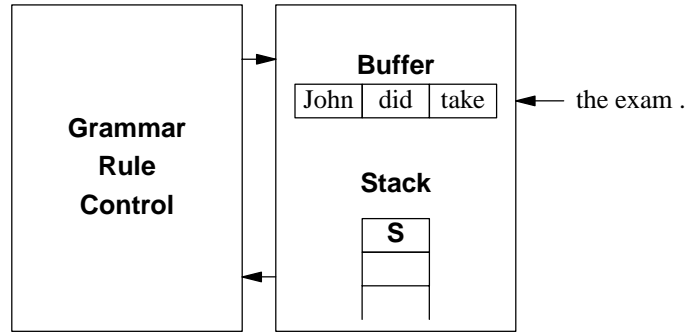


Figure 1: Elements of Deterministic Parsing

2. Hybrid Deterministic Parsing

The Deterministic Parsing model (also called Wait-and-See Parsing) was first introduced by Mitchell Marcus in a rule-based system called PARSIFAL (Marcus, 1980). By permitting the parser to look ahead in the input up to three constituents, parsing is performed deterministically. The constituents are held in a three-place buffer designated for that purpose. A stack is present to permit the recursive processing of embedded structures and to facilitate processing generally. Actions can be performed on the stack and buffer during parsing which build structure and move constituents among the stack and buffer positions. Rules are partitioned into packets which become active or inactive during parsing, but are usually associated with the current (top-level) node of the structure being built. This structure is held on the top of the stack. Figure 1 illustrates the basic elements of a deterministic parser and is shown in the initial configuration for the sentence “John did take the exam.” During parsing, a single processing step consists of selecting a rule from an active rule packet and firing the rule by performing its action. Conflicts are resolved from the static ordering (priority) of rules within the packet. The action potentially makes changes to the stack and buffer and, after a series of processing steps, a termination rule fires which ends the processing. The final structure is left on top of the stack.

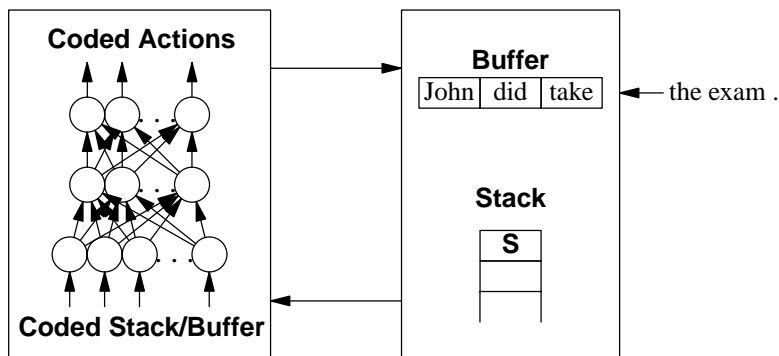


Figure 2: Hybrid Deterministic Parsing

In our hybrid connectionist version of the deterministic parser, the stack and buffer are maintained symbolically exactly as in PARSIFAL. The grammar rule control, however, is replaced by a neural network trained with patterns generated from the rules of the grammar. The network is a layered, feed-forward network similar to ones trained using back-propagation (Rumelhart, et al, 1986), however, we use a modified conjugate-gradient method (Kalman, 1990). We have investigated several training strategies (Kwasny & Faisal, 1990), although space does not permit any further discussion of those techniques here. Figure 2 shows the replacement of the rules with the network.

The buffer and stack of the model are coded for presentation to the network on each iteration of processing. Activation on the output layer of the network is interpreted in a winner-take-all manner and the winning unit designates a rule to execute. The grammar is formulated so that each rule has exactly one action to perform. The network determines which rule, and therefore which action, to perform. In previous versions of our system, the output layer designated an action. This change enables better tracking of the performance of the system by matching the hybrid against the symbolic, deterministic system. It also provides a basis for explanation capabilities that were previously possible only symbolically.

The network can be viewed as defining a mapping from the encoded buffer and top of stack to an action. This mapping is a total function since it has a result for all possible inputs. This is in contrast to the symbolic, rule-based version of the deterministic parser in which the rules do not define a total function since parsing terminates if no rule applies.

Under this scheme, the deterministic parser becomes much more fault-tolerant and one capable of finding syntactic structure even when confronted with imperfect inputs, such as ones containing an unknown word. The exact extent of its robustness is the question that drives this research.

3. Results

Lexical ambiguity is an important problem for a deterministic parser (Milne, 1986; Hindle, 1989). Our model possesses the capability to handle two-way lexically ambiguous items in a sentence (Kwasny & Faisal, 1990). The case of the unknown word is viewed as an extension to that work in which two-way ambiguities have been extended to multi-way ambiguities. An unknown word is totally ambiguous among all categories, features, and classifications required to process it as part of a sentence. The absence of the word from the lexicon makes it impossible to determine any necessary characteristics of the word.

For consistency with our model, an encoding scheme for unknown words is required so that when an unknown word appears in the buffer, it can still be coded for presentation to the network. Furthermore, the encoding scheme chosen is globally defined and should be free from all biases. A coding scheme that appears too similar to a verb, for example, would result in unknown words too often being treated like a verb, even in very unlikely situations. This can lead to extremely disastrous parsing attempts. One of the strengths of our model is the ability to specify items ambiguously using a pattern of activation rather than a set of conflicting features.

Finding a completely unbiased representation for an unknown word has proven to be rather difficult, though not impossible. Once a network is trained and its performance verified for grammatical sentences, experiments are conducted to test its performance on sentences containing an unknown word. Representation choices for an unknown word, which we call the “default code,” can then be evaluated across those situations.

John did <clobber> Mary .	(<i>verb</i>)
John was scheduling <with> Mary .	(<i>preposition</i>)
John and Mary <are> kissing .	(<i>auxiliary verb</i>)

Figure 3: Sentence Forms with Unknown Words (shown in angle brackets < >)

Using a 24-rule grammar as the basis of training, we conducted these experiments. The sentences chosen contain one unknown word each and represent a variety of situations in which the unknown word is required to act like a main verb, a preposition, or an auxiliary verb. Three sample sentences are shown in Figure 3.

From extensive experimentation, we know that a correct choice for the default code in the sentence examples shown consists of a representation in which all units are off (set to a value of -1) except units representing the verb, auxiliary, and preposition feature of the word. The verb and auxiliary units are turned on (set to a value of +1) while the preposition unit is set to +2. (Note that the input layer permits activation values outside the -1 to +1 interval.) This encoding represents an entity that has the potential to be any of the three categories. The need to over-represent the potential of a preposition in the default code is attributed to the sparseness of training patterns containing prepositions used during training.

Perhaps more interesting is what did not work as an appropriate default code. Several choices were explored that represent many ideas and their variations. Setting all units off (-1) seemed to make sense, but that choice for an encoding was already being used to represent an empty buffer position. Another choice explored (along with variations) was to set all units to zero (a middle value) with the hope that other biases would take over and determine how the word should be used. As it turned out, zero made all units look too close to “on” and all unknown words then looked too much like verbs since verbs have the most subcategorization and, therefore, the most units turned on in their encoding.

Many default code choices were tested. Finding one that worked for two of the three sentence cases was generally quite easy, but very few worked for all three. In studying the reason for this, we concluded that the model was being asked to perform well beyond that for which it had been trained. We were, in effect, asking more of the model than was reasonable. There is, in fact, no real basis for the model to generalize in the exact ways we wished since training did nothing specific to suggest how the model should perform under those circumstances. This led us to consider a modification in the training strategy. So, our next set of experiments centered on how best to modify the training to provide the capability desired. In essence, the network had to be trained to expect the unexpected. The answer, as we discovered, was quite simple.

First, we needed to choose a default coding which was distinct from all other codings used in the system. For the experiments reported here, we chose a coding of zeroes for units representing verbs, auxiliaries, and prepositions, and -1 for all the rest. This gave partial activation to the important units and none to the other units. Next, we modified training so that, in addition to the ordinary training patterns, the rules would be used to generate training patterns containing items that were coded using the default coding that we chose. These additional training patterns have the effect of forcing the precise generalization capability we desired without requiring the lexicon to provide definitions for all words. The representative sentence forms worked easily with the network trained in this way.

4. Conclusions

While good generalization capabilities in neural networks are often attained at no additional cost, there are situations in which the nature of the generalization must be shaped through training rather than relegated to chance. Our efforts have suggested a method for training a network to perform a particular kind of generalization when it recognizes that an unknown word is present in the input form.

No lexicon can contain every possible word or word sense. Constraints imposed by syntax and other components of a NLP system should be useful in determining features of unknown words. We have investigated the utility of syntactic constraints in constructing a syntactic interpretation of sentences that contain unknown words.

While there are choices for the representation of unknown words that permit successful processing of such sentences, our investigation argues rather for a modified training schedule that equips the processor with the ability to look for an unknown word and impose syntactic roles from the context of the unknown word. It is a strength of our hybrid deterministic parser that a single, globally-defined, ambiguous encoding lends itself to such processing.

5. Acknowledgements

We thank Laura McCarthy for her preliminary investigation of this problem during an undergraduate research assistantship in summer, 1990. We also thank the sponsors and members of the Center for Intelligent Computer Systems (CICS) at Washington University for their earnest support and interest in our work.

References

- Hindle, Donald. (1989). "Acquiring disambiguation rules from text." *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, (pp. 118-125). Vancouver, British Columbia.
- Kalman, Barry L. (1990). *Super linear learning in back propagation neural networks* (Technical Report WUCS-90-21). St. Louis, MO: Department of Computer Science, Washington University.
- Kwasny, S.C. and K.A. Faisal. (1989). "Competition and learning in a connectionist deterministic parser." In *Proceedings of the 11th Annual Conference of The Cognitive Science Society*, (pp. 690-697). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kwasny, S.C., and Faisal, K.A. (1990). "Connectionism and determinism in a syntactic parser." *Connection Science*, 2, 63-82.
- Marcus, M. P. (1980). *A theory of syntactic recognition for natural language*. Cambridge, MA: MIT Press.
- Milne, R. (1986). "Resolving lexical ambiguity in a deterministic parser." *Computational Linguistics*, 12, 1-12.
- Rumelhart, D. E., G. Hinton, and R.J. Williams. (1986). "Learning Internal Representations by Error Propagation." In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing*, Cambridge, MA: MIT Press, 318-364.